

# HTTP Fundamentals

- HTTP (Hypertext Transfer Protocol) is an application layer protocol for transmitting hypertext documents, such as HTML.
- designed for communication between web browsers and web servers;
- A Client server model and stateless protocol (server doesn't keep any session data between two requests)
- It is an application layer protocol that is sent over TCP or TLS encrypted TCP connection.
- Headers are minimal compared and exchanged between the web server and the web browser that relay all the HTTP messages in the application layer.
- function of protocol headers:
  - > caching
  - > filtering > load balancing
  - > authentication
  - > logging
- why http being stateless causes auth vulnerabilities
- HTTP is stateless, so the server do not retain authentication state between requests so a user's authentication relies on client-side data such as cookies, session IDs or tokens being sent with every request. If these are stolen, reused or manipulated, attackers can impersonate users.
- this statelessness enables vulnerabilities like session hijacking, replay attacks, CSRF & session fixation, man-in-the-middle attacks.

HTTP Header attack surface:	
Header	Attacks
Host	Virtual Host mounting attacks
Cookie	Session Hijacking
Authorization	Token abuse
Origin	CORS exploitation
Referer	CSRF bypass
Content-Type	Validation bypass
User-Agent	Logic flaws / bot detection bypass
X-Forwarded-For	IP Spoofing

Request  
Pretty Raw Hex Requests the root resource [/] using HTTP/2 protocol  
1. GET / HTTP/2 →  
2. Host: www.google.com → http header  
3. Accept: \*/\* → http header  
4. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32; rv:14.0) Gecko/20100101 Firefox/14.0.0 → http header  
5. Accept-Language: en-US,en;q=0.5 → http header  
6. Upgrade-Insecure-Requests: 1 → http header  
7. Sec-Fetch-Dest: document → indicates destination context of request  
8. Sec-Fetch-Mode: navigate → indicates how the page is intended to be used  
9. Sec-Fetch-Site: none → means that the request is not for an external site's resources and was triggered by another site.  
10. Sec-Fetch-User: ?1 → confirms the navigation was initiated by a user action.  
11. Priority: u=0,i → indicates req. importance  
12. T: trailers → indicates client supports HTTP/2 trailer headers.

Cookie Header, sends stored browser cookies to maintain session state.

which content type the client can process.

- HTTP Methods:
  - > GET
  - > POST
  - > PUT
  - > PATCH
  - > DELETE
  - > OPTIONS
  - > HEAD
- Below methods should exist on:
  - > login endpoint - GET, POST
  - > user profile endpoint - GET, PUT, PATCH, POST
- Some frameworks allow overriding HTTP methods using `-method=DELETE` and `x-HTTP-Method-Override` header. This can be used by attackers to send a POST request but force backend to treat it as DELETE/PUT

Safe vs Unsafe HTTP methods	
Safe	Unsafe
• Do not change server state	• Modify server state
• used only for data retrieval	• Must have CSRF protection & auth headers
• GET, HEAD	• POST, PUT, PATCH, DELETE

## Idempotent Vs Non-Idempotent methods

Idempotent	Non-Idempotent
• Multiple identical requests gives same result.	• Each request may cause new state.
• Safe to retry	• Replays are dangerous
• GET, PUT, DELETE	• POST

## HTTP Status Codes

- Categories:
  - 1xx - informational
  - 2xx - Success
  - 3xx - Redirects
  - 4xx - client errors
  - 5xx - Server errors
- Imp codes meaning in security.
  - 200 - Show file failure
  - 202 - Logic flaws
  - 401 - Missing auth
  - 403 - Authorization failure
  - 404 - fake protection
  - 500 - SQLi, RCE hints
- 403 is better for security reasons as it:
  - enforces explicit authentication checks.
  - prevents logic flaws where missing auth.
  - enables proper alerting for unauthorized access attempts
  - Aligns with least privilege b0nA8P guidance.