Advanced Web Application Security

Tejas Mhaske

Cyber Security Trainer

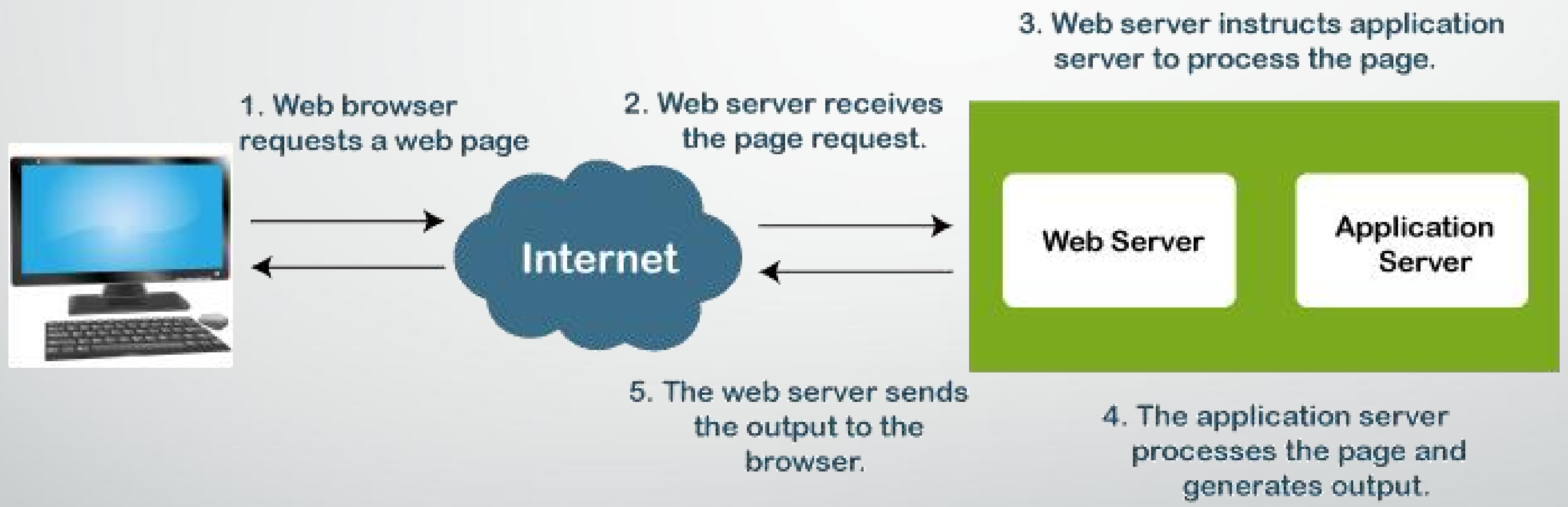
Understanding Web Application

- A web application (web app) is software that is kept on a distant server and distributed via the internet using a browser interface.
- A web-application is an application program that is usually stored on a remote server, and users can access it through the use of Software known as web-browser.
- In general, web applications do not require downloading because, as previously said, they are computer programs that are typically hosted on a remote server. Any user can access it by using a basic web browser such as Google Chrome, Safari, Microsoft Edge, and so on, and the majority of them are free to use.
- A web application is typically programmed in languages that are supported by practically all web browsers, such as HTML and JavaScript, because these languages rely on web browsers to render the program executable.

How Web Applications Works

- In general, a user sends a request to the web-server using web browsers such as Google Chrome, Microsoft Edge, Firefox, etc over the internet.
- Then, the request is forwarded to the appropriate web application server by the web-server.
- Web application server performs the requested operations/ tasks like processing the database, querying the databases; produces the result of the requested data.
- The obtained result is sent to the web-server by the web application server along with the requested data/information or processed data.
- The web server responds to the user with the requested or processed data/information and provides the result to the user's screen .

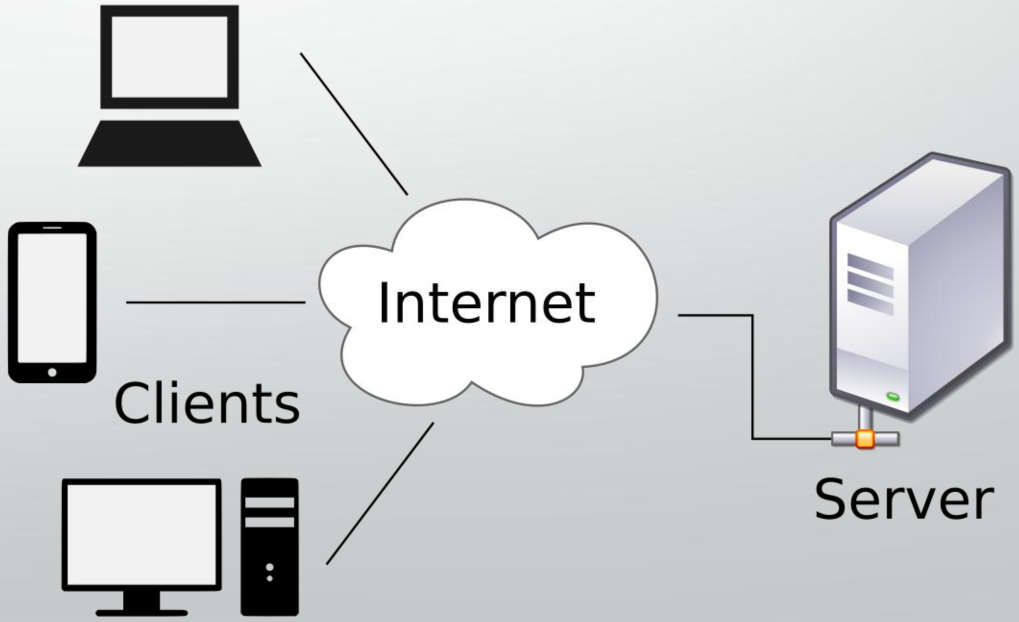
How Web Applications Works



1. Web browser requests a web page
2. Web server receives the page request.
3. Web server instructs application server to process the page.
Internet
Web Server
Application Server
4. The application server processes the page and generates output.
5. The web server sends the output to the browser.

Client-Server Model

- The client-server model describes the communication between two computing entities over a network.



How Clients Server Model Works

- The user enters the URL (Uniform Resource Locator) of the website or file. The Browser then queries the DNS (DOMAIN NAME SYSTEM) server.
- Look up the WEB Server's address using the DNS server.
- The DNS Server answers with the web server's IP address.
- The Browser sends an HTTP/HTTPS request to the WEB Server IP address (given by the DNS server).
- The server sends the website's necessary files.
- The browser then renders the files and displays the website. This rendering is accomplished by the use of the DOM (Document Object Model) interpreter, CSS interpreter, and JS Engine, collectively known as JIT or (Just in Time) Compilers.

DNS

- The Domain Name System (DNS) serves as the Internet's phone book. Humans access information online via domain names such as nytimes.com or espn.com. Web browsers communicate using Internet Protocol (IP) addresses. DNS transforms domain names to IP addresses, allowing browsers to access Internet resources.

How does DNS work?

- DNS resolution is the process of turning a hostname (such as www.example.com) to a computer-friendly IP address (such as 192.168.1.1). Each device on the Internet is assigned an IP address, which is required to locate the relevant Internet device, similar to how a street address is used to locate a specific residence. When a user wishes to load a webpage, there must be a translation between what they type into their web browser (example.com) and the machine-friendly address required to find the example.com webpage.

- To understand the process of DNS resolution, it's vital to learn about the many hardware components that a DNS query must transit through. The web browser performs the DNS lookup "behind the scenes" and does not require any intervention from the user’s computer apart from the initial request.

Web Application Proxy

- Web application proxy servers are servers that facilitate communication between clients and web applications.
- It routes client requests to a web application and returns a response to the client. Organizations employ different servers when using a web application proxy to gain access to web applications in a secure environment.

Web Application Proxy

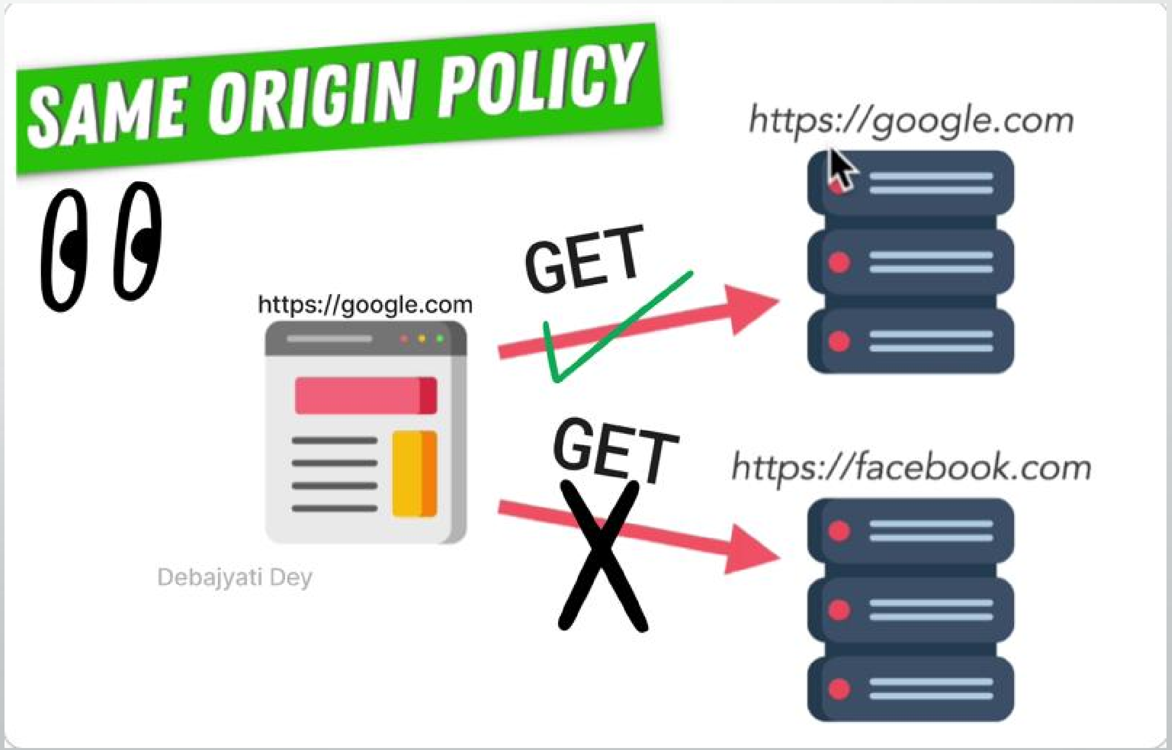
- The online Application Proxy employs Active Directory Federation Services (ADFS) to pre-authenticate access to online applications. This procedure is comparable to how IT administrators use Azure ADFS to authorise access to Azure, Office 365, and other cloud apps.

- Publishing refers to the process of making an application available to other users. When you publish applications through a Web Application Proxy, users can only access the applications that you have published. All of this is accomplished via ADFS, which offers authentication and enforces authorisation for published apps.

Same-Origin Policy (SOP)

- **The Same-Origin Policy (SOP)** is a security feature implemented by web browsers to prevent potentially harmful interactions between different websites.
- **What is "origin"?**

✓ The "origin" of a web page is determined by its protocol (e.g., http or https), domain (e.g., example.com), and port (e.g., :80 or :443). For instance, https://www.example.com:443 has a different origin from http://www.example.com:80.



Same-Origin Policy (SOP)

- **Why is SOP important?**

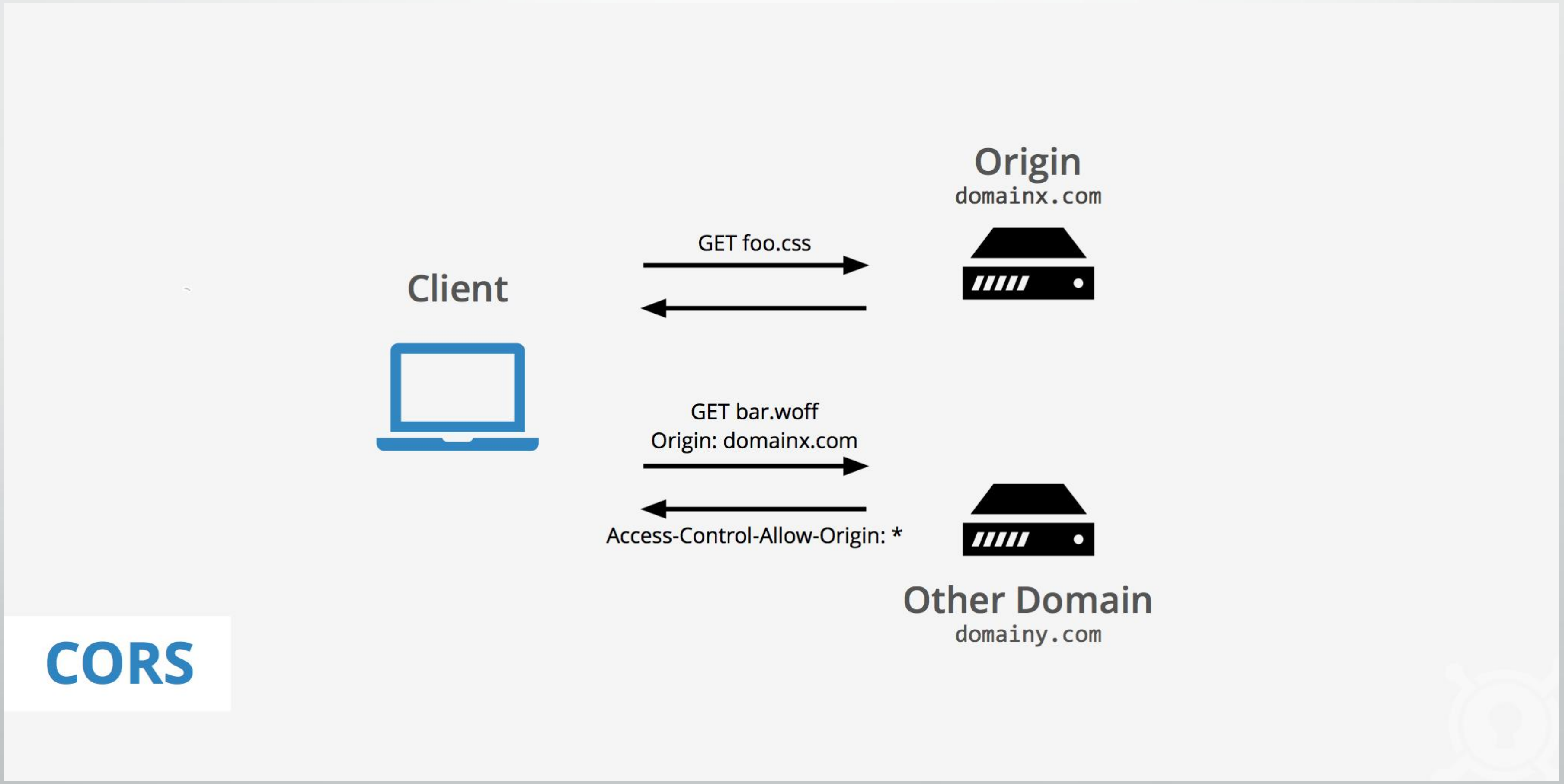
✓ This policy helps protect user data and prevents malicious websites from stealing information from other websites. For example, it stops a malicious site from reading sensitive information from your bank's website if both are open in the same browser.

- **Example**

✓ If you are on https://example.com and it tries to fetch data from https://anotherdomain.com, the browser will block this request due to SOP. However, if anotherdomain.com allows it through mechanisms like Cross-Origin Resource Sharing (CORS), the request can be allowed.

Core Origin policies

- CORS stands for Cross-Origin Resource Sharing. It is a mechanism that uses additional HTTP headers to tell browsers to give a web application running at one origin access to selected resources from a different origin.



Example of CORS

- Suppose you are on https://example.com and your web application needs to fetch data from https://api.anotherdomain.com.
- When your application sends a request to https://api.anotherdomain.com, the server at https://api.anotherdomain.com includes a CORS header in its response, such as Access-Control-Allow-Origin: https://example.com.
- The browser then allows your application to access the response from https://api.anotherdomain.com.

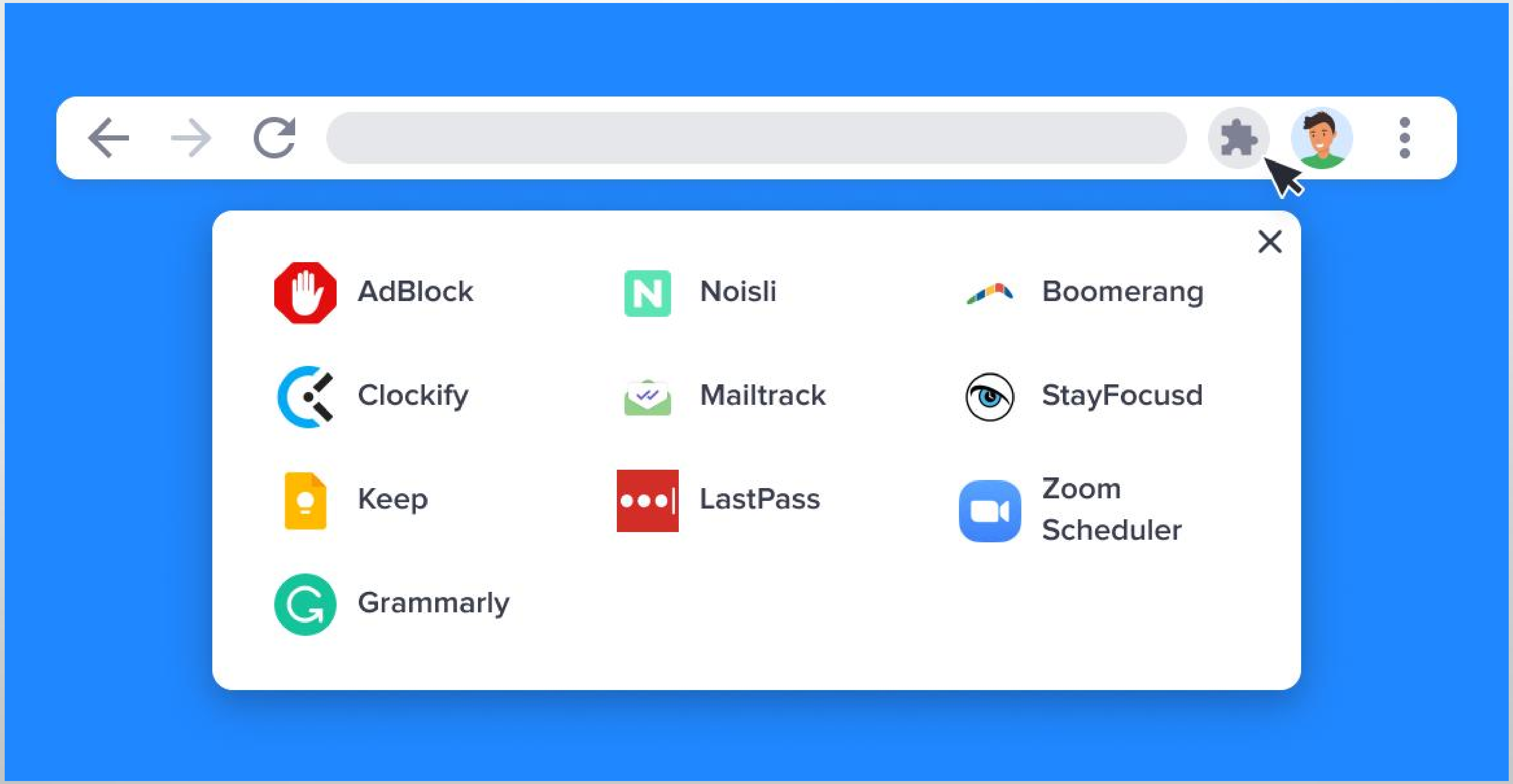
Cookies VS Sessions VS Token

Properties	Cookies	Sessions	Token
Definition	Cookies are small pieces of data stored on the client's device by the web browser. They are used to remember information about the user between sessions.	Sessions refer to a server-side storage of information about a user's interaction with a web application. It persists data across multiple requests from the same user.	Tokens are pieces of data, often encoded as JSON Web Tokens (JWT), used for stateless authentication and authorization.
Data Storage	Stored on the client-side in the browser. Limited in size (typically up to 4KB per cookie).	Stored on the server. The client stores a session ID in a cookie, which is used to retrieve the session data from the server.	Stored on the client-side, usually in local storage, session storage, or cookies. Contains all necessary information within the token itself.

Properties	Cookies	Sessions	Token
Time	Can be set to expire after a specific duration or when the browser session ends. Expiry can range from session-based to years.	Typically lasts for the duration of the user's interaction with the application. Can be configured to expire after a period of inactivity or when the user logs out.	Typically have an expiration time (e.g., 15 minutes to a few hours for access tokens). Can be refreshed using refresh tokens, which have longer expiration times (e.g., days to weeks).
Security	Vulnerable to cross-site scripting (XSS) attacks if not properly secured.	More secure since sensitive data is stored server-side. Vulnerable to session hijacking if session IDs are not properly secured (e.g., by using HTTPS and secure cookies).	Vulnerable to token theft if not properly secured.

Browser Extension

- Small software programs that enhance and customize the functionality of web browsers. They can modify and enhance browser capabilities, integrate with other services, and provide additional features directly within the browser interface.



Google Hacking Database (GHDB):

- Google Hacking Database (GHDB): A repository of search queries designed to find information and vulnerabilities that may be unintentionally exposed on the web using Google's search engine.
- The Google Hacking Database (GHDB) is a collection of search queries, also known as "dorks," that can be used to find information that is not readily accessible through regular searches.
- Example:-

✓ filetype:doc site:example.com confidential

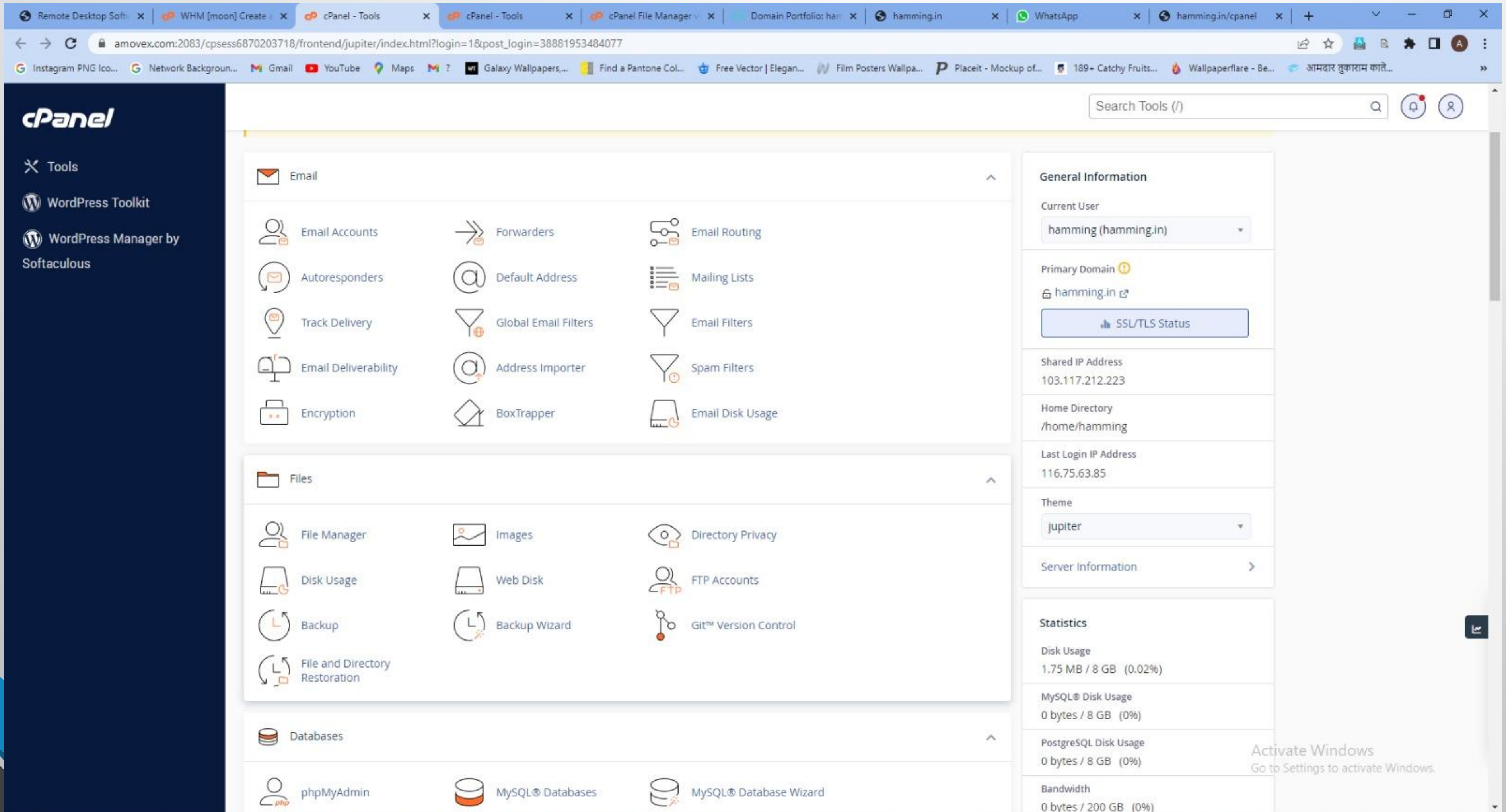
✓ intitle:"Live View / - AXIS(Identifying Security Cameras)

✓ inurl:admin login

✓ intitle:"index of" "ftp(Discovering Vulnerable Servers)

cPanel

- A web hosting control panel software that provides a graphical interface and automation tools to simplify the process of managing a web hosting server. It allows users to manage their websites, email accounts, databases, and other web hosting features easily.



General Information

Current User: hamming (hamming.in)

Primary Domain: hamming.in

Shared IP Address: 103.117.212.223

Home Directory: /home/hamming

Last Login IP Address: 116.75.63.85

Theme: jupiter

Statistics

Disk Usage: 1.75 MB / 8 GB (0.02%)

MySQL® Disk Usage: 0 bytes / 8 GB (0%)

PostgreSQL Disk Usage: 0 bytes / 8 GB (0%)

Bandwidth: 0 bytes / 200 GB (0%)

Functions of cpanel

- **File Management**: Manage website files using the built-in File Manager, FTP accounts, and backup tools.
- **Email Management**: Create and manage email accounts, forwarders, and autoresponders. It also includes tools for filtering spam.
- **Domain Management**: Manage domain names, subdomains, addon domains, and DNS settings.
- **Database Management**: Create and manage MySQL and PostgreSQL databases using phpMyAdmin.
- **Security**: Manage SSL/TLS certificates, configure firewall settings, and set up password-protected directories.
- **Software Management**: Install and manage various software applications, including content management systems (CMS) like WordPress, using Softaculous or other auto-installers.
- **Metrics**: Monitor website performance, bandwidth usage, and access logs.
- **Advanced Tools**: Access to cron jobs, SSH access, and custom error pages.

Web Application VS Cloud Applications

- **Web Application:**
- A web application is software that runs on a web server and can be accessed through a web browser over the internet or an intranet.
- Examples: Gmail, Facebook, online banking systems, and e-commerce sites.
- **Cloud Application:**
- A cloud application is a type of web application that leverages cloud infrastructure for storage, processing, and data management. It can scale efficiently and often uses multiple cloud services.
- Examples: Google Drive, Microsoft Office 365, Dropbox, and Amazon Web Services (AWS).

Future of Web Application

- Pageless Websites
- Blockchain Technology
- Progressive Web Apps
- AI and Chatbots
- Cyber Security
- Motion UI and 3D Elements

धन्यवाद