Web Application Security: Understanding Threats and Mitigating File Upload Risks

Report By Blake Major

Contents

See document outline

Penetration testing, often referred to as “pen testing,” is a crucial aspect of cybersecurity. It involves simulating cyberattacks on a computer system, network, or web application to identify vulnerabilities that could be exploited by malicious actors.

What is Penetration Testing?

Penetration testing is an authorised, simulated attack on a system to evaluate its security. The goal is to find and exploit vulnerabilities in a controlled manner, allowing organisations to understand and fix these weaknesses before they can be exploited by real attackers.

Types of Penetration Testing

1. **Network Penetration Testing**: Focuses on identifying vulnerabilities in network infrastructure, such as firewalls, routers, and switches.
2. **Web Application Penetration Testing**: Targets web applications to find vulnerabilities like SQL injection, cross-site scripting (XSS), and CSRF.
3. **Mobile Application Penetration Testing**: Examines mobile apps for security flaws.
4. **Wireless Penetration Testing**: Assesses the security of wireless networks and devices.
5. **Social Engineering**: Tests the human element by attempting to trick employees into revealing sensitive information.
6. **Physical Penetration Testing**: Involves attempting to gain physical access to facilities and systems.

Phases of Penetration Testing

Planning and Reconnaissance Phase

The first phase of penetration testing is **Reconnaissance**, also known as information gathering or footprinting. This phase is crucial as it lays the groundwork for the entire penetration test.

Objective

The primary goal of the reconnaissance phase is to gather as much information as possible about the target system, network, or application. This information helps penetration testers understand the target's structure, potential vulnerabilities, and how to plan their attack strategy effectively.

Types of Reconnaissance

1. **Passive Reconnaissance**: This involves gathering information without directly interacting with the target. It relies on publicly available data and does not alert the target to the tester's activities.
    - **Examples**:
        - **WHOIS Lookups**: To find domain registration details.
        - **DNS Queries**: To gather information about domain names and IP addresses.
        - **Social Media**: To collect information about employees and organisational structure.
        - **Public Websites**: To find details about the target's infrastructure and technologies used.
2. **Active Reconnaissance**: This involves directly interacting with the target to gather information. It is more intrusive and has a higher risk of detection.
    - **Examples**:
        - **Port Scanning**: To identify open ports and services running on the target.
        - **Network Mapping**: To understand the network topology.
        - **Service Enumeration**: To gather details about the services and their versions.

Techniques and Tools

- **Google Dorking**: Using advanced search operators to find sensitive information exposed on the web.
- **Shodan**: A search engine for Internet-connected devices, useful for finding vulnerable devices.

- **Nmap**: A network scanning tool to discover hosts and services on a network.
- **Maltego**: A tool for gathering and visualising information about the target.
- **Recon-ng**: A web reconnaissance framework with various modules for data collection.

Information Collected

- **Domain Names and IP Addresses**: Basic information about the target's online presence.
- **Network Infrastructure**: Details about routers, switches, firewalls, and other network devices.
- **Operating Systems and Applications**: Information about the software and versions in use.
- **Employee Information**: Names, email addresses, job titles, and other personal details.
- **Security Policies**: Insights into the target's security measures and protocols.

Importance

- **Planning**: Helps in creating a detailed attack plan by understanding the target's environment.
- **Identifying Weaknesses**: Reveals potential entry points and vulnerabilities.
- **Minimising Detection**: Passive reconnaissance helps in gathering information without alerting the target.

Example Scenario

Imagine a penetration tester is tasked with testing the security of a company's web application. During the reconnaissance phase, they might:

1. **Perform a WHOIS lookup** to find the domain registration details.
2. **Use Google Dorking** to find any exposed sensitive files or directories.
3. **Scan the network** with Nmap to identify open ports and running services.
4. **Check social media** for information about employees who might have access to the web application.

By the end of the reconnaissance phase, the tester has a comprehensive understanding of the target's environment, which helps in planning the subsequent phases of the penetration test.

Scanning Phase

The second phase of penetration testing is **Scanning**. This phase involves actively probing the target system to identify open ports, services, and potential vulnerabilities.

Objective

The main goal of the scanning phase is to gather detailed information about the target's network and systems. This helps in identifying potential entry points and understanding the target's security posture.

Types of Scanning

1. **Network Scanning**: Identifies active devices on the network and their IP addresses.
2. **Port Scanning**: Determines which ports are open on the target devices and what services are running on those ports.
3. **Vulnerability Scanning**: Detects known vulnerabilities in the target systems and applications.

Techniques and Tools

1. **Ping Sweeps**: Used to discover live hosts on a network by sending ICMP echo requests.
    - **Tool Example**: Nmap
2. **Port Scanning**: Identifies open ports and the services running on them.
    - **Tool Example**: Nmap, Masscan
3. **Service Enumeration**: Gathers information about the services running on open ports, including version numbers and configurations.
    - **Tool Example**: Nmap, Netcat
4. **Vulnerability Scanning**: Uses automated tools to identify known vulnerabilities in the target systems.
    - **Tool Example**: Nessus, OpenVAS

Steps in the Scanning Phase

1. **Network Scanning**:
    - **Objective**: Identify all active devices on the network.
    - **Method**: Use tools like Nmap to perform a ping sweep and identify live hosts.
    - **Output**: A list of active IP addresses.
2. **Port Scanning**:
    - **Objective**: Identify open ports on the active devices.

    - **Method**: Use tools like Nmap to scan for open ports on the identified IP addresses.
    - **Output**: A list of open ports and the services running on them.
3. **Service Enumeration**:
    - **Objective**: Gather detailed information about the services running on open ports.
    - **Method**: Use tools like Nmap with service detection scripts to identify service versions and configurations.
    - **Output**: Detailed information about the services, including version numbers and potential vulnerabilities.
4. **Vulnerability Scanning**:
    - **Objective**: Identify known vulnerabilities in the target systems.
    - **Method**: Use automated vulnerability scanners like Nessus to scan the identified services for known vulnerabilities.
    - **Output**: A list of vulnerabilities, including severity levels and potential impacts.

Importance

- **Identifying Entry Points**: Helps in finding potential entry points for further exploitation.
- **Understanding Security Posture**: Provides a detailed view of the target's security measures and configurations.
- **Prioritising Vulnerabilities**: Helps in prioritising vulnerabilities based on their severity and potential impact.

Example Scenario

Imagine a penetration tester is tasked with testing the security of a company's network. During the scanning phase, they might:

1. **Perform a network scan** using Nmap to identify all active devices on the network.
2. **Conduct a port scan** on the identified devices to find open ports and running services.
3. **Enumerate services** to gather detailed information about the services, including version numbers.
4. **Run a vulnerability scan** using Nessus to identify known vulnerabilities in the services.

By the end of the scanning phase, the tester has a comprehensive understanding of the target's network, open ports, running services, and potential vulnerabilities, which helps in planning the next phase of the penetration test.

Exploitation Phase

The **Exploitation Phase** in penetration testing is where the actual attacks are carried out to exploit identified vulnerabilities. This phase is critical as it demonstrates the potential impact of the vulnerabilities on the target system.

Objective

The primary goal of the exploitation phase is to gain unauthorised access to the target system by exploiting the vulnerabilities identified during the scanning phase. This phase aims to demonstrate the potential damage that could be caused by an attacker.

Techniques and Methods

1. **Buffer Overflows**: Exploiting software bugs that allow attackers to execute arbitrary code by overflowing the buffer memory.
2. **SQL Injection**: Injecting malicious SQL queries to manipulate the database and gain unauthorised access.
3. **Cross-Site Scripting (XSS)**: Injecting malicious scripts into web pages viewed by other users to steal information or perform actions on their behalf.
4. **Phishing**: Sending deceptive emails to trick users into revealing credentials or downloading malware.
5. **Password Attacks**: Using techniques like brute force, dictionary attacks, or credential stuffing to crack passwords.
6. **Exploiting Misconfigurations**: Taking advantage of improperly configured systems, such as default passwords or open ports.
7. **Social Engineering**: Manipulating individuals into divulging confidential information or performing actions that compromise security.

Tools Used

- **Metasploit**: A powerful exploitation framework that provides various tools for exploiting vulnerabilities.
- **Burp Suite**: A web vulnerability scanner and exploitation tool.
- **SQLmap**: An automated tool for SQL injection and database takeover.

- **John the Ripper**: A password cracking tool.
- **Hydra**: A tool for performing brute force attacks on various protocols.

Steps in the Exploitation Phase

1. **Identify Exploitable Vulnerabilities**: Review the results from the scanning phase to identify vulnerabilities that can be exploited.
2. **Select Exploitation Tools**: Choose appropriate tools and techniques based on the identified vulnerabilities.
3. **Execute Exploits**: Use the selected tools to exploit the vulnerabilities and gain access to the target system.
4. **Establish a Foothold**: Once access is gained, establish a persistent presence on the system (e.g., by installing backdoors or creating new user accounts).
5. **Escalate Privileges**: Attempt to gain higher-level access to increase control over the system.
6. **Pivot and Move Laterally**: Use the compromised system to target other systems within the network.

Importance

- **Demonstrating Impact**: Shows the potential damage that could be caused by an attacker exploiting the vulnerabilities.
- **Identifying Weaknesses**: Helps in identifying weaknesses in the system's defences and access controls.
- **Improving Security Posture**: Provides insights into how to strengthen security measures and prevent similar attacks.

Example Scenario

Imagine a penetration tester is tasked with testing the security of a company's web application. During the exploitation phase, they might:

1. **Identify a SQL injection vulnerability** in the login form.
2. **Use SQLmap** to exploit the vulnerability and gain access to the database.
3. **Extract user credentials** from the database.
4. **Use the credentials** to log in as an administrator.
5. **Install a web shell** to maintain access to the web server.
6. **Pivot to other systems** within the network using the compromised server.

By the end of the exploitation phase, the tester has demonstrated how an attacker could exploit vulnerabilities to gain unauthorised access and control over the target system.

Privilege Escalation Phase

The **Privilege Escalation Phase** in penetration testing is crucial for understanding how an attacker can gain higher levels of access within a system.

Objective

The main goal of the privilege escalation phase is to gain higher-level access than initially obtained. This involves exploiting vulnerabilities to elevate privileges from a lower level (e.g., a regular user) to a higher level (e.g., an administrator or root user). This phase helps in understanding the potential impact of an attacker gaining elevated privileges.

Types of Privilege Escalation

1. **Vertical Privilege Escalation**: This occurs when an attacker gains higher privileges than they currently have. For example, a regular user exploiting a vulnerability to gain administrative access.
2. **Horizontal Privilege Escalation**: This occurs when an attacker gains access to the same level of privileges but for a different user. For example, a user accessing another user's account with similar privileges.

Techniques and Methods

1. **Exploiting Software Vulnerabilities**: Using known vulnerabilities in software to gain higher privileges.
    - **Buffer Overflows**: Exploiting buffer overflow vulnerabilities to execute arbitrary code with elevated privileges.
    - **Kernel Exploits**: Exploiting vulnerabilities in the operating system kernel to gain root access.
2. **Misconfigurations**: Taking advantage of improperly configured systems.
    - **Weak File Permissions**: Accessing files or directories with weak permissions to gain sensitive information or execute privileged commands.
    - **Default Credentials**: Using default usernames and passwords that have not been changed.
3. **Password Cracking**: Cracking passwords to gain access to higher-privileged accounts.
    - **Brute Force Attacks**: Trying all possible combinations to crack a password.

    - **Dictionary Attacks**: Using a list of common passwords to attempt to gain access.
4. **Social Engineering**: Manipulating individuals to gain higher privileges.
    - **Phishing**: Sending deceptive emails to trick users into revealing their credentials.
    - **Impersonation**: Pretending to be a legitimate user to gain access.
5. **Exploiting Trust Relationships**: Taking advantage of trust relationships between systems.
    - **Pass-the-Hash**: Using hashed credentials to authenticate without knowing the actual password.
    - **Token Impersonation**: Using stolen tokens to impersonate higher-privileged users.

Tools Used

- **Metasploit**: A powerful exploitation framework that provides various tools for privilege escalation.
- **PowerSploit**: A collection of PowerShell scripts for post-exploitation tasks, including privilege escalation.
- **Mimikatz**: A tool for extracting plaintext passwords, hashes, PIN codes, and Kerberos tickets from memory.
- **John the Ripper**: A password cracking tool.
- **LinPEAS/WinPEAS**: Tools for enumerating possible privilege escalation paths on Linux and Windows systems.

Steps in the Privilege Escalation Phase

1. **Identify Potential Escalation Paths**: Review the results from the gaining access phase to identify potential paths for privilege escalation.
2. **Select Exploitation Techniques**: Choose appropriate techniques and tools based on the identified paths.
3. **Execute Exploits**: Use the selected techniques to exploit vulnerabilities and gain higher privileges.
4. **Verify Elevated Access**: Confirm that the privileges have been successfully escalated.
5. **Establish Persistence**: Ensure that the elevated access can be maintained over time (e.g., by creating new user accounts or installing backdoors).

Importance

- **Understanding Impact**: Demonstrates the potential damage that could be caused by an attacker gaining elevated privileges.
- **Identifying Weaknesses**: Helps in identifying weaknesses in the system's access controls and configurations.
- **Improving Security Posture**: Provides insights into how to strengthen security measures and prevent similar attacks.

Example Scenario

Imagine a penetration tester has gained initial access to a company's network as a regular user. During the privilege escalation phase, they might:

1. **Identify a misconfigured service** running with administrative privileges.
2. **Use Metasploit** to exploit a vulnerability in the service and gain administrative access.
3. **Verify elevated access** by executing commands that require administrative privileges.
4. **Install a persistent backdoor** to maintain administrative access.
5. **Use the elevated privileges** to access sensitive data and further compromise the network.

By the end of the privilege escalation phase, the tester has demonstrated how an attacker could exploit vulnerabilities to gain higher-level access and control over the target system.

Reporting Phase

The **Reporting Phase** is the final and one of the most critical phases in penetration testing. It involves documenting the findings, providing detailed analysis, and offering recommendations for remediation.

Objective

The primary goal of the reporting phase is to communicate the results of the penetration test to stakeholders in a clear, comprehensive, and actionable manner. The report should provide a detailed account of the vulnerabilities discovered, the methods used to exploit them, and the potential impact on the organisation. It should also include recommendations for mitigating the identified risks.

Key Components of a Penetration Testing Report

1. **Executive Summary**: A high-level overview of the findings, suitable for non-technical stakeholders such as executives and managers.
    - **Purpose**: Summarise the scope, objectives, and key findings of the penetration test.
    - **Content**: Brief description of the most critical vulnerabilities, overall security posture, and business impact.
2. **Methodology**: A detailed description of the testing approach and techniques used during the penetration test.
    - **Purpose**: Provide transparency and ensure that the testing process is reproducible.
    - **Content**: Information about the reconnaissance, scanning, exploitation, and post-exploitation phases, including tools and techniques used.
3. **Detailed Findings**: Comprehensive documentation of each vulnerability discovered during the test.
    - **Purpose**: Provide technical details for IT and security teams to understand and address the vulnerabilities.
    - **Content**:
        - **Vulnerability Description**: Explanation of the vulnerability and how it was discovered.
        - **Impact Analysis**: Assessment of the potential impact on the organisation if the vulnerability were to be exploited.
        - **Proof of Concept**: Evidence of exploitation, such as screenshots, logs, or code snippets.
        - **Risk Rating**: Severity of the vulnerability, often categorised as low, medium, high, or critical.
4. **Recommendations**: Actionable steps to remediate the identified vulnerabilities and improve the overall security posture.
    - **Purpose**: Provide clear guidance on how to fix the vulnerabilities and prevent future occurrences.
    - **Content**: Specific recommendations for each vulnerability, including configuration changes, software updates, and best practices.
5. **Conclusion**: A summary of the overall findings and the effectiveness of the current security measures.
    - **Purpose**: Reinforce the key points and provide a final assessment of the organisation's security posture.
    - **Content**: Recap of the most critical vulnerabilities, overall risk level, and suggested next steps.
6. **Appendices**: Additional information that supports the findings and recommendations.

    - **Purpose**: Provide supplementary details that may be useful for further analysis.
    - **Content**: Raw data from scans, detailed logs, tool outputs, and references to relevant documentation.

Importance

- **Communication**: Ensures that stakeholders understand the findings and their implications.
- **Actionable Insights**: Provides clear and practical recommendations for improving security.
- **Compliance**: Helps organisations meet regulatory requirements and industry standards.
- **Documentation**: Serves as a record of the penetration test and can be used for future reference and audits.

Example Scenario

Imagine a penetration tester has completed a test on a company's web application. The reporting phase might include:

1. **Executive Summary**: Highlighting critical vulnerabilities such as SQL injection and XSS, and their potential impact on customer data.
2. **Methodology**: Describing the tools and techniques used, such as Nmap for scanning and Metasploit for exploitation.
3. **Detailed Findings**: Documenting each vulnerability with descriptions, impact analysis, proof of concept, and risk ratings.
4. **Recommendations**: Providing specific steps to fix the vulnerabilities, such as updating software, implementing input validation, and enhancing access controls.
5. **Conclusion**: Summarising the overall security posture and suggesting regular security assessments.
6. **Appendices**: Including raw scan data, detailed logs, and references to security best practices.

By the end of the reporting phase, the organisation has a clear understanding of its security weaknesses and a roadmap for remediation.

Benefits of Penetration Testing

- **Identifies Vulnerabilities**: Helps in discovering security weaknesses before attackers do.
- **Improves Security Posture**: Provides insights into the effectiveness of existing security measures.
- **Compliance**: Many regulations and standards require regular penetration testing.
- **Risk Management**: Helps in understanding the potential impact of vulnerabilities on business operations.

Tools Used in Penetration Testing

- **Nmap**: Network scanning tool.
- **Metasploit**: Exploitation framework.
- **Burp Suite**: Web vulnerability scanner.
- **Wireshark**: Network protocol analyzer.
- **Nessus**: Vulnerability scanner.

Challenges in Penetration Testing

- **Scope Definition**: Ensuring the test covers all critical areas without disrupting operations.
- **False Positives**: Distinguishing between real vulnerabilities and false alarms.
- **Resource Intensive**: Requires skilled professionals and can be time-consuming.

Best Practices

- **Regular Testing**: Conduct tests regularly to keep up with new vulnerabilities.
- **Comprehensive Reporting**: Provide detailed reports with actionable recommendations.
- **Collaboration**: Work closely with internal teams to understand the environment and potential impacts.

Penetration testing is a proactive approach to cybersecurity, helping organisations to stay ahead of potential threats by identifying and mitigating vulnerabilities before they can be exploited.

Web application attacks

Web Application Attacks Account for Around 39% of All Breaches

Web application attacks are a major contributor to data breaches. According to various cybersecurity reports, approximately **39% of all data breaches involve web application attacks**. This high percentage is due to several factors:

- **Widespread Use**: Web applications are ubiquitous, used by businesses and individuals for a wide range of purposes.
- **Accessibility**: Being accessible over the internet makes them prime targets for attackers.
- **Complexity**: The complexity of web applications often leads to vulnerabilities that can be exploited.

Will Cost the World an Estimated $4 Trillion in 2025

The financial impact of web application attacks is staggering. It's estimated that by 2025, the global cost of cybercrime, including web application attacks, will reach **$4 trillion**. This figure includes:

- **Direct Costs**: Such as ransom payments, legal fees, and fines.
- **Indirect Costs**: Including loss of business, reputational damage, and recovery expenses.
- **Long-Term Costs**: Such as increased cybersecurity measures and insurance premiums.

Are Extremely Accessible Attack Targets

Web applications are particularly accessible targets for several reasons:

- **Internet Exposure**: They are often publicly accessible, making them easy to find and attack.
- **Common Vulnerabilities**: Many web applications share common vulnerabilities, such as SQL injection and XSS, which are well-documented and widely understood by attackers.
- **Automated Tools**: Attackers can use automated tools to scan for and exploit vulnerabilities in web applications, making it easier to launch attacks at scale.

Summary

- **Prevalence**: Web application attacks are a significant portion of all breaches due to their widespread use and inherent vulnerabilities.
- **Financial Impact**: The cost of these attacks is projected to be enormous, highlighting the need for robust security measures.

- **Accessibility**: The ease with which attackers can target web applications underscores the importance of secure coding practices and regular security assessments.

Cross-Site Scripting (XSS) is a type of security vulnerability found in web applications. It allows attackers to inject malicious scripts into web pages viewed by other users. Here's a brief overview:

Common Web Application Attacks

Cross Site Scripting (XSS)

How XSS Works

1. **Injection**: An attacker injects malicious code, usually JavaScript, into a web application.
2. **Execution**: When a user visits the compromised web page, the malicious script is executed in their browser.
3. **Impact**: The script can steal cookies, session tokens, or other sensitive information, and even manipulate the content displayed to the user.

Types of XSS

1. **Reflected XSS**: The malicious script is reflected off a web server, such as in an error message or search result.
2. **Stored XSS**: The malicious script is stored on the server, such as in a database, and is executed when the data is retrieved and displayed.
3. **DOM-Based XSS**: The vulnerability exists in the client-side code rather than the server-side code.

Prevention

- **Input Validation**: Ensure all user inputs are validated and sanitised.
- **Output Encoding**: Encode data before rendering it in the browser.
- **Content Security Policy (CSP)**: Implement CSP to restrict the sources from which scripts can be executed.

Cross Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is a type of attack that tricks a user into performing actions they didn't intend to on a web application where they are authenticated. Here's a brief overview:

How CSRF Works

1. **Trick the User**: An attacker tricks the user into clicking a malicious link or loading a malicious page.
2. **Execute Unwanted Actions**: The malicious request is sent to a web application where the user is authenticated, using their credentials.
3. **Impact**: The web application processes the request as if it came from the user, potentially changing settings, transferring funds, or performing other actions.

Example Scenario

- **User is Logged In**: The user is logged into their bank account.
- **Malicious Link**: The attacker sends the user a link that, when clicked, sends a request to transfer money from the user's account to the attacker's account.
- **Execution**: The bank processes the request because it appears to come from the authenticated user.

Prevention

- **CSRF Tokens**: Include unique tokens in forms and verify them on the server.
- **SameSite Cookies**: Use the SameSite attribute in cookies to prevent them from being sent with cross-site requests.
- **Double Submit Cookies**: Send a cookie with a token and include the same token in a request parameter, verifying both on the server.

SQL Injections

SQL Injection (SQLi) is a type of attack where an attacker inserts malicious SQL code into a query, allowing them to manipulate the database. Here's a brief overview:

How SQL Injection Works

1. **Injection**: An attacker inputs malicious SQL code into a web application's input fields (e.g., login forms, search boxes).
2. **Execution**: The application executes the injected SQL code as part of its query to the database.

3. **Impact**: The attacker can view, modify, or delete data, and in some cases, gain administrative access to the database.

Example Scenario

- **User Input**: A login form asks for a username and password.
- **Malicious Input**: The attacker inputs admin' OR '1'='1 as the username.
- **Resulting Query**: The SQL query becomes SELECT * FROM users WHERE username = 'admin' OR '1'='1' AND password = 'password'.
- **Effect**: The condition OR '1'='1' is always true, allowing the attacker to bypass authentication.

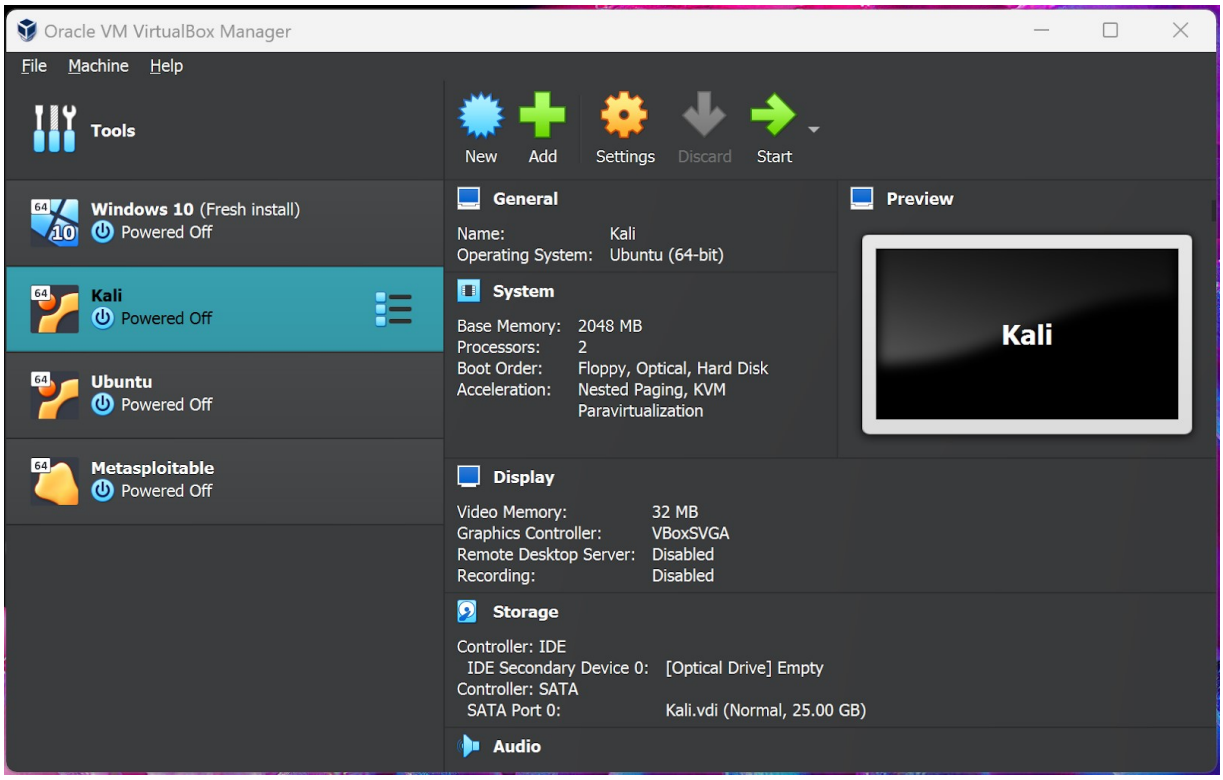
Prevention

- **Parameterized Queries**: Use prepared statements with parameterized queries to separate SQL code from data.
- **Input Validation**: Validate and sanitise all user inputs.
- **Least Privilege**: Limit database user permissions to only what is necessary.

Exploiting Unrestricted Uploads: A practical Example

In this practical walkthrough I created a digital environment utilising virtual machines and various tools associated with them. This environment results in the creation of a personal web server running a vulnerable web application on it. I then use a second machine running an installation of Kali linux to attack the web application via an unrestricted file upload on the site. Using this I am able to upload and then execute a PHP reverse shell on the system which connects back to my attacking Kali machine and gives me access to their target system via a shell.

- To begin we'll open our VM software, in this case I'm using virtualbox, and spin up a Kali linux VM and a metasploitable2 VM.



Metasploitable

Metasploitable is an intentionally vulnerable virtual machine (VM) designed for testing security tools and practising penetration testing techniques. Here's a brief overview:

Key Features

- **Purpose**: Metasploitable is used for training, exploit testing, and general target practice. It provides a safe environment to learn and practice penetration testing without risking real systems.
- **Vulnerabilities**: The VM is pre-configured with numerous vulnerabilities, focusing on operating system and network services rather than custom applications
- **Compatibility**: It is compatible with common virtualization platforms like VMware and VirtualBox
- **Default Credentials**: The default login credentials are msfadmin:msfadmin

Usage

- **Training**: Ideal for security professionals and students to practise and improve their skills.

- **Tool Testing**: Allows users to test the effectiveness of various security tools and techniques.
- **Demonstrations**: Useful for demonstrating common vulnerabilities and exploitation methods.

Setup

- **Download**: Available for download from platforms like SourceForge and Rapid7.
- **Installation**: Can be easily set up on virtualization software by importing the VM image and starting it up.

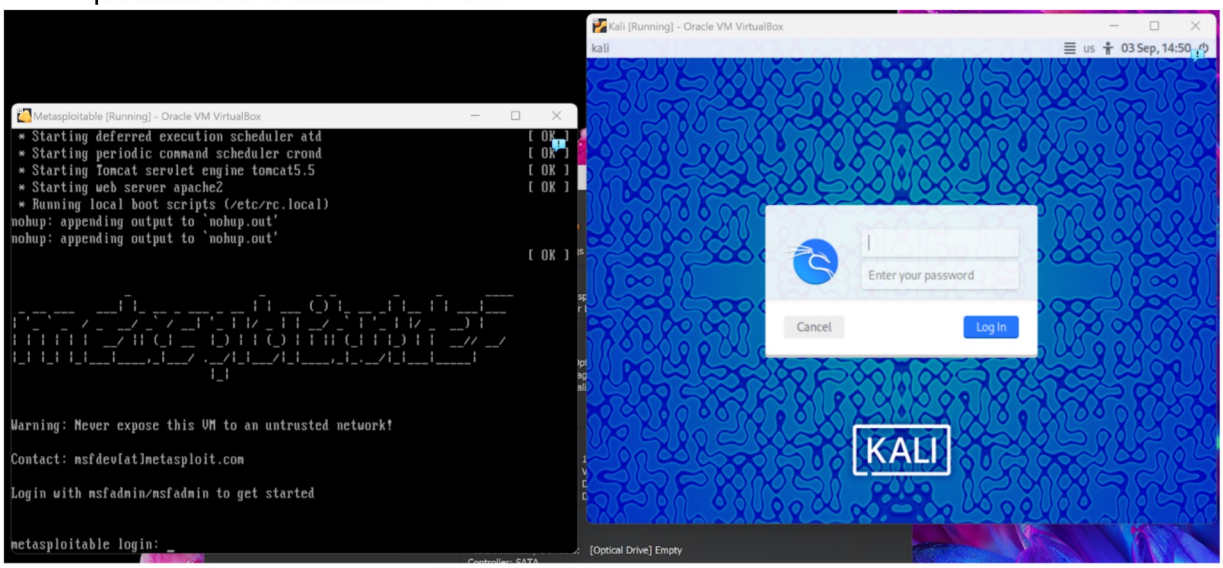
Example Scenario

A security professional might use Metasploitable to:

1. **Practice Exploits**: Test different exploits using tools like Metasploit.
2. **Learn Vulnerabilities**: Understand how various vulnerabilities can be exploited.
3. **Improve Skills**: Enhance their penetration testing skills in a controlled environment.

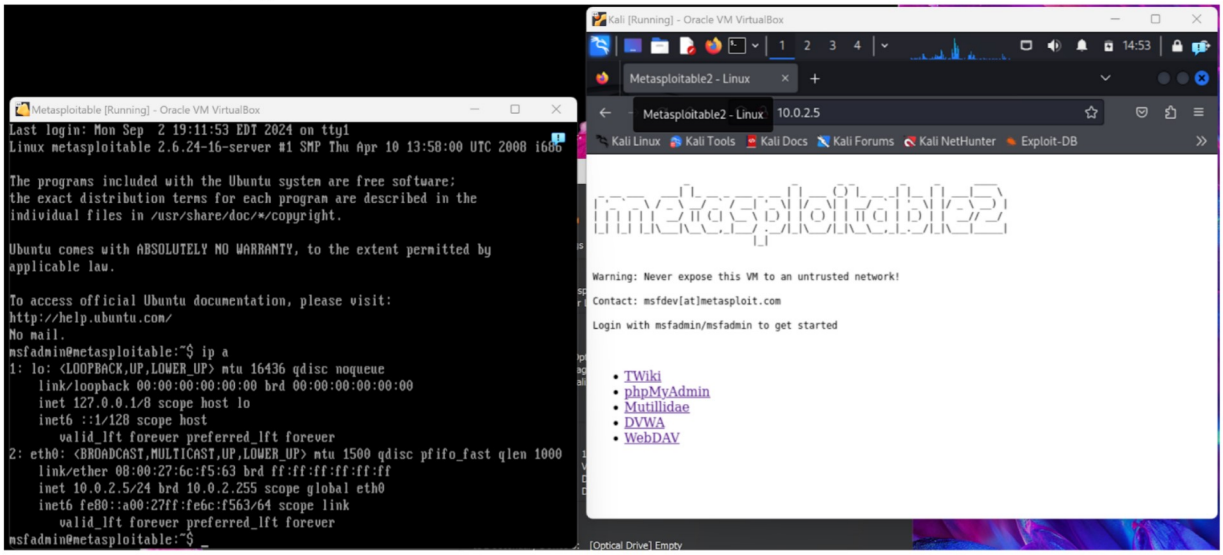
Metasploitable is a valuable resource for anyone looking to gain hands-on experience in penetration testing and cybersecurity.

- Once they're ready you can login to both with whatever credentials you've set. In this case I used the default msfadmin:msfadmin for metasploitable and my own personalised credentials for Kali.



- Firstly, we'll find the IP address of our metasploitable machine by running the "ip a" command on it. Then switch over to our Kali machine and type that IP into our browser,

This should bring up the metasploitable2 page, Finally click on the third link to access the DVWA page.



Damn Vulnerable Web Application - DVWA

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application designed to be intentionally insecure. Its primary purpose is to help security professionals, developers, and students practise and improve their skills in a legal and controlled environment.

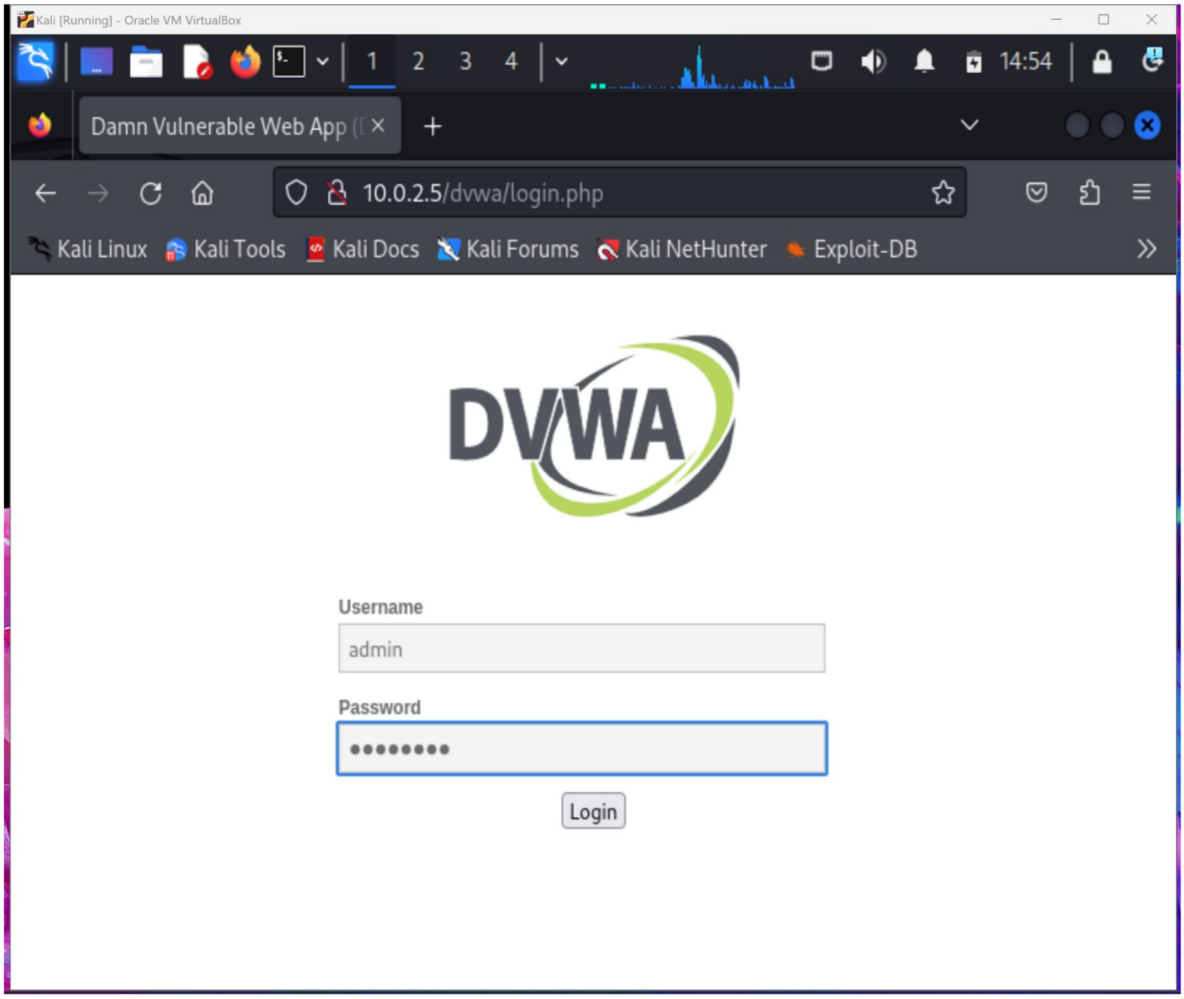
Key Features:

- **Variety of Vulnerabilities**: DVWA includes multiple security flaws, such as SQL Injection, Cross-Site Scripting (XSS), and Command Injection, among others.
- **Difficulty Levels**: It offers different levels of difficulty (low, medium, high, and impossible) to help users gradually improve their skills.
- **Educational Tool**: Ideal for learning and teaching web application security, it provides a hands-on approach to understanding how vulnerabilities can be exploited and how to defend against them.

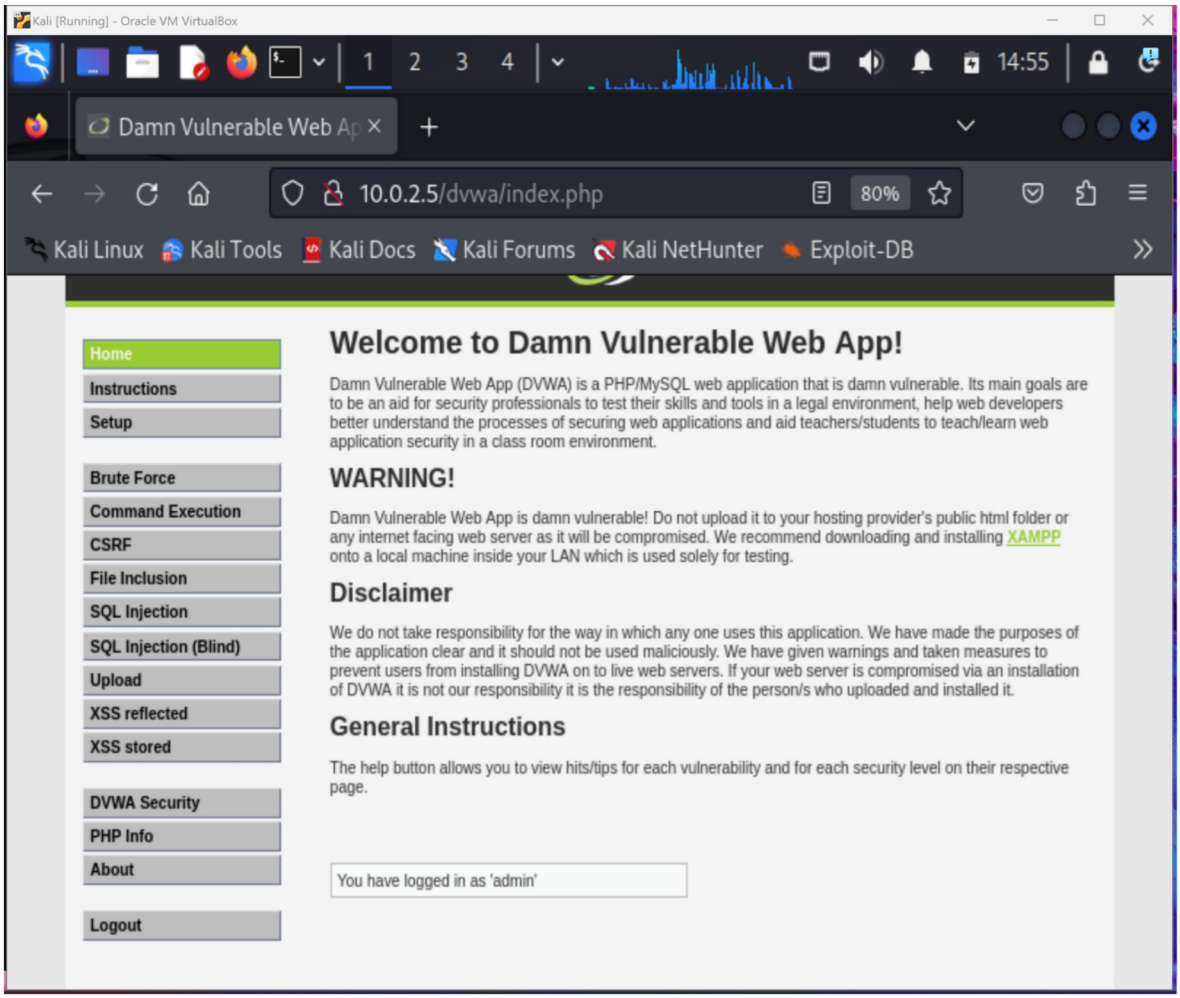
Usage:

- **Penetration Testing Practice**: Users can simulate attacks and test their penetration testing tools and techniques.
- **Security Awareness**: Helps developers understand common security issues and how to mitigate them.
- **Legal and Safe**: Designed for use in a controlled environment, such as a virtual machine, to ensure safe and legal practice.

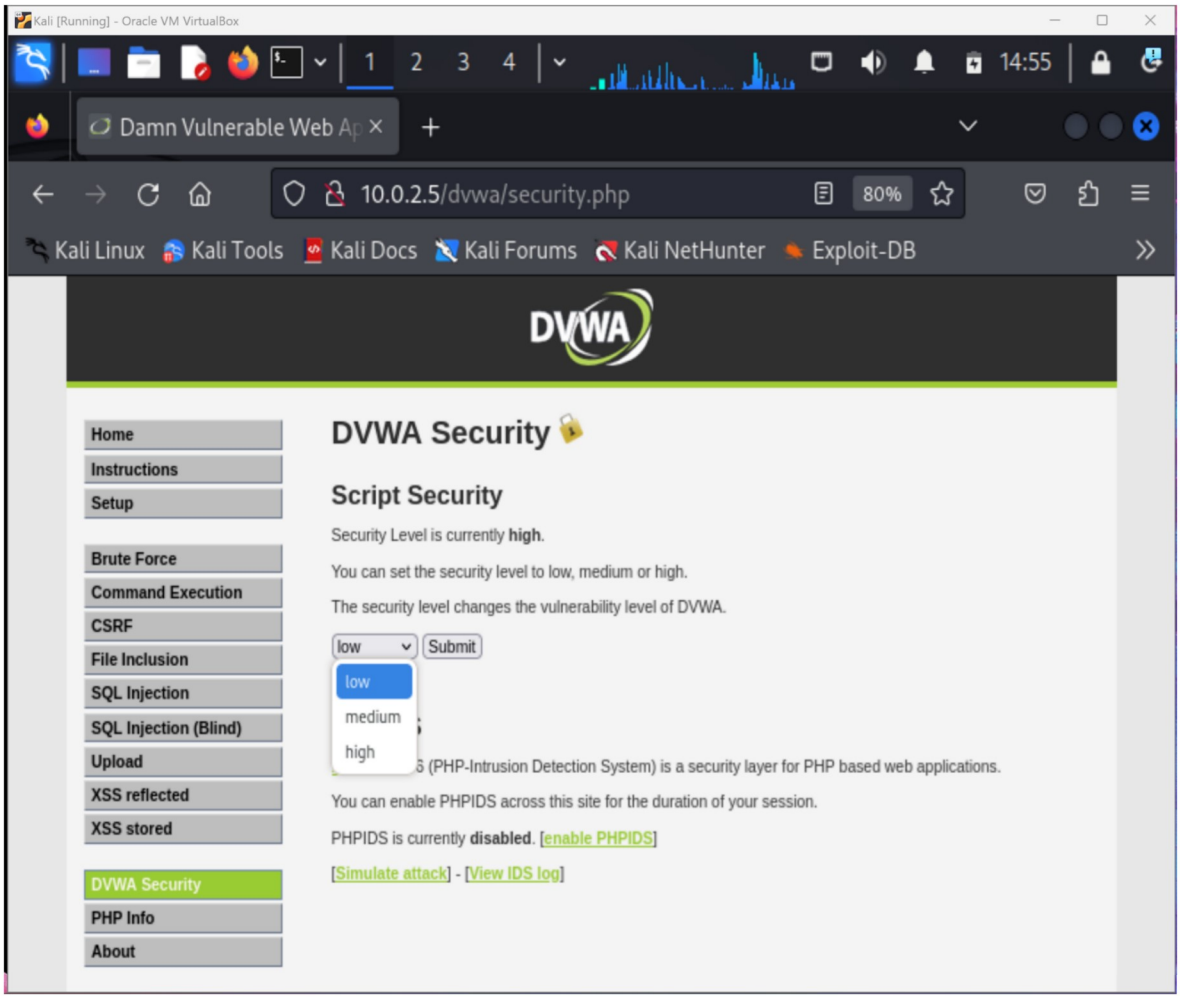
- Now we'll need to login to DVWA with the default credentials: admin:password



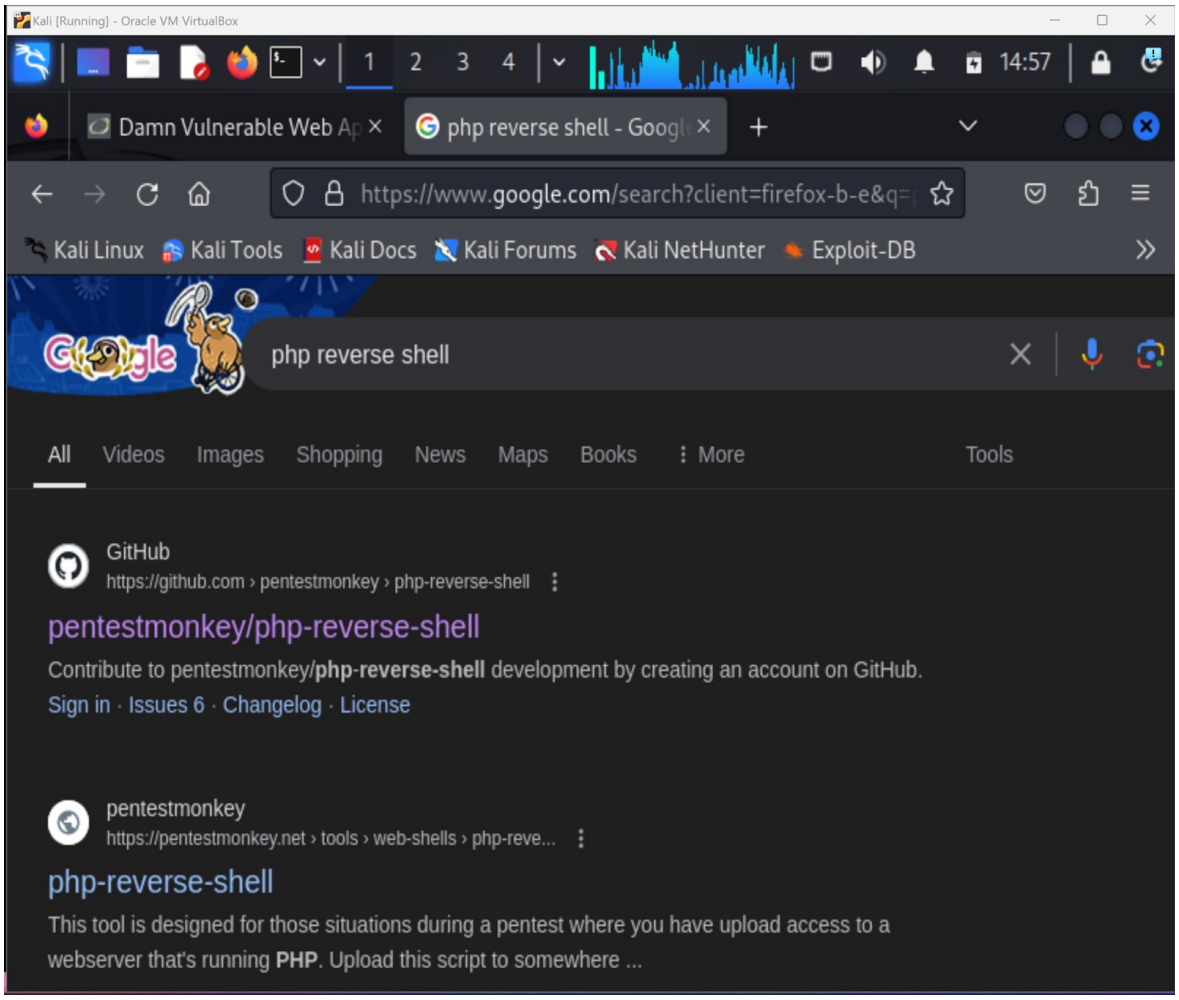
- After that we're greeted with the home page, it has a variety of different attack vectors you can try to exploit and practise with, you'll notice several of the attacks I mentioned previously are available here. To begin we'll navigate to the DVWA security page.



- On this page we need to change the difficulty to low and hit submit. This will change the security settings to be extremely flimsy and this is to highlight the ease an attack like this can be done without preventative measures.



- Now we're going to download our payload to use on the target. I chose to use a PHP reverse shell which was written by pentestmonkey and is available on their github.



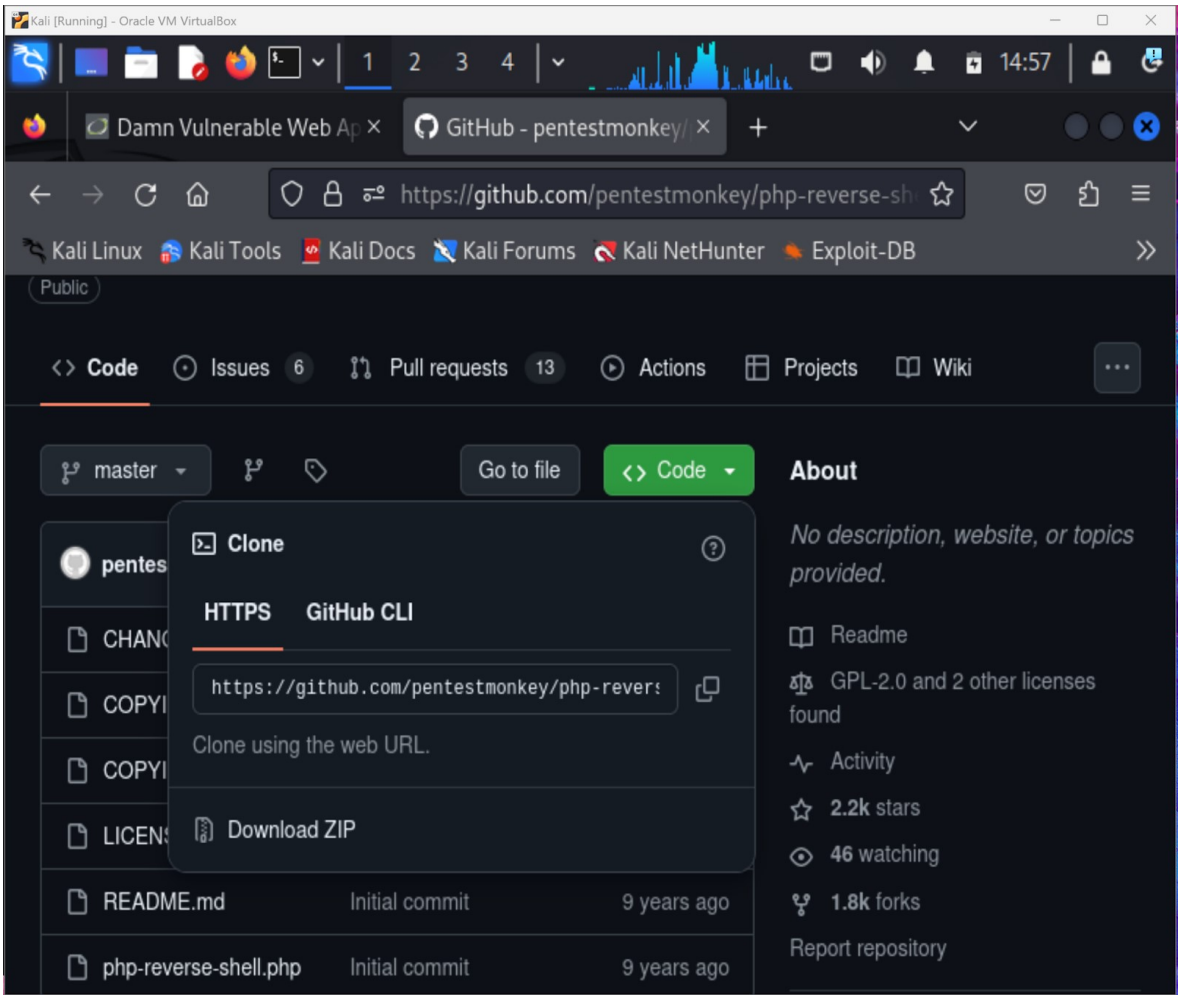
Index

- A payload in this context can be used to refer to any malicious code that can cause harm to the target

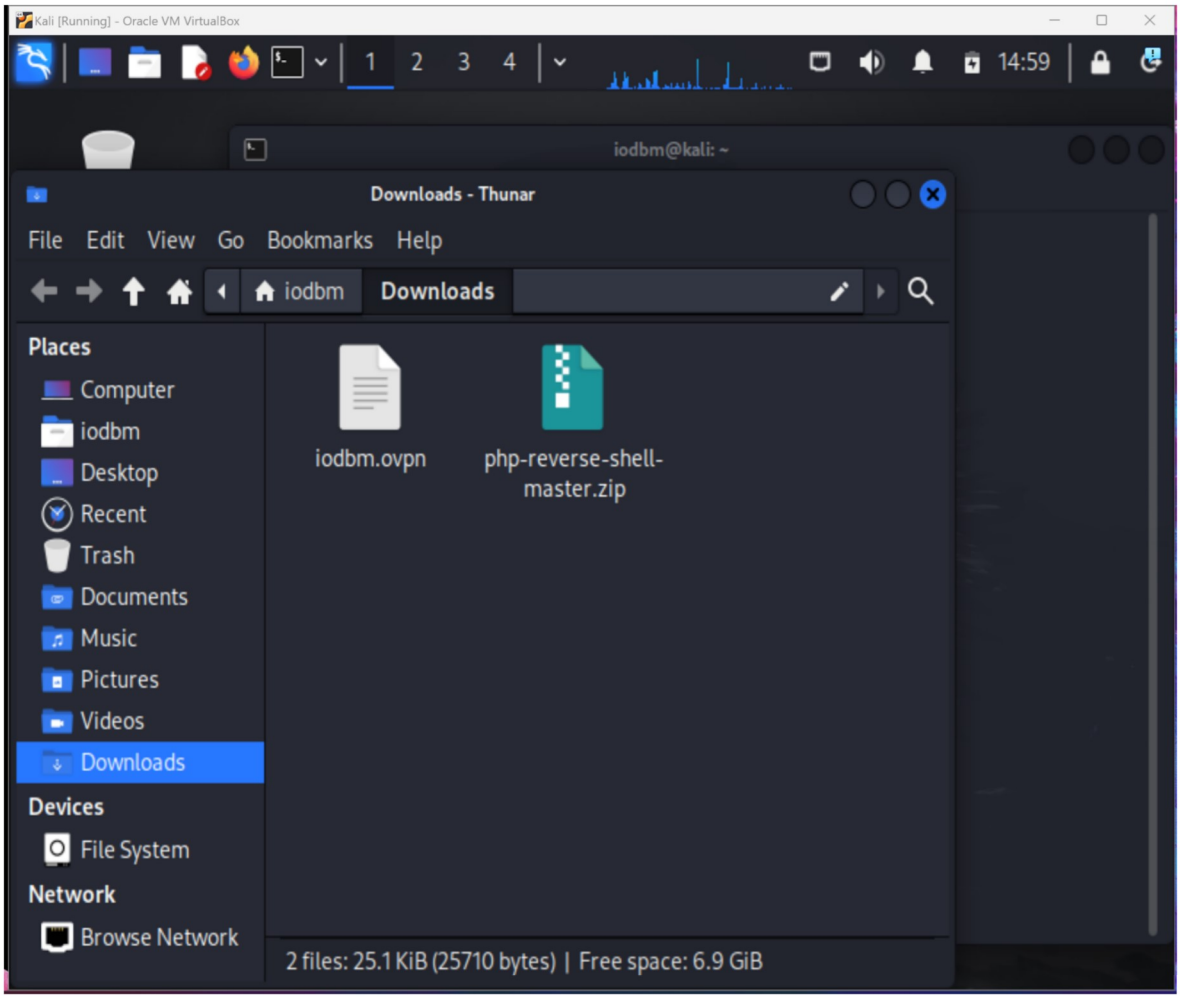
- PHP is a scripting language commonly used with web development which makes it suitable to attack web applications

- A reverse shell is a type of attack where a threat actor connects a target machine to their machine, generally granting them access

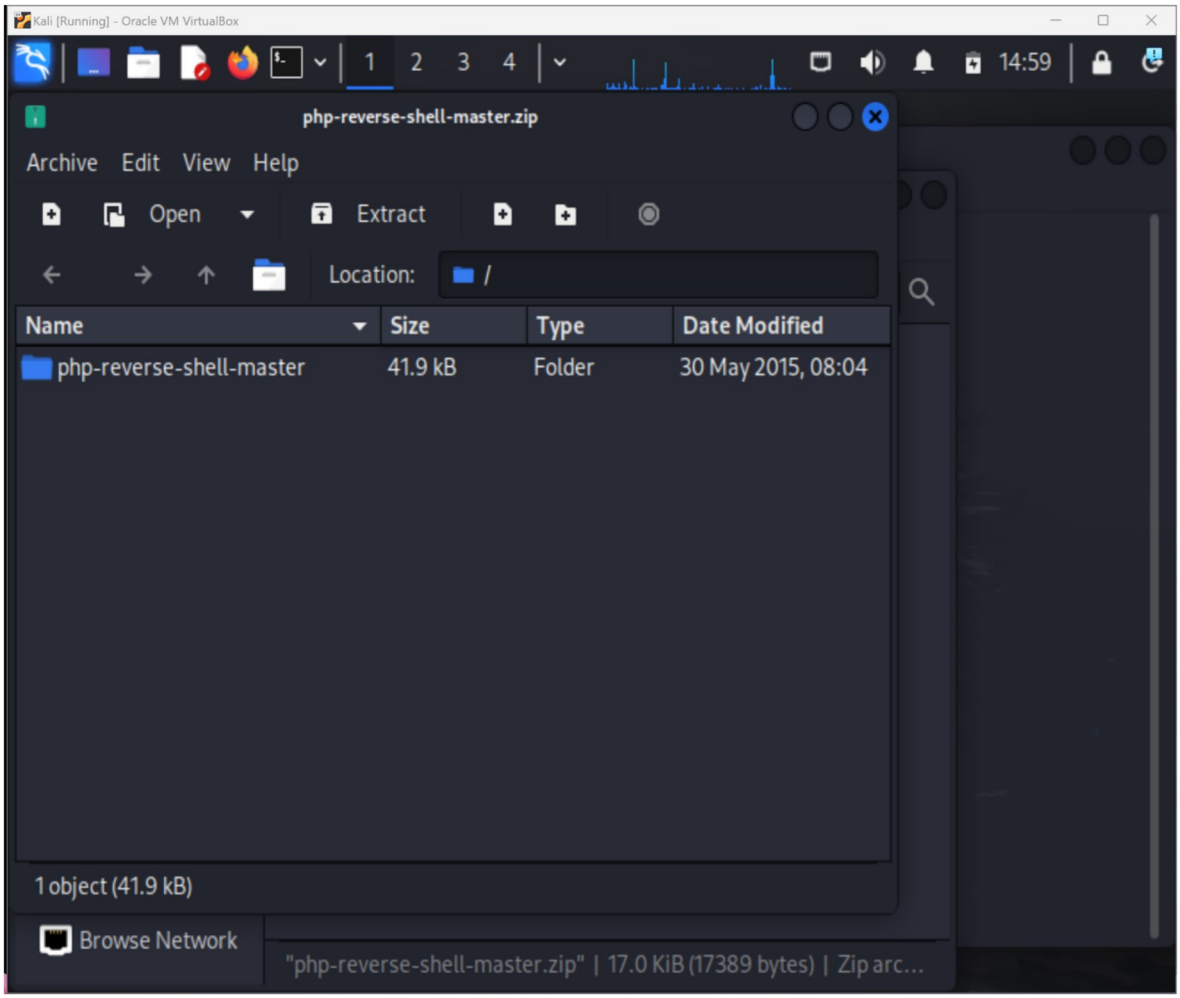
- Once in the site we can click the arrow on the green code button. Then from that menu we click the Download ZIP button. This will begin the download



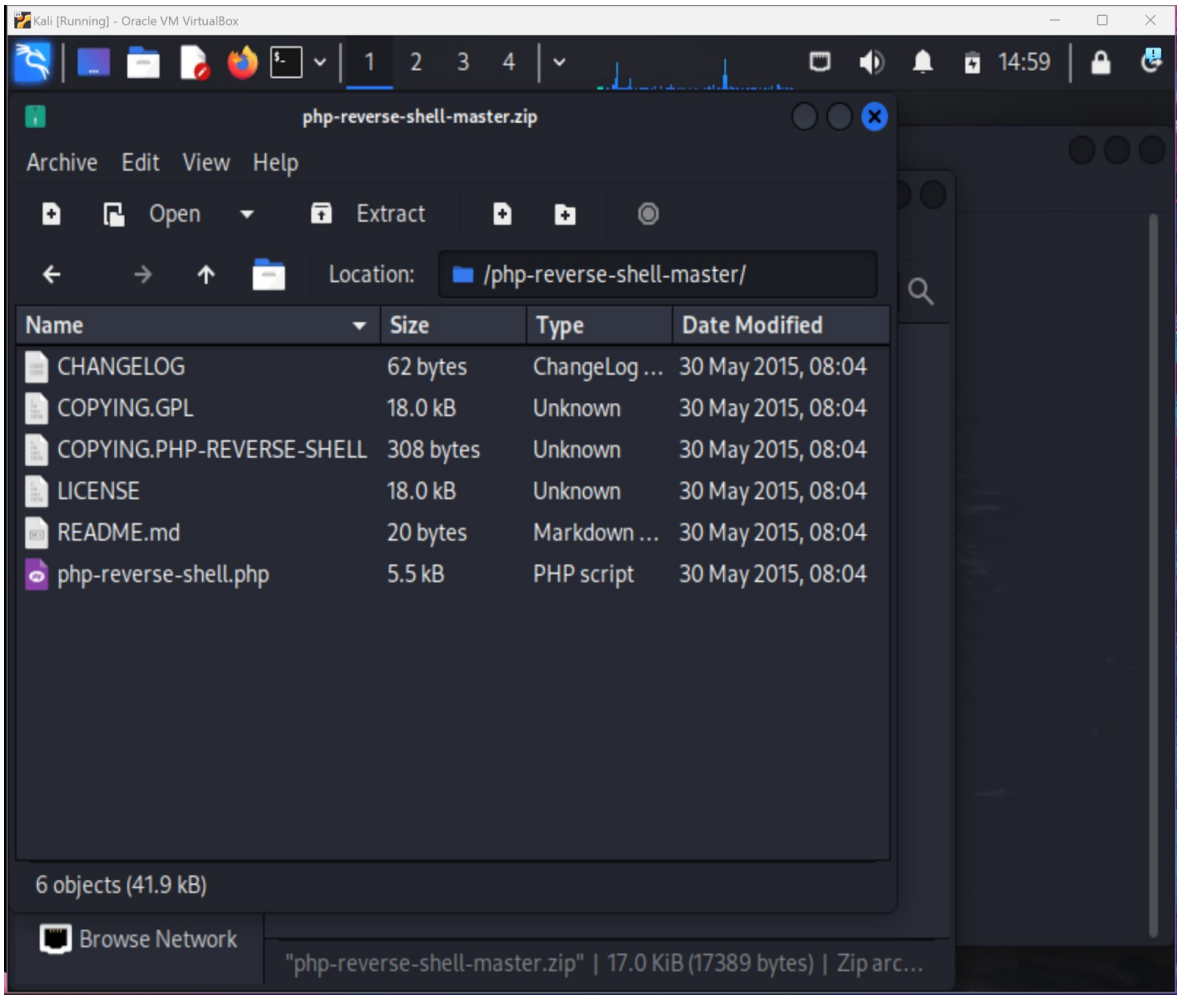
- Now you'll need to go to your downloads folder and open the zip file



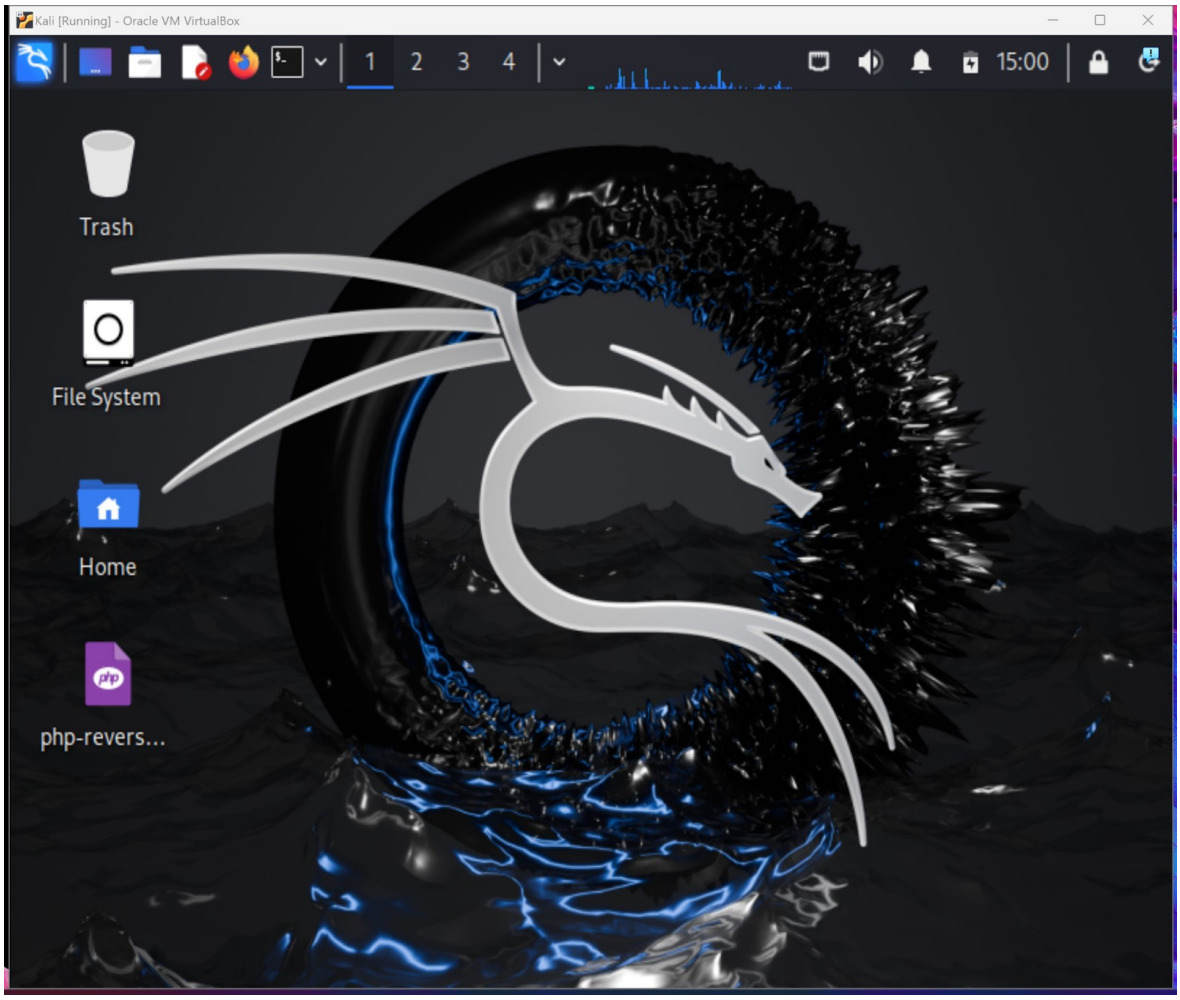
- Inside will be a folder which we need to open as well



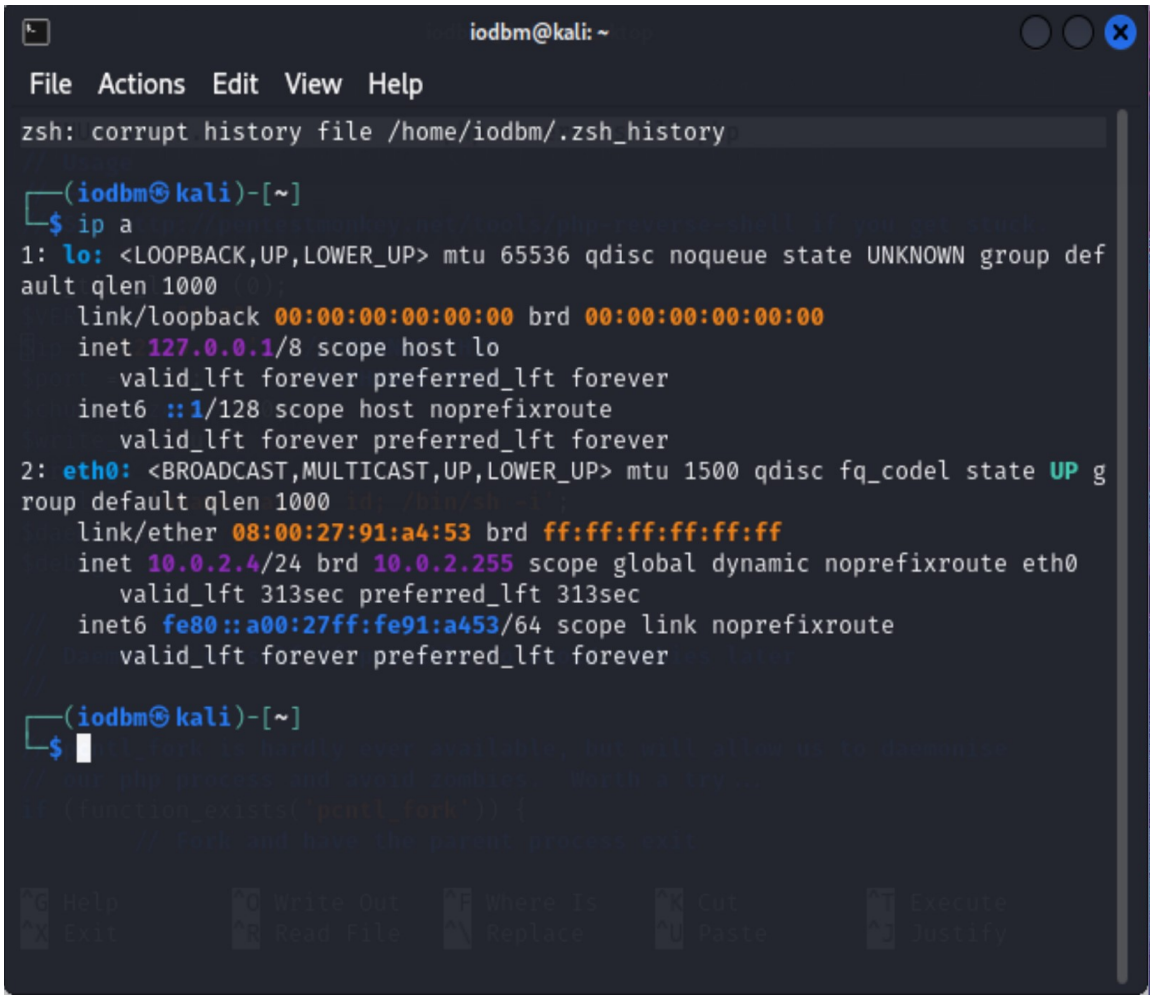
- Inside of this folder there will be several documents which correspond with the program. We will ignore these and instead focus on the purple .php file, this is the payload.



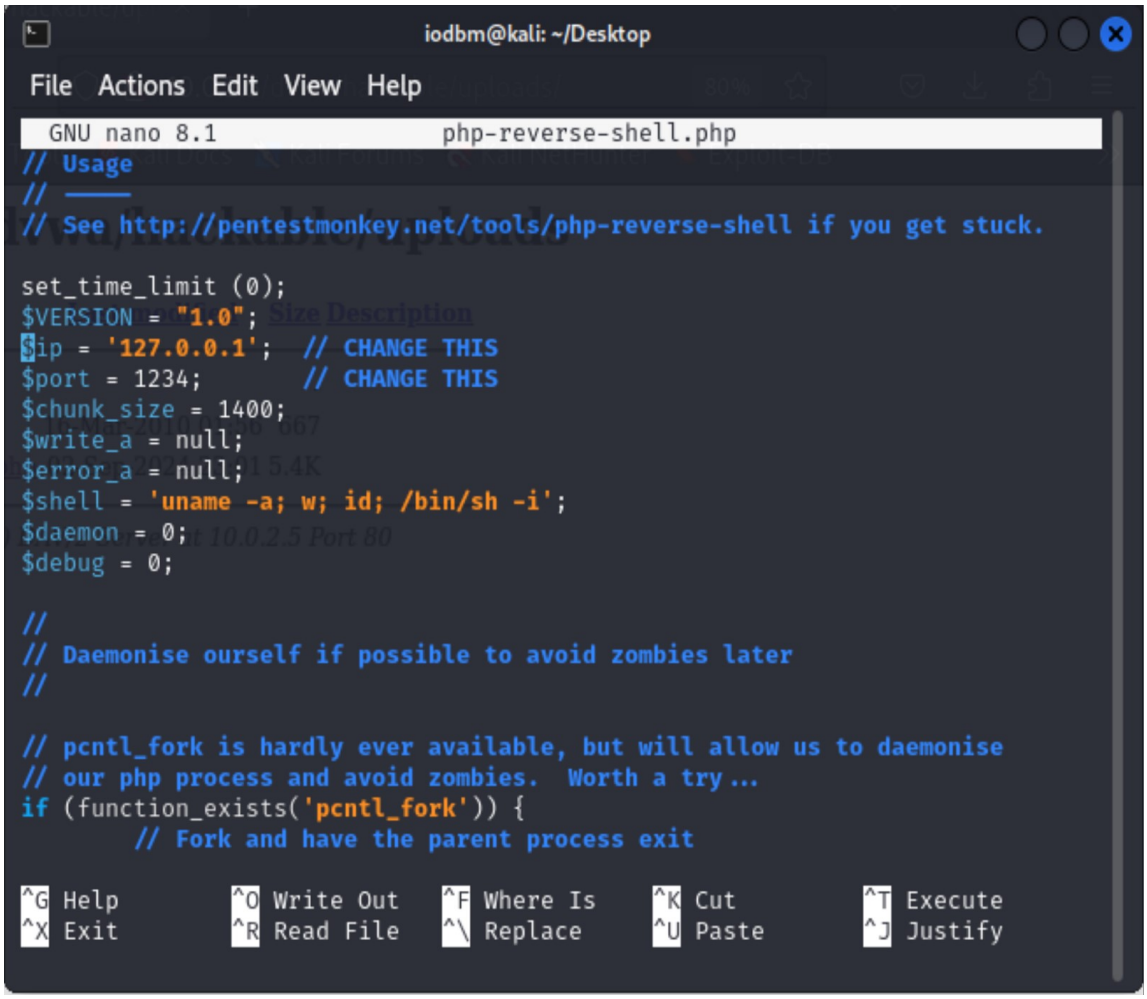
- This next step isnt necessary but I decided to move my payload to the desktop to make it easier to use. After that we need to open a new terminal window.



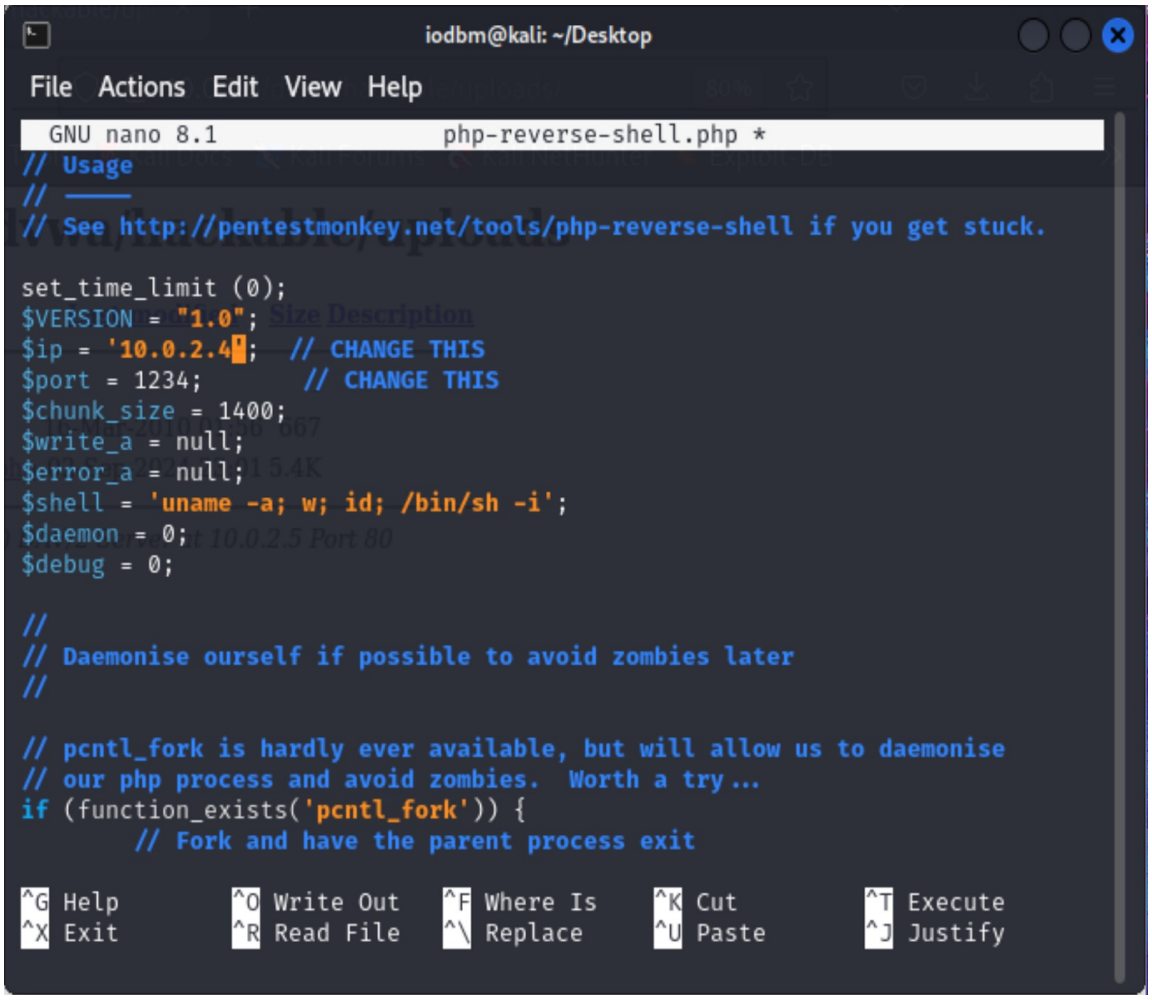
- We can use the “ip a” command again here to find the attacking (Kali) machine's IP in this case it is 10.0.2.4.



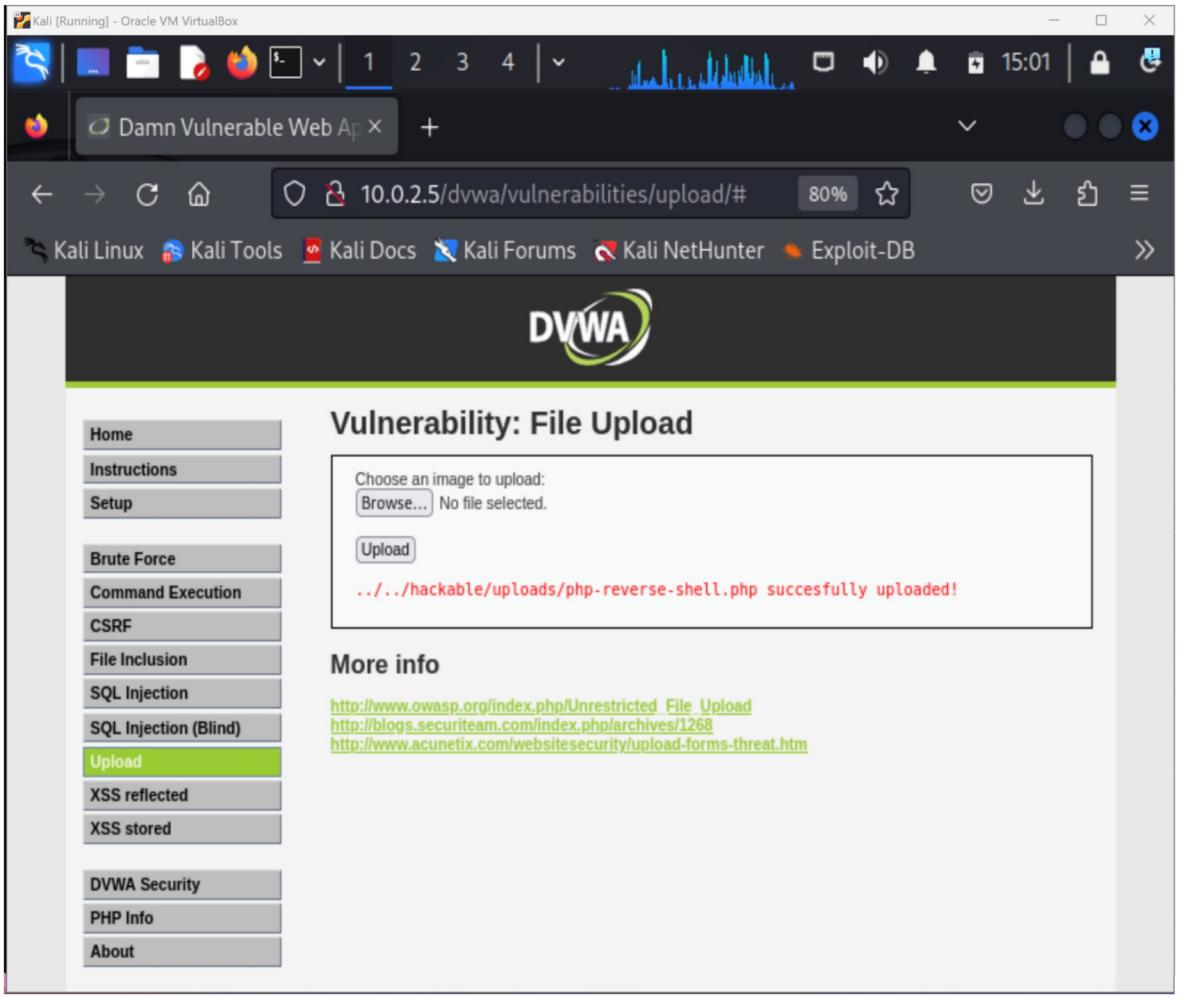
- Now that you have the payload you will need to modify it to make it work on your system. We'll change the ip value to the ip we found in the previous step and we can leave the port as default.



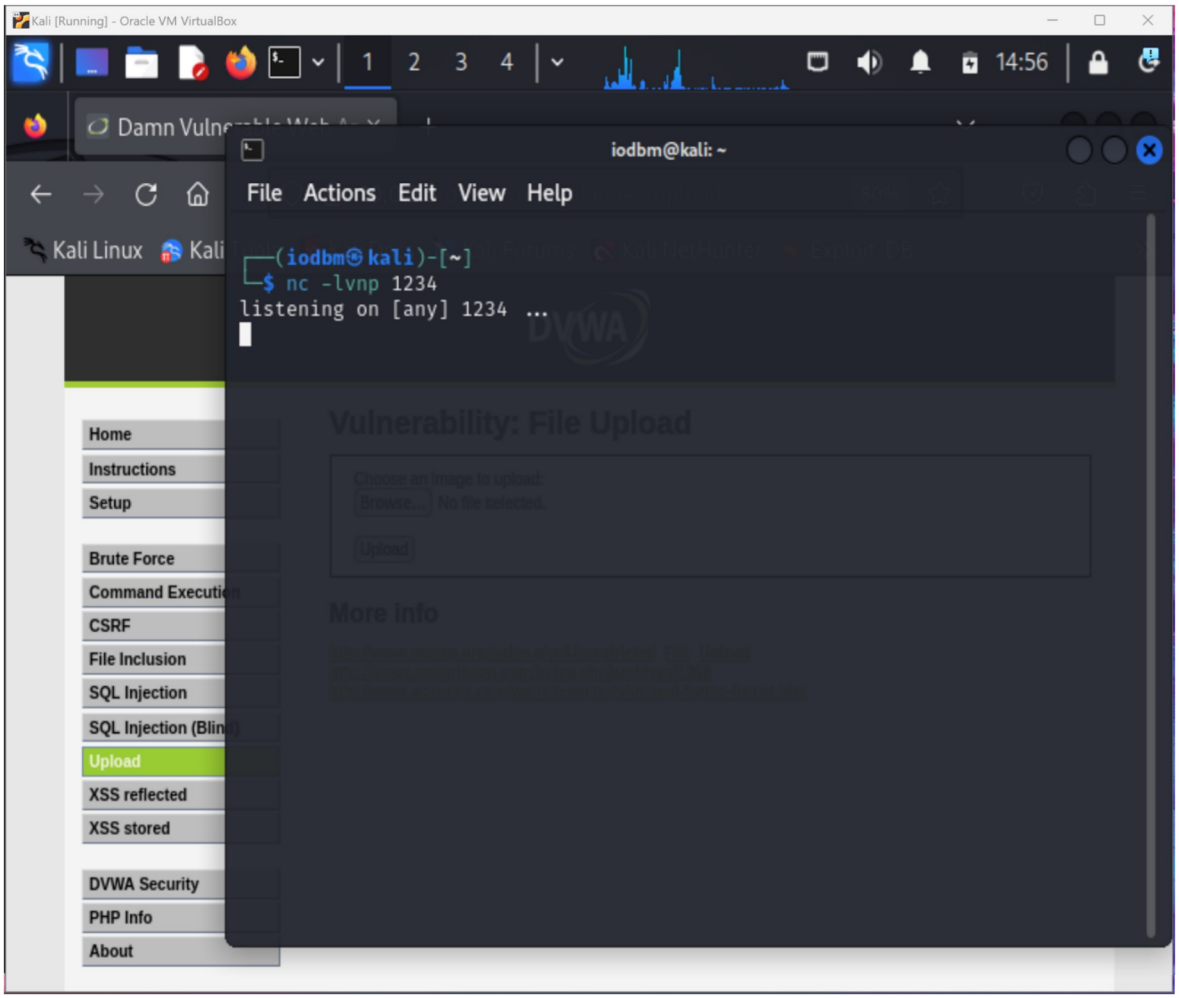
- Now that we've made our changes, we can save and exit the file.



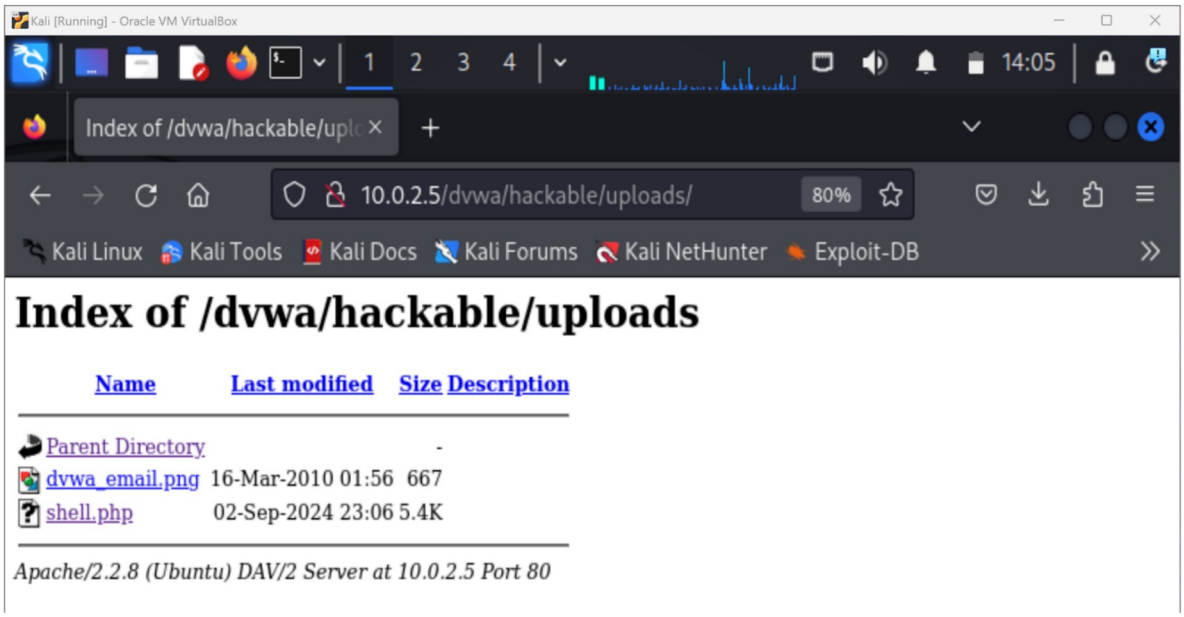
- Next we can move back to the browser and click the browse button, then we'll select the file from wherever we stored it and hit upload to upload it to the site.



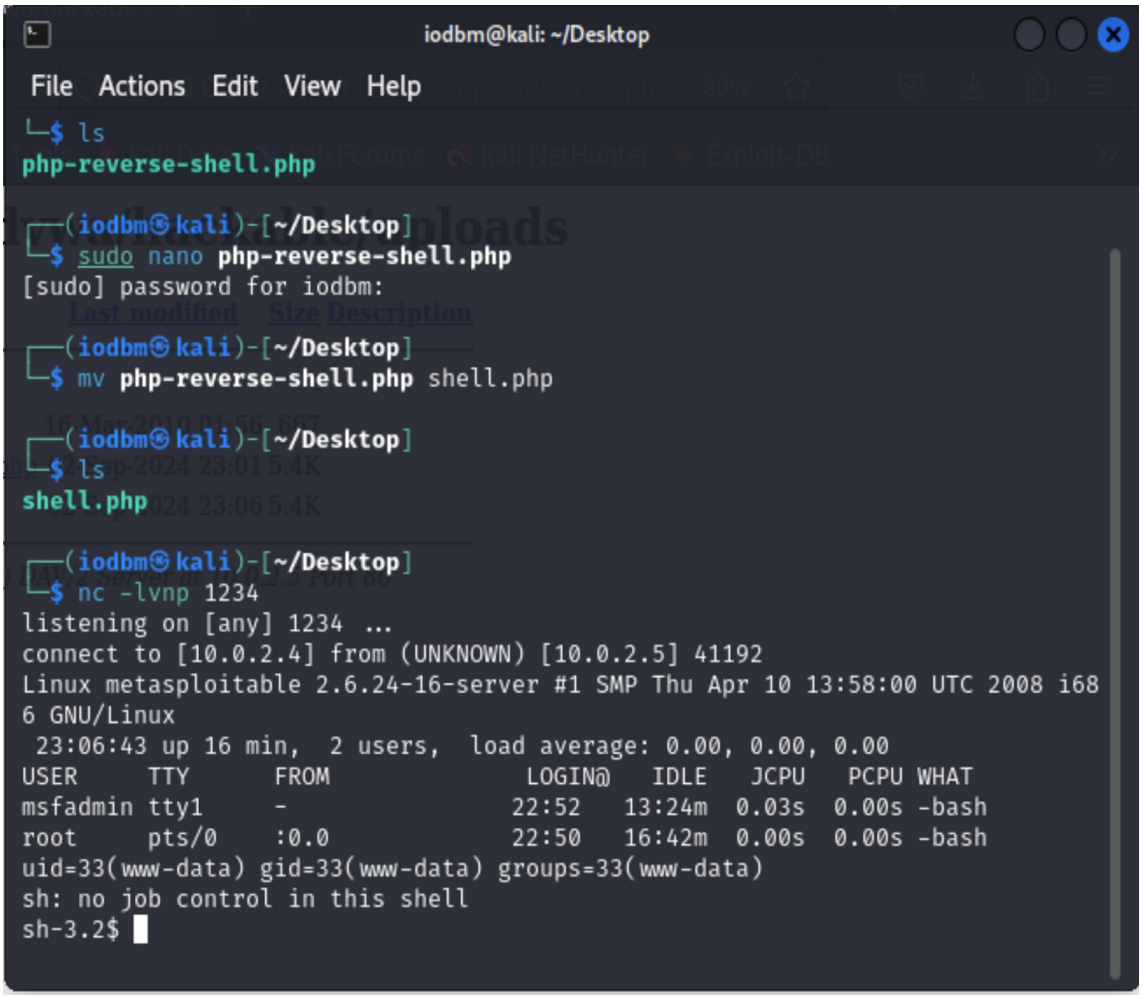
- Once we've uploaded the file we'll go back to our terminal and set up a listener by typing the command nc -lvnp 1234, nc stands for Netcat which is a security tool that we can use to set up listeners on a particular port, a listener is a feature which captures incoming data on the port.



- Now that the file has been uploaded we need to find where it has gone, luckily it tells us the directory here so if we add that to the end of our current url we'll be shown our script. Now with our listener still active, we just need to click the file to run the script.



- After that, when we switch over to our terminal we'll see some new information. Under our command to set the listener it now says there's a connection to our kali IP from the metasploitable IP then some more information about the machine we've connected to and finally a shell command prompt.



With all of that complete we finally have a reverse shell established and can move to the privilege escalation step if we choose to do so. However, for this report I have chosen to omit the last two steps of the general penetration testing procedure as I feel they do not contribute to the topic of this report.

Mitigation

As I have now shown, exploiting a web application's unrestricted uploads can be both very effective and extremely simple without the proper security in place to prevent it. Therefore I have several suggestions to improve security on this specific vulnerability as well as on web applications as a whole.

For file upload vulnerabilities

1. Limit the Type of Files Users Can Upload

Purpose: Prevent the upload of potentially harmful files like PHP scripts, which could be executed on the server and compromise security.

Implementation:

- **File Type Whitelisting**: Only allow specific file types (e.g., .jpg, .png, .pdf) to be uploaded. This can be enforced through both client-side and server-side validation.
- **MIME Type Checking**: Verify the MIME type of the uploaded file to ensure it matches the expected file type.
- **File Extension Checking**: Check the file extension to ensure it is allowed. However, this should not be the sole method of validation as it can be easily bypassed.

2. Store Uploaded Files in Secure Directories

Purpose: Ensure that uploaded files cannot be accessed or executed directly by users, reducing the risk of malicious code execution.

Implementation:

- **Non-Web Accessible Directories**: Store files in directories that are not directly accessible via the web. This prevents users from accessing the files through a URL.
- **Randomised File Names**: Rename uploaded files to random, unique names to prevent guessing and direct access.
- **Access Controls**: Implement strict access controls to ensure only authorised users or processes can access the uploaded files.

3. Scan Incoming Files for Malicious Code or Viruses

Purpose: Detect and prevent the upload of files containing malicious code or viruses that could harm the system.

Implementation:

- **Antivirus Software**: Integrate antivirus software to scan files upon upload. This can be done using APIs provided by antivirus vendors.
- **Malware Scanners**: Use specialised malware scanners to detect and block malicious code.
- **Sandboxing**: Execute files in a sandbox environment to observe their behaviour and detect any malicious activity before allowing them to be stored or processed.

Summary

- **File Type Restrictions**: Prevent harmful files from being uploaded by allowing only specific file types.
- **Secure Storage**: Store files in secure, non-web accessible directories with randomised names and strict access controls.
- **File Scanning**: Use antivirus and malware scanning tools to detect and block malicious files.

These measures collectively enhance the security of your web application by mitigating the risks associated with file uploads.

For web applications as a whole

Policies - Improved Guidelines for Companies and Developers

Improved guidelines for companies and developers are essential to ensure a consistent and secure approach to software development and cybersecurity. These guidelines typically include:

- **Code of Conduct**: Establishes expected behaviour and ethical standards.
- **Data Protection and Privacy Policies**: Ensures compliance with regulations like GDPR and CCPA.
- **Incident Response Plans**: Provides a structured approach to handle security breaches.
- **Software Development Policies**: Includes secure coding standards, code review processes, and version control practices.

Tools & Techniques - Secure Coding Practices

Secure coding practices help developers write code that is less vulnerable to attacks. Key practices include:

- **Input Validation**: Ensuring all input is validated to prevent injection attacks.
- **Output Encoding**: Encoding data before output to prevent cross-site scripting (XSS).
- **Authentication and Password Management**: Implementing strong authentication mechanisms and secure password storage.
- **Session Management**: Properly managing user sessions to prevent hijacking.
- **Access Control**: Ensuring users have appropriate access levels.
- **Cryptographic Practices**: Using strong encryption for data protection.

- **Error Handling and Logging**: Securely handling errors and logging events without exposing sensitive information.

Training - For Employees to Recognize Threats

Training employees to recognize threats is crucial for maintaining cybersecurity. Effective training programs include:

- **Phishing Awareness**: Teaching employees to identify and avoid phishing scams.
- **Password Management**: Educating on creating and managing strong passwords.
- **Safe Internet Practices**: Promoting safe browsing habits and recognizing suspicious links.
- **Incident Reporting**: Encouraging employees to report potential security incidents promptly.
- **Regular Simulations**: Conducting mock phishing attacks and other threat simulations to test and reinforce training.

Frameworks - Such as OWASP or NIST

Frameworks provide structured approaches to managing cybersecurity risks. Two prominent frameworks are:

- **OWASP (Open Web Application Security Project)**: Focuses on improving the security of software through community-led open-source projects, such as the OWASP Top 10, which lists the most critical web application security risks.
- **NIST (National Institute of Standards and Technology)**: Provides comprehensive frameworks like the NIST Cybersecurity Framework, which helps organisations manage and reduce cybersecurity risk through a set of industry standards and best practices

The Dynamic Nature of Cybersecurity: A conclusion

1. **Evolving Threat Landscape**:
    - **Adaptive Adversaries**: Cyber threat actors are constantly evolving their tactics, techniques, and procedures (TTPs) to bypass security measures. For every new defence mechanism, attackers develop countermeasures to circumvent it.
    - **Example**: When organisations implement multi-factor authentication (MFA), attackers may resort to phishing attacks to steal authentication

tokens or use social engineering to trick users into revealing their credentials.

2. **No Perfect Security**:
    - **Inherent Vulnerabilities**: Every system has inherent vulnerabilities due to the complexity of software and hardware. New vulnerabilities are discovered regularly, and not all can be patched immediately.
    - **Resource Constraints**: Organisations often face resource constraints, making it challenging to implement and maintain comprehensive security measures across all systems and applications.

The Importance of an Attacker's Perspective

1. **Proactive Defence**:
    - **Penetration Testing**: By simulating real-world attacks, penetration testers can identify vulnerabilities before malicious actors exploit them. This proactive approach helps in strengthening defences.
    - **Red Team Exercises**: Red teaming involves a group of security professionals who simulate advanced persistent threats (APTs) to test the effectiveness of an organisation's security posture.
2. **Understanding Attack Vectors**:
    - **Threat Modelling**: By understanding how attackers think and operate, organisations can better anticipate potential attack vectors and implement targeted defences.
    - **Example**: If an organisation knows that attackers often exploit unpatched software, they can prioritise patch management and ensure critical systems are updated promptly.
3. **Continuous Improvement**:
    - **Feedback Loop**: Regularly testing and evaluating security measures helps in identifying gaps and areas for improvement. This continuous feedback loop is essential for maintaining a robust security posture.
    - **Learning from Incidents**: Analysing past security incidents and breaches provides valuable insights into how attacks were carried out and what defences were effective or failed.

Sub-Conclusion

While it is impossible to achieve perfect security, adopting an attacker's mindset allows organisations to identify and mitigate vulnerabilities more effectively. By continuously testing and improving security measures, organisations can stay one step ahead of threat actors and reduce the risk of successful exploits.

Bibliography

- https://www.ibm.com/topics/penetration-testing
- https://www.cisco.com/c/en/us/products/security/what-is-pen-testing.html
- https://www.synopsys.com/glossary/what-is-penetration-testing.html
- https://tryhackme.com/r/room/pentestingfundamentals
- https://www.stationx.net/bug-bounty-programs-for-beginners/
- https://tryhackme.com/r/careers/quiz?utm_source=email_marketing&utm_medium=byte&utm_campaign=careers_quiz
- https://github.com/vavkamil/awesome-bugbounty-tools
- https://tryhackme.com/module/learn-burp-suite
- https://cybersapiens.com.au/cyber-awareness/top-15-best-tools-you-need-to-become-a-pro-bug-bounty-hunter/
- https://portswigger.net/burp/documentation/desktop/getting-started
- https://www.zerodayinitiative.com/
    - Particularly the blog
- https://www.reddit.com/r/bugbounty/comments/jw6ie1/how_hard_is_it_to_find_a_bug/
- https://www.reddit.com/r/
- bugbounty/
- https://en.wikipedia.org/wiki/Bug_bounty_program
- https://b0mk35h.medium.com/my-first-bug-hunting-experience-a-journey-from-disappointment-to-success-ae92c222a0d0
- https://www.youtube.com/watch?v=vRBihr41JTo
- https://www.verizon.com/business/resources/reports/dbir/
- https://www.cobalt.io/blog/cybersecurity-statistics-2024
- https://owasp.org/www-community/attacks/csrf
- https://www.youtube.com/watch?v=eWEgUcHPle0
- https://blog.barracuda.com/2024/02/07/threat-spotlight-attackers-targeting-web-applications-right-now
- https://www.cobalt.io/blog/cybersecurity-statistics-2024
- https://www.youtube.com/watch?v=vRBihr41JTo
- https://www.youtube.com/watch?v=56h9hErCvto&t=24s
- Microsoft Copilot
- https://sourceforge.net/projects/metasploitable/
- https://docs.rapid7.com/metasploit/metasploitable-2/
- https://www.vulnhub.com/series/metasploitable,9/
- https://resources.workable.com/tutorial/policies-any-organization-should-have-plus-templates

- https://www.forbes.com/councils/forbeshumanresourcescouncil/2021/08/19/11-ways-to-ensure-your-company-policies-are-beneficial/
- https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/01-introduction/05-introduction
- https://portswigger.net/web-security/sql-injection
- https://www.w3schools.com/sql/sql_injection.asp
- https://www.spiceworks.com/it-security/application-security/articles/what-is-sql-injection/
- https://owasp.org/www-community/attacks/csrf
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- https://portswigger.net/web-security/csrf
- https://owasp.org/www-community/attacks/xss/
- https://www.cloudflare.com/learning/security/threats/cross-site-scripting/
- https://en.wikipedia.org/wiki/Cross-site_scripting