

## 4.1

### The Need for State:

HTTP is stateless, but because of request/response nature, we need a state to hold the state between the client and the server. Earlier, there was static pages, which anyone could view.

Now, a person has to login through authentication, the site remembers preferences, dynamically generated site. This is done through means of session. Without a session, you would have to login on every page. When a session is created, the user is authenticated and all requests are sent to the server as if the user has been authenticated.

## 4.2

### Weakness in Token Generation

#### Meaningful Tokens

#### Predictable Tokens

#### Encrypted Tokens (ECB)

## 4.3

### Weaknesses in Session Token Handling:

* Disclosure of tokens on network
* Disclosure of tokens in logs
* Vulnerable session termination
* Client exposure to token hijacking

### Securing Session Management

* Generate strong tokens
* Protect tokens throughout lifecycle
* Log, Monitor, Alert

Difference between cookie and session

| **Cookie** | **Session** |
| --- | --- |
| Cookies are client-side files on a local computer that hold user information. | Sessions are server-side files that contain user data. |
| Cookies end on the lifetime set by the user. | When the user quits the browser or logs out of the programmed, the session is over. |
| It can only store a certain amount of info. | It can hold an indefinite quantity of data. |
| The browser’s cookies have a maximum capacity of 4 KB. | We can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time. |
| Because cookies are kept on the local computer, we don’t need to run a function to start them. | To begin the session, we must use the session start() method. |