Introduction

Importance of Logs in DFIR
Logs play a critical role in **Digital Forensics and Incident Response (DFIR)** by providing a **chronological record of system activities**, which helps analysts detect, investigate, and respond to security incidents. Logs serve as digital footprints, allowing forensic teams to reconstruct events, identify attack vectors, and attribute malicious activities to specific users or processes. Without proper logging, detecting **unauthorized access, malware execution, or data exfiltration** would be significantly more challenging. Effective log management enables **real-time threat detection, compliance with security policies, and post-incident forensic investigations**, making it a key component of any **Security Operations Center (SOC)** and forensic analysis process.

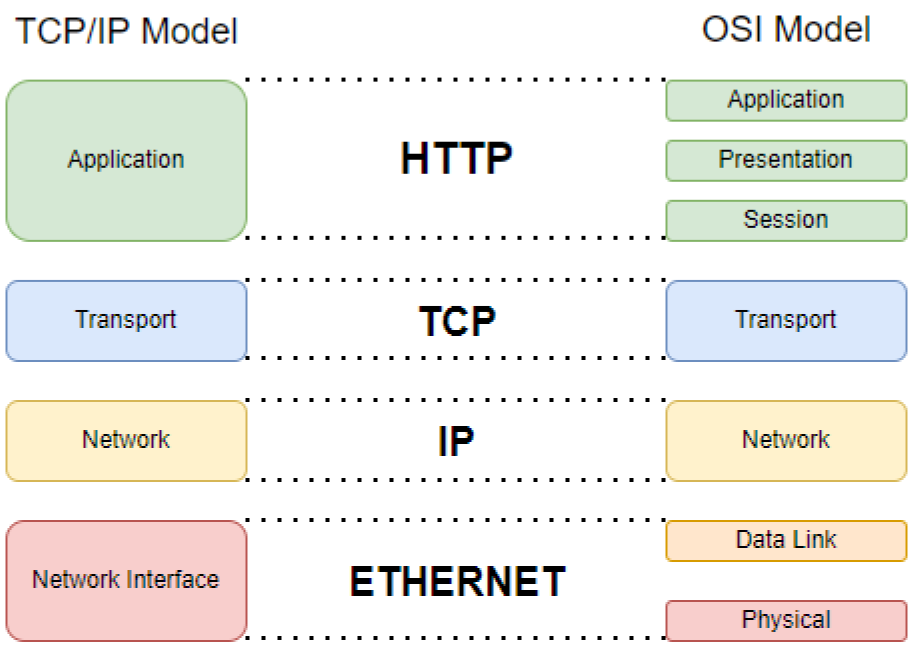
Common Log Types: OS Logs, Web Server Logs, Application Logs
Logs come in various forms, each providing valuable insights into system activity. **Operating System (OS) logs** track user authentication, process execution, system errors, and security-related events, which are essential for detecting unauthorized access or privilege escalation. **Web server logs** record HTTP requests, user interactions, IP addresses, and error codes, helping analysts identify web-based attacks like SQL injection or cross-site scripting (XSS). **Application logs** capture events generated by software applications, including user activity, error messages, and security alerts, which aid in troubleshooting and forensic investigations. Analyzing these logs collectively enhances an organization's ability to **detect anomalies, correlate events, and respond to security threats effectively**.

How Web Applications Work

To identify an anomaly, we should first understand how the technology works. Applications use specific protocols to communicate with each other. In this case, web applications communicate using the Hyper-Text Transfer Protocol (HTTP). Let's take a look at how the HTTP protocol works.

First of all, it's important to know that the HTTP protocol is on layer 7 of the OSI model. This means that protocols such as Ethernet, IP, TCP, and SSL are used before the HTTP protocol.

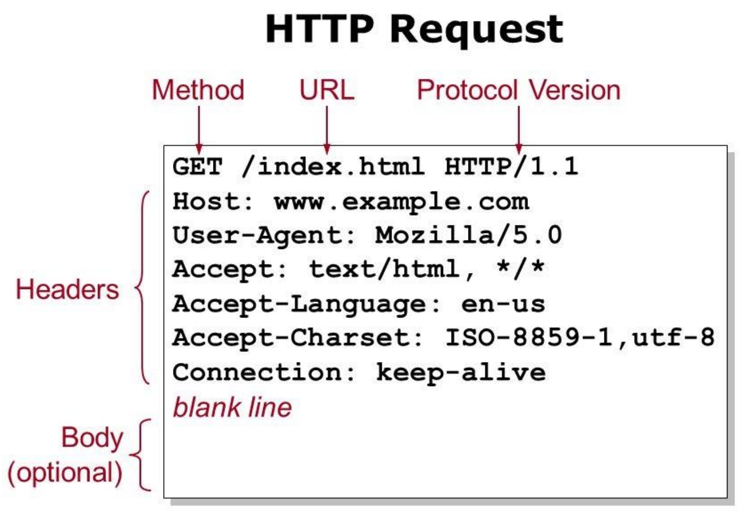
HTTP communication is between the server and the client. First, the client requests a specific resource from the server. The server receives the HTTP request and sends an (HTTP response) back to the client after passing the request through certain controls and processes. The client's device receives the response and displays the requested resource in an appropriate format.



HTTP Requests

An HTTP request is used to retrieve a specific resource from a web server. This resource can be an HTML file, a video, JSON data, etc. The web server's job is to process the response received and present it to the user.

All requests must conform to a standard HTTP format so that web servers can understand the request. If the request is sent in a different format, the web server will not recognize it and will return an error to the user, or the web server may not be able to provide service (which is another type of attack).



```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

An HTTP request consists of a request line, request headers, and a request message body. The request line consists of the HTTP method and the resource requested from the web server. The request headers contain certain headers that the server will process. The request message body contains the data to be sent to the server.

The image above shows an example of an HTTP request. Let's examine this HTTP request line by line.

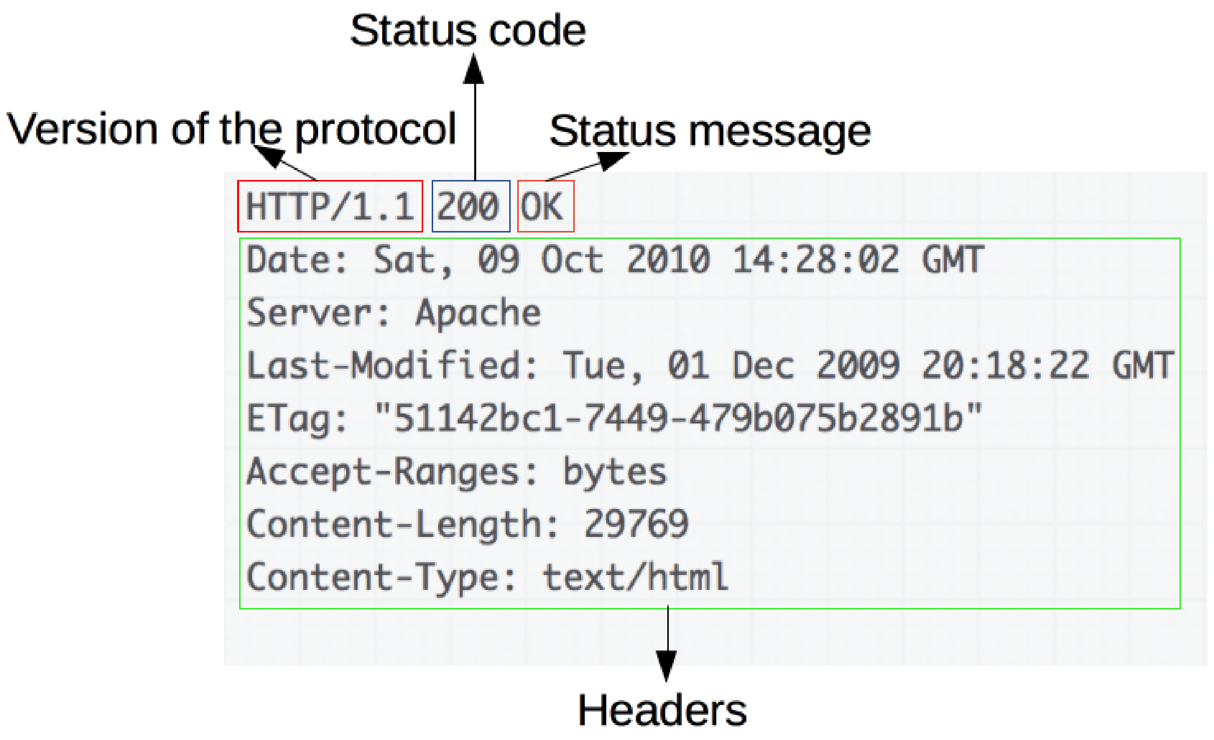
1. The GET method indicates that the resource "/" is being requested from the server. Because there is no name, a symbol like "/" means that the main page of the web server is being requested.
2. Nowadays there are web applications that belong to more than one domain found on a single web server, so browsers use the "Host" header to identify which domain the requested resource belongs to.
3. When a web application wants to store information on the client's device, it stores it in a "cookie" header. Cookies are typically used to store session information. This saves you from having to re-enter your username and password when you visit a web application that requires you to log in.
4. The “Upgrade-Insecure-Requests” header indicates that the client wants to communicate using encryption (SSL).
5. The “User-Agent” header contains information about the client's browser and operating system. Web servers use this information to send specific HTTP responses to the client. You can find some automated vulnerability scanners by looking under this header.
6. The type of data requested is in the “Accept” header.
7. The type of encoding accepted by the client is found in the “Accept-Encoding” header. You can usually find the names of compression algorithms under this header.
8. The “Accept-Language” header contains the client's language information. The web server uses this information to display the prepared content in the client's language.
9. The “Connection” header shows how the HTTP connection is made. If there is data such as "close", it means that the TCP connection will be closed after receiving the HTTP response. If you see "keep-alive", this means that the connection will be maintained.
10. An empty line is inserted between the HTTP request header and the HTTP request message body to create a partition.
11. Any other data to be sent to the web application is in the Request Message Body. If the HTTP POST method is used, then the POST parameters can be found here.

HTTP Responses

When the web server receives an HTTP request, it performs the necessary checks and processes and then sends the requested resource to the client. There is no standard process, as there are many technologies and designs involved. The server may pull data from the database depending on what the requested resource is, or it may process the incoming data. However, the HTTP Response Message must reach the client after all the processing.

An HTTP response message contains a Status Line, Response Headers, and a Response Body. The Status line contains the status code (e.g. 200: OK) and HTTP protocol information. Within the Response Header, some headers are used for a variety of purposes. The Response Body contains information about the requested resource.

If a web page has been requested, there will usually be HTML code in the Response Body. When the client receives the HTML code, the web browser will process the HTML code and display the web page.



```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```