1. File path traversal, simple case:
   * Found the vulnerability in the URL section of the application and forced the API to retrieve the data of the file **passwd** from the backend by writing file traversal syntax in the image tag by using traversal sequence techniques i.e. “..**/../../”**.
2. File path traversal, traversal sequences blocked with absolute path bypass:
   * In this case the traversal sequences are blocked by the application but it can be bypassed by using absolute path traversal techniques or by nested traversal sequence bydirectly going to the desired path.
3. File path traversal, traversal sequences stripped non-recursively:
   * In this case the traversal sequences are blocked by the application but it can be bypassed by using absolute path traversal techniques or by nested traversal sequence i.e. “….// or ….\/”.
4. File path traversal, traversal sequences stripped with superfluous URL-decode:
   * In this case the traversal sequences are decoded from URL so we need to provide the traversal path using the URL encoded symbols like **%252f** instead of **/.**
5. Lab: File path traversal, validation of start of path:
   * In this case the application verifies every data source requested by the user in the URL so we need to enter the entire path from the main head directory in order to exploit the vulnerability.
6. File path traversal, validation of file extension with null byte bypass:
   * Now in order to exploit this vulnerability we need to insert null bytes into the URL parameters in the end because the application has implemented in the frond by using **%00 as NULL Byte.**