AIG Shields Up: Cybersecurity

Forage Job Simulation

Task One

The first steps will be conducting an analysis of the recently disclosed — at the time –Apache Log4j zero-day vulnerability by researching official advisories from CISA. Following this, I will move on to assess AIG's internal infrastructure to identify which systems are affected and determine the appropriate teams responsible. Finally, I will prepare a formal advisory communication to inform the relevant team leads about the vulnerability and recommend urgent remediation steps.

Currently conducting the analysis over the given sources: https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-356a & https://www.cisa.gov/news-events/news/cisa-fbi-nsa-and-international-partners-issue-advisory-ransomware-trends-2021

Concluding notes I have been logging while reading: According to listed CISA sources

CSA: Log4j – Vulnerability = log4shell
Log4shell is a remote code execution (RCE) vulnerability affecting Apache's Log4j library, versions 2.0-beta9 to 2.14.1.

Teams with Log4j may be affected by the vulnerability in question.

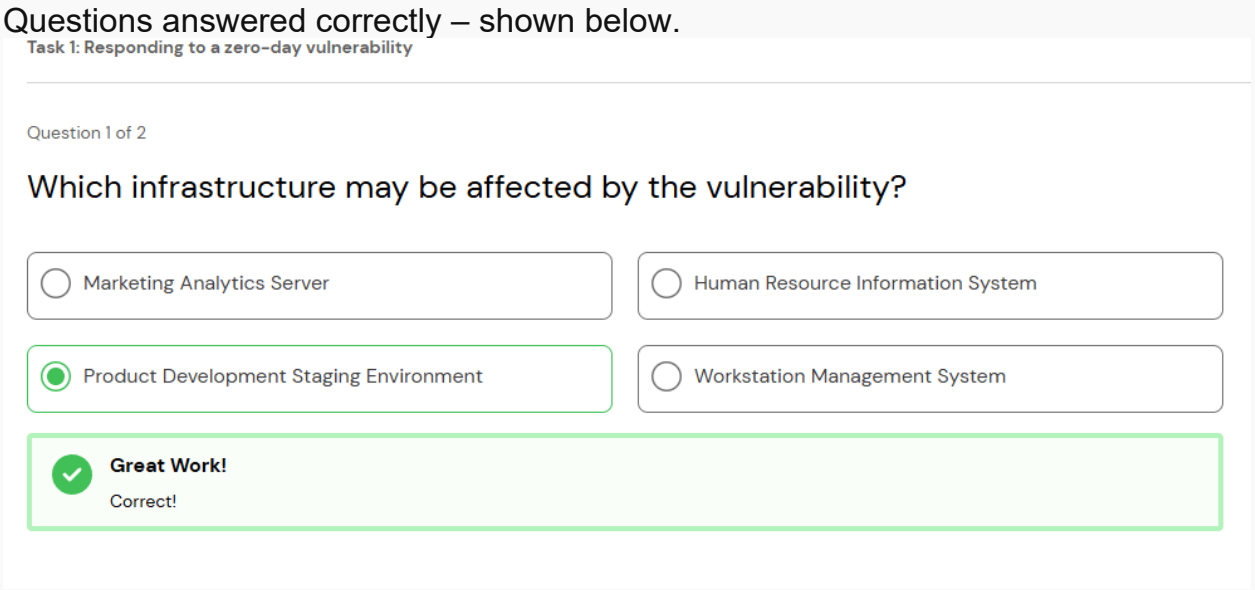
Log4Shell and CVE-2021-45046 listed as critical
and CVE-2021-45105 listed as high

any Java-based service or product that uses vulnerable Log4j versions for logging is at risk.

Based on my analysis and reporting so far, it is clear to see based on the high level infrastructure of teams below, the **Product Development** may be affected by the vulnerability.

Product Team	Product Name	Team Lead	Services Installed
IT	Workstation Management System	Jane Doe (tech@email.com)	OpenSSH dnsmasq lighttpd
Product Development	Product Development Staging Environment	John Doe (product@email.com)	Dovecot pop3d Apache httpd Log4j Dovecot imapd MiniServ
Marketing	Marketing Analytics Server	Joe Schmoe (marketing@email.com)	Microsoft ftpd Indy httpd Microsoft Windows RPC Microsoft Windows netbios-ssn Microsoft Windows Server 2008 R2 - 2012 microsoft ds
HR	Human Resource Information System	Joe Bloggs (hr@email.com)	OpenSSH Apache httpd rpcbind2-4

Questions answered correctly – shown below.



Task 1: Responding to a zero-day vulnerability

Question 2 of 2

Which team has ownership of the affected infrastructure?

- ○ IT team
- ◉ Product Development team
- ○ Marketing team
- ○ HR team

Great Work!

Correct!

Based on the goal of this task, it is time to draft an advisory email to alert the infrastructure owner of the seriousness of this vulnerability. I am going to start by looking back over the reported notes I made, and referring back the infrastructure table and CISA sources as needed.

I just wrote a sample response that I would send to the correct product development team. **Here is the submission:**

Hello Product Development,

AIG Cyber & Information Security Team would like to inform you about critical vulnerabilities recently discovered in the Apache Log4j Java logging library that may affect Log4j.

The vulnerabilities—specifically CVE-2021-44228 (Log4Shell), CVE-2021-45046, and CVE-2021-45105—allow remote attackers to execute arbitrary code or cause denial-of-service conditions on vulnerable systems that use affected versions of Log4j (2.0-beta9 through 2.14.1). Exploitation is possible by submitting specially crafted requests that trigger the vulnerable Log4j logging process, potentially allowing full system compromise.

Given the widespread use of Java and Log4j across IT and operational technology environments, these vulnerabilities pose a high risk, including unauthorized access, data theft, ransomware deployment, and service disruptions.

To mitigate this risk, it is critical to:

Identify all assets and applications using Log4j within your environment.

Immediately upgrade Log4j to version 2.17.0 or later, where these vulnerabilities are patched.

If patching is not immediately possible, apply recommended workarounds or configuration changes from Apache's official guidance to disable JNDI lookups.

Monitor logs and network traffic for signs of exploitation attempts.

Coordinate with Cyber & Information Security Team to report any suspected compromise.

Please confirm once remediation steps have been completed or if assistance is required to identify affected assets or apply fixes.

For any questions or issues, don't hesitate to reach out to us.

Kind regards,
AIG Cyber & Information Security Team

Task Two

After sending off the advisory email, things have taken a turn. An attacker managed to exploit the Log4j vulnerability and tried to deploy ransomware on the affected server. Thankfully, the Incident Detection & Response team jumped in fast and stopped the attack before it could do major damage as only one zip file got encrypted, which the team does not have a backup for.

Now, the leadership at AIG has made it clear they won't be paying the ransom. Instead, they want to see if we can crack the decryption key ourselves. I am now diving into this next challenge which sounds like a perfect mix of technical problem-solving and real-world urgency.

To start the brute force process, I am going to write a script using Python. The task provided the famous **rockyou.txt** file (as the attacker was quite unsophisticated) which I will be utilizing in the script to brute force the unknown password.

This will be my first script written on my own and throughout the coding I will be utilizing AI tools and Google searches to complete this only as needed. The goal for this is to understand the process and be able to re-emulate in the future (or at least be able to quickly re-learn/adapt to a future problem).

Starting inn VSCode and with the template provided, I am currently working with this:

```
from zipfile import ZipFile

# Use a method to attempt to extract the zip file with a given password

# def attempt_extract(zf_handle, password):

#

#

#

def main():

    print("[+] Beginning bruteforce ")

    with ZipFile('enc.zip') as zf:

        with open('rockyou.txt', 'rb') as f:

            # Write your logic here…

            # Iterate through password entries in rockyou.txt

            # Attempt to extract the zip file using each password

            # Handle correct password extract versus incorrect password attempt)

    #print("[+] Password not found in list")

if __name__ == "__main__":

    main()
```

I am more comfortable with iteration and decoding (second section) than the first section, so I will start here. I need to start by iterating over the opened rockyou.txt file using *for password in f:*

Next, confirming my reasoning with AI tools, I need to clean up the password using *.strip*. This seems to be correct and I end up with *pwd = password.strip()* /n *print(pwd)*. Simply put, .strip is removing any leading or trailing **whitespace characters** from the password bytes.

Here is where I got choked up first and confused about the logic. After trial and error and consulting ChatGPT, it was clear that I needed to rewind and focus back on the first section (function) as I would have to call it in the second. After requesting a first step hint, I saw the given function from the template *def attempt_extract(zf_handle, password):* and this is a good start.

Because of the notes given in the template "# Use a method to attempt to extract the zip file with a given password", I knew I was going to utilize *try* and *except*. However, I am not aware of the method to use but a simple Google search "method in python to extract all files from a zip" I got .extractall(), perfect! That leaves me with these lines of code. *try:* /n *zf_handle.extractall(pwd=password)* /n *return True* /n *except:* /n *return False*. This try and except code if extraction succeeds, it returns True. If extraction fails (wrong password or other error), the except block catches the error and returns False.

This concludes the first function and I am nearly done with the script! It is fairly simple from here. I just need to call back to the previous function in an attempt to extract using the current password! The full code will be pasted below! **NOTE** Please ignore any formatting errors as this code runs smoothly through any IDE.

```
from zipfile import ZipFile

def attempt_extract(zf_handle, password):

    try:

        zf_handle.extractall(pwd=password)

        return True

    except:

       # Extraction failed
```

```
        return False
def main():
  print("[+] Beginning bruteforce")
  # Open the encrypted zip file
  with ZipFile('enc.zip') as zf:
     # Open the password list in binary mode
     with open('rockyou.txt', 'rb') as f:
        # Iterate through each password candidate
        for password in f:
           # Clean password
           pwd = password.strip()
           print(pwd)
           # Attempt to extract using the current password
           if attempt_extract(zf, pwd):
              print(f"[+] Password found: {pwd}")
              break
        else:
           print("[-] Password not found in list")
if __name__ == "__main__":
   main()
```