

Section 1.1.2 Computer Networks Fundamentals

HTTP

{Hypertext Transfer Protocol}

Cracking OSCP: Your Roadmap to Ethical Hacking Success

YouTube: HackProKP – Kailash Parshad

Socials: HackProKP

Github: <https://github.com/at0m-b0mb/Cracking-OSCP-Your-Roadmap-to-Ethical-Hacking-Success>

Complete Youtube Playlist:

<https://www.youtube.com/watch?v=MvkNbn8i2so&list=PLyrv3TPH3ejYNZipa0OIUvkdjHeUTRJ3J&index=1&t=0s>

WHAT IS HTTP?

○ HTTP, which stands for **Hypertext Transfer Protocol**, is the foundation of data communication on the World Wide Web. It is an application layer protocol used for transmitting hypermedia documents, such as HTML. HTTP is the protocol used by web browsers and servers to communicate over the Internet.

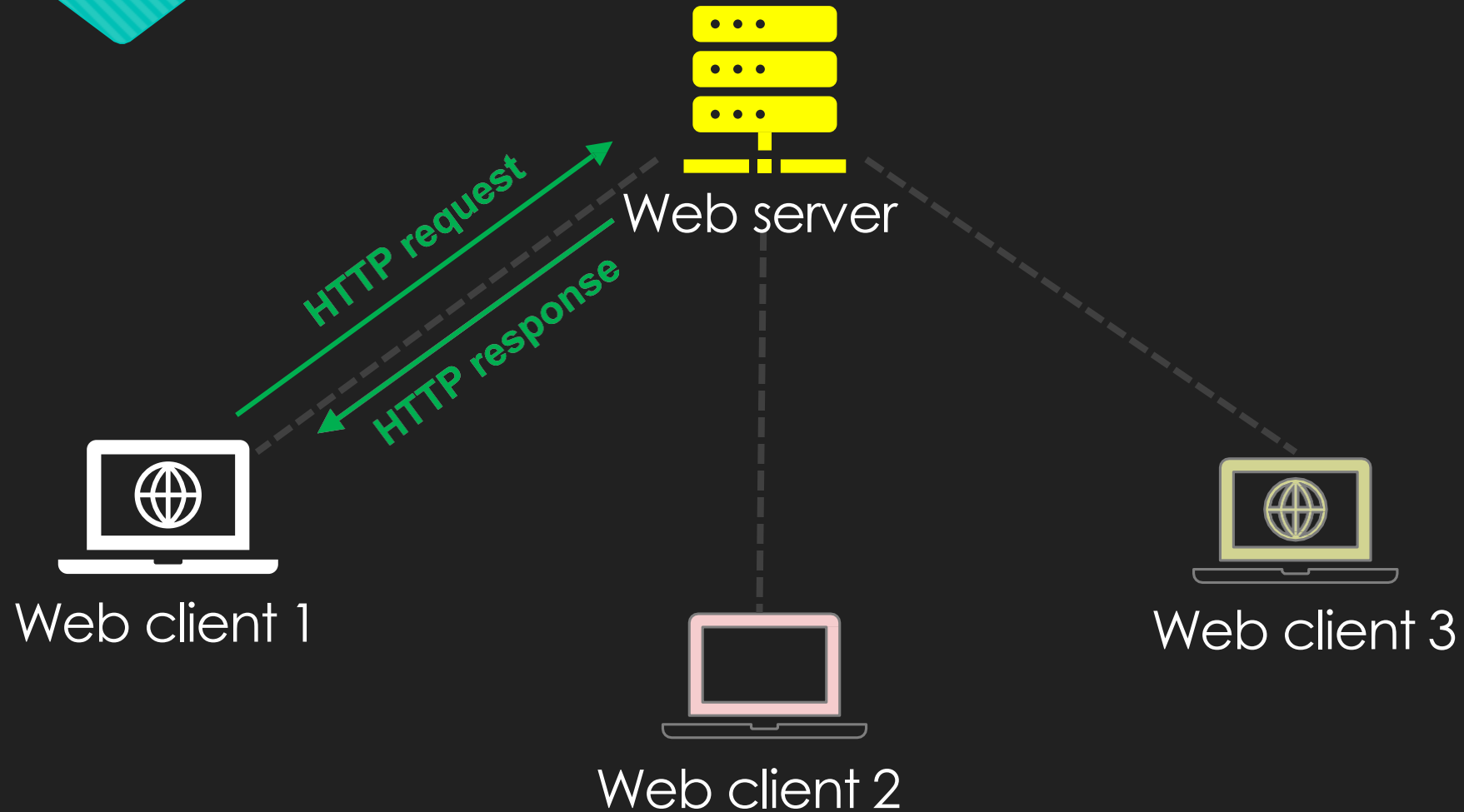
WHAT IS HTTP?

- HTTP is implemented in two programs: a **Client program** and a **Server program**. The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and
- how the client and server exchange the messages.

Client and Server

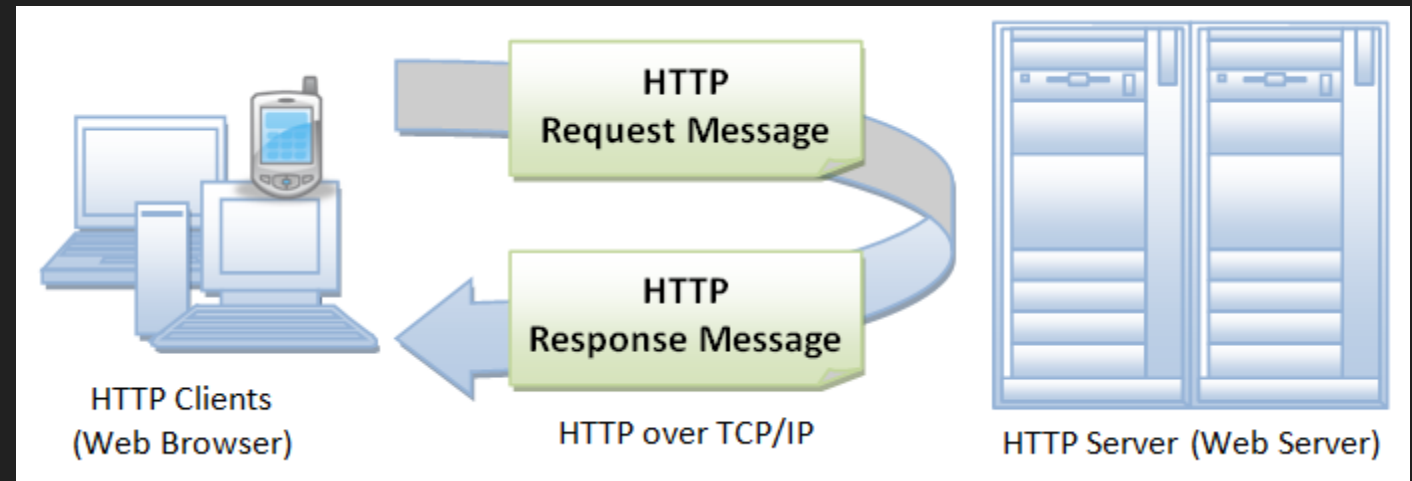
- **Client:** This is typically a web browser or any application that initiates a request to a server.
- **Server:** This is a computer or a system that responds to requests from clients. It hosts resources such as web pages, images, or other content.

CLIENT-SERVER



Request-Response Cycle:

The communication between a client and a server happens in a request-response cycle.



<https://medium.com/geekculture/http-the-request-response-cycle-ddcb5bb70707>

The client sends an HTTP request to the server, and the server responds with the requested data or an error message.

HTTP Methods (Verbs):

- HTTP defines several methods that indicate the desired action to be performed on a resource. Common methods include:
 - **GET**: Retrieve data from the server.
 - **POST**: Submit data to be processed to a specified resource.
 - **PUT**: Update a resource or create a new resource if it doesn't exist.
 - **DELETE**: Request the removal of a resource.
 - **HEAD**: Retrieve only the headers of a resource without the actual data.

HTTP Methods (Verbs):

- **GET:**
 - Used to retrieve data from the server.
 - Parameters are included in the URL.
 - Example:

```
GET /example/path?param1=value1&param2=value2 HTTP/1.1
```


HTTP Methods (Verbs):

- **POST:**
 - Used to submit data to be processed to a specified resource.
 - Parameters are included in the request body.
 - Example:

```
POST /submit/form HTTP/1.1
```

```
Content-Type: application/x-www-form-urlencoded  
param1=value1&param2=value2
```

HTTP Methods (Verbs):

- **PUT:**

- Used to update a resource or create a new resource if it doesn't exist.
- Similar to POST but typically used for updating existing resources.
- Example:

```
PUT /update/resource HTTP/1.1
Content-Type: application/json
{"key": "new_value"}
```

HTTP Methods (Verbs):

- **DELETE:**
 - Used to request the removal of a resource.
 - Often requires additional authentication for security.
 - Example:

```
DELETE /delete/resource HTTP/1.1
```

HTTP Methods (Verbs):

- **HEAD:**

- Similar to GET but only retrieves the headers, not the actual data.
- Useful for checking if a resource has been modified without downloading the entire content.
- Example:

```
HEAD /check/resource HTTP/1.1
```

URL (Uniform Resource Locator):

- URLs are used to identify and locate resources on the web. They consist of a protocol (e.g., "**http**://" or "**https**://"), a domain name, and a path to the resource on the server.

URL (Uniform Resource Locator):

`http://www.example.com/path/resource?param1=value1¶m2=value2`

The diagram shows a URL with four components identified by brackets below them:

- Protocol:** `http`
- Domain/Host:** `www.example.com`
- Path:** `/path/resource`
- Query Parameters:** `?param1=value1¶m2=value2`

- **Protocol:** Specifies the protocol used (e.g., http, https).
- **Domain/Host:** Identifies the server.
- **Path:** Specifies the location of the resource on the server.
- **Query Parameters:** Additional data sent to the server.

Headers

- Both **requests** and **responses** include headers, which provide additional information about the request or the response. Headers can include details like **content type**, **length**, **caching directives**, etc.

Headers

- **Common request headers:**
 - **Host:** Specifies the domain name of the server.
 - **User-Agent:** Identifies the client making the request.
 - **Content-Type:** Specifies the type of data in the request body.
- **Common response headers:**
 - **Content-Type:** Specifies the type of data in the response body.
 - **Status Code:** Indicates the result of the request.

HTTP REQUEST

```
GET /course_x/test.docx HTTP/1.1
Host: www.udemy.com
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Request

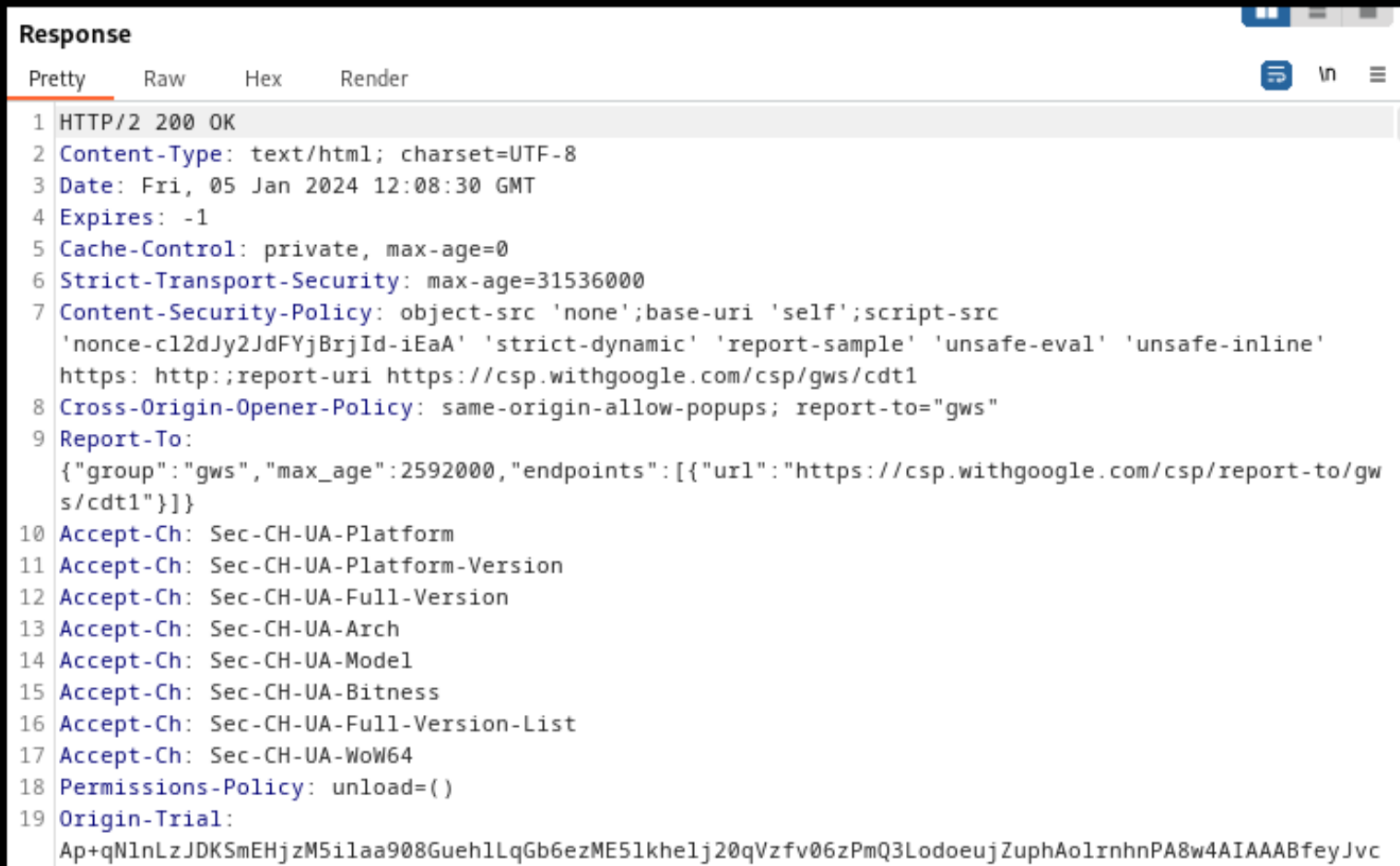
Pretty Raw Hex

```
1 GET /search?q=What+is+HTTP%3F&oq=What+is+HTTP%3F&gs_lcrp=
  EgZjaHJvbWUyBggAEEUY0dIBCTIyNzkzajBqMagCALACAA&sourceid=chrome&ie=UTF-8 HTTP/2
2 Host: www.google.com
3 Cookie: NID=
  511=g8C5ATko6nskPxZ22G0xwSWRyFUy0I0NqiGA2Wh2m1FAbR1kH0f1dT4aNrHe4xXpb09npUrP_0
  [REDACTED]
4 Sec-Ch-Ua:
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: ""
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
  apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 X-Client-Data: CLThygE=
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17
```

HTTP RESPONSE

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Nov 2022 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Nov 2022 15:11:03 GMT
Content-Length: 1088
Content-Type: text/html
```

```
(data data data data data ...)
```



Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Date: Fri, 05 Jan 2024 12:08:30 GMT
4 Expires: -1
5 Cache-Control: private, max-age=0
6 Strict-Transport-Security: max-age=31536000
7 Content-Security-Policy: object-src 'none';base-uri 'self';script-src
  'nonce-cl2dJy2JdFYjBrjId-iEaA' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'
  https: http;;report-uri https://csp.withgoogle.com/csp/gws/cdt1
8 Cross-Origin-Opener-Policy: same-origin-allow-popups; report-to="gws"
9 Report-To:
  {"group":"gws","max_age":2592000,"endpoints":[{"url":"https://csp.withgoogle.com/csp/report-to/gws/
  s/cdt1"}]}
10 Accept-Ch: Sec-CH-UA-Platform
11 Accept-Ch: Sec-CH-UA-Platform-Version
12 Accept-Ch: Sec-CH-UA-Full-Version
13 Accept-Ch: Sec-CH-UA-Arch
14 Accept-Ch: Sec-CH-UA-Model
15 Accept-Ch: Sec-CH-UA-Bitness
16 Accept-Ch: Sec-CH-UA-Full-Version-List
17 Accept-Ch: Sec-CH-UA-WoW64
18 Permissions-Policy: unload=()
19 Origin-Trial:
  Ap+qNlnLzJDKSmEHjzM5ilaa908GuehlLqGb6ezME51khe1j20qVzfV06zPmQ3LodoeujZuphAo1rnhnPA8w4AIAAABfeyJvc
```

Status Codes:

- Status codes indicate the result of the HTTP request.

- Categories:

- **1xx**: Informational.
- **2xx**: Success.
- **3xx**: Redirection.
- **4xx**: Client errors.
- **5xx**: Server errors.

```
HTTP/1.1 200 OK
Content-Type: text/html
```

```
<html>
...
</html>
```

- **200 OK**: The request was successful.
- **404 Not Found**: The requested resource could not be found.
- **500 Internal Server Error**: Something went wrong on the server.

HTTP Versions:

- There are different versions of the HTTP protocol. **HTTP/1.1** and **HTTP/2** are common versions used today. HTTP/2 introduces improvements in performance and efficiency over HTTP/1.1.
- HTTP/1.1 is widely used, but HTTP/2 introduces improvements in performance and efficiency, allowing multiple requests and responses to be multiplexed over a single connection.

Statelessness:

- HTTP is a stateless protocol, meaning each request from a client to a server is independent, and the server doesn't retain information about the client between requests. Sessions and cookies are often used to maintain state.
- HTTP is stateless, meaning each request is **independent**.
- **Sessions** and **cookies** are used to maintain state across multiple requests.

Security:

- HTTPS (Hypertext Transfer Protocol Secure) is a secure version of HTTP that encrypts the data exchanged between the client and server, providing a more secure communication channel.
- HTTPS encrypts the data exchanged between the client and server using **Transport Layer Security (TLS)**, providing a secure communication channel.

Websockets:

- HTTP is traditionally request-response based, WebSockets provide a full-duplex communication channel over a single, long-lived connection, allowing real-time communication between the client and server.

Authentication:

- HTTP supports various authentication mechanisms, such as Basic Authentication and OAuth, to secure access to resources.

Thank You!



Please Like 👍 and Subscribe 📖 to
never miss a video 🎥

And to help me with the Algorithm
🤖

And It's completely Free!!! 💰

Love you guys ❤️