INDEX

Report

1. Project Title

Hunt3r : An Automated Penetration Testing Framework

2. Abstract

Hunt3r stands at the forefront of security testing innovation, offering an advanced automated penetration testing framework. Its sophisticated design integrates seamlessly with organizational structures, aiming to revolutionize security processes. By leveraging state-of-the-art techniques and a modular architecture, Hunt3r enhances efficiency, empowering enterprises to stay ahead in identifying and addressing vulnerabilities. This proactive approach ensures robust network, application, and system security. With Hunt3r, organizations gain a comprehensive solution that streamlines testing procedures and provides actionable insights. It embodies a commitment to continuous improvement, enabling businesses to fortify their defenses and safeguard against evolving cyber threats effectively.

3. Introduction

In today's dynamic threat landscape, organizations confront escalating hurdles in upholding resilient cybersecurity defenses. Amidst this backdrop, penetration testing emerges as a crucial tool for identifying and mitigating security risks. By simulating real-world attacks and evaluating the efficacy of existing security measures, it provides invaluable insights into vulnerabilities. However, traditional manual penetration testing methodologies often prove laborious, demanding significant resources, and susceptible to human error.

Enter the Hunt3r automated penetration testing framework, a pioneering solution tailored to address these pressing challenges. Hunt3r revolutionizes the testing landscape by automating key phases of the testing lifecycle, streamlining processes, and enhancing overall efficiency. Its advanced algorithms and modular architecture enable organizations to conduct thorough security assessments with unparalleled precision and speed.

By automating repetitive tasks and leveraging cutting-edge techniques, Hunt3r empowers security teams to focus their efforts on strategic analysis and response planning. This shift from manual to automated testing not only accelerates the identification of vulnerabilities but also ensures consistent and reliable results across diverse environments.

Moreover, Hunt3r's adaptability and scalability make it suitable for organizations of all sizes and industries. Whether assessing network infrastructure, web applications, or cloud environments, Hunt3r provides a comprehensive solution that aligns with evolving security needs.

Furthermore, Hunt3r enhances collaboration among security professionals by facilitating the sharing of insights and best practices through its intuitive interface and reporting capabilities.

This collaborative approach fosters a culture of continuous improvement and proactive risk management within organizations.

In essence, Hunt3r represents a paradigm shift in the realm of penetration testing, empowering organizations to stay ahead of emerging threats and safeguard their digital assets with confidence. By automating and optimizing security assessment processes, Hunt3r enables organizations to fortify their defenses and maintain resilience in the face of evolving cyber threats.

4. Literature Review

- **Securing the Internet of Things: A Comprehensive Framework for IoT Penetration Testing**
  [1] present a Penetration Testing Framework for IoT at the IIAI-AAI Congress. Focused on the unique challenges posed by IoT security, the study proposes a comprehensive framework to assess and enhance the security of IoT devices and networks. This research contributes to addressing the growing concerns surrounding IoT security by offering practical solutions for vulnerability assessment and mitigation.

- **Redefining Security Assessments: The Case for Automated Penetration Testing Solutions**
  Research indicates manual penetration tests as labor-intensive, redundant, and error-prone. [2] showed that human involvement in testing processes leads to increased time and resource requirements, as well as higher chances of inaccuracies. Organizations face challenges in maintaining consistent and reliable results across tests, highlighting the need for automated solutions like the Hunt3r framework to streamline security assessments efficiently.

- **Navigating the Cyber Terrain: The Imperative Role of VAPT in Addressing Escalating Cybersecurity Risk**
  The exponential growth of internet-connected devices correlates with a parallel increase in cybersecurity consequences. This trend underscores the critical importance of Vulnerability Assessment and Penetration Testing (VAPT). [3] highlights the escalating risks posed by cyber threats in tandem with the expanding digital landscape. VAPT emerges as a vital strategy for organizations to proactively identify and address vulnerabilities, mitigating potential cybersecurity incidents effectively.

- **Advancing Cybersecurity Through Machine Learning: A Study on Automated Penetration Testing Techniques**
  [4] explore the utilization of machine learning in automating penetration testing. Their study, presented at ICE3IS, highlights the efficacy of machine learning techniques in enhancing the automation of penetration testing processes. The research contributes to advancing cybersecurity strategies by leveraging artificial intelligence for proactive vulnerability assessment and mitigation

5. Problem Statement

Existing manual penetration testing methods are plagued by time constraints, errors, and scalability issues. There's a pressing demand for a user-friendly, adaptable tool that provides comprehensive assessments and customizable features. This tool must efficiently pinpoint vulnerabilities across varied environments, offering actionable insights for effective remediation. By bridging these gaps, such a solution would significantly enhance cybersecurity efforts, ensuring robust protection against evolving threats in today's dynamic digital landscape.

6. Objectives

- **Automate key pen testing tasks**: Reduce manual effort, freeing up security experts for strategic work.
- **Enable continuous security monitoring**: Schedule regular scans to stay ahead of emerging vulnerabilities and track remediation progress.
- **Improve security assessment efficiency**: Reduce time and resource demands for comprehensive security testing.
- **Increase vulnerability detection accuracy**: Minimize false positives and prioritize critical vulnerabilities for faster remediation.
- **Foster proactive security posture**: Enable organizations to identify and address vulnerabilities before attackers exploit them.

7. Technology Used

Python: Python will serve as the primary programming language for developing the project.

Subprocess: The subprocess module is used to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.

Time: The time module is used for various time-related functions, such as waiting or sleeping.

Nmap: The nmap library is used for network exploration and security auditing. It allows scanning of networks for open ports, service versions, and other details.

Colorama: The colorama library is used for adding color and style to the output in the terminal.

Cryptography: The cryptography library is used for encryption and decryption purposes. It provides cryptographic recipes and primitives.

WHOIS: The whois command-line tool is used for querying WHOIS databases to obtain information about domain names and IP addresses.

DNSRecon: The dnsrecon command-line tool is used for DNS reconnaissance, such as querying DNS servers for information about domain names.

Sublist3r: The sublist3r command-line tool is used for subdomain enumeration, helping to discover subdomains associated with a target domain.

Gobuster: The gobuster command-line tool is used for directory and file brute-forcing. It helps in discovering hidden files and directories on web servers.

Searchsploit: The searchsploit command-line tool is used for searching and displaying exploits from the Exploit Database.

Ping: The ping command-line tool is used for checking network connectivity by sending ICMP echo requests to a target host.

Waf00f: The waf00f command-line tool is used for detecting web application firewalls (WAFs) and filtering appliances.

8. Methodology

Pre-Engagement and Planning:

Gather information: Define target scope, network architecture, and risk tolerance.
Customize scan parameters: Adjust scan depth, intensity, and excluded areas based on specific needs.
Schedule scans: Set up regular scans for continuous monitoring or one-time assessments.

Reconnaissance and Information Gathering:

Passive reconnaissance: Collect publicly available information about the target network and potential vulnerabilities.
Active discovery: Identify hosts, services, and open ports with tools like nmap and Netdiscover.
Fingerprint systems and services: Gather version and operating system information for vulnerability targeting.

Vulnerability Scanning and Exploitation:

Identify vulnerabilities: Utilize automated scanners like OpenVAS, Nexpose, or Qualys to identify known vulnerabilities.
Exploit vulnerabilities: Employ tools like Metasploit and custom scripts to attempt exploitation based on scan results.
Assess impact and severity: Evaluate the potential impact of identified vulnerabilities and prioritize critical issues.

Post-Exploitation and Reporting:

Gather post-exploitation information: Collect evidence of successful exploitation and potential access gained.
Generate detailed reports: Provide comprehensive reports with vulnerability descriptions, severity levels, remediation steps, and CVSS scores.
Visualize results: Offer report for easy understanding of vulnerabilities and scan findings.

9. IMPLEMENTATION

Initialization

- Set up testing environment
- Install and configure required tool - cryptography, python-nmap, colorama

Reconnaissance

- Automated gathering of information about the target system and network by using OSINT tools such as Whois
- Whois collects the publicly available information from open sources

Scanning

- Automated scanning for vulnerabilities using tools like Nmap, Nessus
- Nmap provides the relevant information regarding open ports and versions of the applications/software running on the target

Exploitation

- Automated exploitation of identified vulnerabilities using tools like Metasploit, ExploitDB, Searchsploit

Post-Exploitation

- Automated analysis of compromised systems for further exploration and lateral movement.

Reporting

- Generate comprehensive reports with findings and recommendations for exploits to penetrate into the system/network.
- Generate encrypted report that is not readable for unauthorized users, encryption using Fernet

Implemented Python Code

```
def whois_scan(target):
    print("\n\033[92m" + "_"*100 + "\033[0m\n")
    print(f"{Fore.CYAN}\n\n[*] Performing whois scan...\n{Style.RESET_ALL}")
    f = open(f"{target}_scans.txt", "a")
    f.write("\n\033[92m" + "_"*100 + "\033[0m\n")
    f.write("\n\nWhois scan result: \n")
    f.close()
    target1 = target
    if(target1[:2]=='10' or target1[:3]== '172'or target1[:3]== '192'):
        print(f"{Fore.RED}\nSorry, You're Trying to scan Private IP Address\n{Style.RESET_ALL}")
        f = open(f"{target}_scans.txt", "a")
        f.write("No Report Availvale for this scan :)\n\n")
        f.close()
    else:
        whois_result = run_command(f"whois {target}",timeout=15)
        if(whois_result==""):
            print(f"{Fore.RED}\nNo Result Available\n{Style.RESET_ALL}")
            f = open(f"{target}_scans.txt", "a")
            f.write("No Report Availvale for this scan :)\n\n")
            f.close()
        elif(whois_result[:7]=='Timeout'):
            print(f"{Fore.RED}\nTimed Out...\n{Style.RESET_ALL}")
            f = open(f"{target}_scans.txt", "a")
            f.write("No Report Availvale for this scan :)\n\n")
            f.close()
        else:
            print(whois_result)
            f = open(f"{target}_scans.txt", "a")
            f.write("Information about domain names, IP addresses, and other resources registered with the Interr
            f.write(whois_result)
```

Figure 10.1: Code for whois scan

```
def check_connection(target):
    temp = run_command(f"ping -c 2 {target}",timeout=10)
    if(len(temp)>100):
        return 1
    else:
        return 0


def main():
    t1 = run_command(f"figlet Hunt3r",timeout=2)
    print("\033[94m"+t1+"\033[0m")
   # print(run_command(f"figlet Hunt3r",timeout=2))

    check = input(f"{Fore.GREEN}If you want to decrypt a report enter '-d'\nIf you want to scan a new target ente
'-ns'\n: {Style.RESET_ALL}")
    if(check=='-d'):
        # Load the key from the file
        filename = input("Enter the file name: ")
        key = load_key("key.key")
            # Decrypt a file
        decrypt_file(filename, key)

    elif(check=='-ns'):
        target = input(f"{Fore.RED}\nEnter the target domain or IP address: {Style.RESET_ALL}")
        if(check_connection(target)):
            print("\n")
            print("\033[92m"+"Target is up"+"\033[0m")
            f = open(f"{target}_scans.txt", "w")
            f.write(f"Scanning results for {target}    \n")
```

Figure 10.2: Code for checking connection and implementing main function

```
    # Print and write the results
    output = ''
    a1 = ''
    a2 = ''
    a3 = ''
    for host in nmScan.all_hosts():
        output += 'Host : %s (%s)\n' % (host, nmScan[host].hostname())
        output += 'State : %s\n' % nmScan[host].state()
        for proto in nmScan[host].all_protocols():
            output += 'Protocol : %s\n' % proto
            lport = nmScan[host][proto].keys()
            for port in lport:
                output += 'Port : %s\tService : %s\tVersion : %s\tCPE : %s\n' % (port, nmScan[host][proto][port]
['name'], nmScan[host][proto][port]['version'], nmScan[host][proto][port]['cpe'])
                # v_lst.append(nmScan[host][proto][port]['name']+" "+nmScan[host][proto][port]['version'])
                v_lst.append(nmScan[host][proto][port]['cpe'])
                a = nmScan[host][proto][port]['cpe']
                index_of_third_colon = find_third_index(a, char_to_find)
                index_of_fourth_colon = find_fourth_index(a, char_to_find)
                if(index_of_third_colon≠-1 & index_of_fourth_colon≠-1):
                    a1 = a[index_of_third_colon+1:index_of_fourth_colon-1]
                    a2 = a[index_of_fourth_colon+1:]
                elif(index_of_third_colon≠-1 & index_of_fourth_colon==-1):
                    a1 = a[index_of_third_colon+1:index_of_fourth_colon-1]
                # print("a1: ",a1)
                # print("a2: ",a2)

                if(index_of_third_colon≠-1 & index_of_fourth_colon≠-1):
                    a3 = a1+" "+a2
                    final_lst.append(a3)
                if(a3==""):
                    if(a1==""):
                        a3 = a
                        final_lst.append(a3)
                    else:
                        a3 = a1
                        final_lst.append(a3)

                # print("a3: ",a3)

    print(output)
```

Fig 10.3: Code for analysing the scan results

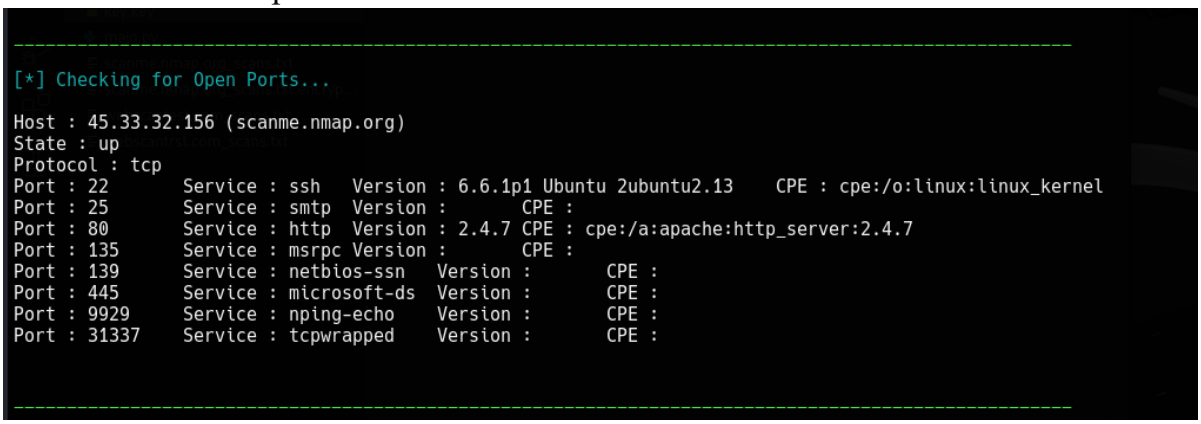
```
def searchForExploits(target):
    print(f"{Fore.CYAN}\n\n\nFor Exploit searching... \n\n{Style.RESET_ALL}",final_lst)
    f = open(f"{target}_scans.txt", "a")
    f.write("\n\033[92m" + "_"*100 + "\033[0m\n")
    f.write("\n\nExploits found for the target: \n")
    f.close()
    print("CPE List: ",v_lst)
    print("\n")
    f = open(f"{target}_scans.txt", "a")
    for a in final_lst:
        print(a)
        f.write(a)

        #    f.write(f"Scanning results for {target}....\n")
        result = run_command(f"searchsploit {a}",timeout=5)
        print(result)
        if(result==""):
            f.write("No Exploits found\n")
        else:
            f.write("Exploits found:\n")
            f.write(result)
    f.close()
```

Figure 10.4: Code for searching the exploits

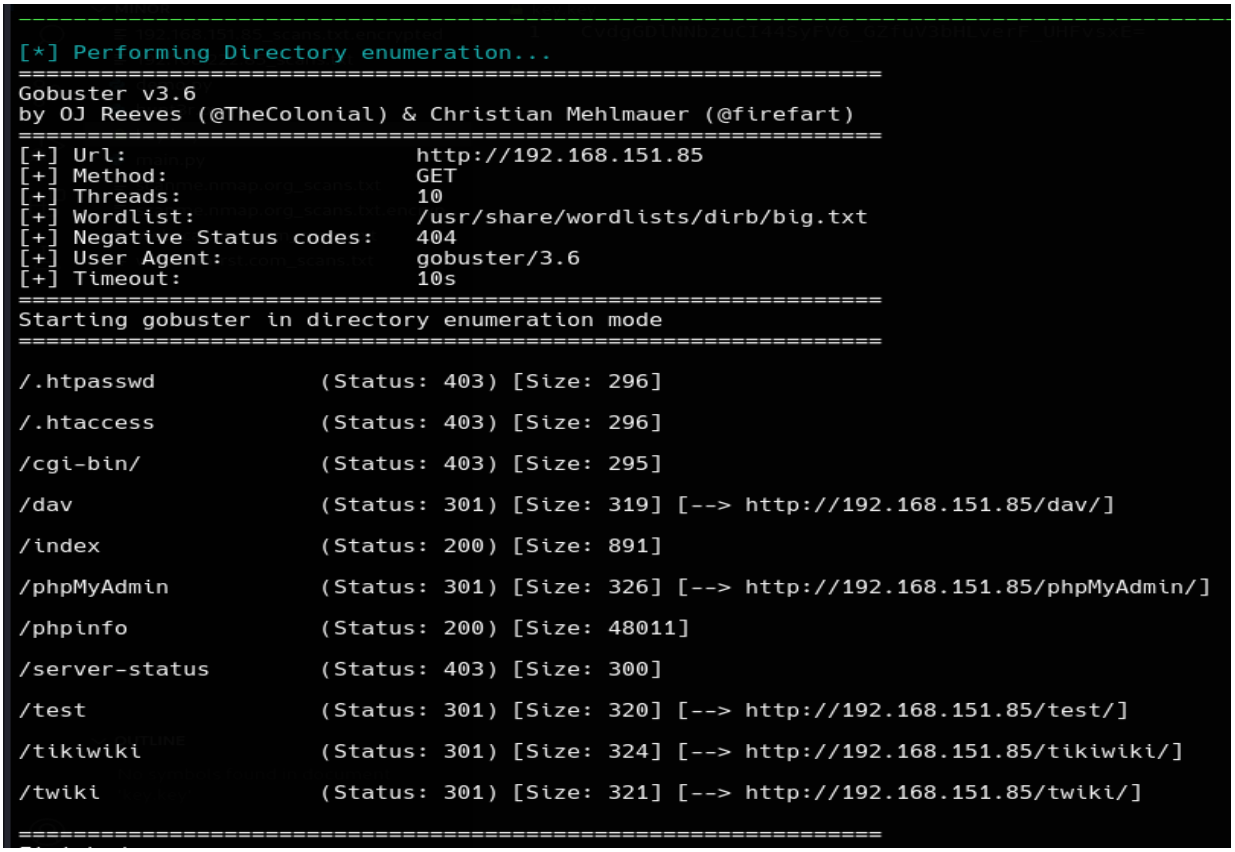
10. Results

- Checking the target state up or down

- Ports, Services ,Versions and their CPE's Found
- Open ports found:
    - 22-ssh
    - 25-smtp
    - 80-http



- Directory Enumeration by finding directories and files on the web application
- uses a wordlist to perform directory enumeration on the target
- By finding the directories, it increases the attack surface against the target
- Found Directories:
    - /.htpasswd
    - /.htaccess
    - /index
    - /phpinfo
    - /test
    - /tikiwiki
    - /phpMyAdmin
    - /server-status



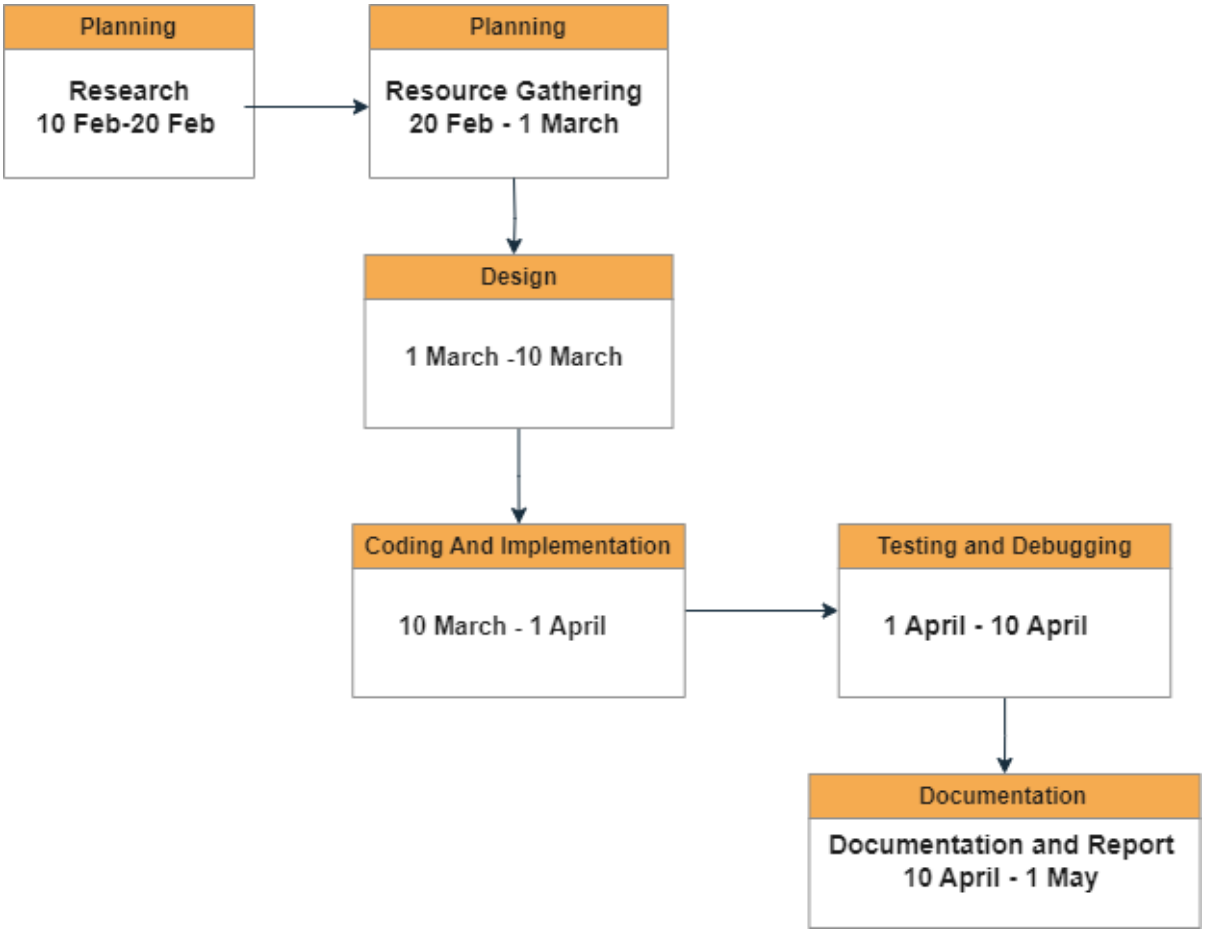
- Finding Related Exploits of all services on the network.
- Using Searchsploit, it directly provides the exploit for the attack

```
For Exploit searching...

 ['vsftp 2.3.4', 'bin 9.4.2', 'http_serve 2.2.8', 'proftp 1.3.1', 'mysq 5.0.51a-3ubuntu5', 'postgresq 8
CPE List:  ['cpe:/a:vsftpd:vsftpd:2.3.4', 'cpe:/o:linux:linux_kernel', 'cpe:/o:linux:linux_kernel', 'cp
, 'cpe:/a:samba:samba', 'cpe:/o:linux:linux_kernel', '', '', '', '', '', 'cpe:/a:proftpd:proftpd:1.3.1'
ealircd', '', 'cpe:/a:apache:coyote_http_connector:1.1']


vsftp 2.3.4
------------------------------------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------------------------------------
vsftpd 2.3.4 - Backdoor Command Execution
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)
------------------------------------------------------------------------------------------------------------
Shellcodes: No Results

bin 9.4.2
------------------------------------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------------------------------------
BIND 9.4.1 < 9.4.2 - Remote DNS Cache Poisoning (Metasploit)
------------------------------------------------------------------------------------------------------------
Shellcodes: No Results

http_serve 2.2.8
Exploits: No Results
Shellcodes: No Results

proftp 1.3.1
Exploits: No Results
Shellcodes: No Results

mysq 5.0.51a-3ubuntu5
Exploits: No Results
Shellcodes: No Results

postgresq 8.3
------------------------------------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------------------------------------
PostgreSQL 8.2/8.3/8.4 - UDF for Command Execution
PostgreSQL 8.3.6 - Conversion Encoding Remote Denial of Service
```

- Performed AES-128 Encryption on the .txt file(report) generated.
- Key generates everytime, save the key in a different file

```
Encoded Report File created as: 192.168.151.85_scans.txt.encrypted


┌──(kali㉿kali)-[~/minor]
└─$ ls
192.168.151.85_scans.txt.encrypted  key.key                              webscantest.com_scans.txt
192.168.222.85_scans.txt            main.py                              webscantrst.com_scans.txt
demo.py                             scanme.nmap.org_scans.txt
hunt3r.py                           scanme.nmap.org_scans.txt.encrypted
```

10. PERT Chart



11. References

[1] G. Yadav, A. Allakany, V. Kumar, K. Paul and K. Okamura, "Penetration Testing Framework for IoT," 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI), Toyama, Japan, 2019, pp. 477-482, doi: 10.1109/IIAI-AAI.2019.00104

[2] "penetration test", Adv. Comput. Sci. an Int. J., vol. 3, no. 6, pp. 107-121, 2014.

[3] Y. Tyagi, S. Bhardwaj, S. Shekhar and A. P, "Efficient Vulnerability Assessment and Penetration Testing: A Framework for Automation," 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), Greater Noida, India, 2023, pp. 553-557, doi: 10.1109/CISES58720.2023.10183397.

[4] Clintswood, D. G. Lie, L. Kuswandana, Nadia, S. Achmad and D. Suhartono, "The Usage of Machine Learning on Penetration Testing Automation," 2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), Yogyakarta, Indonesia, 2023, pp. 322-326, doi: 10.1109/ICE3IS59323.2023.10335188.