Title Screen

Contents

Challenge: GET aHEAD

Description:

* We have been tasked with finding a flag contained within a specific server that has been supplied to us by PicoCTF.
* The URL of this Server is <http://mercury.picoctf.net:28916/>

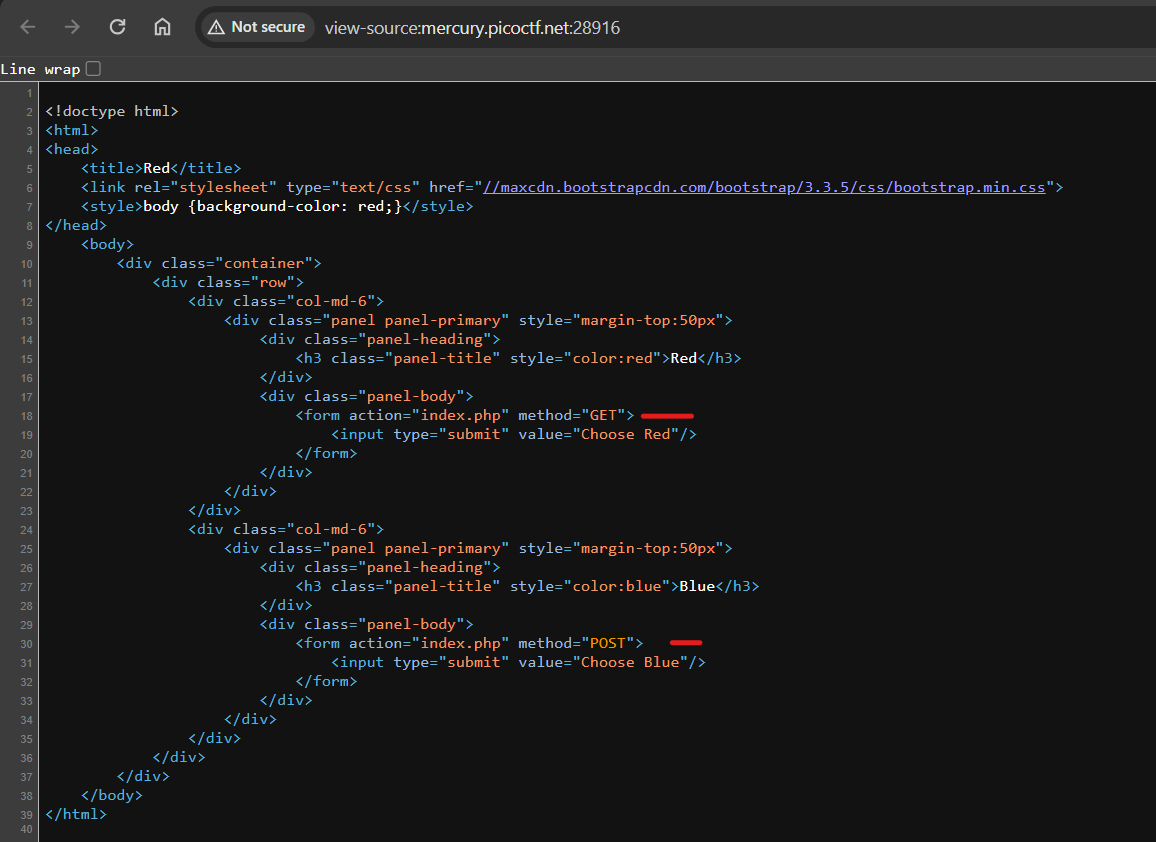
Walkthrough:

* Navigating to the URL provided by PicoCTF directed us to a web page. This page contained two box’s. Within each box contained two buttons, linked to the colours: Red or Blue. When clicking on a button, the background would change to the corresponding colour I.E If the “Choose Blue” button was clicked, the background colour would change to blue.



Source Code Check:

* From reviewing the source code of this web page, I noticed a number of interesting features:
  + **The button logic was different depending on the colour picked** - Looking closely within the source code, we can see that the button that chooses Red does so by sending a “GET” request to index.php, while the button that chooses Blue does so using a “POST” request.



Actions:

* Judging by the title of the challenge, we can assume that there will be data within the HTTP header.
  + Using CURL, we are able to complete a HTTP head request, which is very similar to a GET request, however the server only returns the HTTP headers.
    - Command: curl -I HEAD -i <http://mercury.picoctf.net:53554/index.php>
    - Result: picoCTF{r3j3ct\_th3\_du4l1ty\_2e5ba39f}

Explanation:

* The -i Flag basically means that we want to get the HTTP response headers in the output once we submit this curl command.
* The -I Flag states that we want to specifically send a HEAD request.

Background information:

* What is a HTTP Head Request?
  + HTTP HEAD is one of 9 standard request methods supported by the HTTP protocol. For HEAD requests, the server sends a response identical to the GET request but without the response body. HEAD requests are used to get meta information about a resource, such as the type and size of the resource. The HTTP HEAD method should not change the server state and must be idempotent, which means that multiple identical HEAD requests must have the same effect as a single request.

Further Reading:

* What is and How do we use [cURL](https://blog.hubspot.com/website/curl-command#:~:text=Client%20URL%20(cURL%2C%20pronounced%20%E2%80%9C,send%20to%20that%20server%20URL.)

HTTP in-Depth:

The Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. It is a protocol for fetching resources such as HTML documents. It is a client-server protocol: requests are sent by a client (usually a web browser) to a server, which hosts resources such as HTML files, and the server responds with the content of the requested resource or an error message if unable to do so. Below, I'll provide an in-depth overview of HTTP, including its operations, versions, methods, status codes, and how secure HTTP (HTTPS) works.

Basics of HTTP

Client-Server Model: HTTP follows the client-server model, where a client sends a request to a server, and the server responds. This interaction is stateless, meaning the server does not retain session information between requests.

Stateless Protocol: Each request from any client contains all the information the server needs to fulfill the request. The server does not retain session information, though web applications can implement session management on top of HTTP.

Components of HTTP

* Request: Initiated by the client, a request includes:
  + Method: Indicates the desired action (e.g., GET, POST).
  + URI: Specifies the resource on which to apply the method.
  + HTTP Version: Indicates the HTTP version (e.g., HTTP/1.1).
  + Headers: Provide information about the request or about the client itself.
  + Body: Contains data sent by the client (not present in all requests, such as GET).

Response: Sent by the server, a response includes:

* + Status Line: Includes the protocol version, a status code, and a status message.
  + Headers: Provide information about the server or about the returned resource.
  + Body: Contains the resource requested by the client.

HTTP Methods

* GET: Requests a representation of the specified resource.
* POST: Submits data to be processed to the specified resource.
* PUT: Replaces all current representations of the target resource with the request payload.
* DELETE: Removes the specified resource.
* HEAD: Similar to GET, but without the response body.
* OPTIONS: Describes the communication options for the target resource.
* PATCH: Applies partial modifications to a resource.

Status Codes

* 1xx: Informational responses.
* 2xx: Success (e.g., 200 OK, 201 Created).
* 3xx: Redirection (e.g., 301 Moved Permanently).
* 4xx: Client errors (e.g., 404 Not Found).
* 5xx: Server errors (e.g., 500 Internal Server Error).

HTTP Versions

* HTTP/1.0: The first widely used version, which operates on a single request-per-connection basis.
* HTTP/1.1: Introduced persistent connections, pipelining, and chunked transfer encoding.
* HTTP/2: Provides reduced latency by enabling full request and response multiplexing.
* HTTP/3: The upcoming version, based on QUIC, designed to be more efficient and secure.

Secure HTTP (HTTPS)

* HTTPS is an extension of HTTP that uses SSL/TLS to encrypt the data transmitted between the client and server. This encryption enhances security, particularly important for sensitive transactions.