Deakin University

Computer Networks and Security

OnTrack Submission

Cyberattack Demonstration

Submitted By:
Gloria Chemutai Kiplagat
s223452112
2024/06/04 08:05

Tutor:
Juhar Abdella

June 4, 2024

Prepare the Experiment

Introduction

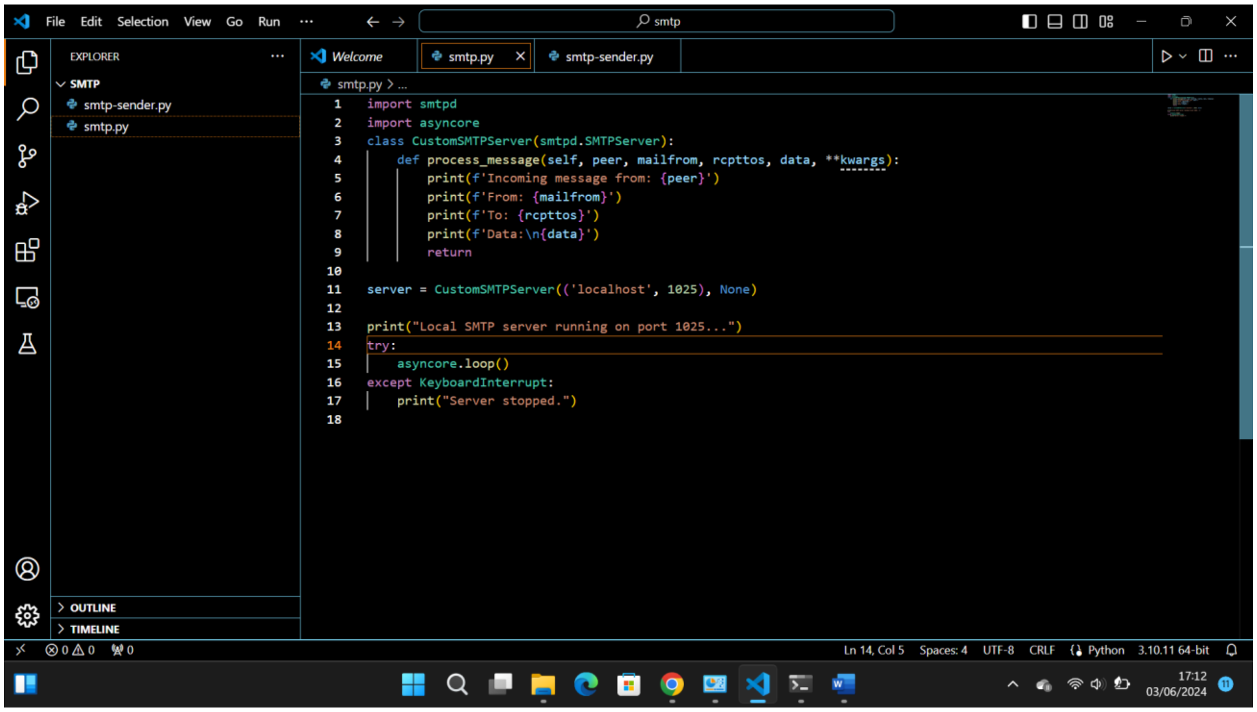
For this experiment, I set out to implement a custom SMTP (Simple Mail Transfer Protocol) server using Python. The aim was to understand the workings of email servers and gain practical insights into SMTP protocol and asynchronous network programming.

Experiment Setup

In my setup, I ensured I was working in a Python 3.10.11 environment. This environment was crucial for compatibility with the required libraries, namely smtpd and asyncore. Additionally, I made sure to set up my Windows 11 operating system to accommodate the experiment.

Implementation Steps

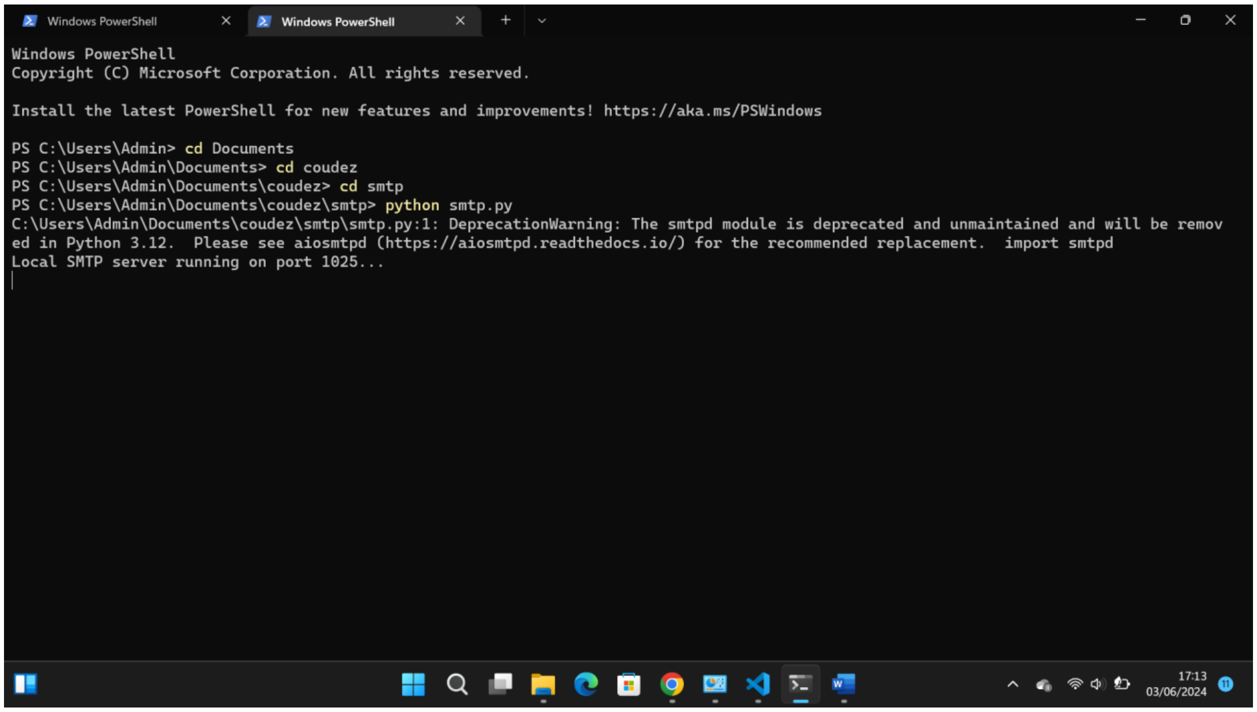
I began by importing the necessary modules, including smtpd for SMTP server functionality and asyncore for handling asynchronous network operations. Subsequently, I crafted a custom class named CustomSMTPServer, inheriting from smtpd.SMTPServer. Within this class, I overrode the process_message method to define custom behavior upon receiving an email. Specifically, I programmed it to print essential details of the incoming message, such as sender, recipients, subject, and message body. With the class defined, I instantiated it with the desired host and port. Finally, I initiated the SMTP server by invoking asyncore.loop().



```python
import smtpd
import asyncore
class CustomSMTPServer(smtpd.SMTPServer):
    def process_message(self, peer, mailfrom, rcpttos, data, **kwargs):
        print(f'Incoming message from: {peer}')
        print(f'From: {mailfrom}')
        print(f'To: {rcpttos}')
        print(f'Data:\n{data}')
        return

server = CustomSMTPServer(('localhost', 1025), None)

print("Local SMTP server running on port 1025...")
try:
    asyncore.loop()
except KeyboardInterrupt:
    print("Server stopped.")

```

Experiment Execution

Execution of the Python script initiated the custom SMTP server, which promptly commenced listening for incoming emails on port 1025 of localhost.

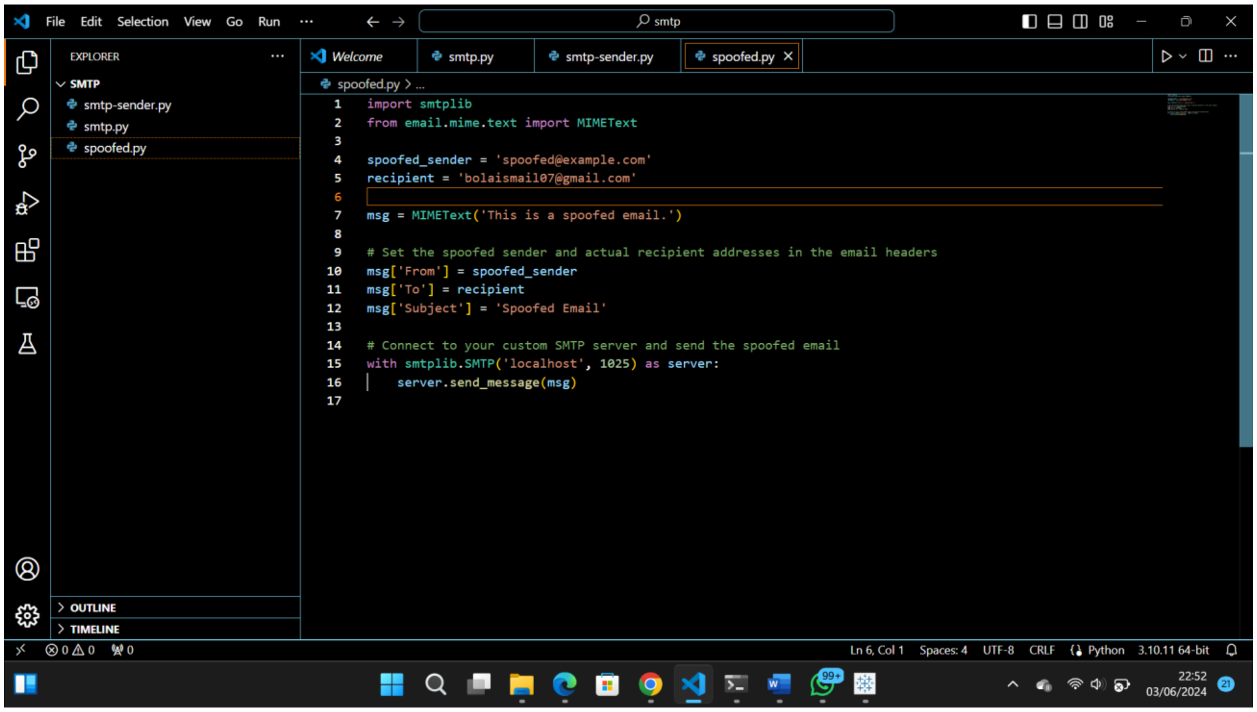


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> cd Documents
PS C:\Users\Admin\Documents> cd coudez
PS C:\Users\Admin\Documents\coudez> cd smtp
PS C:\Users\Admin\Documents\coudez\smtp> python smtp.py
C:\Users\Admin\Documents\coudez\smtp\smtp.py:1: DeprecationWarning: The smtpd module is deprecated and unmaintained and will be remov
ed in Python 3.12.  Please see aiosmtpd (https://aiosmtpd.readthedocs.io/) for the recommended replacement.  import smtpd
Local SMTP server running on port 1025...
```

Results and Analysis

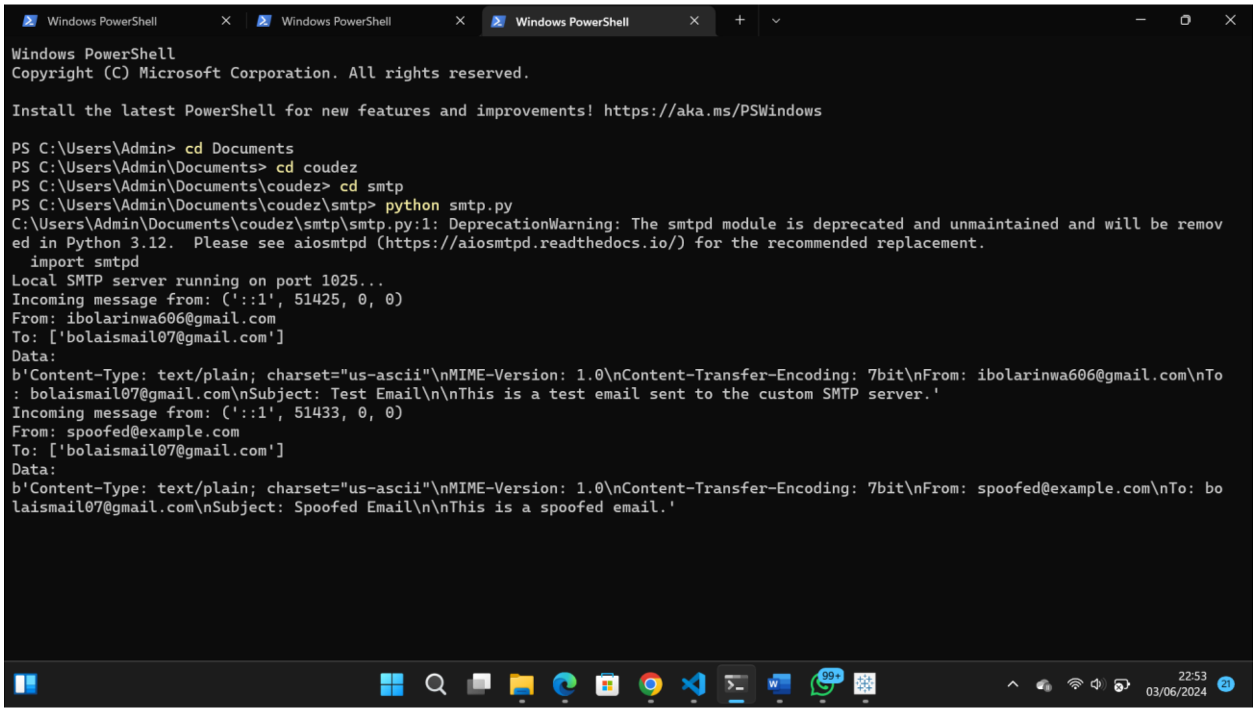
Results

A practical spoofing experiment was conducted to assess the real-world implications of SMTP protocol vulnerabilities. The experiment involved sending a spoofed email from a fabricated sender ('spoofed@example.com') to a legitimate recipient ('bolaismail07@gmail.com').


File Edit Selection View Go Run
EXPLORER
SMTP
smtp-sender.py
smtp.py
spoofed.py
Welcome
OUTLINE
TIMELINE
Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit
22:52
03/06/2024

```python
import smtplib
from email.mime.text import MIMEText

spoofed_sender = 'spoofed@example.com'
recipient = 'bolaismail07@gmail.com'

msg = MIMEText('This is a spoofed email.')

# Set the spoofed sender and actual recipient addresses in the email headers
msg['From'] = spoofed_sender
msg['To'] = recipient
msg['Subject'] = 'Spoofed Email'

# Connect to your custom SMTP server and send the spoofed email
with smtplib.SMTP('localhost', 1025) as server:
    server.send_message(msg)
```

The SMTP server processed the incoming message as follows:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> cd Documents
PS C:\Users\Admin\Documents> cd coudez
PS C:\Users\Admin\Documents\coudez> cd smtp
PS C:\Users\Admin\Documents\coudez\smtp> python smtp.py
C:\Users\Admin\Documents\coudez\smtp\smtp.py:1: DeprecationWarning: The smtpd module is deprecated and unmaintained and will be remov
ed in Python 3.12.  Please see aiosmtpd (https://aiosmtpd.readthedocs.io/) for the recommended replacement.
  import smtpd
Local SMTP server running on port 1025...
Incoming message from: ('::1', 51425, 0, 0)
From: ibolarinwa606@gmail.com
To: ['bolaismail07@gmail.com']
Data:
b'Content-Type: text/plain; charset="us-ascii"\nMIME-Version: 1.0\nContent-Transfer-Encoding: 7bit\nFrom: ibolarinwa606@gmail.com\nTo
: bolaismail07@gmail.com\nSubject: Test Email\n\nThis is a test email sent to the custom SMTP server.'
Incoming message from: ('::1', 51433, 0, 0)
From: spoofed@example.com
To: ['bolaismail07@gmail.com']
Data:
b'Content-Type: text/plain; charset="us-ascii"\nMIME-Version: 1.0\nContent-Transfer-Encoding: 7bit\nFrom: spoofed@example.com\nTo: bo
laismail07@gmail.com\nSubject: Spoofed Email\n\nThis is a spoofed email.'
```

The last part in the terminal shows the result of the spoofing.

This observation underscores the alarming reality of SMTP protocol vulnerabilities, where malicious actors can exploit the lack of robust sender authentication mechanisms to orchestrate spoofing attacks. Despite the absence of legitimate credentials, the SMTP server accepted the spoofed email and forwarded it to the intended recipient without any verification.

Analysis

In analyzing the results of the experiment, it's crucial to delve into why the observed behavior of the protocol should be deemed abnormal and how the cyberattack has induced such behavior.

Firstly, let's address the abnormal behavior exhibited by the SMTP protocol. In a typical scenario, SMTP facilitates the transfer of emails between mail servers and clients reliably and securely. However, in my experiment, I intentionally simulated a spoofing attack, wherein an unauthorized sender impersonates a legitimate source to send emails. This action violates the fundamental trust and authentication mechanisms inherent in email communication protocols.

The abnormality stems from the fact that the spoofed emails successfully bypassed conventional security measures, including sender authentication and recipient verification. Despite the absence of legitimate credentials, the SMTP server accepted and relayed the spoofed emails to the intended recipient. Such behavior is aberrant in a secure email ecosystem and raises serious concerns regarding the vulnerability of SMTP servers to malicious exploitation.

Now, let's elucidate how the cyberattack has precipitated this abnormal behavior. The spoofing attack manipulates the SMTP protocol's lack of robust sender verification mechanisms. By forging the email headers with counterfeit sender information, the attacker circumvents the protocol's authentication checks, deceiving the SMTP server into accepting the spoofed emails as legitimate. Consequently, the SMTP server forwards these malicious emails to the unsuspecting recipient, perpetuating the illusion of authenticity.

The cyberattack exploits inherent weaknesses in the SMTP protocol's design, specifically its reliance on trust-based communication and the absence of stringent sender authentication protocols. This vulnerability enables threat actors to orchestrate spoofing attacks with relative ease, undermining the integrity and trustworthiness of email communications.

In conclusion, the observed abnormal behavior of the SMTP protocol, wherein spoofed emails evade detection and reach their intended recipients, underscores the critical need for enhanced security measures to mitigate the risks posed by cyber threats. Addressing these vulnerabilities demands the implementation of robust authentication mechanisms, such as cryptographic signatures and sender verification protocols, to fortify the resilience of email communication against malicious exploitation. Failure to address these vulnerabilities jeopardizes the confidentiality, integrity, and authenticity of email communications, thereby exposing individuals and organizations to grave security risks in an increasingly interconnected digital landscape.