Spring Framework: Critical Vulnerability Spring4Shell

https://campus.barracuda.com/doc/96770678/

This article provides updates on recently discovered vulnerabilities (CVE-2022-22963 and CVE-2022-22965) in Spring Framework.

The following table provides key information about the vulnerabilities.

CVE Number	Commonly Known/Associated As	Criticality & CVSS Score	Exploit Type	Software Firmware Versions	Prerequisite to Exploit Vulnerability	Barracuda WAF Affected
CVE-2022-22965	Spring4Shell Relates to old CVE-2010-1622	Zero-day	RCE	Spring MVC and Spring WebFlux applications running on JDK 9+	Application running on Tomcat as WAF deployment	NO
CVE-2022-22963	SpEL (Spring Expression Language)	Critical	ELV->RCE	Spring Cloud Function versions : 3.1.6, 3.2.2 and older unsupported versions		NO

Description

Spring Framework is an application framework and inversion of control container for the Java platform. Recently, two vulnerabilities were discovered in Spring Framework (CVE-2022-22965) and in Spring Cloud Function (CVE-2022-22963).

Spring4Shell is a misnomer for all these vulnerabilities combined (CVE-2022-22965, CVE-2022-22950 & CVE-2022-22963). Spring4Shell refers to CVE-2022-22965. Also, note that Spring4Shell has no relation with the log4shell vulnerability.

The following sections list the difference between these vulnerabilities, along with their effects and mitigation.

CVE-2022-22963

Description

CVE-2022-22963 was reported on March 29, 2022. It affects Spring Cloud functions only, which is not in Spring Framework. Spring has already released a newer version to take care of this.
CVE-2022-22963 uses routing functionality to provide specially crafted Spring Expression Language (SpEL) as a routing expression to access local resources and perform RCE. It uses a specific HTTP request header: spring.cloud.function.routing-expression.

Barracuda WAF is not affected by this vulnerability.

Exploit

This is an RCE, and a malicious actor can provide a specially crafted SpEL as a routing expression that may result in access to local resources.

Mitigations

- On the Barracuda WAF, you can manually perform the configuration changes mentioned under **Barracuda WAF Manual Mitigation Configuration** to protect against this vulnerability

Barracuda WAF Manual Mitigation Configuration

Enable “OS Command Injection” and “OS Command Injection Strict” patterns on the **Security Policies > Parameter Protection** page.

Additional Configuration Steps

- Create a header ACL with the header name of *spring.cloud.function.routing-expression* and ensure the *os-command-injection-medium* and *os-command-injection-strict* are associated with this header ACL for inspection.

This may result in some false positives, depending on how the application names other parameters. The administrator can create the pattern initially in passive mode and evaluate the Web Firewall Logs generated.

Recommendation

You can update your infrastructure as follows:

- Vendor Advisory: https://tanzu.vmware.com/security/cve-2022-22963
- Users of affected versions should upgrade to 3.1.7, 3.2.3. No other steps are necessary. Releases that have fixed this issue include:
- Spring Cloud Function

  - 3.1.7
  - 3.2.3

CVE-2022-22965

Description

This vulnerability affects Spring MVC and Spring WebFlux applications running on JDK 9+. The specific exploit requires the application to run on Tomcat as a WAR deployment and will not work if the Spring Boot executable is in jar deployment. So by default, the deployed application is not vulnerable to this exploit.

Barracuda WAF is not affected by this vulnerability.

Exploit

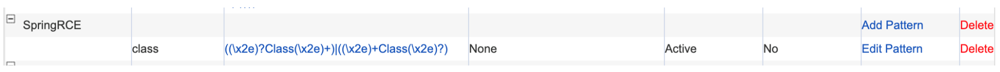
This is an RCE vulnerability, in Spring Core version 5.3.17 or earlier (for 5.3.x) and version 5.2.19 or earlier (for 5.2.x). It appears to be a bypass of protections set up for CVE-2010-1622.

Mitigations

- On the Barracuda WAF, you can manually perform the following configuration changes to protect against this vulnerability.

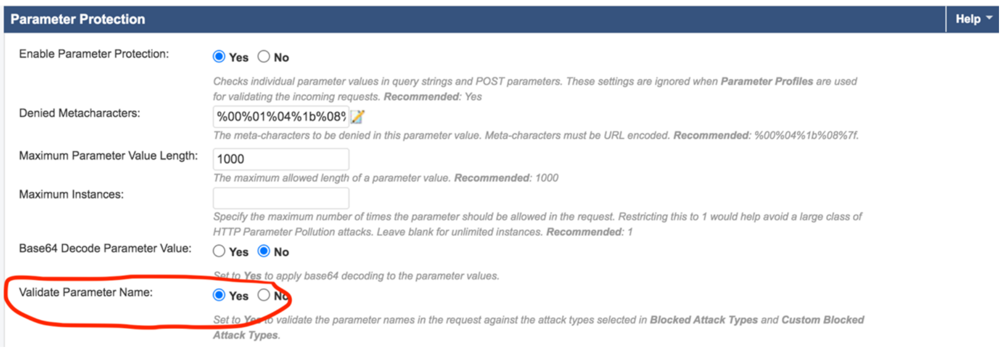
Configuration Steps

- As a very strict approach, it is advised to create a pattern ((\x2e)?Class(\x2e)+)|((\x2e)+Class(\x2e)?) and use this pattern to validate the parameter name. Set **Case Sensitivity** to **No**.



⊟ SpringRCE						Add Pattern	Delete
	class	((\x2e)?Class(\x2e)+)|((\x2e)+Class(\x2e)?)	None	Active	No	Edit Pattern	Delete

  Then, bind it to the Custom Blocked Attack Types on the **Parameter Protection** page. For this to work, you must scroll down to the **Validate Parameter Name** setting and select **Yes**.



Parameter Protection Help

Enable Parameter Protection: [✓ Yes] No

*Checks individual parameter values in query strings and POST parameters. These settings are ignored when **Parameter Profiles** are used for validating the incoming requests. **Recommended**: Yes*

Denied Metacharacters: %00%01%04%1b%08%

*The meta-characters to be denied in this parameter value. Meta-characters must be URL encoded. **Recommended**: %00%04%1b%08%7f.*

Maximum Parameter Value Length: 1000

*The maximum allowed length of a parameter value. **Recommended**: 1000*

Maximum Instances:

*Specify the maximum number of times the parameter should be allowed in the request. Restricting this to 1 would help avoid a large class of HTTP Parameter Pollution attacks. Leave blank for unlimited instances. **Recommended**: 1*

Base64 Decode Parameter Value: Yes [✓ No]

*Set to **Yes** to apply base64 decoding to the parameter values.*

Validate Parameter Name: [✓ Yes] No

*Set to **Yes** to validate the parameter names in the request against the attack types selected in **Blocked Attack Types** and **Custom Blocked Attack Types**.*

- As a general recommendation, we suggest to keep os-command-injection-medium and strict patterns enabled while keeping an eye on possible false positives.

This may result in false positives, depending on how the application names other parameters. Administrators can configure the pattern initially in passive mode and evaluate the Web Firewall Logs generated.

Vendor advisory: https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement

- Spring Framework 5.3.18 and 5.2.20, which contain the fixes, have been released.
- Spring Boot 2.6.6 and 2.5.12 that depend on Spring Framework 5.3.18 have been released.
- Apache Tomcat has released versions 10.0.20, 9.0.62, and 8.5.78, which close the attack vector on Tomcat’s side. See Spring Framework RCE, Mitigation Alternative.

Further Reading:

- https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement
- https://spring.io/blog/2022/
- https://thehackernews.com/2022/03/security-patch-releases-for-critical.html
- https://securityboulevard.com/2022/04/critical-alert-spring4shell-rce-cve-2022-22965-in-spring/
- https://www.databreachtoday.com/springshell-spring-cloud-function-bugs-need-urgent-patching-a-18822?rf=2022-04-01_ENEWS_ACQ_DBT__Slot1_ART18822&mkt_tok=MDUxLVpYSS0yMzcAAAGDhN0UVdOzrHrKZ5NyWSdphDsc9RZujJR2Ql6F_wl1E76Mg_Jl9KwJ1UwpdfwvOvnl_wfRLxm9p9ZCw5fXY2mVj8lKo2bgINjkzUq4o191m9OkRoKzMQ

Figures