Web Application Security

Dr. Digvijaysinh Rathod
Associate Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in

Session

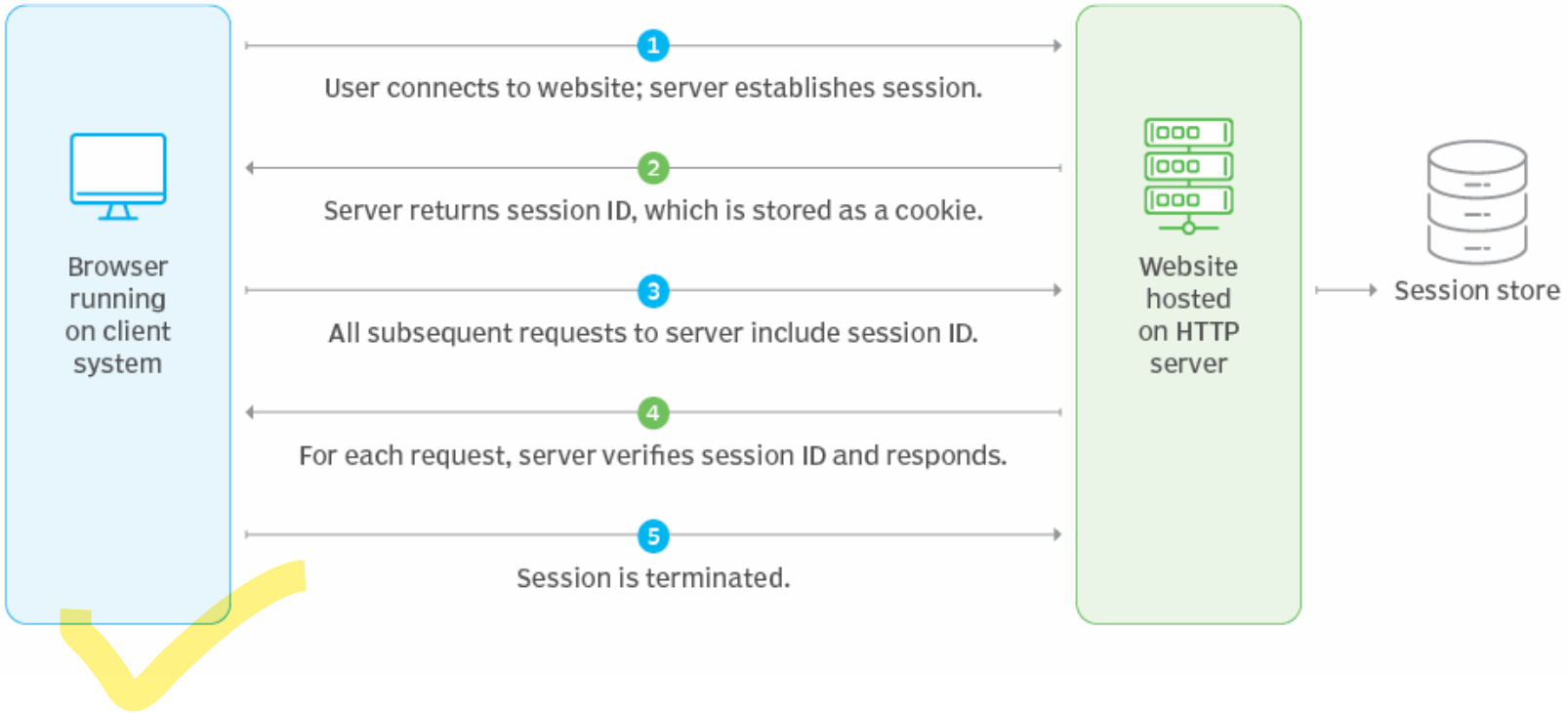
- ✓ HTTP is a “stateless” protocol.
- ✓ Which means there is no “built-in” standard to keep track of interrelated requests. **Each request is treated as independent.**
- ✓ These web applications are very advanced and usually handle complex operations which take **more than one pair of request/response to complete**, this requires something to track the present state of operation

- ✓ These apps also present tailored content to each user.
- ✓ **This requires identifying a user across multiple requests.**
- ✓ So there must be something that adds an **“enhancement” to HTTP.**

- ✓ There are solutions that are the results of clever thinking by web programmers.
- ✓ **HTTP uses client-server architecture and uses TCP as its transmission protocol and multiple requests can be sent over just one TCP connection,**
- ✓ but these are also considered independent by both client and server.

- ✓ There are solutions that are the results of clever thinking by web programmers.
- ✓ **HTTP uses client-server architecture and uses TCP as its transmission protocol and multiple requests can be sent over just one TCP connection,**
- ✓ but these are also considered independent by both client and server.

How session IDs are assigned



Session

- ✓ **HttpSession.** Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
- ✓ An HTTP session is a mechanism for maintaining stateful communication between a web server and a client over the stateless HTTP protocol.
- ✓ In a stateless protocol like HTTP, each request from a client to the server is independent, meaning the server has no inherent knowledge of previous requests.

- ✓ However, HTTP sessions enable the server to associate multiple requests from the same client as part of a logical session, allowing for the retention of user-specific data across requests.
- ✓ **Key Components of an HTTP Session:**

  1. Session Identifier: A unique identifier assigned to each session, typically in the form of a session ID or token. This identifier is used to associate subsequent requests with the correct session data.

Session

- ✓ **Session Data:** Information specific to a particular session, such as user authentication credentials, preferences, or items in a shopping cart.
- ✓ **Session Management:** Mechanisms for creating, maintaining, and destroying sessions. This includes techniques for session initialization, expiration, and storage.

✓ Implementation of HTTP Sessions:

HTTP sessions can be implemented using various techniques, including:

1. Cookies: One of the most common methods, where a unique session identifier is stored in a cookie on the client-side. Subsequent requests from the client include this cookie, allowing the server to identify the session.

2. URL Rewriting: Session identifiers are appended to URLs as query parameters, enabling the server to track sessions.

- ✓ **Hidden Form Fields:** Session identifiers can be embedded within HTML forms as hidden fields, allowing for session tracking during form submissions.
- ✓ **Server-Side Storage:** Session data can be stored on the server-side, either in-memory, in a database, or through other persistent storage mechanisms.

Security Considerations:

- ✓ While HTTP sessions facilitate stateful communication, they also introduce security considerations:
- ✓ **Session Hijacking:** Attackers may attempt to steal session identifiers to impersonate legitimate users. Implementing secure session management techniques, such as using HTTPS and employing secure cookies, helps mitigate this risk.
- ✓ **Session Expiry:** Properly managing session expiration helps minimize the window of opportunity for attackers to exploit inactive sessions.

Security Considerations:

- ✓ **Session Fixation:** Attackers may attempt to fixate a user's session identifier, allowing them to hijack the session once the user authenticates. Techniques like session regeneration after authentication can help prevent this.
- ✓ **Cross-Site Scripting (XSS):** XSS vulnerabilities can be exploited to steal session identifiers or manipulate session data. Sanitizing user input and implementing proper output encoding can mitigate this risk.

Mobile Phone Security

Dr. Digvijaysinh Rathod
Associate Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in