INTERNSHIP PROGRAM

BY

CENTRAL UNIVERSITY OF JAMMU

TOPIC ALLOTED:

Development of Web/Application-Based Automated Cyber Security Assessment Tool

TIME PERIOD: 6 Weeks

NAME: ANKIT LAMBA

ROLL No. 22BECCS06

Content

Objective and goals of Project

The project focuses on leveraging various libraries to develop a comprehensive web application capable of performing several network-related functions, such as Port scanning which is the process of probing a network host for open ports to identify services and potential security vulnerabilities, Host discovery which is the process of identifying active devices on a network to map its structure and detect potential security threats, and Footprinting, it is the process of collecting data about a target network or system to identify vulnerabilities and prepare for further attacks. To achieve these capabilities, we will go through a range of tools, including `socket` for low-level network operations, `python-nmap` for more advanced network scanning tasks, and various web frameworks such as Django and Flask for building the web interface and handling server-side operations.

The choice between these frameworks will depend on the specific requirements of our project, such as the complexity of the application, the need for built-in features, and the development timeline. By carefully evaluating these tools and frameworks, we aim to select the most suitable ones to ensure our web application is efficient, scalable, and user-friendly. Through this project, we aim to provide a robust solution for network analysis and management, leveraging the strengths of each library and framework to meet our goals.

INTRODUCTION

In this project, we have explored a range of advanced cybersecurity tools, enhancing their technical acumen in digital protection. They will develop a sophisticated Flask-based web portal for comprehensive security assessments, integrating APIs to automate critical tasks such as host discovery and port scanning. This portal will serve as a robust platform for conducting and managing security evaluations, providing a seamless and efficient assessment process. Additionally, students will learn to analyze assessment results, generate detailed reports, and recommend mitigation strategies. Through this initiative, they will gain practical expertise in identifying vulnerabilities and securing systems, equipping them with the skills necessary to tackle real-world cybersecurity challenges.

IMPORTANCE

1. **Hands-On Experience:** Provides practical, hands-on experience with cutting-edge cybersecurity tools, enhancing learning beyond theoretical knowledge.

2. **Real-World Skills:** Equips students with skills directly applicable to real-world cybersecurity challenges, making them job-ready.

3. **Automation Proficiency:** Teaches the automation of security assessments, increasing efficiency and accuracy in identifying vulnerabilities.

4. **Web Development Integration:** Combines cybersecurity with web development, fostering a versatile skill set.

5. **API Integration:** Familiarizes students with API integration, a crucial aspect of modern cybersecurity and software development.

6. **Comprehensive Assessments**: Enables comprehensive security assessments, covering critical areas such as host discovery and port scanning.

7. **Enhanced Security Practices:** Promotes best practices in cybersecurity, contributing to more secure digital infrastructures.

8. **Critical Thinking:** Develops analytical and critical thinking skills through the analysis of assessment results and the recommendation of mitigation strategies.

9. **Career Preparation:** Prepares students for various roles in cybersecurity, increasing their employability in a high-demand field.

10. **Innovation and Adaptability:** Encourages innovation and adaptability by challenging students to create and improve cybersecurity solutions, preparing them for evolving technological landscapes.

OBJECTIVES

The objective of this project is to provide students with a comprehensive understanding of advanced cybersecurity tools and techniques through practical, hands-on experience. By developing a sophisticated Flask-based web portal, students will learn to conduct and manage security assessments efficiently, automating tasks such as host discovery and port scanning. This approach not only enhances their technical skills but also integrates critical aspects of web development and API utilization, fostering a versatile and robust skill set.

Additionally, the project aims to cultivate students' analytical abilities by teaching them to interpret assessment results, generate detailed reports, and recommend effective mitigation strategies. Through this process, students will gain valuable insights into identifying and addressing vulnerabilities, preparing them for real-world cybersecurity challenges. Ultimately, the objective is to equip students with the knowledge and skills necessary to protect digital infrastructures and excel in the rapidly evolving field of cybersecurity.

RELEVENT RESEARCH

WASCA is a novel and intelligent system capable of aiding organizations in drawing conclusions about cyber-attacks allowing them to amend their cybersecurity risk management measures with minimal human effort. WASCA was researched and developed in two stages:

Dictionary based

WASCA-D (Web Application Security Assessment and Defence) is a comprehensive framework aimed at evaluating and enhancing the security of web applications. This approach systematically identifies vulnerabilities, analyses risks, and implements robust defence mechanisms. The assessment phase employs various tools and techniques, including automated scanners and manual testing, to uncover potential security flaws such as SQL injection, cross-site scripting (XSS), and insecure authentication methods. By prioritizing these vulnerabilities, organizations can effectively address security issues through secure coding practices, configuration management, and regular security audits.

The framework emphasizes the importance of continuous monitoring and updating security measures to adapt to evolving threats. WASCA-D not only focuses on immediate threat mitigation but also on long-term security resilience. By integrating this proactive approach, organizations can ensure their web applications are fortified against cyber-attacks, protecting sensitive data and maintaining user trust. In today's digital landscape, where threats are increasingly sophisticated, adopting WASCA-D is essential for maintaining robust web application security.

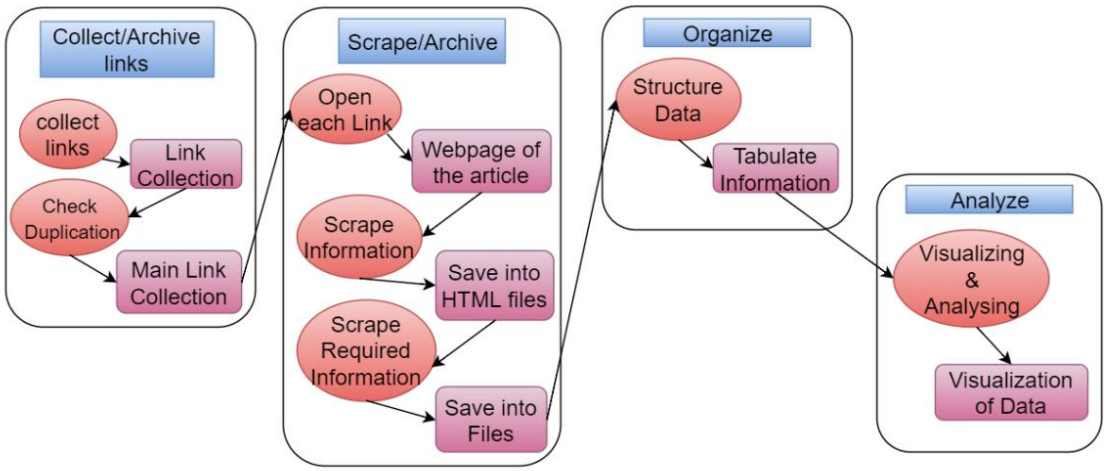


FIGURE 3.1: *WASCA-D Architecture*

NLP-based

WASCA-NLP (Web-based Automated Security and Compliance Assessment using NLP) is an innovative approach that leverages natural language processing (NLP) to enhance security and compliance assessments. By integrating advanced NLP algorithms, WASCA-NLP can automatically analyse large volumes of security documentation, identify compliance issues, and generate detailed reports. This system significantly reduces the manual effort required in security assessments, making the process more efficient and accurate. The web-based nature of WASCA-NLP ensures accessibility and ease of use, allowing security professionals to conduct thorough assessments from anywhere.

Furthermore, WASCA-NLP is designed to stay updated with the latest security standards and regulations, ensuring that assessments are always aligned with current requirements. This adaptability is crucial in the ever-evolving field of cybersecurity, where new threats and compliance mandates frequently emerge. By automating the analysis and reporting process, WASCA-NLP not only saves time but also minimizes human error, providing reliable and

consistent results. This technology represents a significant advancement in the realm of automated security assessments, enhancing both efficiency and effectiveness in maintaining robust security postures.

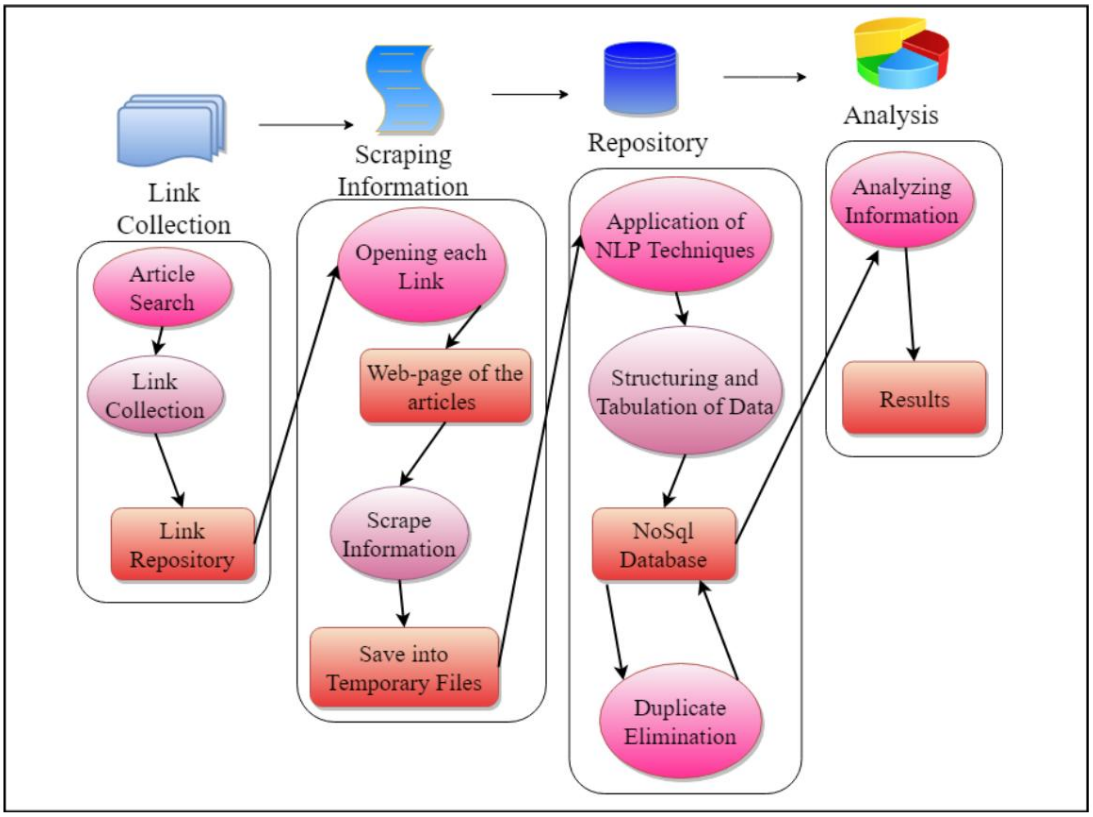
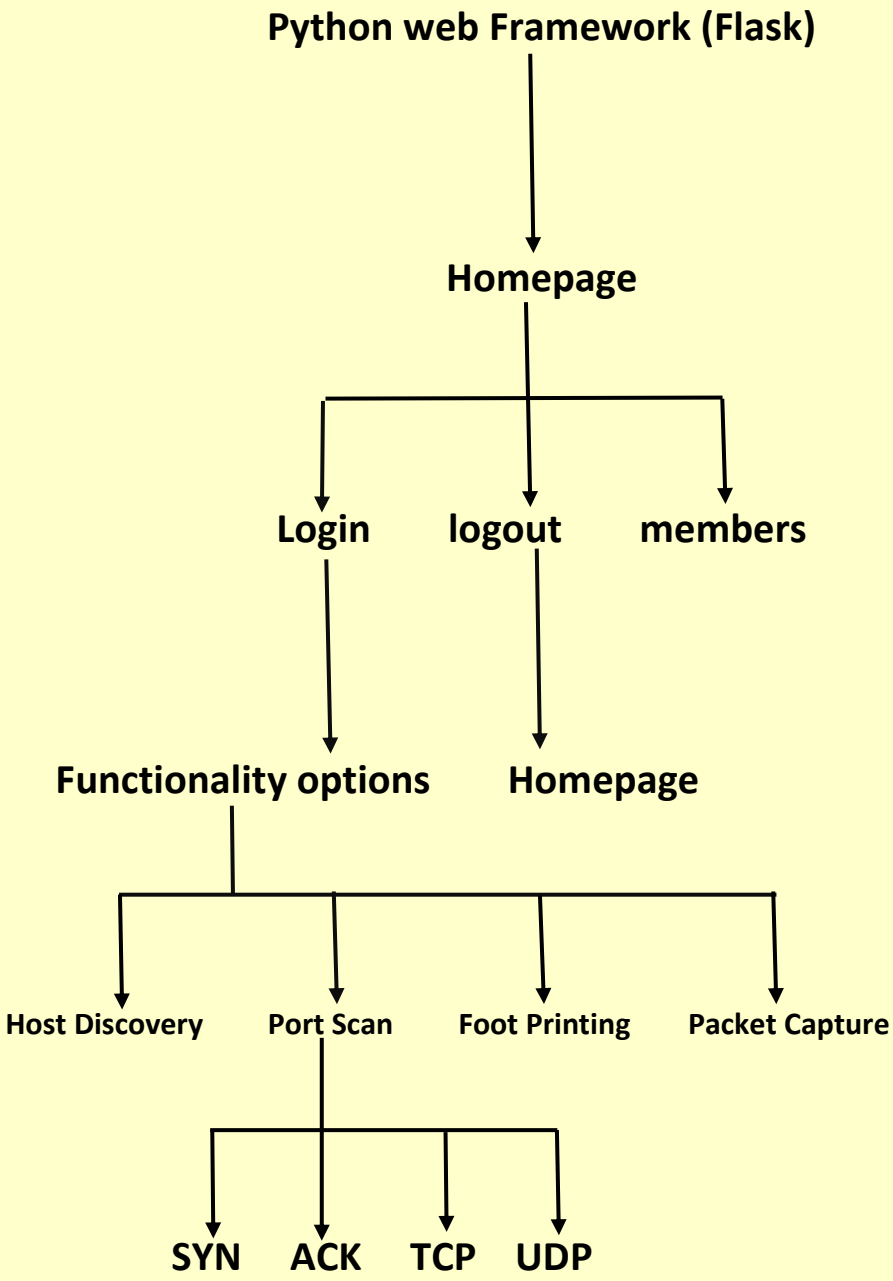


FIGURE 4.1: *WASCA-NLP Architecture*

Methodology

- **Selecting a Web Development Framework and Creating Pages:** Begin by choosing an appropriate web development framework to create a comprehensive webpage. Develop essential pages such as a login interface, logout functionality, and a project members page. After users log in, offer a range of options including advanced functionalities like port scanning and footprinting.
- **Enhancing User Interface with HTML and CSS:** Utilize HTML and CSS to enhance the user interface and improve overall site usability. These front-end tools play a crucial role in creating a visually appealing and intuitive website design. Implement responsive design principles to ensure the webpage functions seamlessly across various devices and screen sizes.
- **Implementing Python Tools (python-nmap, socket, scapy):** Implement custom functions using powerful Python tools such as python-nmap for network scanning, socket for network communication, and scapy for packet manipulation. These functions will respond dynamically to user requests, executing tasks like port scanning, footprinting, and other network-related operations.
- **Displaying Results in Plain Text Format:** Ensure that the output from the Python functions is presented in plain text format on subsequent pages. This approach enhances readability and facilitates easy interpretation of the results by users.

Flow Chart Representation



Create a webpage

For creating this webpage, we are going to use the Flask framework instead of Django framework as Flask is a small framework by most standards, small enough to be called a "micro framework." It is small enough that once you become familiar with it, you will likely be able to read and understand all of its source code. But being small does not mean that it does less than other frameworks. Flask was designed as an extensible framework from the ground up; it provides a solid core with the basic services, while extensions provide the rest. Because you can pick and choose the extension packages that you want, you end up with a lean stack that has no bloat and does exactly what you need.

Flask has two main dependencies. The routing, debugging, and Web Server Gateway Interface (WSGI) subsystems come from Werkzeug, while template support is provided by Jinja2. Werkzeug and Jinja2 are authored by the core developer of Flask.

A basic code for "hello world" flask application is shown in the figure below:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return '<h1>Hello World!</h1>'

if __name__ == '__main__':
    app.run(debug=True)
```

The name that is being used for the web application is "WELLtol". The Home page of WELLtol contains a logo and a nav bar with options HOME, LOGIN, MEMEBERS and

LOGOUT. With the use of html and CSS we have made the page a bit interactive and had applied a typing animation.

In the login page we have two input bars one will take the input as USERNAME and the second one will take the PASSWORD. The function “tester” will be called at the back end to check if the username and password are matching. If it’s a prefect match it will redirect to the next page other wise will show the login error.

```
@app.route('/tester', methods=['POST','GET'])
def tester():
    if request.method == 'POST':
        user:str = request.form['username']
        pasw:str = request.form['password']
        if user == usern and pasw == passw:
            return redirect("optionList")
        else:
            return f"login failed ERROR 404"
```

After the successful login we will be taken to the page with a number of options of different types of activates one can perform on our application. Options are provided in the check box format so only one option can be selected at a time. Options given to the user are Port scan, Host discovery, Packet capturing and Footprinting.

Integrate tools with the web application

To provide functionalities like port scan, host discovery and footprinting we are going to use a tool called python-nmap. Through this module we can use all the functionalities of nmap in the python functions, it also helps us to use a range of different filters and constraints.

Like for the port scan we can use -p 10-20. Here -p means that it's a port scan and 10-20 is the range of ports it's going to scan. In addition to this we can specify type of scan we want like SYN-ACK, TCP or UDP.

SYN-ACK SCAN: Syn-Ack scan is a type of TCP scan used in network security to identify open ports on a target system. It works by sending TCP ACK packets to a target machine without initiating a full TCP handshake. The response from the target helps determine the state of the post.

TCP SCAN: It involves sending TCP packets to each port and waiting for a response. A response indicates an open port, while no response or a reset packet suggests a closed port.

UDP SCAN: It sends UDP packets to each port and checks for responses. Since UDP is connectionless, the absence of a response may indicate an open port, making it less reliable but faster.

PORT scanning and its Types

```
if scan_type == "SYN_ACK":
    scanner.scan(ip_address,'1-1024','-v -sS')
    c = scanner.scaninfo()
    d = f"Ip Status: , {scanner[ip_address].state()}"
    e = f"{scanner[ip_address].all_protocols()}"
    f = f"Open Port: , {scanner[ip_address]['tcp'].keys()}"
    return render_template("result.html",a=a,b=b,c=c,d=d,e=e,f=f)

elif scan_type == "UDP":
    scanner.scan(ip_address,'1-1024','-v -sU')
    c = scanner.scaninfo()
    d = f"Ip Status: , {scanner[ip_address].state()}"
    e = f"{scanner[ip_address].all_protocols()}"
    f = f"Open Port: , {scanner[ip_address]['udp'].keys()}"
    return render_template("result.html",a=a,b=b,c=c,d=d,e=e,f=f)

elif scan_type == "TCP":
    scanner.scan(ip_address,'1-1024','-v -sT')
    c = scanner.scaninfo()
    d = f"Ip Status: , {scanner[ip_address].state()}"
    e = f"{scanner[ip_address].all_protocols()}"
    f = f"Open Port: , {scanner[ip_address]['tcp'].keys()}"
    return render_template("result.html",a=a,b=b,c=c,d=d,e=e,f=f)
```

Host scanning, in host scanning we scan our network to find all the active host, for this purpose we have -sn, here -sn perform a ping scan and check if the hosts are up or not.

HOST Discovery

```
@app.route("/Hdscan", methods = ['POST'])
def hds():
    ip_add = request.form["IP"]
    p = request.form["bits"]
    ip = str(ip_add) + "/" + str(p)
    nm = nmap.PortScanner()
    nm.scan(hosts=ip,arguments="-sn")
    nm.scan(hosts=ip, arguments = "-sn")
    lis = [(x,nm[x]['status']['state']) for x in nm.all_hosts()]
    return render_template('hdresult.html',lis=lis)
```

Footprinting, in this we thoroughly scan a target and try to gather as much info as possible like OS details, services running and their versions, in addition to the open ports. To

accomplish this task, we have -A which is used for aggressive scan which will provide us with all the information about OS and versions of the services running on the target.

FOOTPRINTING

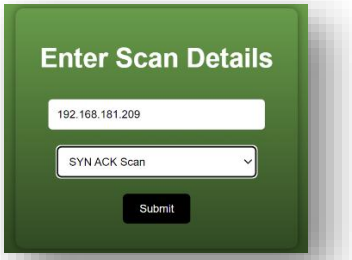
```
@app.route("/footing", methods = ['POST'])
def foting():
    target = request.form["IP"]
    nm = nmap.PortScanner()
    nm.scan(hosts=target, arguments='-A')

    for host in nm.all_hosts():
        lis=[]
        a = f"Host: {host} ({nm[host].hostname()})"
        b = f"State: {nm[host].state()}"
        for proto in nm[host].all_protocols():
            c = f"Protocol: {proto}"
            lport = nm[host][proto].keys()
            for port in sorted(lport):
                bis = []
                d = f"Port: {port}\tState: {nm[host][proto][port]['state']}"
                if 'version' in nm[host][proto][port]:
                    e = f"Version: {nm[host][proto][port]['version']}"
                if 'product' in nm[host][proto][port]:
                    f = f"Product: {nm[host][proto][port]['product']}"
                if 'extrainfo' in nm[host][proto][port]:
                    g = f"Extra Info: {nm[host][proto][port]['extrainfo']}"
                if 'reason' in nm[host][proto][port]:
                    h = f"Reason: {nm[host][proto][port]['reason']}"
                if 'cpe' in nm[host][proto][port]:
                    i = f"CPE: {nm[host][proto][port]['cpe']}"
```

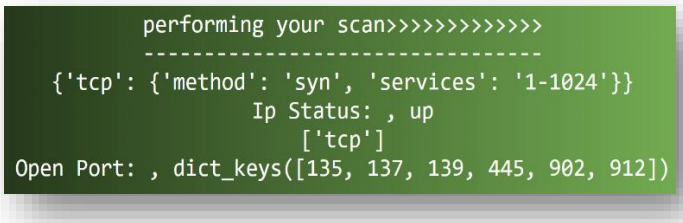
Now, for the task of packet capturing we will use the scapy module of the python. Scapy in itself is a phenomenal tool which provides a number of functionalities like packet formation, packet capturing and more. And there are even a range of attacks that one can perform only using scapy like DOS attack and attack on spanning tree protocol, etc.

```
@app.route("/pkgcap", methods = ['POST'])
def pcap():
    c = int(request.form["PAK"])
    sniff(count = c, prn=packet_display)
    return render_template("pkgresult.html",Pak=Pak)
```

Results and Discussion

Results for the Port scan using the SYN-ACK scan:

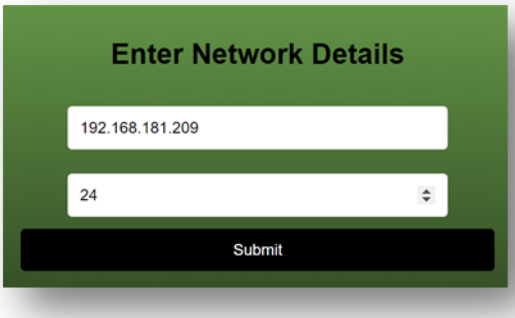




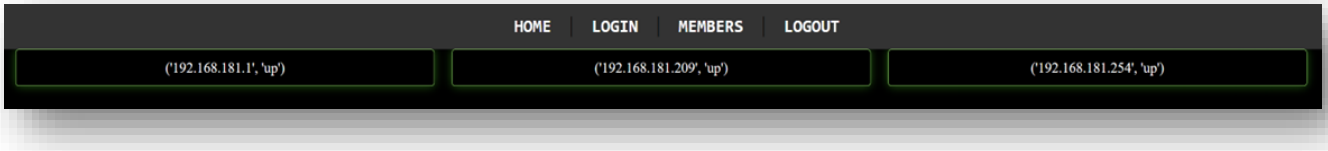
Using this scan we get a number of information like:

- Method used: syn
- Number of ports scanned: 1-1024
- Ip status of target: up/Alive
- Open ports: 135,137,139,445,902,912
- Protocol used for detect open ports: TCP

Results for the host discovery:

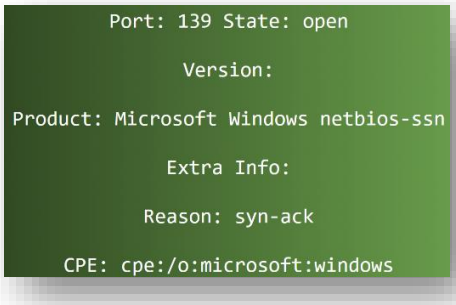


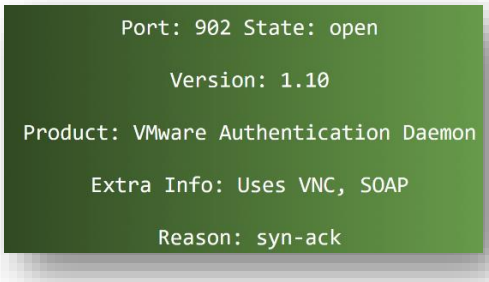
First we insert the details like our Network address or our IP address and then we ask for the number of bits used to identify the network.



Upon submitting the details asked we will get the IP address of all the hosts which are up in these blocks like structure shown in the above image. In the blocks we have the IP address and the status of that IP.

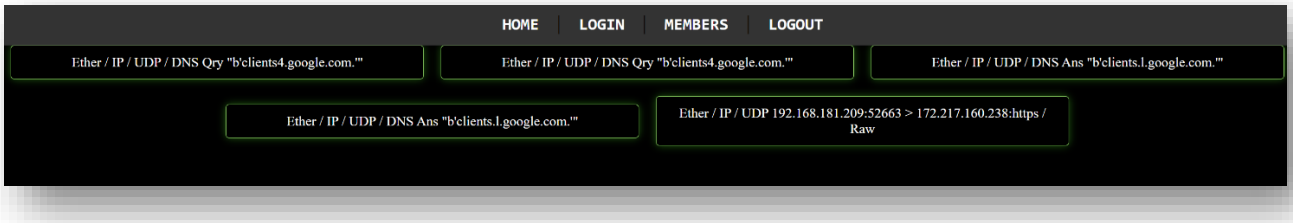
Result of the Footprinting:





As you can see here, footprinting provides much more information compared to a port scan. We obtain details related to open ports, services, the version of the service running, OS details, and some extra information. This comprehensive data allows for a deeper understanding of the target system's vulnerabilities and potential entry points.

Result of the packet capturing:



The captured packet summary using Scapy reveals detailed information about network communication. It shows DNS queries and responses, including a query for "b'clients4.google.com.'" and a corresponding answer for "b'clients.l.google.com.'" Additionally, it includes a UDP packet from a local IP address (192.168.181.209) to Google's HTTPS service (172.217.160.238). This packet summary highlights the resolution of domain names to IP addresses and the subsequent data exchange, providing insight into the network's activity and the interaction between the local system and external servers.

Conclusion

Key Achievements

- The very first thing is gaining a better understanding of Flask, which is a lightweight and beginner-friendly web framework. We learned how to call different functions when an action is performed on the web page, making it easier to handle user interactions and dynamic content. This foundational knowledge is crucial for developing robust and interactive web applications efficiently.
- We learned how to use the python-nmap module to perform a number of tasks like port scanning, host discovery, and footprinting, which are essential for network security analysis and vulnerability assessment. This knowledge enables us to identify potential vulnerabilities, monitor network activity, and enhance overall network security measures.
- Next, we learned about scapy, scapy is a wonderful tool which can be used for a number of things like DOS attack, packet formation, we can perform a number of different attacks like a attack on spanning tree protocol. But we used scapy to capture packets and see the summery of the captured packets.
- Finally, we were able to successfully create a comprehensive web application that performs port scanning, footprinting, packet capturing, and various other network security tasks. This tool integrates multiple functionalities to enhance our network analysis and vulnerability assessment capabilities.

limitations of the project

- The very first limitation is that in port scanning, we can only scan ports in the range of 1-1024. This restriction prevents us from analyzing higher-numbered ports, which could also be critical for security assessments.
- The second limitation is that we can only show the summary of the captured packet and not the entire content of the packet. This restricts our ability to analyze detailed packet information and might limit thorough investigations.
- One of the major limitations is that we cannot store or download the output we have gathered from the function. This restriction hinders our ability to retain and analyze the data for future reference or detailed examination.
- In port scanning, there are typically three primary methods available: TCP connect scans, SYN scans, and UDP scans. Each method serves different purposes and has varying levels of stealth and effectiveness depending on the network environment and the goals of the scan.
- In the backend of the website, there is currently no connection to a database for fetching or storing any data. This setup limits the functionality related to dynamic content management and user-specific data storage, which are essential for many interactive and personalized web applications. Integrating a database would allow for efficient data management, retrieval, and storage, enabling features such as user profiles, content management systems, e-commerce transactions, and more, thereby enhancing the overall functionality and usability of the website.

References

- https://www.youtube.com/watch?v=FrT_n5BkpK4&t=431s
- https://www.youtube.com/watch?v=yefgBA1CecI
- https://www.youtube.com/watch?v=Yh23ZtfYOSs&t=1671s
- https://scholarworks.unr.edu/bitstream/handle/11714/6007/Pillai_unr_0139M_12949.pdf?sequence=1