**FTP Brute Force Attack using Metasploit with Threat Detection via Wireshark**

This project demonstrates a brute-force attack on an FTP service using Metasploit, a penetration testing framework. The goal is to understand how brute-force attacks work, how attackers can exploit weak FTP credentials, and how to monitor and detect such attacks using a network analysis tool like Wireshark.

**What is FTP?**

FTP (File Transfer Protocol) is a standard network protocol used to transfer files between a client and a server.

* Operates over TCP/IP networks.
* Used to upload and download files from servers.
* Default port: 21.
* Does not encrypt data (credentials and files are sent in plaintext).

**How FTP Works**

1. The client connects to the server on port 21.
2. The server requests a username and password.
3. If credentials are valid, the client gains access.
4. Files can be listed, uploaded, downloaded, or deleted.

**Why FTP is Used?**

* Easy way to share files between systems.
* Supported by most operating systems.
* However: Since FTP transmits data in plaintext, it is highly vulnerable to eavesdropping and brute-force attacks.

**What is Metasploit?**

Metasploit is a popular penetration testing framework used by cybersecurity professionals, ethical hackers, and attackers to test and exploit vulnerabilities in computer systems.

**Key Features**

* Exploitation Framework → Ready-made modules to exploit services.
* Auxiliary Modules → Includes brute-force scanners like the FTP login scanner.
* Payloads → Gain access to victim machines.
* Post-Exploitation Tools → Privilege escalation, data theft, etc.
* Community & Updates → Regularly updated with new modules.

**Why Metasploit is Used?**

* Automates brute-force attacks.
* Provides systematic penetration testing.
* Simulates real-world attacks.
* Helps identify weak credentials and improve defenses.

**Project Objective**

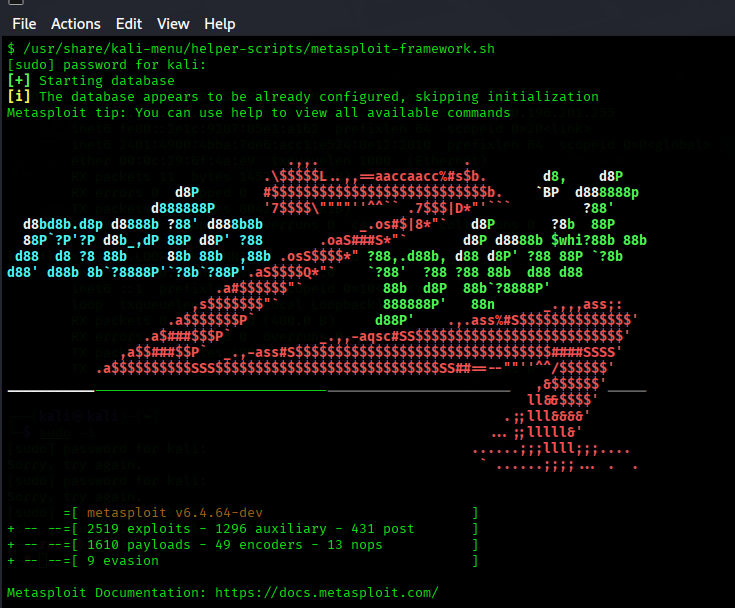
To demonstrate an FTP brute-force attack against a Linux victim machine using Metasploit, identify valid credentials, and monitor the attack traffic using Wireshark.

**Tools and Environment Setup**

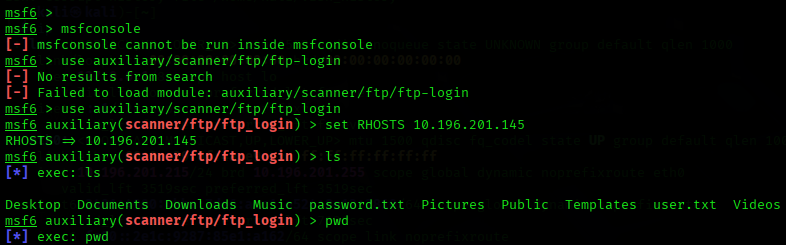
* Attacker Machine: Kali Linux (IP:10.196.201.215) running Metasploit.
* Victim Machine: Ubuntu Linux (IP: 10.196.201.145) with FTP service enabled.
* Monitoring Tool: Wireshark to capture and analyse FTP brute-force traffic.

**Running the Metasploit Attack**

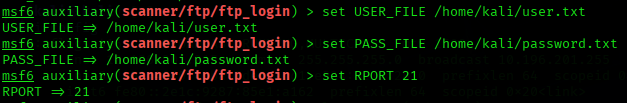
Here is the step-by-step breakdown of the commands you used in **msfconsole**:



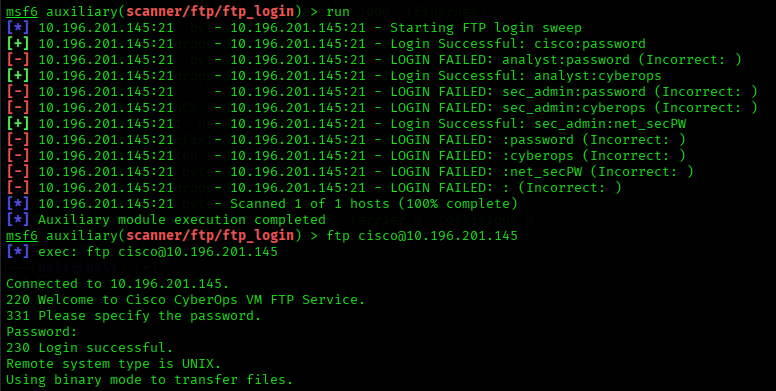
1. **use auxiliary/scanner/ftp/ftp\_login**
   * Loads the FTP brute-force module in Metasploit.
   * This module is designed to try multiple username and password combinations against an FTP service.
2. **set RHOSTS 10.196.201.145**
   * Sets the target victim machine’s IP address (the Linux machine running FTP).
   * Here, 192.168.1.71 is the victim system.
3. **ls and pwd**
   * These are basic Linux commands executed from within msfconsole.
   * **ls** → lists files in the current directory (to check if username/password files exist).
   * **pwd** → shows the current working directory (/root in this case).



1. **set USER\_FILE /root/user.txt**
   * Tells Metasploit where to find the list of possible usernames.
   * The file user.txt contains a list of usernames that the attacker will try (e.g., cisco, analyst, sec\_admin).
2. **set PASS\_FILE /root/passwords.txt**
   * Specifies the list of possible passwords to test.
   * passwords.txt contains a collection of common or guessed passwords (e.g., password, cybercops, net\_secPW).



1. **set RPORT 21**
   * Defines the port number for the FTP service.
   * FTP runs on port 21 by default, so this confirms the module will attack the correct service.
2. **run**
   * Starts the brute-force attack.
   * Metasploit begins testing all possible username-password combinations from user.txt and passwords.txt.
   * When a valid pair is found, Metasploit displays a success.
3. **ftp cisco@10.196.201.145**
   * After finding valid credentials (ftp user as username and its password), the attacker uses a standard FTP client to log into the victim machine manually.
   * This confirms that the brute-force attack was successful.
   * At this point, the attacker has access to the victim’s FTP server.



1. **vi hacked.txt**
   * Once logged into the victim system, the attacker uploads a file named hacked.txt.
   * This step is used as proof-of-access:
     + It shows the attacker was able to log in successfully.
     + The file may contain a message like “your Machine has been attacked”.
   * In real-world attacks, instead of leaving a harmless file, attackers could upload malware, steal data, or create backdoors.

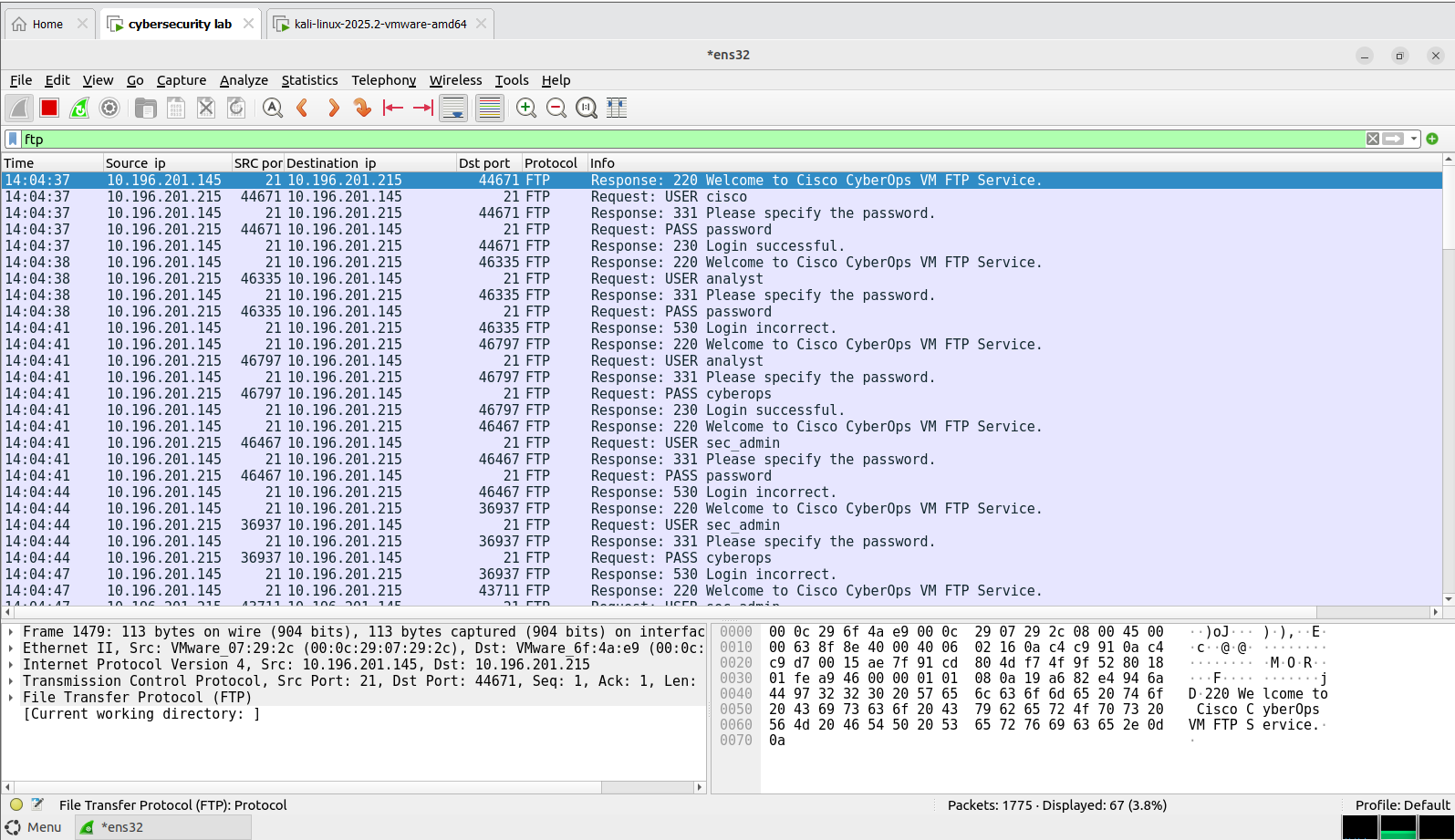
**Threat Detection**

**1. Monitoring with Wireshark**

Wireshark captured and analyzed the attack traffic:

* Multiple USER and PASS requests were observed (brute-force attempts).
* Cleartext credentials were visible in packet captures.
* Successful login detected when the server replied with 230 Login successful.

**2. Evidence in Captures**



* Wireshark shows the brute-force attack on the FTP server.
* USER = username attempt.
* PASS = password attempt.
* 530 Login incorrect = failed login.
* 230 Login successful = attacker found the right credentials and logged in.
* Since FTP is plaintext, usernames and passwords are clearly visible in the capture.
* Repeated traffic pattern indicating automated brute-force.
* Successful login confirmation followed by file transfer (STOR hacked.txt).

**Summary**

This project demonstrates an FTP brute-force attack using Metasploit. The attacker successfully identified valid credentials and uploaded a proof-of-access file (hacked.txt) to the victim machine. Wireshark captured clear evidence of the attack, including multiple failed attempts and eventual success.

This highlights:

* The insecurity of FTP due to plaintext credentials.
* The risk of weak usernames and passwords.
* The importance of monitoring network traffic to detect brute-force attempts.

**Author Details**

Name: Priyanka H S  
Project Title: FTP Brute Force Attack using Metasploit with Monitoring via Wireshark