spara

Spring4Shell RCE Vulnerability Analysis

Release date: 2022/04/06

Contents

1. Overview

As one of the world's most popular Java lightweight open-source frameworks, Spring allows developers to focus on business logic and simplifies the development cycle of Java enterprise applications.

In recent days, the "Spring4Shell" zero-day vulnerability was published, and by exploiting that, the attacker could execute any code remotely without authentication. Due to the popularity and the release of the framework's exploit, most researchers and security companies have expressed concerns about this vulnerability.

Exploitation requires an endpoint with DataBinder enabled (e.g. a POST request that decodes data from the request body automatically) and depends heavily on the servlet container for the application. For example, when Spring is deployed to Apache Tomcat, the WebAppClassLoader is accessible, which allows an attacker to call getters and setters to ultimately write a malicious JSP file to disk. However, if Spring is deployed using the Embedded Tomcat Servlet Container, the classloader is a LaunchedURLClassLoader which has limited access.

Attempts to exploit Spring4Shell, which is typically intended to deliver a **webshell**, have also been observed by Palo Alto Networks, the SANS Institute and Kasada. The attacks appears to have started before the fixes were released. Chinese company "Qihoo 360" also reported seeing exploit attempts in the past week, including by the Mirai botnet, but the company’s blog post appears to have been deleted, which could indicate an analysis is underway.

CVE	CVE-2022-22965
CPE	• cpe:/a:pivotal_software:spring_framework • cpe:/a:vmware:spring_framework
Affected Spring Frameworks	< 5.2.20 5.3.x < 5.3.18
Risk Factor	Critical
Exploit Prerequisites	• JDK 9 or higher • Apache Tomcat as the Servlet container • Packaged as WAR • spring-webmvc or spring-webflux dependency • Spring Framework versions 5.3.0 to 5.3.17، 5.2.0 to 5.2.19، and older versions
Mitigation	Upgrade to Spring Framework version 5.2.20 or 5.3.18 or later
Patch Publication Date	3/31/2022

2. Affected Software and Vendors

Companies are evaluating the impact of the Spring vulnerability, dubbed Spring4Shell, on their products. While some vendors have started releasing patches, many have determined that their products do not appear to be affected. The developers of Spring which is owned by VMware, announced patches for vulnerabilities last week.

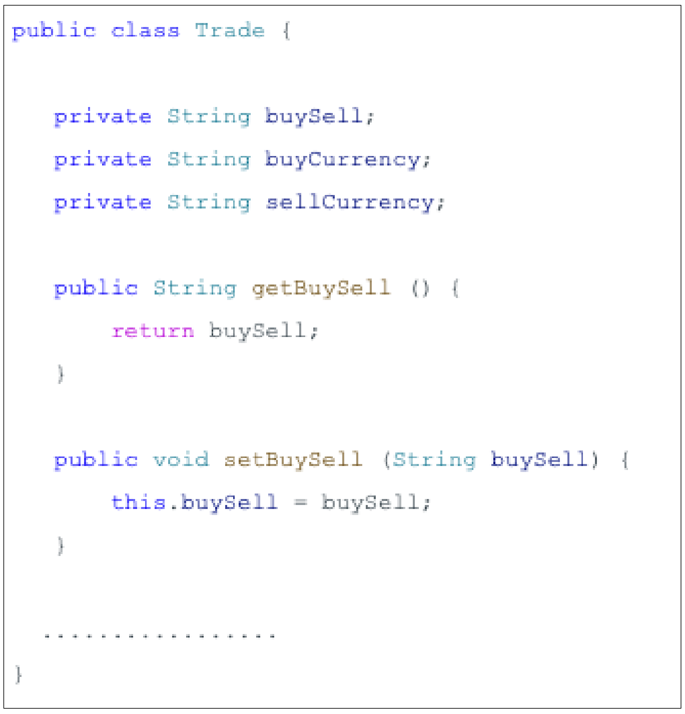
Vendor	Is Affected	More Information
Vmware-Tanzu	Yes	Vmware
Atlassian	Yes	Atlassian
Cisco	Yes	Cisco
PTC	Yes	PTC
Blueriq	Yes	Blueriq
JAMF	Yes	JAMF
Red Hat	Decision Manager 7, JBoss A-MQ 7, JBoss Fuse 7, Process Automation 7 and Virtualization 4	Red Hat
Debian	Yes	Debian
ForgeRock	No	ForgeRock
SonicWall	No	Sonicwall
Commvault	No	Commvault
Sangfor	No	Sangfor
Veritas	No	Veritas
Acunetix	No	Acunetix
Okta	No	Okta
Jenkins	No	Jenkins
Dynatrace	No	Dynatrace
Metabase	No	Metabase
Aerospike	No	Aerospike
McAfee	No	
F5	No	F5

3. Mitigation to fix the Vulnerability

The preferred response is to update to Spring Framework **5.3.18** and **5.2.20** or greater. If you have done this, then no other workarounds are necessary. However, in some cases upgrading is not quickly applicable. For such cases the following links could help to mitigate quickly:

4. Root Cause Analysis for CVE-2022-22965

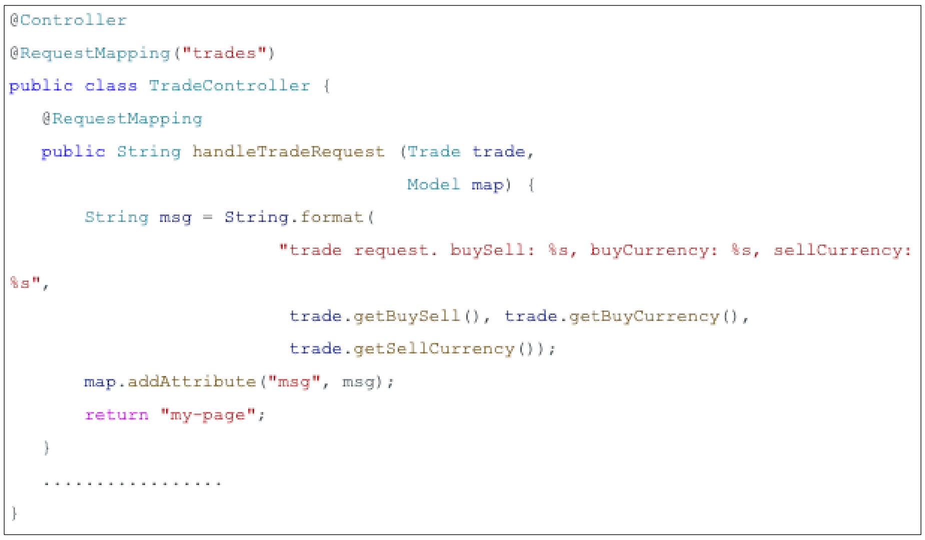
The vulnerability is caused by the "**getCachedIntrospectionResults**" method of the Spring framework, wrongly exposing the class object when binding the parameters.
The default Spring data binding mechanism allows developers to bind HTTP request details to application-specific objects. For example, there is a simple classical application scenario in which the developer creates a trade object to capture request parameters as shown in Figure 1.



```
public class Trade {

   private String buySell;
   private String buyCurrency;
   private String sellCurrency;

   public String getBuySell () {
       return buySell;
   }

   public void setBuySell (String buySell) {
       this.buySell = buySell;
   }


  .................
}
```

Figure 1. Example of a trade object.

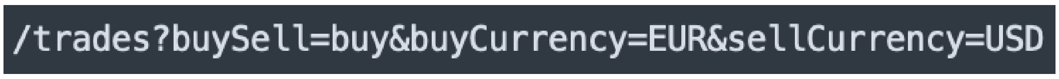
Then the developer creates a controller to use the object trade as shown in Figure 2.



```
@Controller
@RequestMapping("trades")
public class TradeController {
   @RequestMapping
   public String handleTradeRequest (Trade trade,
                                     Model map) {
       String msg = String.format(
                         "trade request. buySell: %s, buyCurrency: %s, sellCurrency:
%s",
                          trade.getBuySell(), trade.getBuyCurrency(),
                          trade.getSellCurrency());
       map.addAttribute("msg", msg);
       return "my-page";
   }
   .................
}
```

Figure 2. Example of a controller using the trade object.

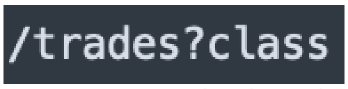
After that, the developer usually creates a request builder for the trade controller, which allows the web user to access the trade object remotely as shown in Figure 3.



```
/trades?buySell=buy&buyCurrency=EUR&sellCurrency=USD
```

Figure 3.Accessing a normal object.

When web users access trade object properties, the binding process (bindRequestParameters) in the Spring framework implementation will call the getCachedIntrospectionResults method to get and set the object property in the cache. However, the return object of the getCachedIntrospectionResults method includes a class object. This means that web users can get a class object remotely by simply submitting a URL as shown in Figure 4.



```
/trades?class
```

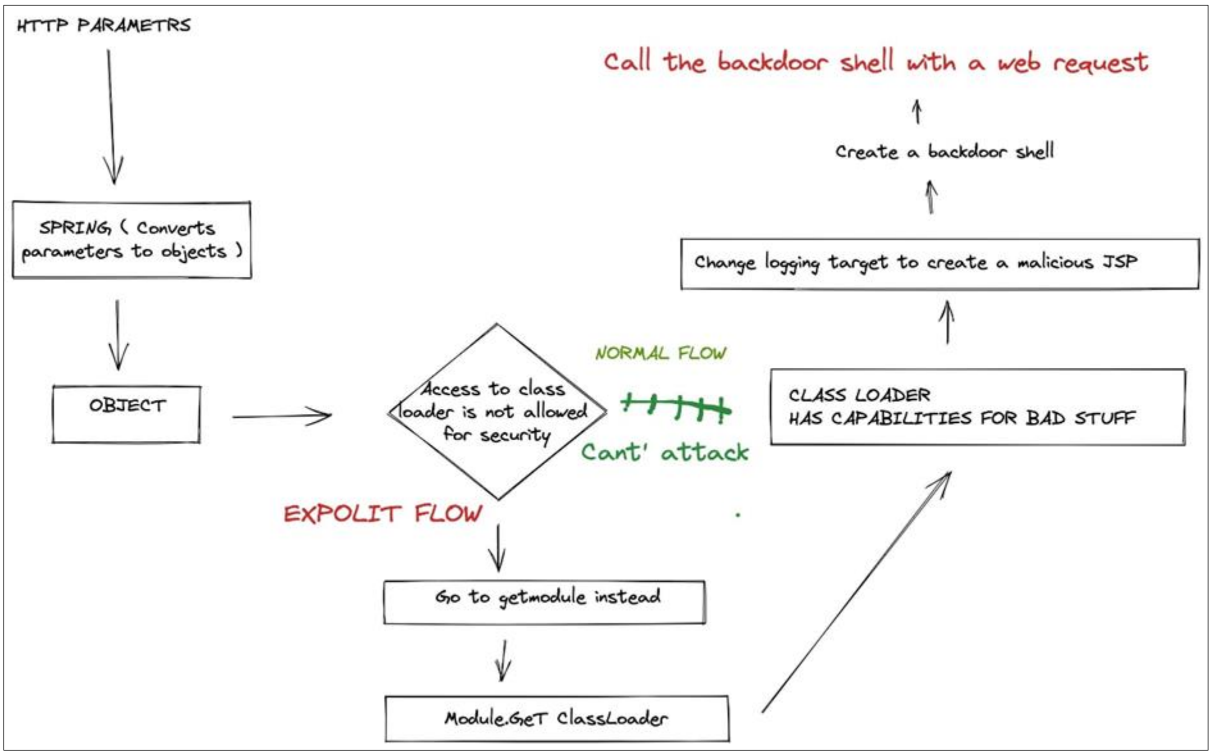
Figure 4. Accessing the class object.

Exposing the class object to web users is very dangerous and can lead to RCE in many ways. The class loader is often used by exploiting payloads to load some sensitive classes dynamically for object modification and code execution.

5. Spring4Shell Exploit Code

Exploit code for this remote code execution vulnerability has been made publicly available.

https://github.com/craig/SpringCore0day/blob/main/exp.py


HTTP PARAMETRS
SPRING (Converts parameters to objects)
OBJECT
Access to class loader is not allowed for security
NORMAL FLOW
Cant' attack
CLASS LOADER HAS CAPABILITIES FOR BAD STUFF
Change logging target to create a malicious JSP
Create a backdoor shell
Call the backdoor shell with a web request
EXPOLIT FLOW
Go to getmodule instead
Module.GeT ClassLoader

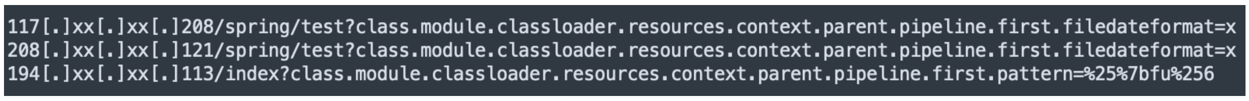
6. Spring4Shell Vulnerability Scan

Scanning for Spring4Shell vulnerability could be done with commercial products such as Tenable products as well as open source solutions such as Nmap. Some useful scanning solutions has been shown in the following table.

Scanner Tool	How to prepare and scan Spring4Shell
Tenable Nessus Scanner	Update plugins (which contains 159374 plugin ID)
Nmap	Use this link(https://github.com/j0hnf/nmap-scripts/blob/main/http-spring4shells.nse)
curl	curl host: port/path?class.module.classLoader.URLs%5B0%5D=0 sending the above request if return 400 HTTP status code, indicates vulnerability
Simple local Spring vulnerability scanner	Use this script(https://github.com/hillu/local-spring-vuln-scanner)

7. Indicators of Compromise

Unit 42 reported some Http requests scanning for this vulnerability which comes below.



```
117[.]xx[.]xx[.]208/spring/test?class.module.classloader.resources.context.parent.pipeline.first.filedateformat=x
208[.]xx[.]xx[.]121/spring/test?class.module.classloader.resources.context.parent.pipeline.first.filedateformat=x
194[.]xx[.]xx[.]113/index?class.module.classloader.resources.context.parent.pipeline.first.pattern=%25%7bfu%256
```

Figure 5, Scanning traffic from PAN-DB cloud logs.

The following filenames would store the webshell contents on the server in the event of successful exploitation:

```
0xd0m7.jsp
myshell.jsp
shell.jsp
tomcatwar.jsp
wpz.jsp
```

There are at least two variants of the webshell. One uses the "pwd" parameter for authentication (password is always j) and the "cmd" parameter for the command to execute. The second variant does not use a parameter for authentication and uses id for the command to execute. Table 1, shows the parameters that the webshell saved to the server would use for authentication and command.

URL Parameters	Authentication	Command
&id=<command>		id
&pwd=j&cmd=<command>	pwd	cmd

Table 1 Parameters used by webshells seen in hits on "Spring Core Remote Code Execution Vulnerability" signature.

The unique commands listed below were submitted to webshells

```
ls
nslookup%20[redacted].test6.ggdd[.]co[.]uk
nslookup+[redacted].test6.ggdd[.]co[.]uk
ping%20[redacted].test6.ggdd[.]co[.]uk
ping+[redacted].test6.ggdd[.]co[.]uk
whoami
cat%20/etc/passwd
cat+/etc/passwd
id
ifconfig
ipconfig
ping%20[redacted].burpcollaborator[.]net
```

❖ Kaspersky Report

Table 2 shows some IOCs extracted from a Kaspersky report.

MD5 Hash	7e46801dd171bb5bf1771df1239d760c	3de4e174c2c8612aebb3adef10027679
File-Name	shell.jsp	exploit.py

Table 2

8. Spring4Shell Detection Rule

You can use the following detection rules to search for possible malicious activities.

❖ Spring4Shell detection rules in Splunk (SIEM)

- **Webshell -- PoC Exploit:**

```
  index=web_apps (sourcetype=nginx OR
sourcetype=webapp_firewall) status=200
  |rex field=url "(?<webShell>\/.*\.(jsp|class)\?.*=.*)"
  |rex field=uri "(?<webShell>\/.*\.(jsp|class)\?.*=.*)"
  |where isnotnull(webShell)
  |eval Domain = mvappend(dest, domain), fullURL = mvappend(url,
uri)
  |table _time src_ip Domain status method webShell fullURL
sourcetype
```

- **PoC Exploit attempting to change Tomcat logging:**

```
  index=web_apps (sourcetype=nginx OR
sourcetype=webapp_firewall) (method=POST OR method=GET)
url="*?class.module.classloader.resources.context.parent.pipelin
e.first.pattern"×=
  |table _time src_ip dest_domain url headers user_agent status
```

- **Linux audit logs (untested):**

If you find suspicious files being created, correlate with type=CWD & check the "cwd" path. Check if the path is your webapps' ROOT/ directory. Correlate with other above detection rules (nginx/web app firewall) to identify source IP.

```
  index=linux_index sourcetype="linux:auditd" type=PATH
name="*.jsp"
  |table _time host type name
```

- **PoC Exploit**

```
  index=web_apps (sourcetype=nginx OR
sourcetype=webapp_firewall) status=200 (method=POST OR
method=GET) (url=*.jsp* OR url=*.class*)
  | table _time src_ip dest_domain url headers
```

❖ Spring4Shell detection rules in Suricata (NIDS)

```
  alert http any any -> [$HOME_NET,$HTTP_SERVERS] any
(msg:"ET EXPLOIT Possible SpringCore RCE/Spring4Shell
Stage 1 Pattern Set Inbound (Unassigned)";
flow:to_server,established; http.method; content:"GET";
http.uri; content:"pipeline.first.pattern="; fast_pattern;
classtype:attempted-admin; sid:2035674; rev:1;
metadata:attack_target Server, created_at 2022_03_31,
deployment Perimeter, deployment Internal, former_category
EXPLOIT, signature_severity Major, tag Exploit, updated_at
2022_03_31;)
```

```
  alert http any any -> [$HOME_NET,$HTTP_SERVERS] any
(msg:"ET EXPLOIT Possible SpringCore RCE/Spring4Shell
Stage 2 Suffix Set Inbound (Unassigned)";
flow:to_server,established; http.method; content:"GET";
http.uri; content:"pipeline.first.suffix="; fast_pattern;
classtype:attempted-admin; sid:2035675; rev:1;
metadata:attack_target Server, created_at 2022_03_31,
deployment Perimeter, deployment Internal, former_category
EXPLOIT, signature_severity Major, tag Exploit, updated_at
2022_03_31;)
```

```
  alert http any any -> [$HOME_NET,$HTTP_SERVERS] any
(msg:"ET EXPLOIT Possible SpringCore RCE/Spring4Shell
```

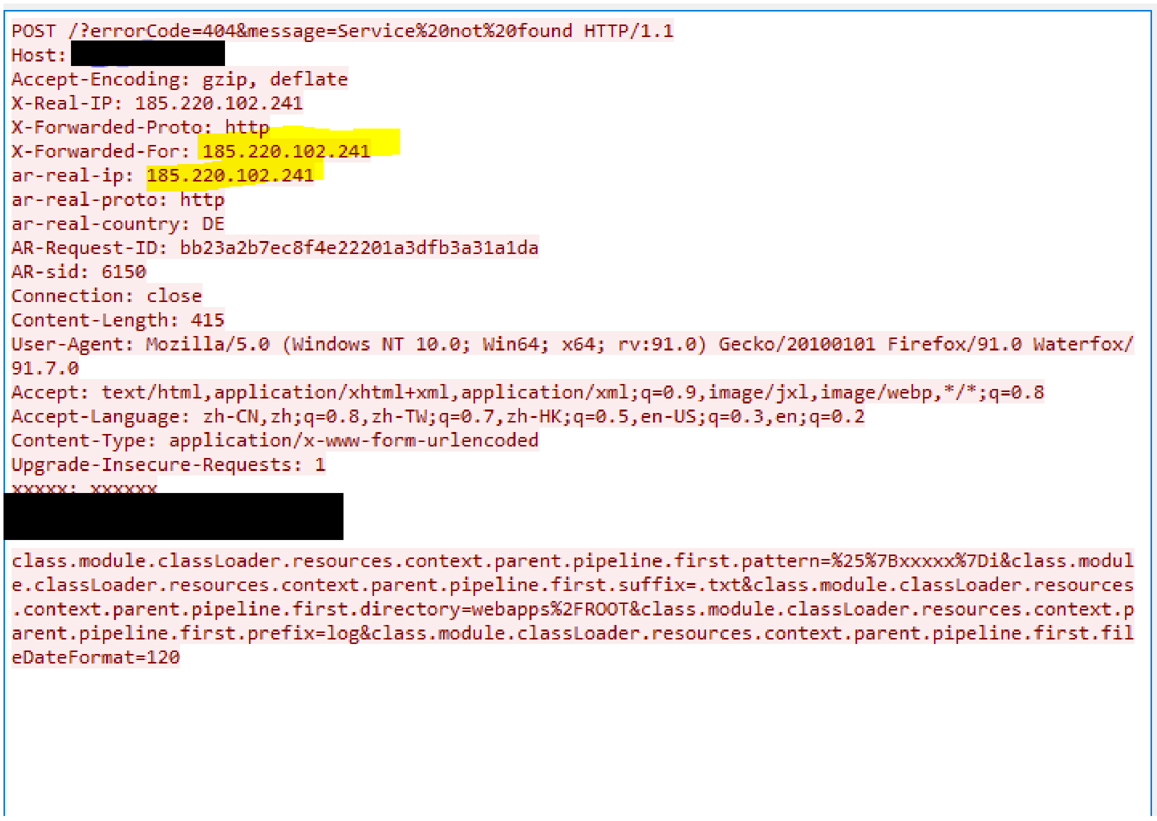
	Stage 3 Directory Set Inbound (Unassigned)"; flow:to_server,established; http.method; content:"GET"; http.uri; content:"pipeline.first.directory="; fast_pattern; classtype:attempted-admin; sid:2035676; rev:1; metadata:attack_target Server, created_at 2022_03_31, deployment Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at 2022_03_31;)
	alert http any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Possible SpringCore RCE/Spring4Shell Stage 4 Prefix Set Inbound (Unassigned)"; flow:to_server,established; http.method; content:"GET"; http.uri; content:"pipeline.first.prefix="; fast_pattern; classtype:attempted-admin; sid:2035677; rev:1; metadata:attack_target Server, created_at 2022_03_31, deployment Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at 2022_03_31;)
	alert http any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Possible SpringCore RCE/Spring4Shell Inbound (Unassigned)"; flow:to_server,established; http.method; content:"POST"; http.request_body; content:"pipeline.first.pattern="; fast_pattern; content:"pipeline.first.suffix="; content:"pipeline.first.directory="; content:"pipeline.first.prefix="; classtype:attempted-admin; sid:2035678; rev:1; metadata:attack_target Server, created_at 2022_03_31, deployment Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at 2022_03_31;

❖ Spring4Shell detection rules in YARA (File)

https://github.com/Neo23x0/signature-base/blob/master/yara/expl_spring4shell.yar

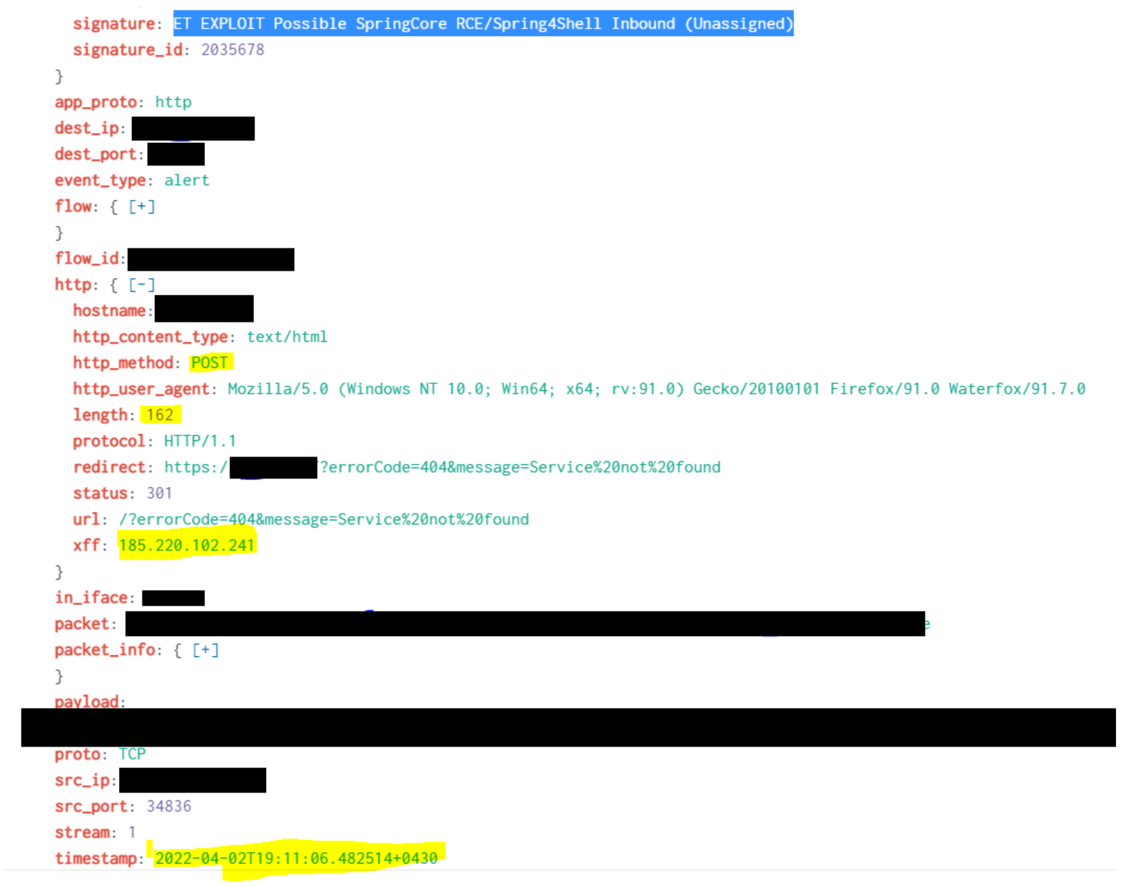
9. Spara Analysis and Monitoring Team Results

The Spara SOC team was prepared to deal with potential threats regarding "Sping4shell: CVE-2022-22965" vulnerability soon after the disclosing of the zero-day. Therefore, the team started searching the relevant systems and found some exploitation attempts that originated from overseas.
Threats appeared just one day after the zero-day publication. The below image shows one of the attacker's attempts to exploit web-based systems.



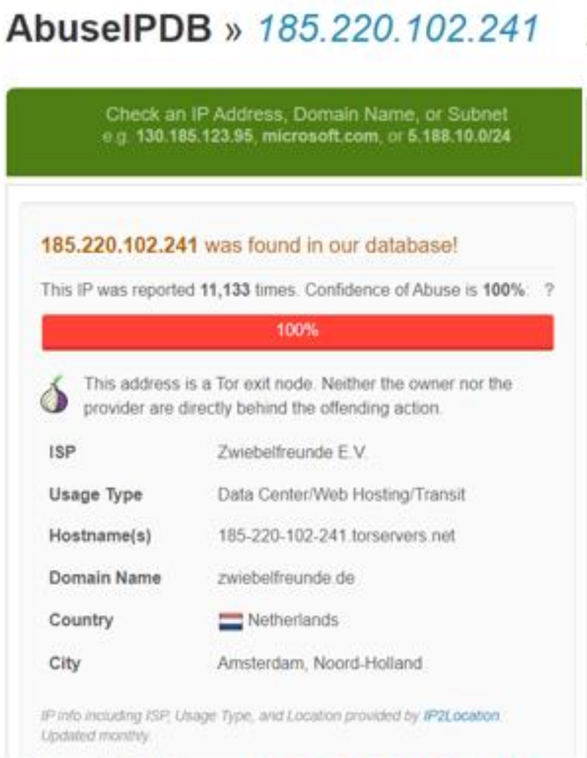
```
POST /?errorCode=404&message=Service%20not%20found HTTP/1.1
Host: [redacted]
Accept-Encoding: gzip, deflate
X-Real-IP: 185.220.102.241
X-Forwarded-Proto: http
X-Forwarded-For: 185.220.102.241
ar-real-ip: 185.220.102.241
ar-real-proto: http
ar-real-country: DE
AR-Request-ID: bb23a2b7ec8f4e22201a3dfb3a31a1da
AR-sid: 6150
Connection: close
Content-Length: 415
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0 Waterfox/
91.7.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/jxl,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
xxxxx: xxxxxx
[redacted]

class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bxxxxx%7Di&class.modul
e.classLoader.resources.context.parent.pipeline.first.suffix=.txt&class.module.classLoader.resources
.context.parent.pipeline.first.directory=webapps%2FROOT&class.module.classLoader.resources.context.p
arent.pipeline.first.prefix=log&class.module.classLoader.resources.context.parent.pipeline.first.fil
eDateFormat=120
```

The following image shows another malicious IP:



```
    signature: ET EXPLOIT Possible SpringCore RCE/Spring4Shell Inbound (Unassigned)
    signature_id: 2035678
  }
  app_proto: http
  dest_ip: [redacted]
  dest_port: [redacted]
  event_type: alert
  flow: { [+]
  }
  flow_id: [redacted]
  http: { [-]
    hostname: [redacted]
    http_content_type: text/html
    http_method: POST
    http_user_agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0 Waterfox/91.7.0
    length: 162
    protocol: HTTP/1.1
    redirect: https:/[redacted]?errorCode=404&message=Service%20not%20found
    status: 301
    url: /?errorCode=404&message=Service%20not%20found
    xff: 185.220.102.241
  }
  in_iface: [redacted]
  packet: [redacted]
  packet_info: { [+]
  }
  payload:
[redacted]
  proto: TCP
  src_ip: [redacted]
  src_port: 34836
  stream: 1
  timestamp: 2022-04-02T19:11:06.482514+0430
```

Some related malicious activities have been observed in one of our customer's environment between 1st to 5th April 2022. All of this malicious activity was conducted by source IP addresses which were in AbuseIPDB's bad reputation lists.



AbuseIPDB » 185.220.102.241

Check an IP Address, Domain Name, or Subnet
e.g. **130.185.123.95**, **microsoft.com**, or **5.188.10.0/24**

185.220.102.241 was found in our database!

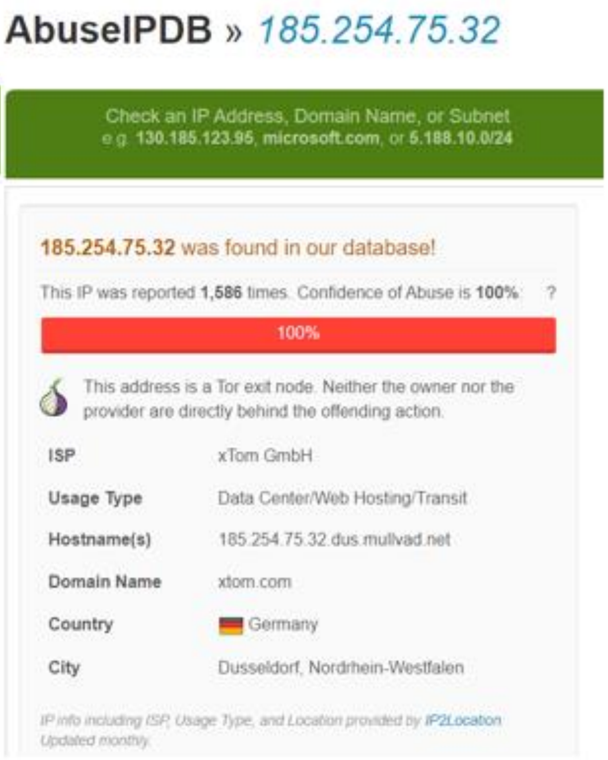
This IP was reported **11,133** times. Confidence of Abuse is **100%**: ?

100%

This address is a Tor exit node. Neither the owner nor the provider are directly behind the offending action.

ISP	Zwiebelfreunde E.V.
Usage Type	Data Center/Web Hosting/Transit
Hostname(s)	185-220-102-241.torservers.net
Domain Name	zwiebelfreunde.de
Country	Netherlands
City	Amsterdam, Noord-Holland

IP info including ISP, Usage Type, and Location provided by IP2Location. Updated monthly



AbuseIPDB » 185.254.75.32

Check an IP Address, Domain Name, or Subnet
e.g. **130.185.123.95**, **microsoft.com**, or **5.188.10.0/24**

185.254.75.32 was found in our database!

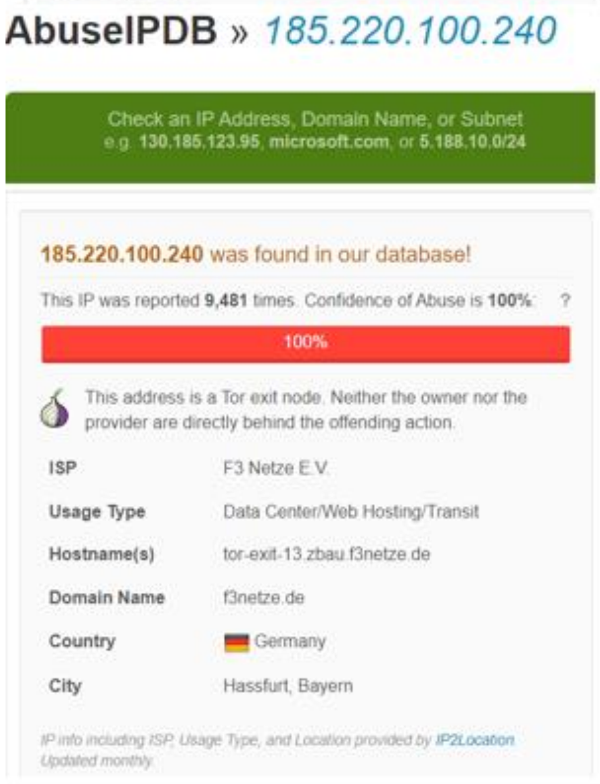
This IP was reported **1,586** times. Confidence of Abuse is **100%**: ?

100%

This address is a Tor exit node. Neither the owner nor the provider are directly behind the offending action.

ISP	xTom GmbH
Usage Type	Data Center/Web Hosting/Transit
Hostname(s)	185.254.75.32.dus.mullvad.net
Domain Name	xtom.com
Country	Germany
City	Dusseldorf, Nordrhein-Westfalen

IP info including ISP, Usage Type, and Location provided by IP2Location. Updated monthly



AbuseIPDB » 185.220.100.240

Check an IP Address, Domain Name, or Subnet
e.g. **130.185.123.95**, **microsoft.com**, or **5.188.10.0/24**

185.220.100.240 was found in our database!

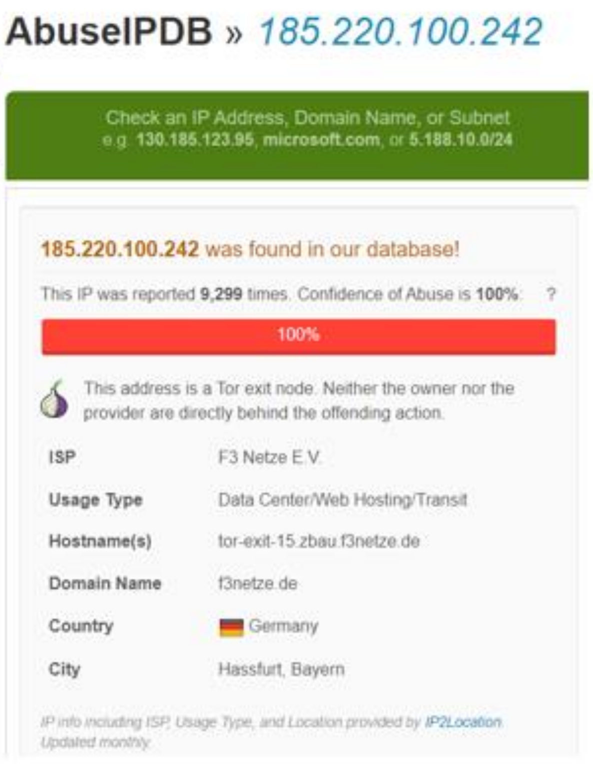
This IP was reported **9,481** times. Confidence of Abuse is **100%**: ?

100%

This address is a Tor exit node. Neither the owner nor the provider are directly behind the offending action.

ISP	F3 Netze E.V.
Usage Type	Data Center/Web Hosting/Transit
Hostname(s)	tor-exit-13.zbau.f3netze.de
Domain Name	f3netze.de
Country	Germany
City	Hassfurt, Bayern

IP info including ISP, Usage Type, and Location provided by IP2Location. Updated monthly



AbuseIPDB » 185.220.100.242

Check an IP Address, Domain Name, or Subnet
e.g. **130.185.123.95**, **microsoft.com**, or **5.188.10.0/24**

185.220.100.242 was found in our database!

This IP was reported **9,299** times. Confidence of Abuse is **100%**: ?

100%

This address is a Tor exit node. Neither the owner nor the provider are directly behind the offending action.

ISP	F3 Netze E.V.
Usage Type	Data Center/Web Hosting/Transit
Hostname(s)	tor-exit-15.zbau.f3netze.de
Domain Name	f3netze.de
Country	Germany
City	Hassfurt, Bayern

IP info including ISP, Usage Type, and Location provided by IP2Location. Updated monthly

Identified malicious IPs are as follows.

185.220.100.242	185.220.100.240	185.254.75.32	185.220.102.241

10. Conclusion

Spring4Shell is officially assigned CVE-2022-22965, and the patch was released on March 31، 2022. Since exploitation is straightforward and all the relevant technical details have already gone viral on the internet, it's only a matter of time until Spring4Shell becomes fully weaponized and abused on a larger scale. Developers and users who have projects or products based on JDK9+ and the Spring Framework (or its derivatives) are strongly urged to patch as soon as possible.

11. Resources

https://bestofmalagasyblogs.com/vendors-assessing-the-impact-of-the-spring4shell-vulnerability/
https://securelist.com/spring4shell-cve-2022-22965/106239/
https://github.com/west-wind/Spring4Shell-Detection
https://unit42.paloaltonetworks.com/cve-2022-22965-springshell/
https://www.cyberkendra.com/2022/03/springshell-rce-0-day-vulnerability.html

spara