

## **Introduction web application.**

A web application, also known as a web app, is a software application that runs on a web server and is accessed by users through a web browser. It is designed to provide specific functionality or services to users, similar to traditional desktop applications, but with the advantage of being accessible through a web browser on various devices.

*The basic working principle of a web application involves client-server architecture and the use of standard web technologies.*

The basic components of a web application typically include:

1. *Front-end: This is the part of the web application that the user interacts with, such as the user interface (UI) and user experience (UX) design. It usually consists of **HTML, CSS, and JavaScript**, which are used to create the visual elements and enable user interaction.*
2. *Back-end: This is the part of the web application that runs on the server and is responsible for processing data and generating responses to client requests. It includes server-side scripting languages such as **PHP, Python, or Ruby**, and databases such as **MySQL or PostgreSQL**.*
3. *Web server: The web server is responsible for receiving and responding to HTTP requests from clients. Popular web servers include **Apache, Nginx, and IIS**.*
4. *APIs: Application Programming Interfaces (APIs) allow different software components to communicate with each other. Web applications may use APIs to integrate with other services or to enable third-party developers to build applications that interact with their data.*
5. *Security: Security is a critical component of any web application, as it helps protect user data and prevent unauthorized access. This*

*includes measures such as* **SSL/TLS encryption, secure authentication, and access control.**

How web application work...

1. The user enters the URL for the web application or clicks a link to access it in a web browser.
2. The web browser sends a request to the web server. The request contains information such as the HTTP method (e.g., GET, POST), the requested URL, and any additional data.
3. The web server receives the request and determines the appropriate handler or controller based on the requested URL. This routing process is often managed by a web framework or server-side scripting language.
4. Once the web server identifies the appropriate handler, it processes the request. This may involve fetching data from a database, performing computations, or executing various tasks based on the nature of the request.
5. The web application's business logic handles the processing of the request. It includes tasks such as handling user input, performing calculations, and interacting with databases or external systems.
6. If the web application needs to retrieve or store data, it interacts with a database. User information, application data, and other relevant data are stored and managed in the database. A database management system (DBMS) like MySQL, PostgreSQL, or MongoDB may be used for data operations.
7. Once the server has processed the request and obtained the necessary data, it generates a response. The response typically includes HTML, CSS, JavaScript, and other resources required to render the web page.
8. The web server sends the response back to the user's web browser.
9. The user's web browser receives the response and renders the web page based on the received HTML, CSS, and JavaScript.
10. The user can now interact with the web application by clicking buttons, filling out forms, or performing other actions.

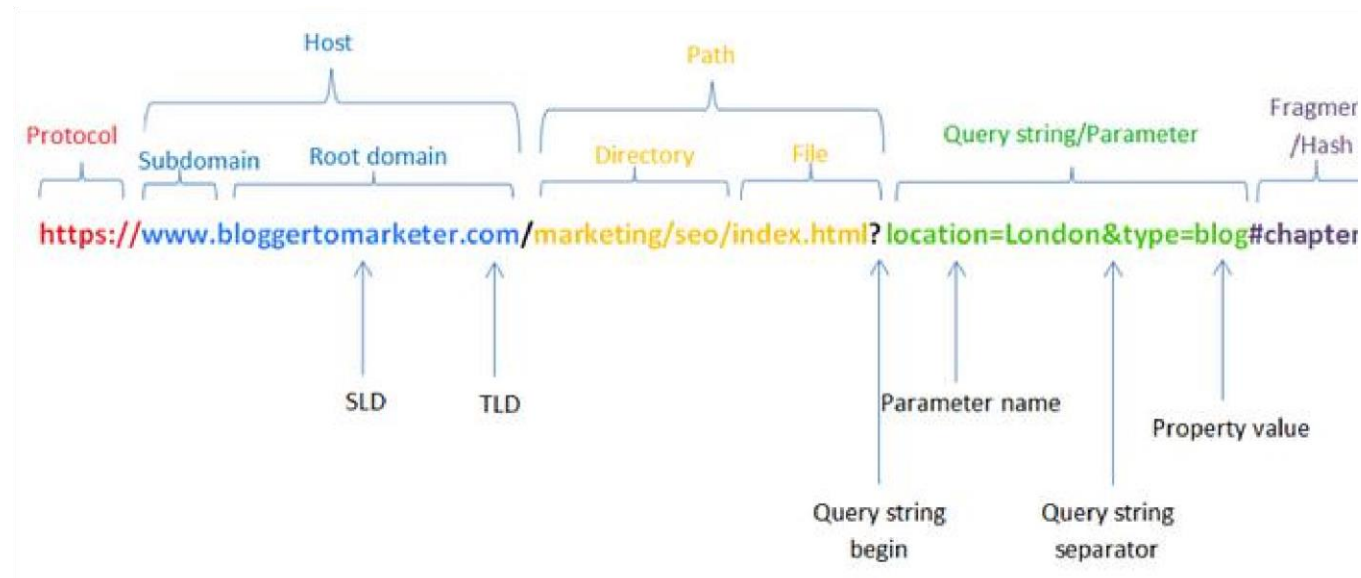
11. User interactions trigger additional requests to the server, initiating a new cycle of processing. These requests follow the same client-server architecture and request-response cycle.
12. Web applications often need to maintain user-specific data or state across multiple requests. Techniques like cookies, sessions, or local storage are used to manage and maintain this state.
13. Web applications can utilize technologies like AJAX or WebSocket to provide real-time updates or enable bidirectional communication between the client and server. AJAX allows for dynamic content updates without requiring a full page reload, while WebSocket facilitates continuous communication for real-time notifications or live data streaming.

Basic communication for client and server.

1. The client, which could be a web browser or a mobile application, *sends a request to the server*. This request is typically sent using the **HTTP (Hypertext Transfer Protocol) protocol**.
2. The request contains information such as the **URL of the requested resource**, **any parameters needed for the request**, and other metadata such as *cookies or authentication tokens*.
3. The *server receives the request and processes it*. This could involve *retrieving data from a database, executing some business logic, or generating a response dynamically based on the request*.
4. The server then *generates a response to the request*. This response contains the requested data or confirmation that the requested action was completed, along with additional metadata such as **status codes, headers, and cookies**.
5. The server sends the response back to the client, again *using the HTTP protocol*.
6. The client receives the response and processes it. This could involve rendering the data on a web page or performing some other action based on the response.
7. If necessary, the client may send additional requests to the server to retrieve more data or perform further actions, starting the process over again.

**There are two ways for a server to get input from a user in a web application.**

**1. Header/URL Parameters:** In this method, input is passed to the server through the URL of the requested web page. This is typically done by appending parameters to the end of the URL, separated by question marks and ampersands. *For example, a URL with parameters might look like this:*



`http://example.com/page?param1=value1&param2=value2`. The server can then extract these parameters from the URL and use them to generate the appropriate response.

## Understanding URLs

**2. Body:** The HTTP body, also known as the message body, is part of an HTTP request or response that carries the actual data being sent or received. It is located after the headers in an HTTP message and is separated from the headers by a blank line.

### HTTP Request and Response Header...

**1. HTTP Request Headers:** HTTP request headers are *sent by the client (typically a web browser or an application) to the server to provide additional information about the request.*

method	path	protocol
GET	/tutorials/other/top-20-mysql-best-practices/	HTTP/1.1
<pre>Host: net.tutsplus.com User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q= Accept-Language: en-us,en;q=0.5 Accept-Encoding: gzip,deflate Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300 Connection: keep-alive Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120 Pragma: no-cache Cache-Control: no-cache</pre>		

### HTTP headers as Name: Value

## HTTP Request Header

*Some commonly used request headers include:*

*Host:* Specifies the hostname of the server the client wants to communicate with.

*User-Agent:* Identifies the client application or browser making the request.

*Accept:* Indicates the media types or content formats that the client can handle or prefers in the response.

*Content-Type:* Specifies the media type of the content in the request body (if any).

*Authorization:* Provides credentials or authentication tokens for accessing protected resources.

*Cookie:* Contains cookies previously stored by the client for the server to identify the user or maintain session state.

**2. HTTP Response Headers:** HTTP response headers are *sent by the server back to the client as part of the HTTP response*. They provide information about the response and additional instructions to the client.

*Some commonly used response headers include:*

*Content-Type*: Specifies the media type or format of the content in the response body.

*Content-Length*: Indicates the size of the response body in bytes.

*Cache-Control*: Provides instructions for caching the response at the client or intermediary servers.

*Set-Cookie*: Sets cookies on the client to be stored for subsequent requests.

*Location*: Redirects the client to a new URL if the response is a redirection.

*Server*: Specifies the software or server name that generated the response.

protocol	status code
HTTP/1.1	200 OK
Transfer-Encoding: chunked	
Date: Sat, 28 Nov 2009 04:36:25 GMT	
Server: LiteSpeed	
Connection: close	
X-Powered-By: W3 Total Cache/0.8	
Pragma: public	
Expires: Sat, 28 Nov 2009 05:36:25 GMT	
Etag: "pub1259380237:gz"	
Cache-Control: max-age=3600, public	
Content-Type: text/html; charset=UTF-8	
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT	
X-Pingback: http://net.tutsplus.com/xmlrpc.php	
Content-Encoding: gzip	
Vary: Accept-Encoding, Cookie, User-Agent	

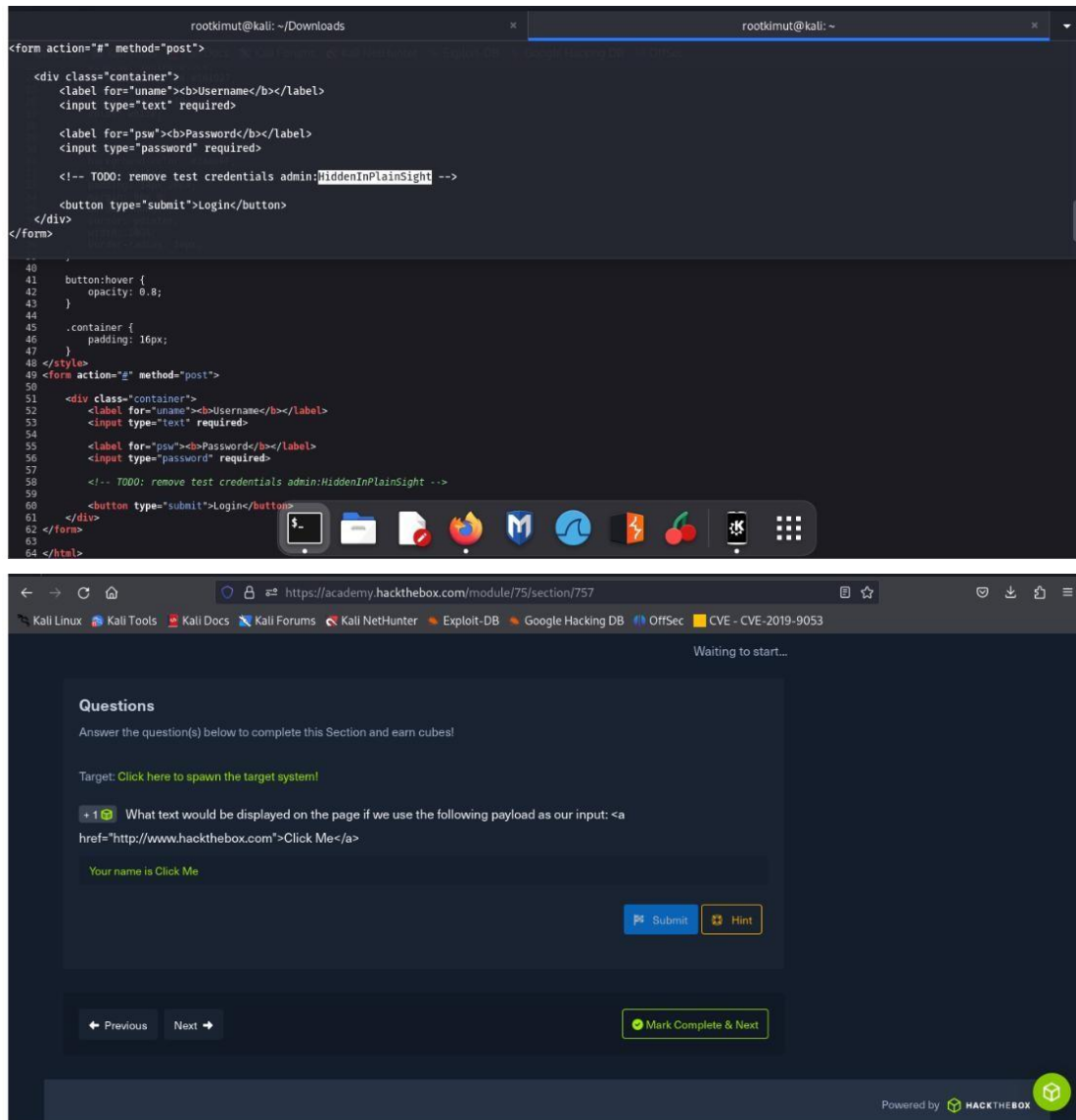
**HTTP headers as Name: Value**

HTTP Response Headers.

HTTP Request and Response Body...

1. **HTTP Request Body**: The HTTP request body is part of an HTTP request where you can *include additional data or information to be sent to the server*. It is used in **HTTP methods** such as **POST**, **PUT**, and **PATCH** to send data to the server for processing or storage. The request body is typically used to send data in structured formats such as *JSON (JavaScript Object Notation)* or *XML (eXtensible Markup Language)*.
2. **HTTP Response Header**: The HTTP response body is the part of an HTTP response *that contains the data or information sent by the server back to the*

*client in response to an HTTP request.* It is used to convey the requested resource, such as HTML content, JSON data, images, or any other type of data.



cs-sa07-24019  
John Mutave

← → ↺ 🏠

🔒 https://academy.hackthebox.com/module/75/section/758

🔖 ☆

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec CVE - CVE-2019-9053

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target: [Click here to spawn the target system!](#)

+ 1 🎁 Try to use XSS to get the cookie value in the above page

XSSisFun

Submit


Hint


← Previous

Next →

🟢 Mark Complete & Next

Powered by

 HACKTHEBOX



The screenshot shows the HackTheBox Academy web interface. At the top, there's a navigation bar with various links like 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', 'OffSec', and 'CVE - CVE-2019-9053'. Below this is a search bar with the text 'Microsoft', 'Office365', 'Skype', 'Stack Overflow', and 'De11'. The main content area has a heading 'Questions' and a subheading 'Answer the question(s) below to complete this Section and earn cubes!'. There is a single question: '+1 🟢 If a web server returns an HTTP code 201, what does it stand for?'. Below the question is a text input field with the placeholder 'Created'. To the right of the input field are two buttons: 'Submit' and 'Hint'. At the bottom of the interface, there are two buttons: 'Previous' and 'Next', and a button 'Mark Complete & Next' with a green checkmark icon. The footer of the page says 'Powered by HACKTHEBOX' with a logo.

The screenshot shows a web browser window with the URL <https://academy.hackthebox.com/module/75/section/761>. The browser's address bar and tabs are visible at the top. The main content area has a dark background. A code block with the title 'Code: php' contains the following PHP code:

```
while($row = $result->fetch_assoc() ){  
    echo $row["name"]. "<br>";  
}
```

Below the code block, a paragraph of text reads: "This basic example shows us how easy it is to utilize databases. However, if not securely coded, database code can lead to a variety of issues, like [SQL Injection vulnerabilities](#)." Further down, a section titled 'Questions' contains the instruction "Answer the question(s) below to complete this Section and earn cubes!". Below this is a question card with a '+1' icon and the text "What type of database is Google's Firebase Database?". The answer "NoSQL" is entered in the text field below the question. At the bottom right of the question card are two buttons: "Submit" and "Hint".



cs-sa07-24019  
John Mutave

The image shows a Kali Linux terminal window and a web browser displaying a HackTheBox Academy module page.

**Terminal Window:**

```
rootkimut@kali: ~/Downloads
rootkimut@kali: ~
rootkimut@kali: ~$ curl -X GET "http://83.136.253.251:36879/index.php?id=1"
superadmin
rootkimut@kali: ~$
```

**Web Browser (Top):**

URL: <https://academy.hackthebox.com/module/75/section/754>

**Questions**

Answer the question(s) below to complete this Section and earn cubes!

Target: 83.136.253.251:36879 🏆

Life Left: 88 minute(s)

+1 🏆 Use GET request '/index.php?id=0' to search for the name of the user with ID number 1?

Submit your answer here...

**Web Browser (Bottom):**

URL: <https://academy.hackthebox.com/module/75/section/764>

We will see these vulnerabilities again and again in our learning journey and real-world assessments. It is important to become familiar with each of these as even a basic understanding of each will give us a leg up in any information security realm. Later modules will cover each of these vulnerabilities in-depth.

**Questions**

Answer the question(s) below to complete this Section and earn cubes!

+1 🏆 To which of the above categories does public vulnerability 'CVE-2014-6271' belongs to?

Command Injection

Submit Hint

Previous Next

Mark Complete & Next

Powered by HACKTHEBOX

Shareable link: <https://academy.hackthebox.com/achievement/1296187/75>