**Hacking Web Applications**

**Introduction to Web Applications:**

**Web Application:**

A web application is a software program that runs on a web server rather than being installed on a user's device. It is accessed through a browser and performs various tasks such as data processing, user interaction, and content management, offering dynamic features and functionalities online.

**How Web Application Works:**

Web applications function by sending requests from the user's browser to a web server. The server processes the request, communicates with databases or external APIs, and returns the requested data in HTML, CSS, or JavaScript format. This is then rendered in the browser, allowing users to interact with dynamic content, update data, or perform tasks, creating a seamless and interactive experience without needing to install software locally.

**Web Services:**

A web service is an application or software that is deployed over the Internet and uses standard messaging protocols such as SOAP, UDDI, WSDL, and REST to enable communication between applications developed for different platforms.

**Types of Web Services:**

1. SOAP Web services: It is based on the XML format and is used to transfer data between a service provider and requestor.
2. RESTful Web services: It is based on a set of constraints using underlying HTTP concepts to improve performance.

**OWASP top 10 application security risks:**

The **OWASP Top 10** is a list of the most critical security risks to web applications. Here are the latest 2021 OWASP Top 10 application security risks:

1. **Broken Access Control**  
   Inadequate restrictions on authenticated users, allowing unauthorized actions.
2. **Cryptographic Failures**  
   Weak encryption practices, such as improper storage or transmission of sensitive data.
3. **Injection**  
   Flaws such as SQL, NoSQL, and command injection, where attackers manipulate data queries.
4. **Insecure Design**  
   Risky design decisions leading to vulnerabilities, often overlooked in early stages of development.
5. **Security Misconfiguration**  
   Default settings, incomplete setups, or exposed sensitive information due to misconfigurations.
6. **Vulnerable and Outdated Components**  
   Using outdated libraries or components with known vulnerabilities, leaving the app exposed.
7. **Identification and Authentication Failures**  
   Weak or improper authentication mechanisms, such as broken password recovery processes.
8. **Software and Data Integrity Failures**  
   Insecure software updates or flaws in data integrity, allowing for tampering or malware installation.
9. **Security Logging and Monitoring Failures**  
   Insufficient logging and monitoring, hindering the ability to detect and respond to incidents.
10. **Server-Side Request Forgery (SSRF)**  
    An attacker tricks the server into making requests to internal systems or external resources, bypassing security controls.

These risks highlight common vulnerabilities that developers and organizations must address to secure their applications and protect user data.

**Cryptographic Failures:**

**Cryptographic Failures** occur when sensitive data, such as passwords or financial information, is inadequately protected through weak encryption methods or improper key management. This can include using outdated algorithms, storing unencrypted data, or transmitting sensitive information over insecure channels, making it vulnerable to unauthorized access and attacks.

**Injection flaws**

**Injection flaws** occur when untrusted data is sent to an interpreter (e.g., SQL, NoSQL, or command shell) as part of a command or query. Attackers can manipulate this data to execute unauthorized commands, leading to data leakage, unauthorized access, or system compromise, posing serious security risks.

**Command injection attacks:**

**Command injection attacks** occur when an attacker injects malicious commands into an application that executes system commands. This can happen when user input is improperly validated, allowing the attacker to execute arbitrary commands on the host operating system. It can lead to unauthorized access, data manipulation, or system compromise.

1. Shell injection
2. HTML embedding
3. File injection

**Security misconfiguration:**

**Security misconfiguration attacks** occur when an application, server, or database is improperly configured, leaving it vulnerable to exploitation. This can involve default settings, unnecessary services running, exposed sensitive data, or overly permissive access controls. These misconfigurations can be exploited by attackers to gain unauthorized access or cause damage.

1. Unvalidated inputs
2. Parameter/Form tampering
3. Improper Error handling
4. Insufficient Transport Layer Protection
5. Improper Restriction of XXE

**Vulnerable and Outdated Components:**

**Vulnerable and Outdated Components** refer to using outdated software libraries, frameworks, or components with known security flaws. These outdated elements may not receive security patches, making the application susceptible to attacks. Failure to update or patch these components can expose the app to exploitation and compromise user data.

**Identification and Authentication Failures:**

**Identification and Authentication Failures** occur when an application’s mechanisms for verifying user identities are weak or improperly implemented. This includes issues like weak password policies, improper session handling, or failure to enforce multi-factor authentication. Attackers can exploit these flaws to gain unauthorized access to sensitive systems or user accounts.

1. Session ID in URLs
2. Password Exploitation
3. Timeout Exploitation

**Security Logging and Monitoring Failures:**

**Security Logging and Monitoring Failures** occur when an application lacks proper logging or monitoring of security events. This makes it difficult to detect and respond to attacks or security breaches in real-time. Without sufficient logs, organizations are unable to analyze incidents, identify vulnerabilities, or prevent future attacks effectively.

**Server-Side Request Forgery (SSRF)**:

**Server-Side Request Forgery (SSRF)** occurs when an attacker manipulates a server to make requests to internal or external resources, bypassing security controls. This can lead to unauthorized access to internal systems, data exfiltration, or interaction with unintended resources. SSRF vulnerabilities often arise from improper input validation.



**Unvalidated Redirects and Forwards:**

**Unvalidated Redirects and Forwards** occur when an application redirects or forwards users to a URL based on untrusted input without proper validation. Attackers can exploit this by manipulating the URL, redirecting users to malicious websites or phishing sites, potentially leading to data theft, malware installation, or unauthorized access.

**Watering Hole attack:**

A **Watering Hole attack** is a targeted cyberattack where hackers compromise a legitimate website that is frequently visited by a specific group or organization. The attackers inject malicious code into the site, which then infects visitors' devices, often leading to data theft, system compromise, or further attacks on the organization.

**Cross-site Request Forgery (CSRF) attack:**

A **Cross-site Request Forgery (CSRF)** attack tricks a user into performing unintended actions on a web application where they are authenticated. By exploiting the user's session, the attacker can send malicious requests, such as changing account settings or transferring funds, without the user's consent, potentially causing harm or data loss.

**Cookie/Session Poisoning:**

**Cookie/Session Poisoning** is an attack where an attacker manipulates or alters the session cookies stored on a user's device to impersonate the user or gain unauthorized access. By tampering with session data, such as modifying session identifiers, attackers can hijack sessions, bypass authentication, and access sensitive information.

**Hidden Field Manipulation Attack:**

**Hidden Field Manipulation** is an attack where an attacker modifies hidden form fields in a web page to alter data sent to the server. These hidden fields, typically used for storing sensitive information like user IDs or authentication tokens, can be tampered with to manipulate server-side logic, leading to unauthorized actions or access.

**Clickjacking:**

**Clickjacking** is a malicious technique where an attacker tricks a user into clicking on something different from what they perceive, essentially by overlaying transparent or disguised elements on a webpage. This can lead to unintended actions, such as liking a post, enabling permissions, or making fraudulent transactions, without the user's knowledge.

**Pass the Cookie:**

A **Pass the Cookie** attack occurs when an attacker steals or intercepts a user's session cookie, which is used for authentication, and then uses it to impersonate the user. This allows the attacker to bypass authentication mechanisms and gain unauthorized access to a web application or sensitive user data.

**Foot printing Web Infrastructure: Detecting Web Application Firewalls and proxies on target site**

**Foot printing Web Infrastructure** refers to the process of gathering information about a target website's infrastructure, including servers, technologies, and security mechanisms, to identify potential vulnerabilities. This can involve techniques like DNS querying, scanning for open ports, or examining publicly available data to understand how the web application is structured.

**Detecting Web Application Firewalls (WAFs)** involves identifying protective mechanisms that filter or block malicious web traffic. Techniques include sending known attack patterns to observe if the WAF responds, checking for error messages or altered responses, or using tools that specifically test for the presence of WAFs in a target application.

Tools like WAFW00F can help to do so.

**Detecting Proxies** involves identifying intermediary servers that route traffic between users and the target site. Methods include checking for unusual latency, IP address patterns, or inspecting HTTP headers for signs of proxy servers. Attackers may try to bypass proxies to target the backend server directly.

Tools like HTTP/1.1 TRACE can help to do so.

**Detecting Load Balancers** involves identifying the presence of systems that distribute incoming traffic across multiple servers to balance the load. Techniques for detection include:

1. **Analyzing Response Headers:** Load balancers often add specific headers like "X-Load-Balancer" or "Via," which can be clues.
2. **Behavioral Analysis:** By observing traffic patterns, such as sudden shifts in response time or server behavior, attackers can infer the presence of a load balancer.
3. **IP Address Patterns:** Detecting multiple IP addresses with similar response times may indicate a load balancing setup.

Tools like dig and load balancing detector help to do so.

**Analyse Web Applications:**

Analyse the active application's functionality and technologies to identify exposed attack surfaces

1. Identify Entry Points for User Input: Review the generated HTTP request to identify the user input entry points
2. Identify Server-Side Technologies: Fingerprint the technologies active on the server using various fingerprint techniques such as HTTP fingerprinting
3. Identify Server-Side Functionality: Observe the applications revealed to the client to identify the server-side structure and functionality
4. Identify Files and Directories: Identify misconfigured web applications that expose critical files and directories over the Internet
5. Identify Web Application Vulnerabilities: Identify exploitable vulnerabilities in the underlying web technologies
6. Map the Attack Surface: Identify the various attack surfaces uncovered by the applications and their associated vulnerabilities

**Identify Entry Points for User Input:**

* Examine URL, HTTP Header, query string parameters, POST data, and cookies to determine all user input fields
* Identify HTTP header parameters that can be processed by the application as user inputs such as User-Agent, Referer, Accept, Accept-Language, and Host headers
* Determine URL encoding techniques and other encryption measures implemented for secure web traffic such as SSL

Tools: Burp Suite, OWASP Zed Attack Proxy, WebScarab, httprint

**Identify Server-Side Technologies:**

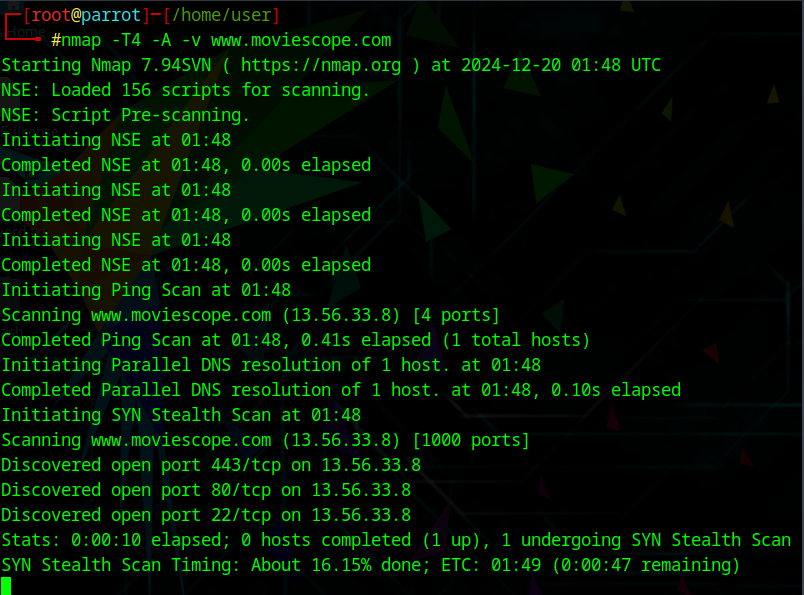
* Perform a detailed server fingerprinting, analyze HTTP headers and HTML source code to identify server-side technologies
* Examine URLs for file extensions, directories, and other identification information
* Examine the error page messages
* Examine session tokens: JSESSIONID - Java, ASPSESSIONID - IIS server, ASP.NET\_Sessionld - ASP.NET, PHPSESSID - PHP
* Use tools such as httprint and WhatWeb to identify server-side technologies

**Identify Web Application Vulnerabilities:**

* Attackers use various techniques to detect vulnerabilities in target web applications hosted on web servers either to gain administrator level access to the server or to retrieve sensitive information stored on the server.
* Comprehensive vulnerability scanning can disclose security flaws associated with executables, binaries, and technologies used in a web application
* Attackers can use tools such as Vega to the vulnerabilities of target web applications

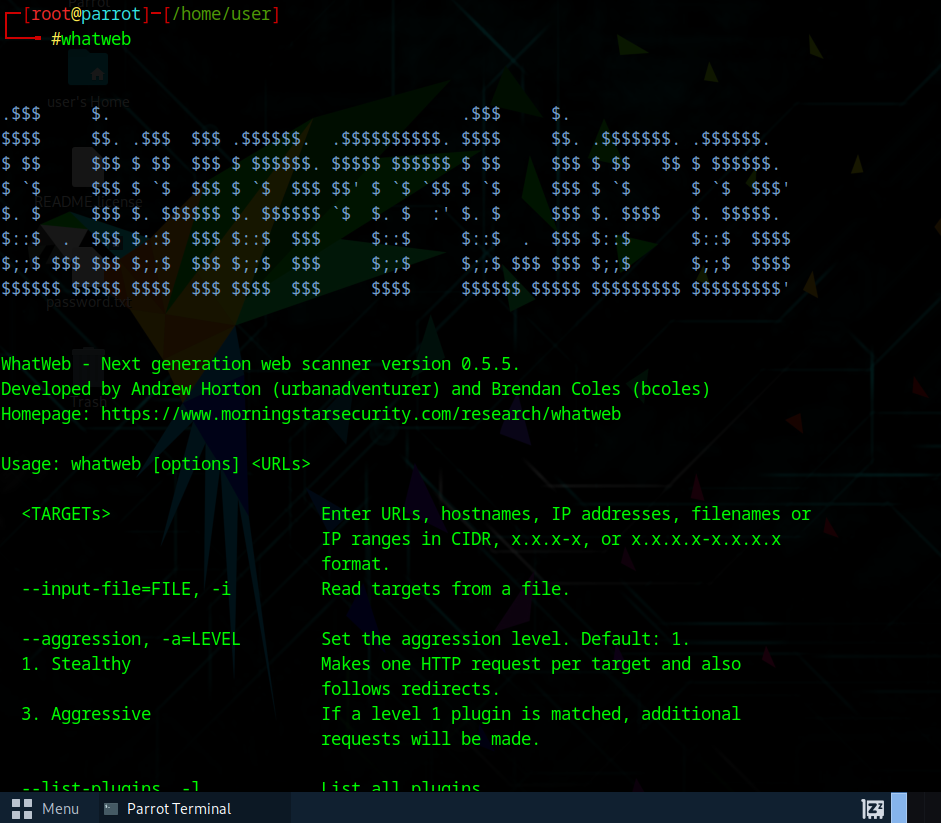
Tools: Vega, Arachni, appsider, uniscan

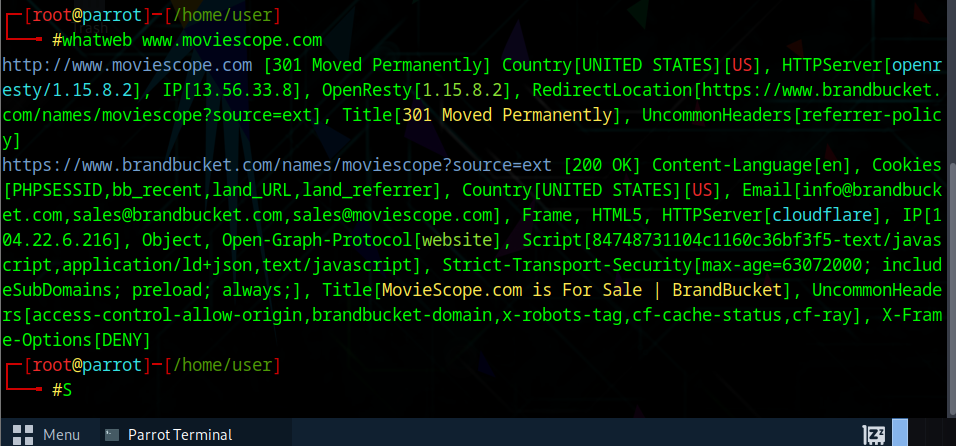
**Perform Web Application Reconnaissance using Nmap and Telnet:**

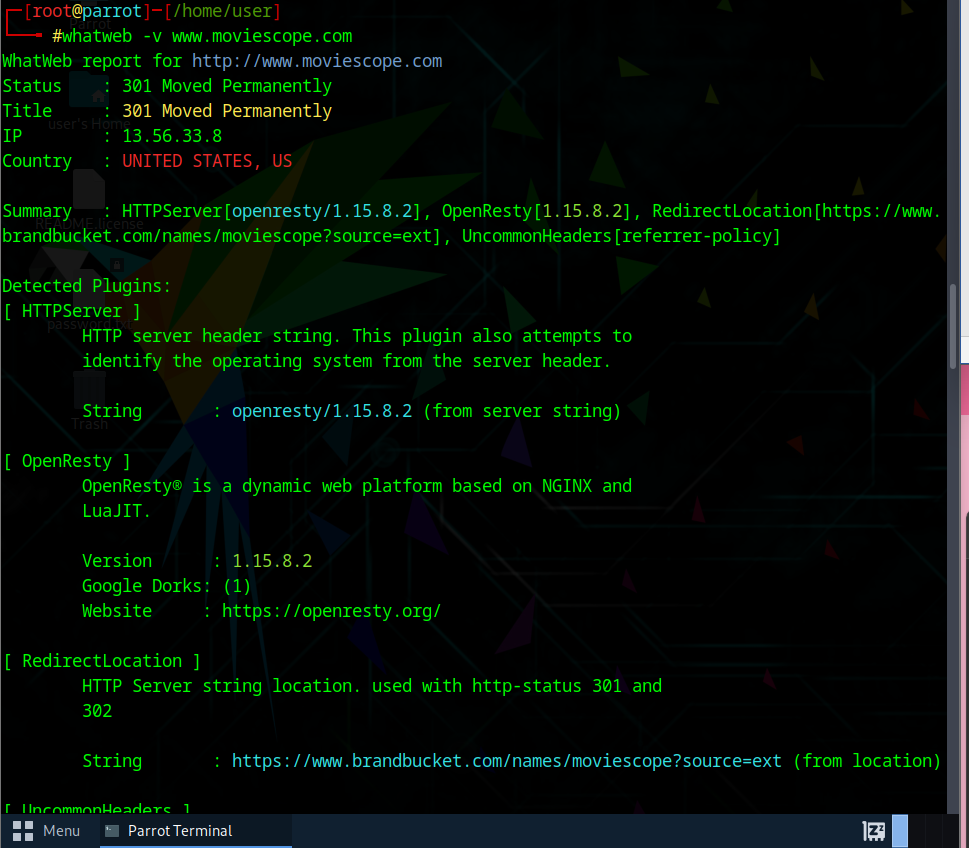
****

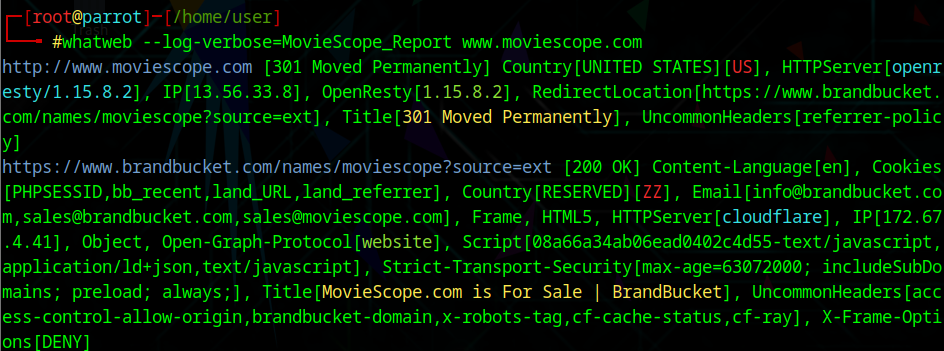
****

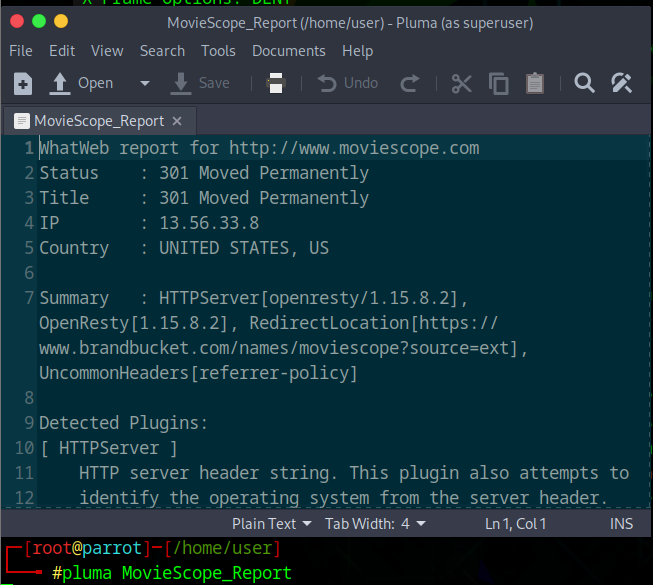
**Perform Web Application Reconnaissance using WhatWeb:**

****

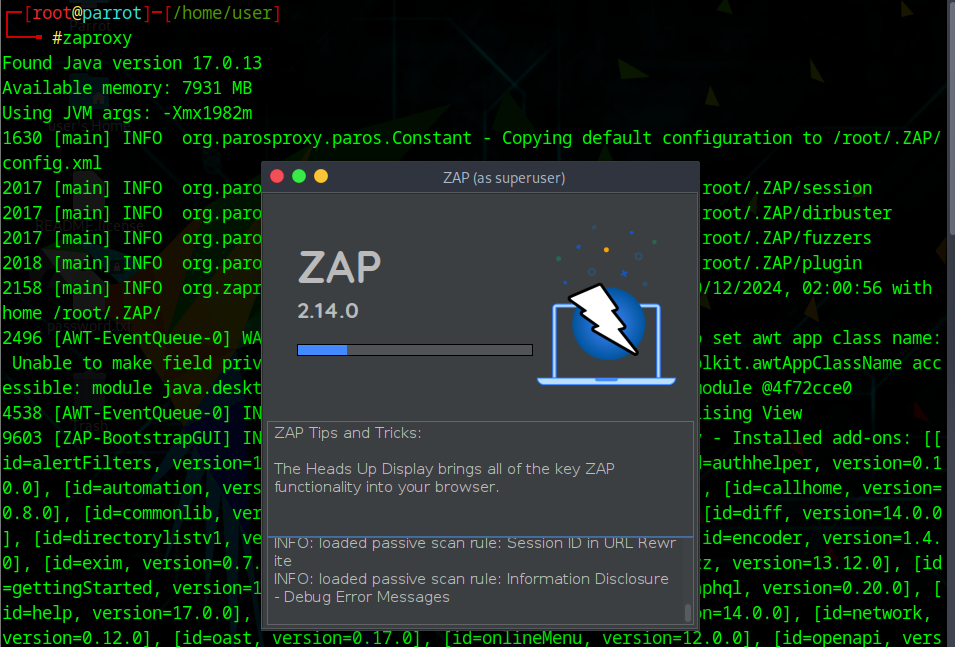
****

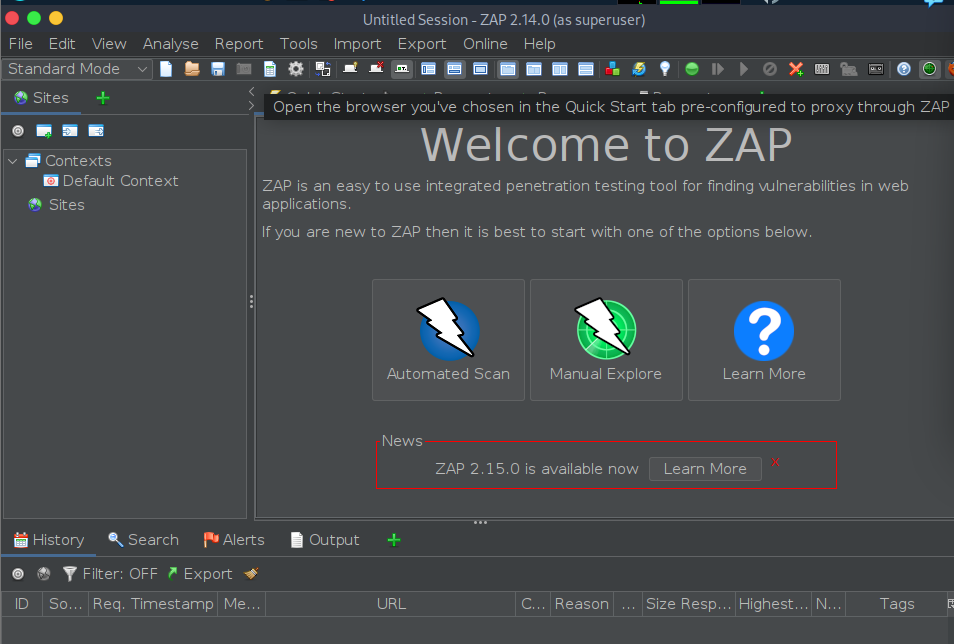
****

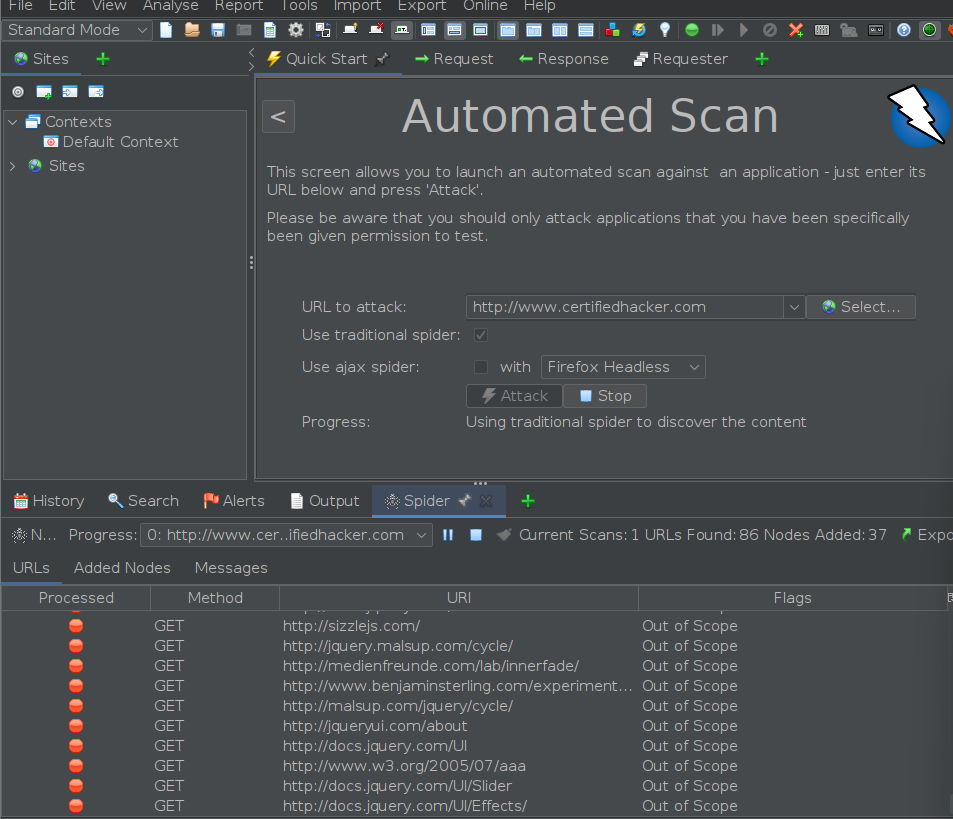
****

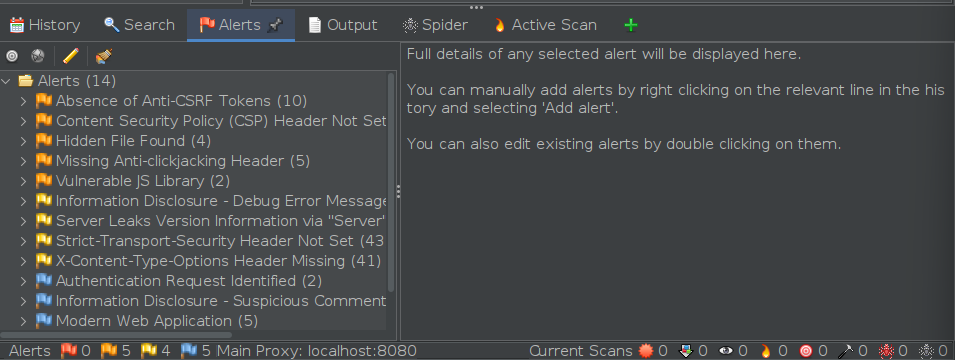
****

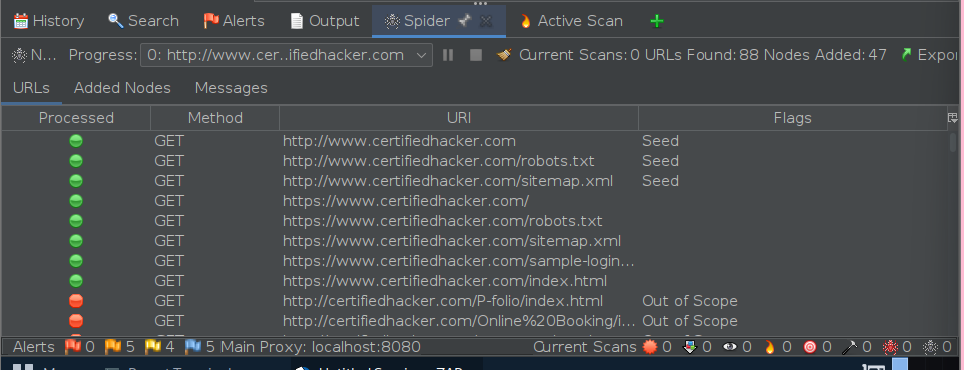
**Perform Web Spidering Using OWASP ZAP:**

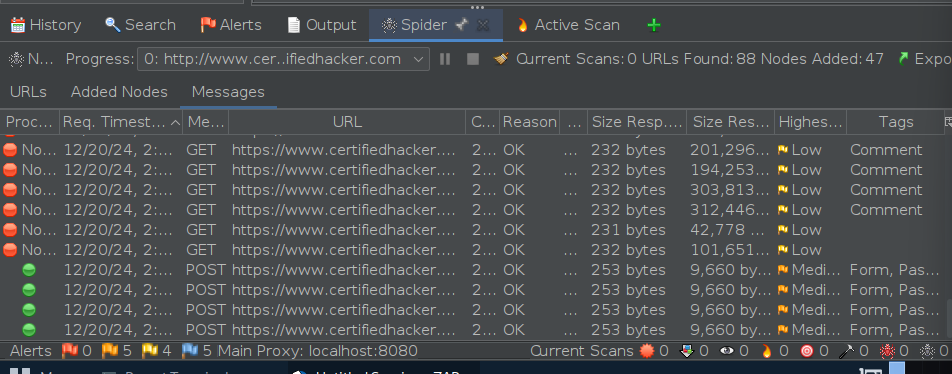
****

****

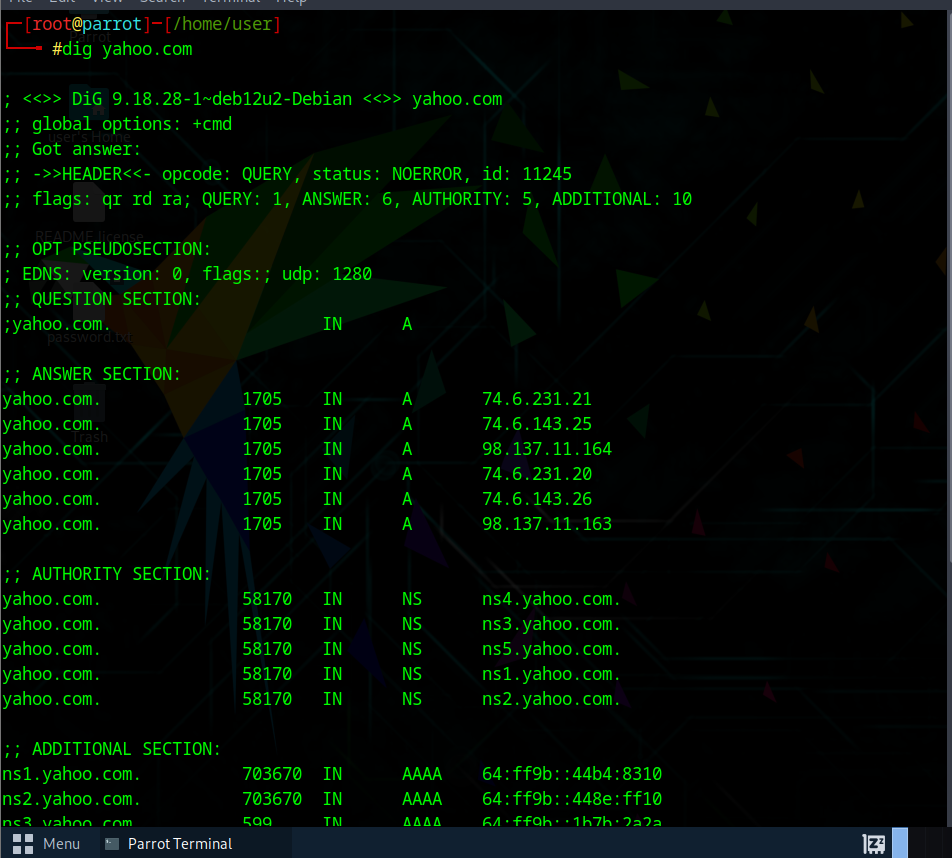
****

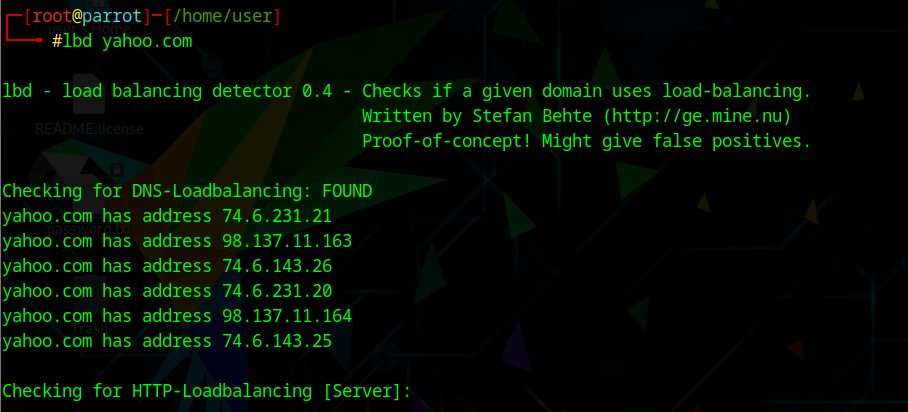
****

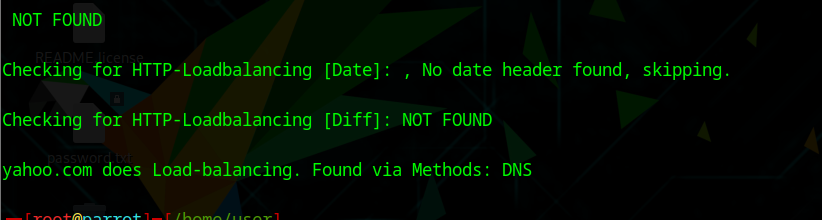
****

****

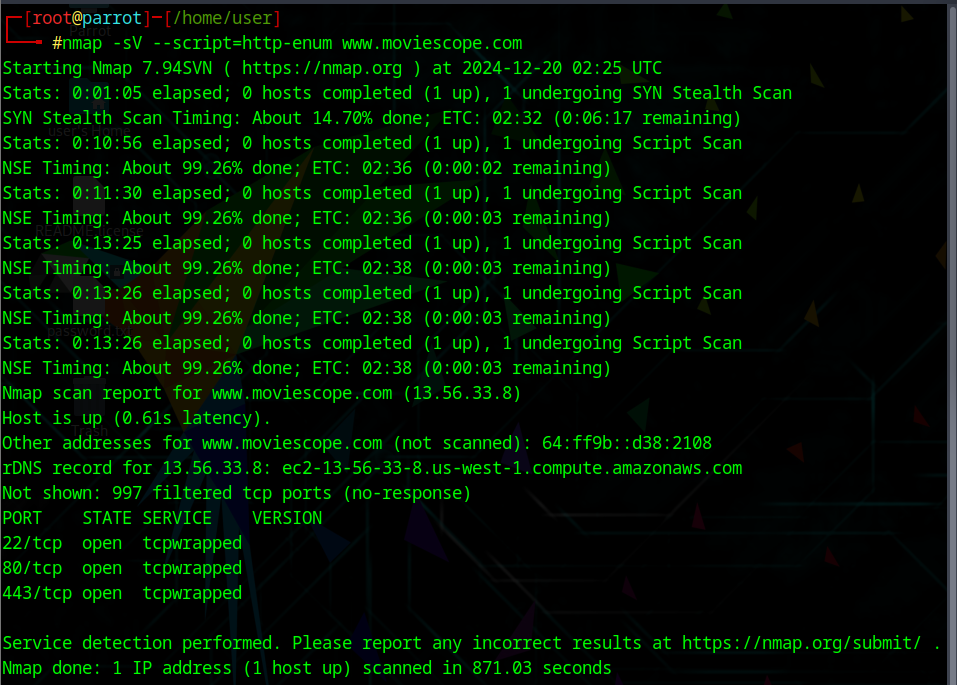
**Detect Load Balancers Using Various Tools:**

****

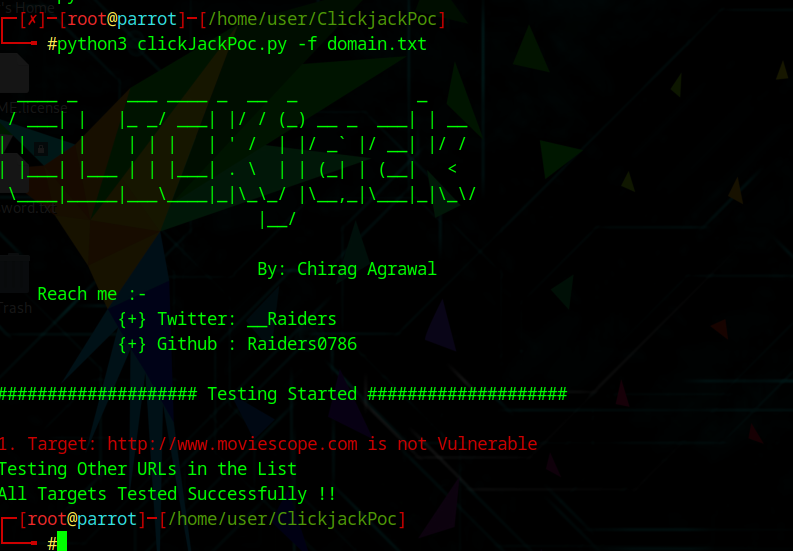
****

****

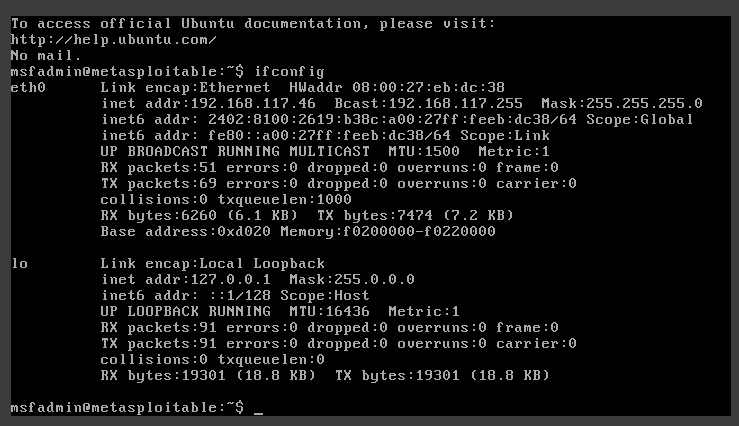
**Identify Web Server Directories using Various Tools:**

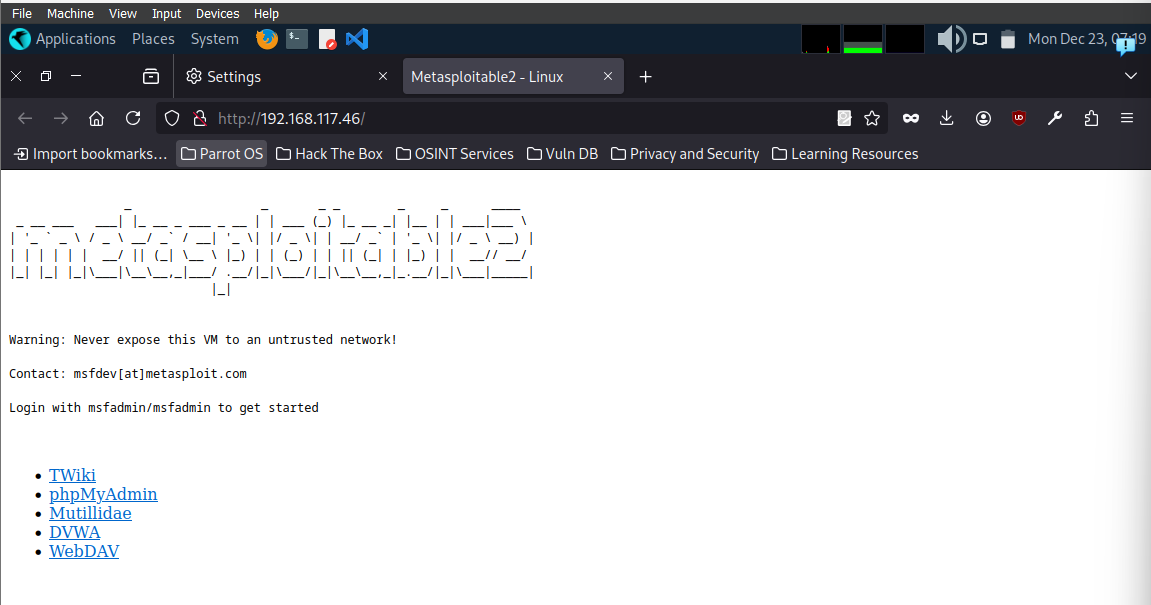
****

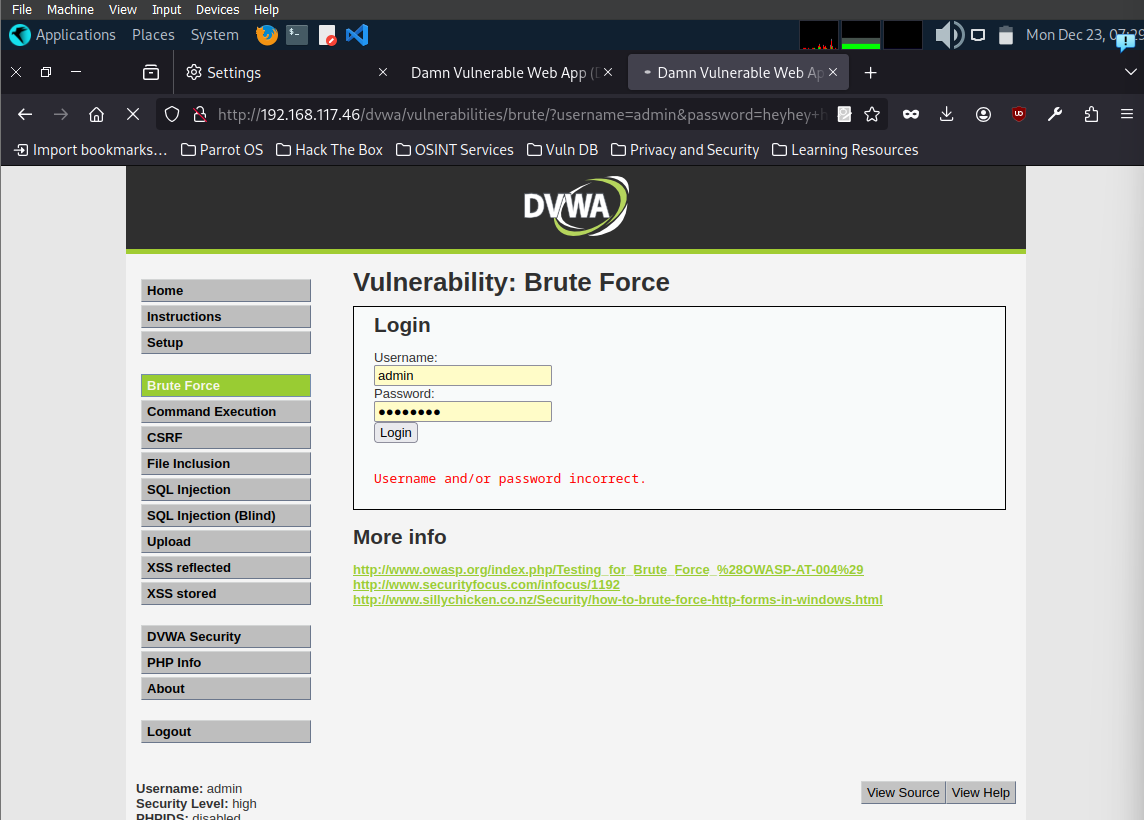
**Identifying Clickjacking Vulnerability using ClickjackPoc:**

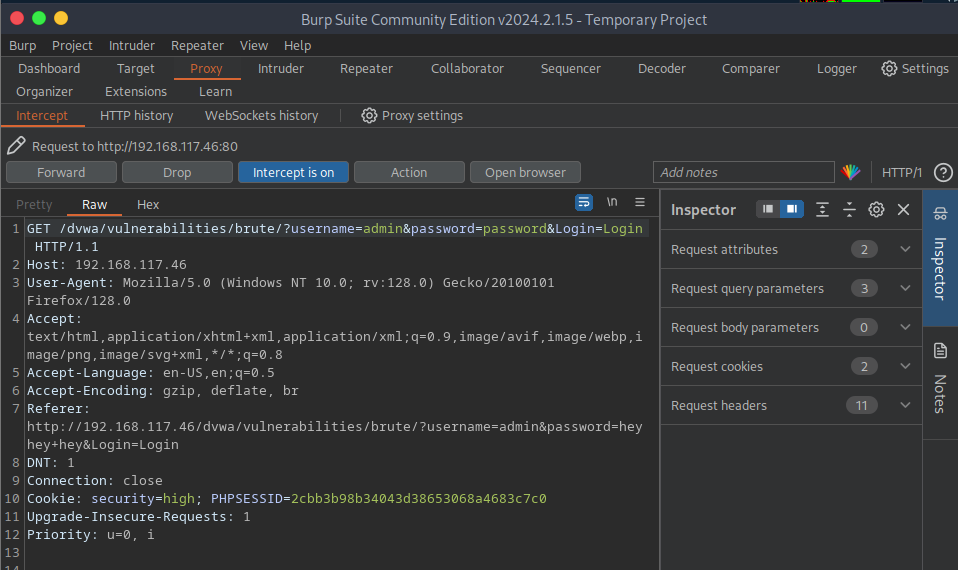
****

**Perform a Brute-force attack using Burp Suite:**

****

****

****

****

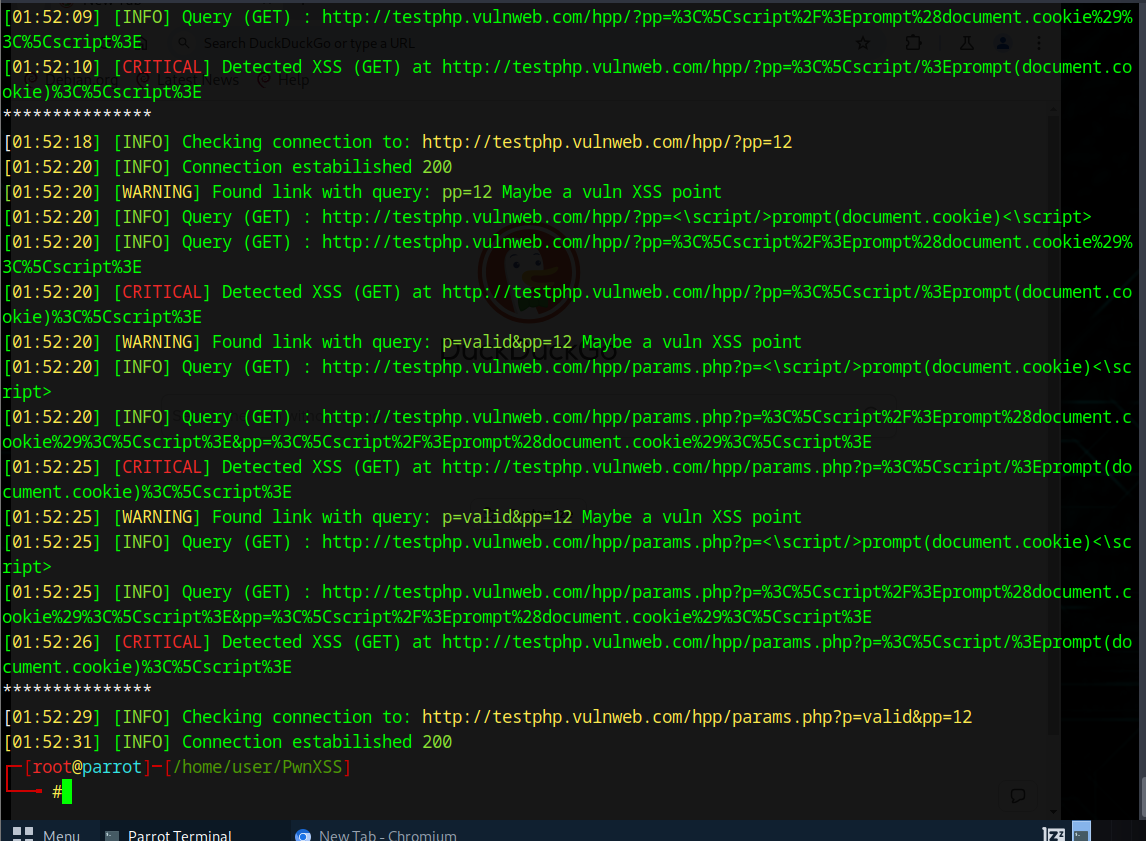
**Perform Parameter Tampering using Burp-Suite:**

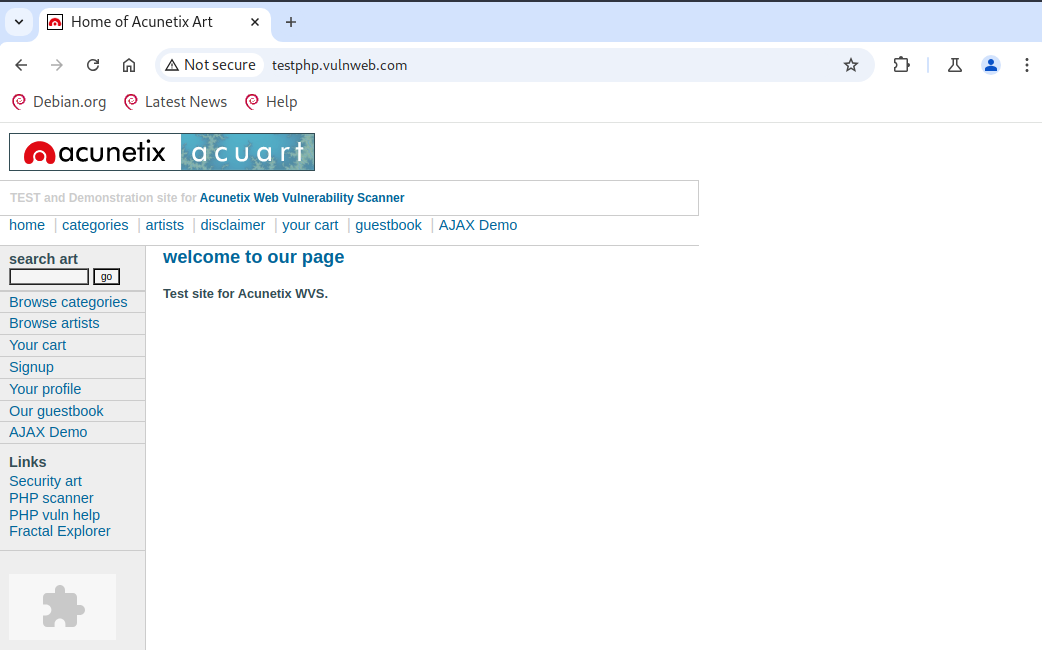
Ahh! I can’t manage to find such websites to test it.

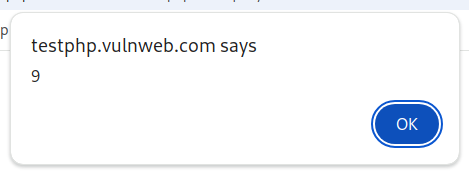
**Identifying XSS vulnerabilities in web applications using PwnXSS:**

****

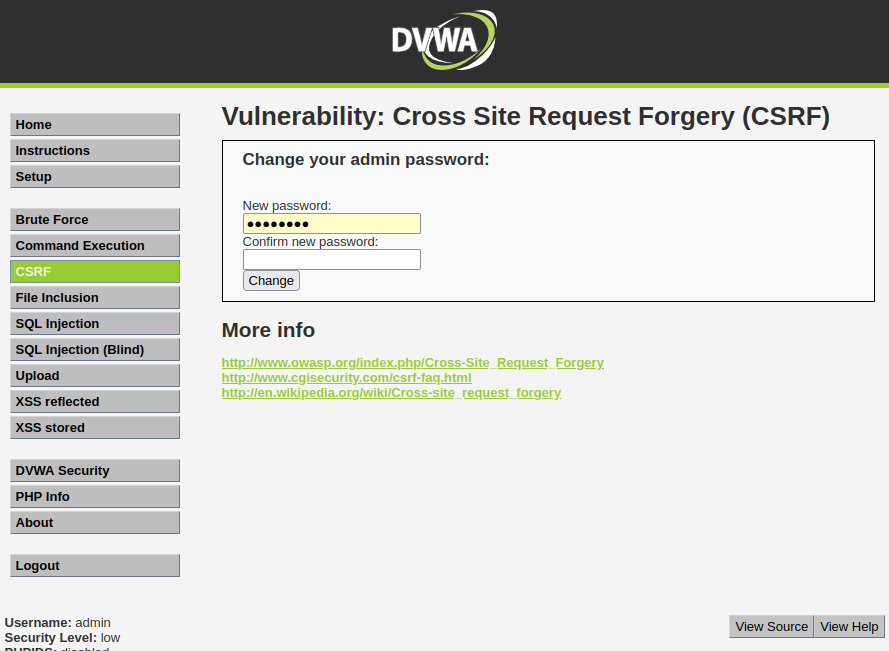
****

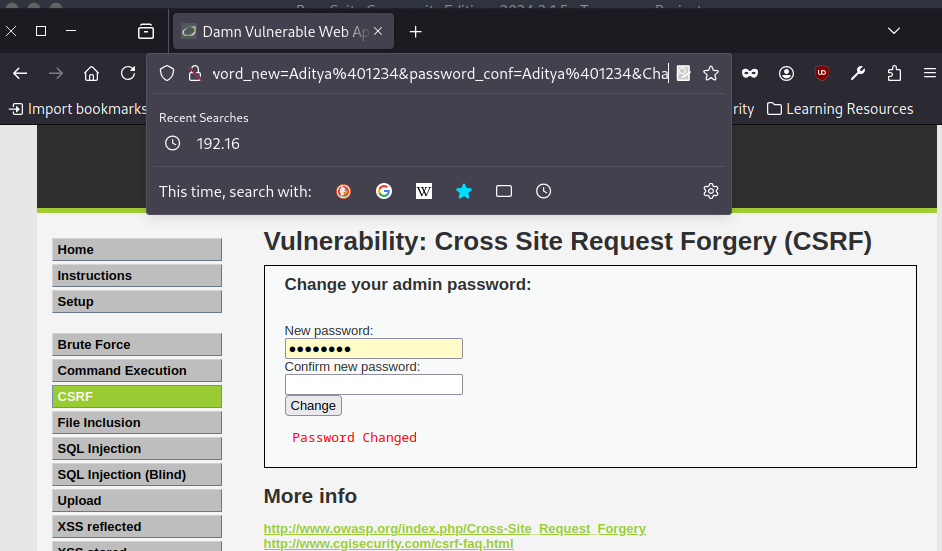
****

****

****

**Perform Cross-site Request Forgery (CSRF) attack:**

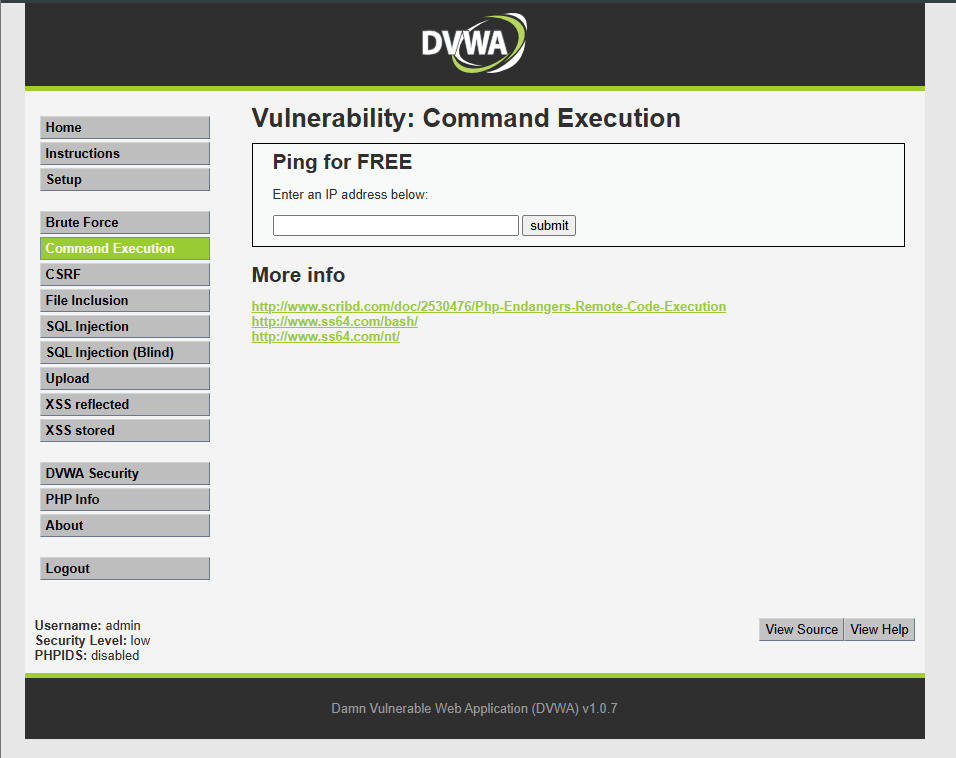
****

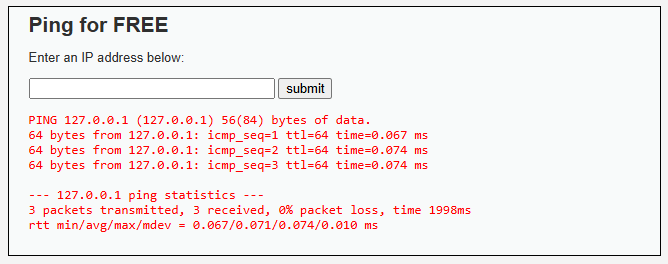
****

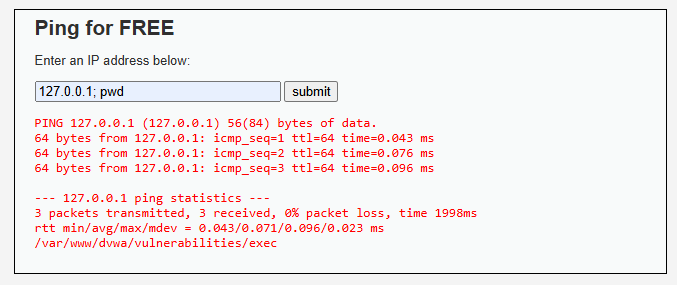
**Enumerate and hack a web application using WPScan and Metasploit:**

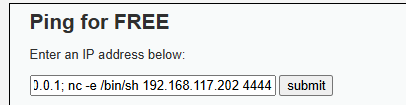
****

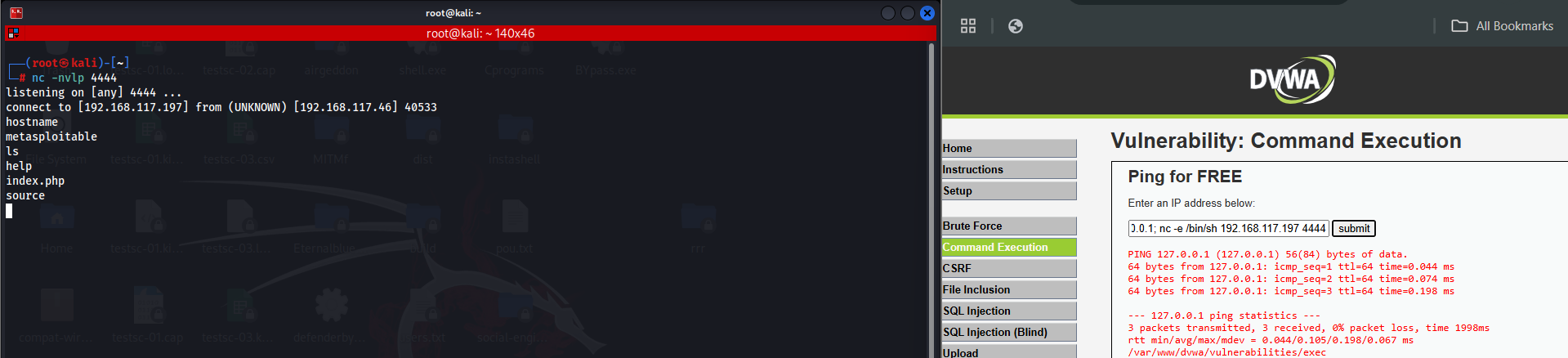
**Exploit a remote command execution vulnerability to compromise a target web server:**

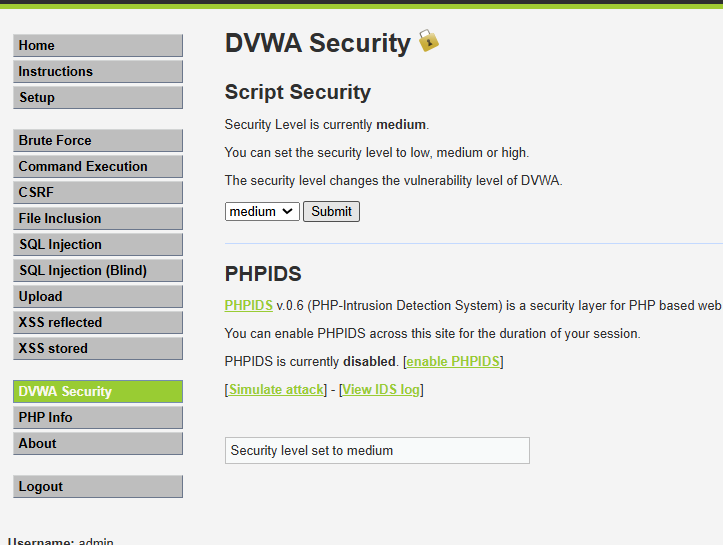
****

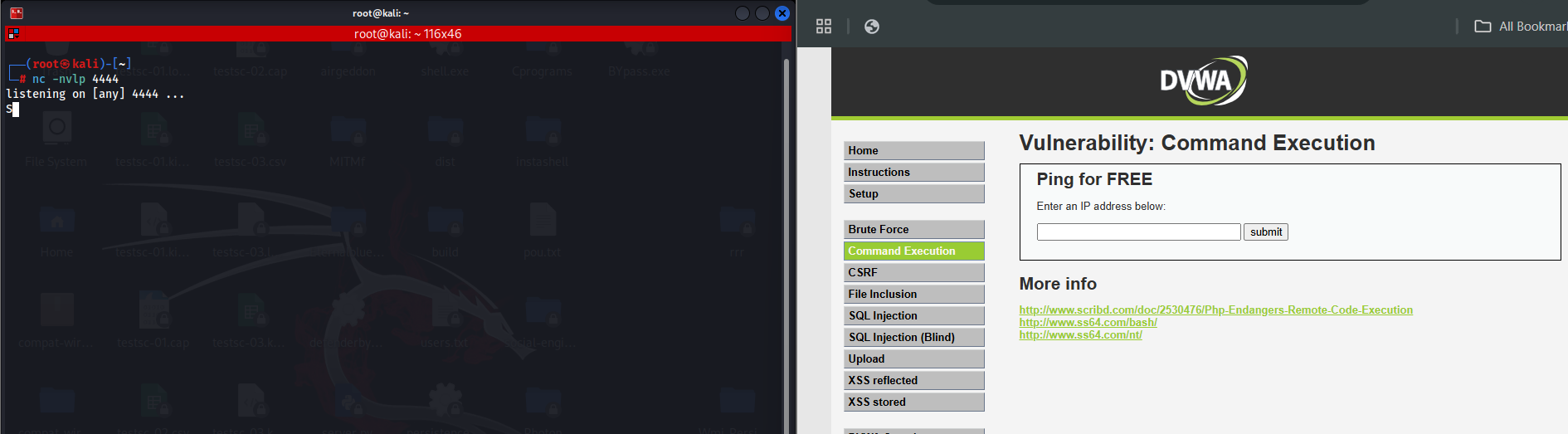
****

****

****

****

****

****