Project Summary: Log4j Advisory and Brute-Force Decryption using Python

This project involved two main tasks: drafting a security advisory email about a critical vulnerability and modifying a Python brute-force script to decrypt a password-protected zip file.

Task 1: Log4j Security Advisory Email

The first task was to draft a security advisory email for the Product Development Team regarding the newly discovered Log4j vulnerability. This vulnerability (CVE-2021-44228 and CVE-2021-45046) allows unauthenticated attackers to perform remote code execution (RCE) on affected systems running Log4j versions 2.0-beta9 to 2.15.0, which could lead to unauthorized access and potential data exfiltration.

Key Elements of the Advisory:

- **Vulnerability Overview**: Explained the critical nature of the Log4j vulnerability and provided references to NIST advisories for further information.
- **Affected Products**: Listed specific vulnerable versions of Log4j (2.0-beta9 to 2.15.0).
- **Risk and Impact**: Highlighted the Remote Code Execution threat, stressing the urgency of mitigating the risk to avoid unauthorized access or data breaches in the staging environment.
- **Remediation Steps**: Provided actionable steps to update Log4j to secure versions 2.16.0 (Java 8) or 2.12.2 (Java 7), alongside monitoring for any exploitation attempts.

This advisory informed the development team about the vulnerability and provided clear steps to secure the infrastructure, ensuring proactive cybersecurity measures.

Task 2: Python Brute-Force Decryption Script

The second task was to work with a partially complete Python script designed to brute-force decrypt a zip file (`enc.zip`) using a wordlist (`rockyou.txt`). This task involved understanding and completing the Python script to automate the password discovery process, unzipping the file only if the correct password was found.

```
from zipfile import ZipFile

def attempt_extract(zf_handle, password):
    try:
        zf_handle.extractall(pwd=password)
        return True
    except:
        return False

def main():
    print("[+] Beginning bruteforce ")
    with ZipFile('enc.zip') as zf:
        with open('rockyou.txt', 'rb') as f:
            for p in f:
                password = p.strip()
                if attempt_extract(zf, password):
                    print("[+] Correct password: %s" % password)
                    exit(0)
                else:
                    print("[-] Incorrect password: %s" % password)

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

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

Explanation of the Python Script:

1. **Setting up the Brute-Force Function**: The script defines an `attempt_extract` function, which uses a try-except block to attempt file extraction with each password entry from `rockyou.txt`. If the password is correct, it returns `True`; otherwise, it returns `False`.
2. **Brute-Force Loop**: In the `main()` function, the script opened `enc.zip` and `rockyou.txt`, then iterated through each password in the wordlist. For each password, it is called `attempt_extract`. If successful, the correct password was printed, and the script terminated. If no password was found, the script ended with a message indicating failure.
3. **Completion of the Script**: Added print statements to give feedback on each password attempt and a termination condition for successful decryption. This allowed for efficient password checking and user-friendly output to monitor the decryption progress.

Results:

By modifying the Python script to handle password checking, the zip file was successfully decrypted, achieving a 100% success rate in accessing encrypted files through brute-force. The code demonstrated effectiveness in real-world applications of decryption and password testing, verifying cybersecurity and forensics skills.

Conclusion:

Through this project, I effectively communicated a critical vulnerability advisory to stakeholders and implemented a brute-force Python script to decrypt an encrypted file. These tasks demonstrated strong capabilities in cybersecurity incident communication, vulnerability response, and practical programming to solve security challenges. The Python script's successful decryption verified the project's success, confirming proficiency in both technical and communication aspects of cybersecurity.