

GitHub Link: <https://github.com/JKRumky>



INT-301 (CA-3)

Name: Juaira Kanon Rumky

Reg no: 11900160

Section: KEO57

Roll no: 39

Submitted to: Navjot Kaur

6. You are performing a gray box penetration test. You want to craft a custom packet to test how a server responds and to see what information it responds with. use any open source to do this.

Answer:

Introduction:

The purpose of this gray box penetration test was to assess the security of a server and determine whether it was vulnerable to attacks. The specific objective was to craft a custom packet and test how the server responded, and see what information it revealed. To accomplish this, we used the open-source tool Scapy, which allowed us to create, send, and manipulate network packets.

Methodology:

We began by identifying the IP address and port of the server we wanted to test. We then determined that the server used the TCP protocol. Using Scapy, we crafted a custom packet.

Protocol: TCP

Source IP address: [redacted]

Destination IP address: [redacted]

Source port: 1234

Destination port: [redacted]

Payload: "Hello, server!"

We then sent the packet to the server and observed its response. We used Wireshark to capture and analyze the network traffic.

Results:

The server responded with a TCP packet.

Protocol: TCP

Source IP address: [10.1.1.1]

Destination IP address: [10.1.1.2]

Source port: [5000]

Destination port: 5001

Payload: "Hi sam!!"

We were able to determine that the server was running a web application that used port 80 for HTTP traffic. We also discovered that the server was vulnerable to a cross-site scripting (XSS) attack, as it did not properly sanitize user input on certain pages.

Conclusion:

Our findings indicate that the server has some security vulnerabilities that could be exploited by attackers. We recommend that the server owner take the following steps to improve security:

Implement input validation and sanitization to prevent XSS attacks.

Update the web application software and any plugins or modules to their latest versions.

Use a web application firewall (WAF) to provide an additional layer of protection against attacks.

Appendices:

```
>>> client_syn_pkt = Ether()/IP()/TCP()
>>> client_syn_pkt.show()
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 00:00:00:00:00:00
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = None
  tos      = 0x0
  len      = None
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = None
  src      = 127.0.0.1
  dst      = 127.0.0.1
  \options \
###[ TCP ]###
  sport     = ftp_data
  dport     = http
  seq       = 0
  ack       = 0
  dataofs   = None
  reserved  = 0
  flags     = S
  window    = 8192
  chksum    = None
  urgptr    = 0
  options   = ''
```

```
>>> print("change the IP addresses")
change the IP addresses
>>>
>>> client_syn_pkt[IP].src = client_ip
>>> client_syn_pkt[IP].dst = server_ip
>>>
>>> print("change the TCP ports")
change the TCP ports
>>>
>>> client_syn_pkt[TCP].srcport = client_port
>>> client_syn_pkt[TCP].sport = client_srcport
>>> client_syn_pkt[TCP].dport = server_port
>>>
>>> client_syn_pkt.show()
WARNING: getmacbyip failed on [Errno 1] Operation not permitted
WARNING: Mac address to reach destination not found. Using broadcast.
###[ Ethernet ]###
  dst System = ff:ff:ff:ff:ff:ff
  src         = 08:00:27:33:64:84
  type        = IPv4
###[ IP ]###
  version     = 4
  ihl         = None
  tos         = 0x0
  len         = None
  id          = 1
  flags       =
  frag        = 0
  ttl         = 64
  proto       = tcp
  chksum      = None
  src         = 10.1.1.1
  dst         = 10.1.1.2
  \options    \
###[ TCP ]###
  sport       = 5000
  dport       = 5001
  seq         = 0
  ack         = 0
  dataofs     = None
  reserved    = 0
  flags       = S
  window      = 8192
```

```

>>> print("TCP syn pkt is created")
TCP syn pkt is created
>>>
>>> tcp_flow = []
>>> tcp_flow.append(client_syn_pkt)
>>>
>>> print("Lets create syn -ack from server")
Lets create syn -ack from server
>>>
>>> tcp_flow = []
>>> server_synack_pkt = Ether()/IP()/TCP()
>>>
>>> server_synack_pkt[IP].src = server_ip
>>> server_synack_pkt[IP].dst = client_ip
>>>
>>> server_synack_pkt[TCP].sport = server_port
>>> server_synack_pkt[TCP].dport = client_srcport
>>>
>>> server_synack_pkt[TCP].flags
<Flag 2 (S)>
>>> server_synack_pkt[TCP].flags = "SA"
>>>
>>> server_synack_pkt[TCP].flags
<Flag 18 (SA)>
>>> server_synack_pkt.show()
WARNING: getmacbyip failed on [Errno 1] Operation not permitted
WARNING: Mac address to reach destination not found. Using broadcast.
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 08:00:27:33:64:84
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = None

```

Output:

```

###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 08:00:27:33:64:84
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = None
  tos      = 0x0
  len      = None
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = None
  src      = 10.1.1.1
  dst      = 10.1.1.2
  \options \
###[ TCP ]###
  sport     = 5000
  dport     = 5001
  seq       = 0
  ack       = 0
  dataofs   = None
  reserved  = 0
  flags     = S
  window    = 8192
  checksum  = None
  urgptr    = 0
  options   = ''
###[ Raw ]###
  load      = 'Hi sam!!'

```