**Student Name: Jun Wang Weight:**

**Student ID: 000791814 Marks:**

Assignment 1

Answer all of the following questions. Type your answer below each question and then submit this document using the Assignment tool in Brightspace by the due date (see your Course Schedule).

1. Describe the following approaches to input handling:
   1. Reject Known Bad

This approach typically employs a blacklist containing a set of literal strings or

patterns that are known to be used in attacks. The validation mechanism blocks

any data that matches the blacklist and allows everything else. This is regarded as the least effective approach to validating user input.

* 1. Accept Known Good

This approach employs a whitelist containing a set of literal strings or patterns,

or a set of criteria, that is known to match only benign input. The validation mechanism allows data that matches the whitelist and blocks everything else. In cases where this approach is feasible, it is regarded as the most effective

way to handle potentially malicious input.

* 1. Sanitization

This approach recognizes the need to sometimes accept data that cannot be guaranteed as safe. Instead of rejecting this input, the application sanitizes it

in various ways to prevent it from having any adverse effects. Potentially mali-

cious characters may be removed from the data, leaving only what is known to

be safe, or they may be suitably encoded or “escaped” before further processing

is performed.

* 1. Safe Data Handling

Many web application vulnerabilities arise because user-supplied data is processed in unsafe ways. Vulnerabilities often can be avoided not by validating

the input itself but by ensuring that the processing that is performed on it is

inherently safe.

* 1. Semantic Checks

With some vulnerabilities the input supplied by the attacker is identical to the input that an ordinary, nonmalicious user may submit. What makes it malicious is the different circumstances under which it is submitted.

1. Explain the following HTTP headers:

**General Headers**

* 1. Connection

tells the other end of the communication whether it should close the TCP connection after the HTTP transmission has completed or keep it open for further messages.

* 1. Content-Encoding

specifies what kind of encoding is being used for the content contained in the message body.

* 1. Content-Length

specifies the length of the message body, in bytes (except in the case of responses to HEAD requests, when it indicates the length of the body in the response to the corresponding GET request).

* 1. Content-Type

specifies the type of content contained in the message body, such as text/html for HTML documents.

* 1. Transfer-Encoding

specifies any encoding that was performed on the message body to facilitate its transfer over HTTP. It is normally used to specify chunked encoding when this is employed.

**Request Headers**

* 1. Accept

tells the server what kinds of content the client is willing to accept, such as image types, office document formats, and so on.

* 1. Accept-Encoding

tells the server what kinds of content encoding the client is willing to accept.

* 1. Authorization

submits credentials to the server for one of the built-in HTTP authentication types.

* 1. Cookie

submits cookies to the server that the server previously issued.

* 1. Host

specifies the hostname that appeared in the full URL being requested.

* 1. If-Modified-Since

specifies when the browser last received the requested resource. If the resource has not changed since that time, the server may instruct the client to use its cached copy, using a response with status code 304.

* 1. If-None-Match

specifi es an entity tag, which is an identifi er denoting the contents of the message body. The browser submits the entity tag that the server issued with the requested resource when it was last received. The server can use the entity tag to determine whether the browser may use its cached copy of the resource.

* 1. Origin

is used in cross-domain Ajax requests to indicate the domain from which the request originated.

* 1. Referer

specifies the URL from which the current request originated.

* 1. User-Agent

provides information about the browser or other client software that generated the request.

**Response Headers**

* 1. Access-Control-Allow-Origin

indicates whether the resource can be retrieved via cross-domain Ajax requests.

* 1. Cache-Control

passes caching directives to the browser.

* 1. Etag

specifies an entity tag. Clients can submit this identifier in future requests for the same resource in the If-None-Match header to notify the server which version of the resource the browser currently holds in its cache.

* 1. Expires

tells the browser for how long the contents of the message body are valid. The browser may use the cached copy of this resource until this time.

* 1. Location

is used in redirection responses (those that have a status code starting with 3) to specify the target of the redirect.

* 1. Pragma

passes caching directives to the browser.

* 1. Server

provides information about the web server software being used.

* 1. Set-Cookie

issues cookies to the browser that it will submit back to the server in subsequent requests.

* 1. WWW-Authenticate

is used in responses that have a 401 status code to provide details on the type(s) of authentication that the server supports.

* 1. X-Frame-Options

indicates whether and how the current response may be loaded within a browser

frame.