Web Application Basics

Front End: Surface. Such as a web application or something the user interacts with.

- HTML: Hyper Text Markup Language.
- CSS: Cascading Style Sheets.
- JS: Javascript.

Back End: Inner Core. Things that aren't so visually present.

- DATABASE: Information can be stored, modified and received.
- INFRASTRUCTURE: Web servers, storage, networking devices.
- WAF: Web Application Firewall: Filters out dangerous traffic or malicious harmful things.

1. Which component on a computer is responsible for hosting and delivering content for web applications?
   Web Server
2. Which tool is used to access and interact with web applications?
   Web Browser
3. Which component acts as a protective layer, filtering traffic and blocking malicious attacks?
   Web Application Firewall

Uniform Resource Locator [URL]

Is a web address that allows you to access online content.

The anatomy of a URL is broken up into multiple sections. Such as:
Scheme: Protocol to access the website, HTTP or HTTPS.
User: May include a login query to identify the user.
Host/Domain: Most important part of the URL as it tells you what the destination / web address is.
Port: Port number helps direct the browser to the correct service. It acts as a doorway to the conversation, HTTP = Port 80 HTTPS = Port 443.
Path: Points to the specific file on the page, like a roadmap directing the browser where to go.
Query String: Part of the URL that begins with a "?" often used to search things such as inputs. But can also be used for SQL injections.
Fragment: Starts with a "#" and helps point to a specific section of a webpage.

1. Which protocol provides an encrypted communication to ensure secure data transmission from the web browser and web server?

HTTPS

2. What term describes the practice of registering domain names that are misspelt variations of popular websites to exploit user errors and potentially engage in fraudulent activities?
   Typosquatting
3. What part of the URL is used to pass additional information such as search terms or form inputs to the web server?
   Query String

HTTP Messages

When we talk about HTTP we talk about a message request from an end device to a server somewhere.
Such as an end device: **HTTP Request**
With a server responding: **HTTP Response**

HTTP Is comprised of a few factors such as:
Start Line: Tells you what kind of message is being sent. Such as a request or a response.
Header: Made up of value key-pairs that provide extra message. They give instructions to both the client and the server handling requests. Anything from security, content and types go through this.
Empty Line: Divider that separates the header from the body. It shows where the header stops and then the body of the data.
Body: This is where the actual data is stored. Could be API or web data.
*REMEMBER API = **Application Programming Interface.**

1. Which HTTP message is returned by the web server after processing a clients request?
   HTTP Response
2. What follows the header in a HTTP message?
   Empty Line

HTTP CONTINUED

Request line: This is essentially the beginning of the conversation, tells the server what kind of request it is dealing with. There are 3 parts to the request line:

1. **HTTP Method**
2. **URL Path**
3. **HTTP Version**

HTTP METHODS: Tell the server what kind of actions the user wishes to perform on the resource identified by the URL path.

GET Request: This is used to fetch data from the server without making any changes.
POST Request: Sends the data to the server, to create or update something.
PUT Request: Replaces or updates something on the server.
DELETE Request: Removes something from the server.
PATCH Request: Update part of a resource. Making small changes.
HEAD Request: Works like GET but only retrieves headers.
OPTIONS Request: Tells you what methods are available for a specific resource.
TRACE Request: Similar to options but only shows which method is allowed.
CONNECT Request: Used to create a secure connection, like HTTPS.

URL PATH: Tells the server where to find the resource the user is asking for.
It is crucial to:

- Validate URL path to prevent unauthorized access.
- Sanitise the path to avoid injection attacks.
- Protect sensitive data by conducting privacy and risk assessments.

Following the above will prevent common attacks.

HTTP VERSION: The version shows the protocol version that is used to communicate between the client and the server.

HTTP/0.9 (1991) First version that only supports GET requests.
HTTP/1.0 (1996) Added headers and better support for different content, improved caching.
HTTP/1.1 (1997) Brought persistent connections, chunked transfer encoding.
HTTP/2 (2015) Introduced multiplexing, header compression and faster performance.
HTTP/3 (2022) Built on HTTP2 but uses QUIC for quicker / secure connections.
HTTP/1.1 is still commonly used today however as it is well-supported and works with most existing setups. However upgrading to HTTP/2 or HTTP/3 can introduce better encryption and better security.

1. Which HTTP protocol version became widely adopted and remains one of the most commonly used versions for web communication, known for introducing features like persistent connection chunked faster transfer encoding?
   HTTP/1.1
2. Which HTTP request method describes the communication options for the target resource, allowing clients to determine which HTTP methods are supported by the web server?
   Options
3. In a HTTP request, which component specifies the specific resource or endpoint on the web server that the client is requesting, typically appearing after the domain name in the URL?

URL Path

Request Headers: When we talk about request headers, it allows us to identify more information such as:

Host: Host: tryhackme.com | Specifies name of the webserver it is for.
User-Agent: User-Agent: Mozilla/5.0 | Shares information about the web browser.
Referrer: Referrer: Https://www.google.com | Indicates the URL from which the request came from.
Cookie: Cookie: user_type=student; room=introtowebapplication;room_status=in_progress | Information the web server previously asked the web browser to store.
Content-Type: Content-Type: application/json | Describes type of format of data inside that request.

1. Which HTTP request header specifies the domain name of the web server to which the request is being sent?
   Host
2. What is the default content type for form submissions in a HTTP request where data is being encoded as key=value pairs in a query string format?
   Application/x-www-form-urlencoded
3. Which part of the HTTP request contains additional information like host, user agent, guiding how the web server should process the request?
   Request Headers

HTTP Status and Codes:

HTTP Response = Status line.

- **HTTP Version:** Tells you which HTTP version is being used.
- **Status Code:** Three digit number telling you the outcome of the request.
- **Reason Phrase:** A short message explaining the status code.

Information Responses (100-199) Server has received part of the request and is waiting for the rest. **"Keep Going"** Signal.

Successful Responses (200-299) Everything has worked as expected.

Redirection Messages (300-399) Tells you that the resource you located has moved to a different location.

Client Error Responses (400-499) Indicate a problem with the request. Maybe the URL is wrong or you are missing information such as authentication.

Server Error Responses (500-599) Means the server has encountered an error while trying to fulfil the request.

100 = Continue
200 = OK
301 = Moved permanently
404 = Not Found
500 = Internal Server Error

1. What part of the HTTP response provides the HTTP version, status code, explanation of the responses outcome?
   Status Line
2. Which category of HTTP response code indicates that the web server encountered an internal issue or is unable to fulfil the clients request?
   Server Error Responses
3. Which HTTP status code indicates that the request resource could not be found on the web server?
   404

HTTP Response Headers: Basically key value - pairs. Provide important information about the response and tell the client (browser) how to handle it.

Requires response headers:
Date: Such as the exact date and time the request was made.
Content Type: Tells the client what type of data they are requesting, eg HTML or JSON.
Server: What kind of server software is handling the request.

Other common headers:
Set-Cookie: Sends cookies from the server to the client. Stores them when handling future request.
Cache Control: Tells the client how long it can cache the response before checking again.
Location: Tells the client where to go if the resource has moved.

1. Which HTTP response header can reveal information about the web servers software and version, potentially exposing it to security risks if not removed?
   Server

2. Which flag should be added to cookies in the set-cookie HTTP response header to ensure they are only transmitted over HTTPS, protecting them from being exposed during unencrypted transmissions?
   Secure
3. Which flag should be added to cookies in the set-cookie HTTP response header to prevent them from being accessed via JavaScript, thereby enhancing security against XSS attacks?
   HttpOnly

Security Headers: Security headers help improve overall security of the web application by providing mitigations against attacks like XSS (Cross site scripting).

Content Security Policy (CSP): A CSP header is an additional security layer that can help mitigate common attacks.
An example of such a header is here:
Content-Security-Policy: default-src 'self': script-src 'self' https://cdn.tryhackme.com; style-src 'self'

Defaut-src = Specifies default policy of self.
Script-src = Specifies where scripts can be located from.
Style-src = Specifies policy where CSS style sheets can be loaded from.

Strict-Transport-Security (HSTS): A HTST header ensures that the web browser will always connect over HTTPS.
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

Max-age: Expiry time in seconds.
IncludeSubDomains: Additional setting browser applies to all subdomains.
Preload: Allows website to include all preloaded lists.

X-Content-Type-Options: Can be used to instruct browsers not to guess the MIME time of a resource but only use the content-type-header.
X-Content-Type-Options: nosniff
Nosniff = Instructs browser not to sniff or guess MIME.

Referrer-Policy: Controls the amount of information sent to the destination web server.
Referrer-Policy: no-referrer
Referrer-Policy: same-origin
Referrer-Policy: strict-origin
Referrer-Policy: strict-origin-when-cross-origin

No-referrer = Completely disables any information being sent by the referrer.
Same-Origin = Policy will only send referrer information when the destination is part of the same origin.
Strict-Origin = Policy only sends the referrer as the origin when the protocol stays the same eg HTTPS to HTTPS.
Strict-Origin-When-Cross-Origin = Similar to strict origin except same for origin requests. Where it sends the full URL path in the origin header.

1. In a content Security Policy configuration which property can be set to define where scripts can be loaded from?
   Script-src
2. When confirming the Strict-Transport-Security header to ensure that all subdomains of a site also use HTTPS, which directive should be included to apply the security policy to both the main domain and subdomains?
   IncludeSubDomains
3. Which HTTP header directive is used to prevent browsers from interpreting files as a different MIME type than what is specified by the server, thereby mitigating content type sniffing attacks?
   Nosniff