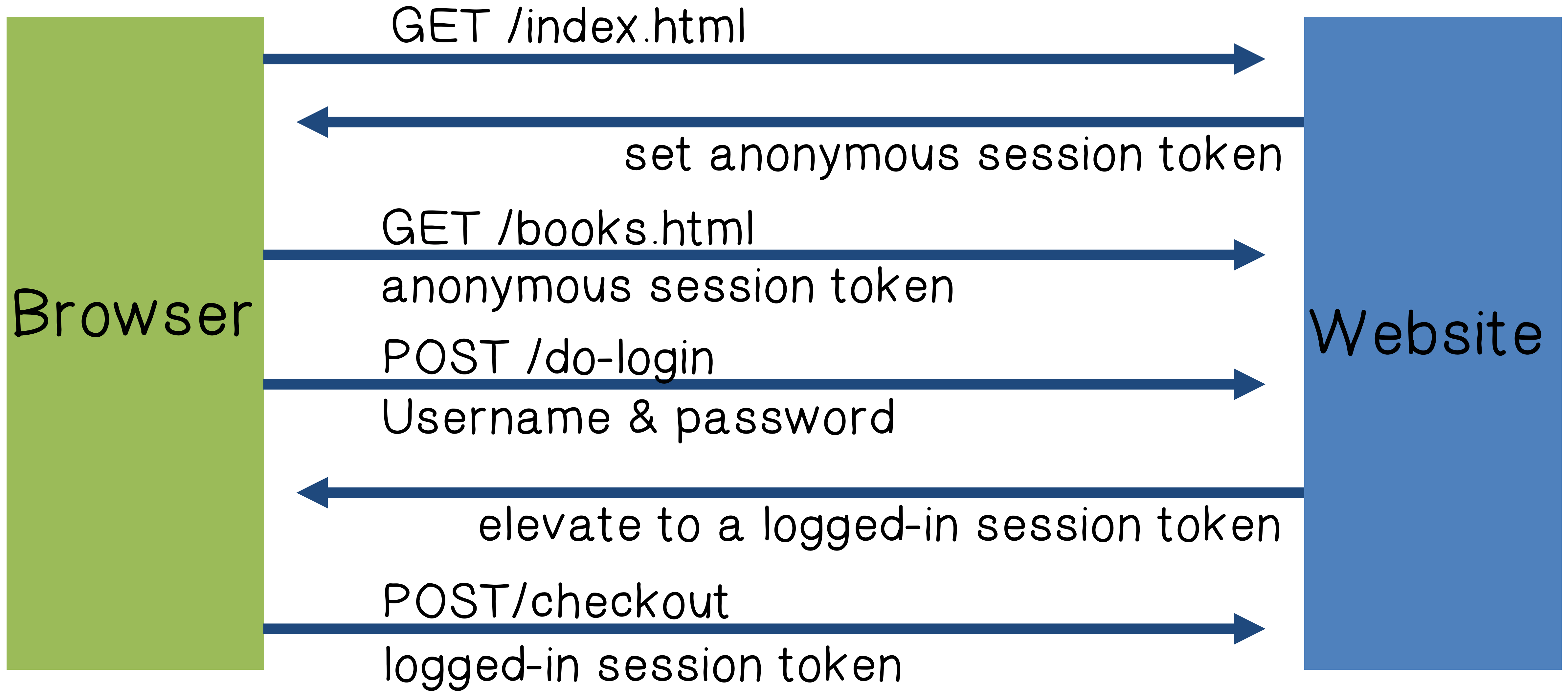
Sessions

- A sequence of requests and responses from one browser to one (or more) sites
  - Session can be long (e.g., Gmail) or short
  - Without session management, users would constantly re-authenticate
- Session management: authorize user once; all subsequent requests are tied to user

Session Tokens



Storing Session Tokens

Browser cookie:

- Set-Cookie: SessionToken=fduhye63sfdb

Embed in all URL links:

- https://site.com/checkout ? SessionToken=kh7y3b

In a hidden form field:

- <input type="hidden" name="sessionid" value="kh7y3b">

Storing Session Tokens

Best Method: a combination of all 3:

- Browser cookie, embed in URL, hidden form field

The HTTP Referer Header

Shows the page you are coming from- your referer

```
Host      slogout.espncricinfo.com
User-Agent  Mozilla/5.0 (Windows NT  6.1; rv:5.0) Gecko/20100101
Firefox/5.0
Accept  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
Connection keep-alive
Referer http://slogout.espncricinfo.com/index.php?page=index.php?page
=index&level=login
```

The HTTP Referer Header

Problem:

Referer leaks URL session token to 3rd parties

Solution: Referer Suppression

not sent when HTTP site refers to an HTTP site

in HTML5: <a rel="noreferrer" href=www.example.com>

Session Token Security- Logout Procedure

Web sites must provide a logout function:

Functionality

- Let user login as different user.

Security

- Prevent others from abusing content

Session Token Security- Logout Procedure

What happens during a logout:

1. Delete SessionToken from client
2. Mark session token as expired on server

! Problem: Many web sites do 1 but not 2!

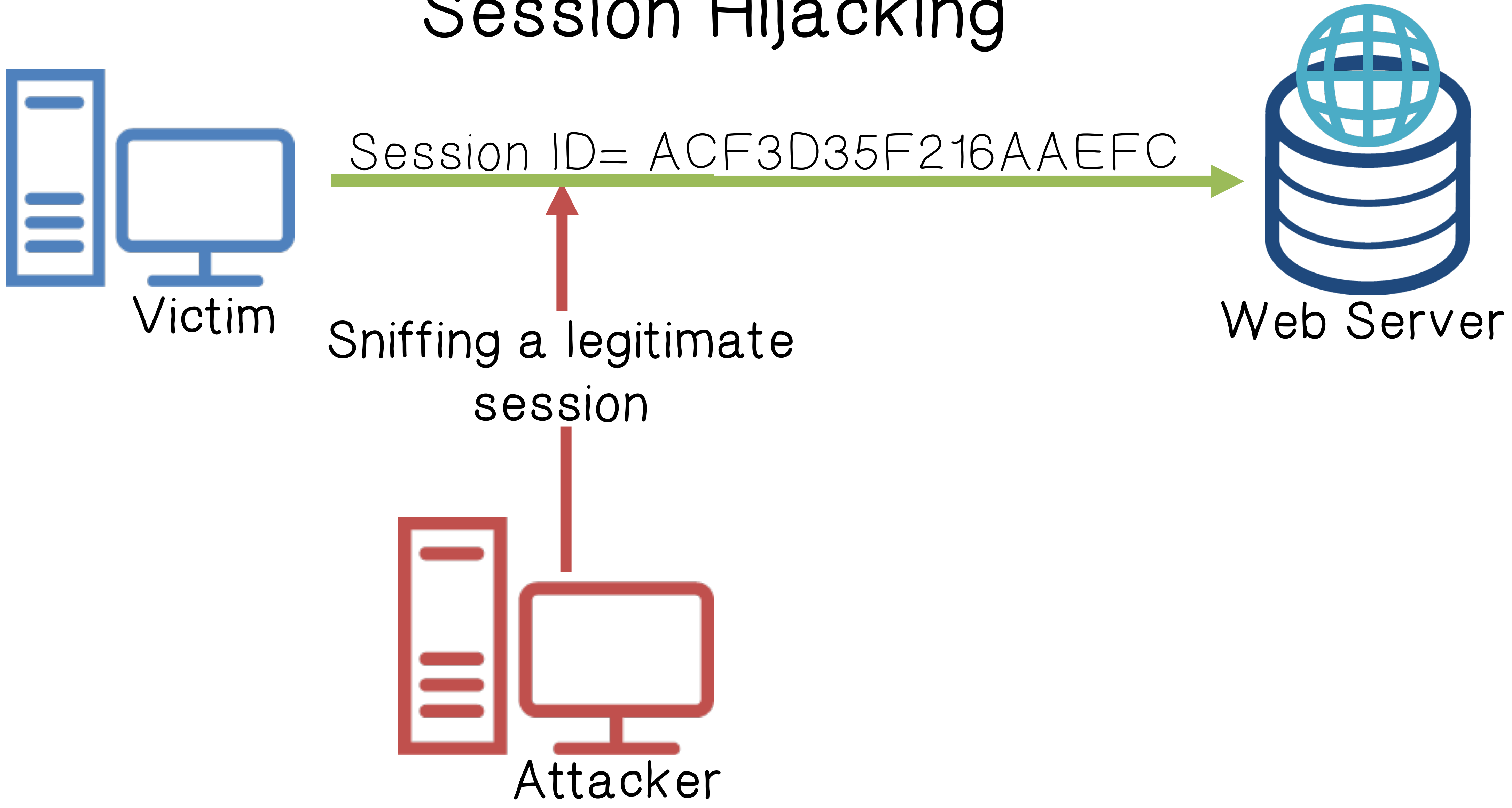
Especially risky for sites who fall back to HTTP after login

Session Token Quiz Solution

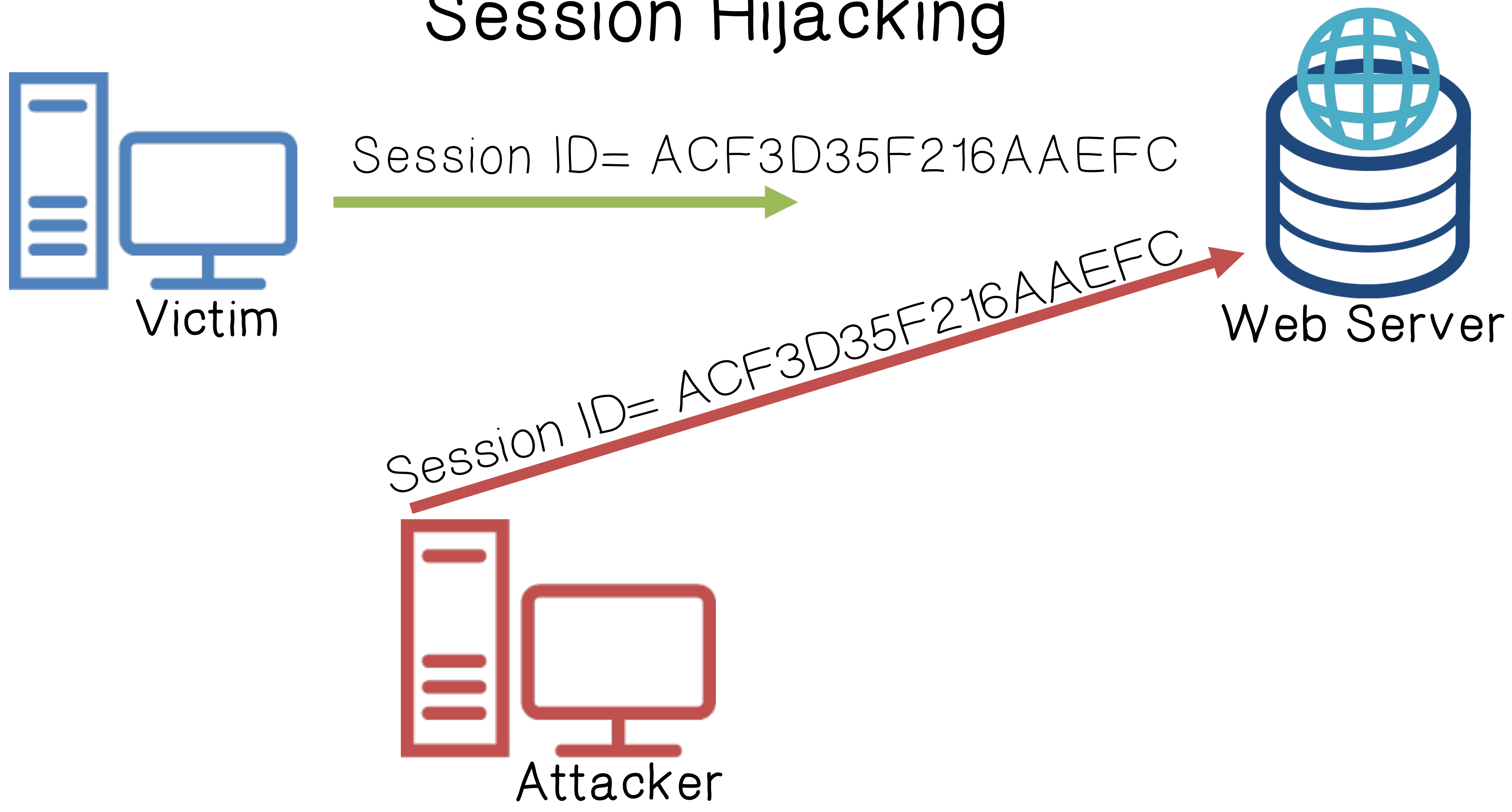
Check all the statements that are true:

- [✓] The token must be stored somewhere
- [✓] Tokens expire, but there should still be mechanisms to revoke them if necessary
- [ ] Token size, like cookie size, is not a concern

Session Hijacking

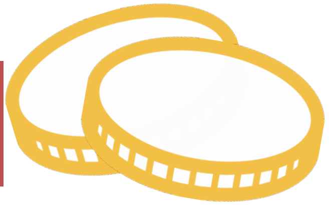


Session Hijacking



Session Hijacking

Beware of predictable tokens!



Example 1: Counter:

User logs in, gets counter value, can view sessions of other users

Example 2: Weak MAC token:

Weak MAC exposes secret key from a few cookies, gets counter value, can view sessions of other users

Apache Tomcat: generateSessionId()

Returns random session ID

[server retrieves client state based on session-id]

Session Hijacking

Session tokens must be unpredictable to attacker

To generate: Use underlying framework
(e.g., ASP, Tomcat, Rails)

Rails: Token=MD5(current time, random nonce)

Session Token Theft

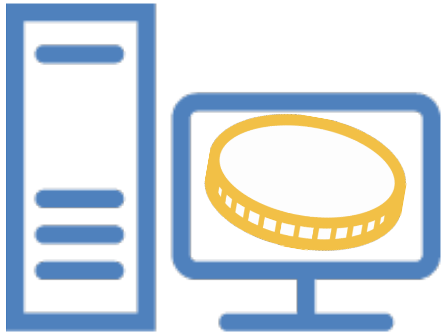
- Example 1: Login over HTTPS, but subsequent HTTP
  - Enables cookie theft at wireless café (e.g., Firesheep)
  - Other ways network attacker can steal token:
    - Site has mixed HTTPS/HTTP pages- token sent over HTTP
    - Man-in-the-middle attacks on SSL

Session Token Theft

- Example 2: Cross Site Scripting (XSS) exploits:
  - Amplified by poor logout procedures
    - Logout must invalidate token on server

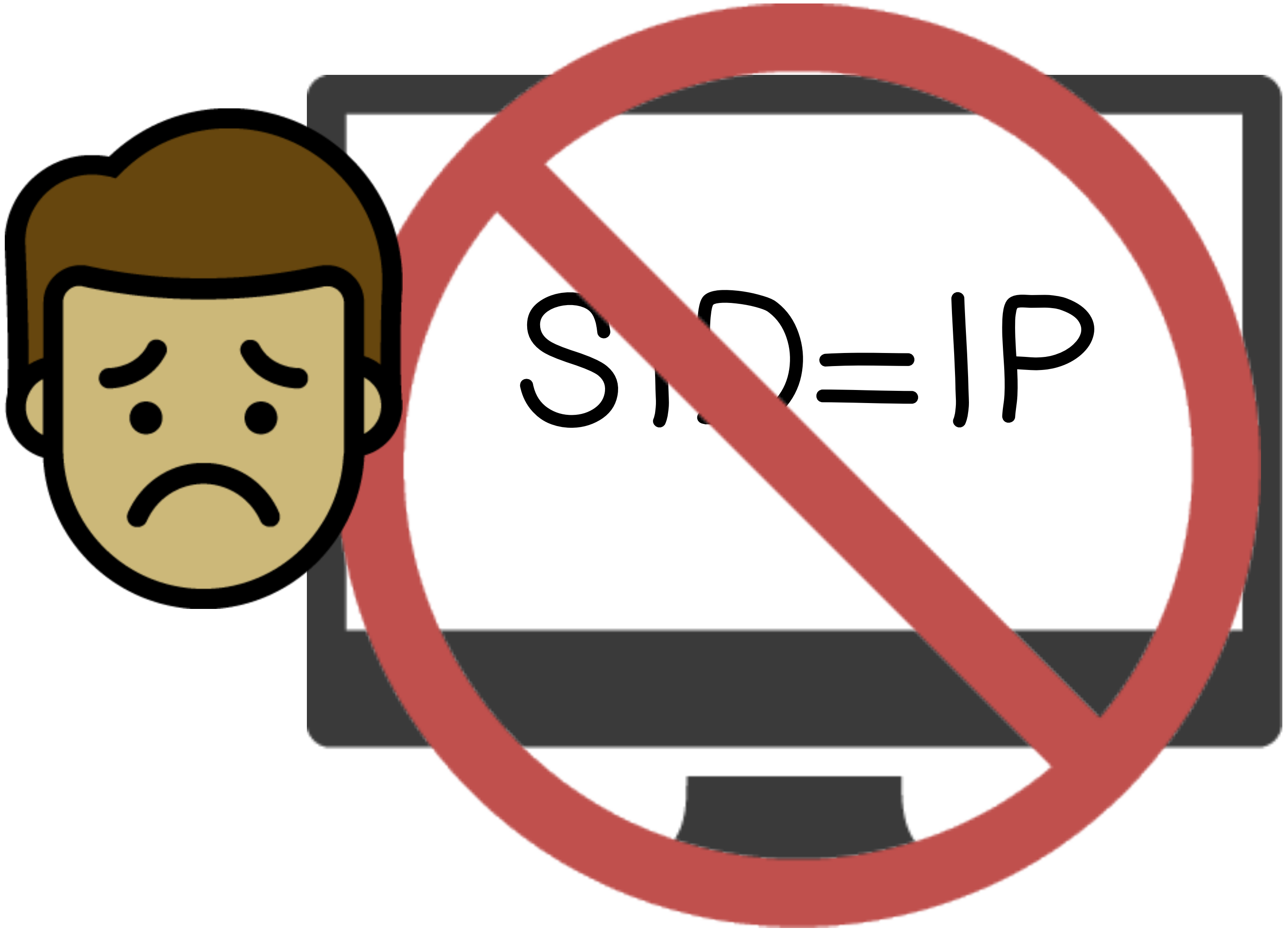
Session Hijacking

Binding SessionToken to client's computer

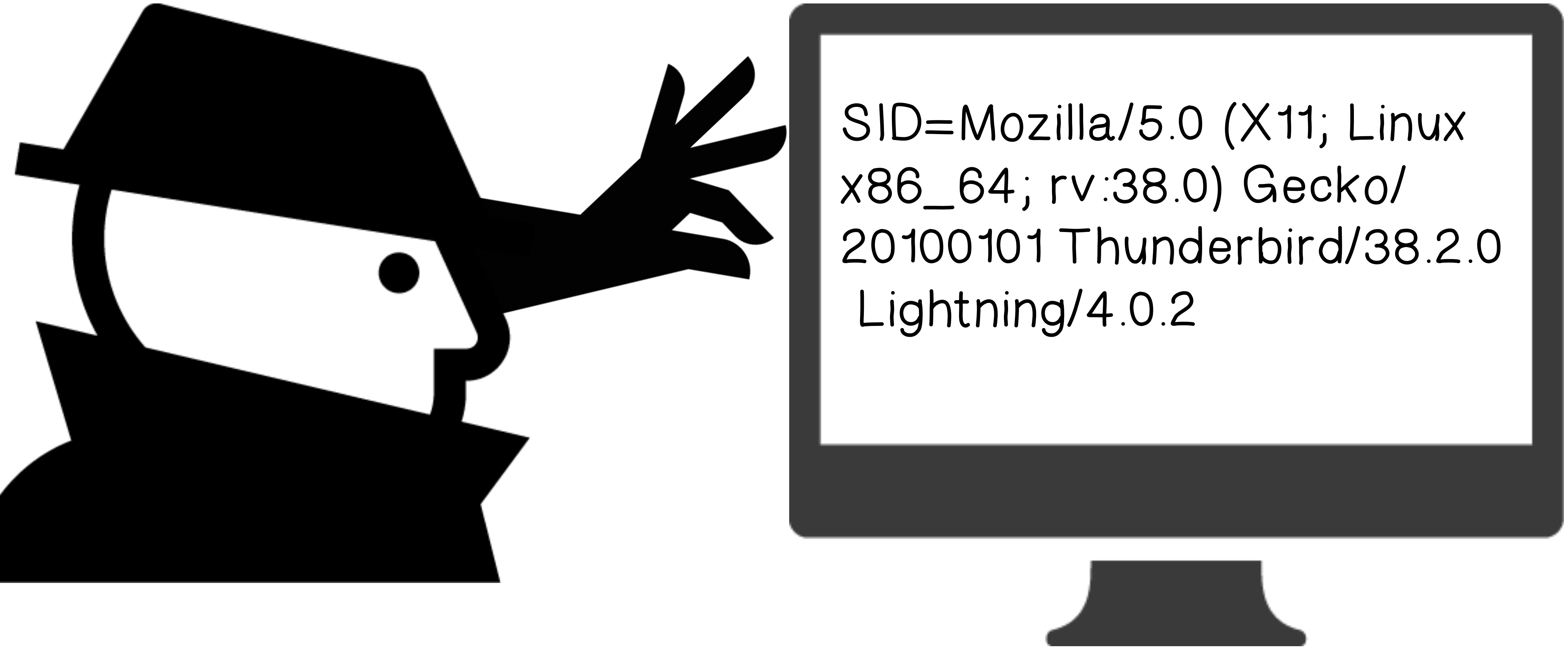


☞ A common idea: embed machine specific data in SID

Session Hijacking



Session Hijacking



SID=Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Thunderbird/38.2.0 Lightning/4.0.2

Session Fixation Attacks

☞ Suppose attacker can set the user's session token:

For URL token, trick user into clicking URL

For cookie tokens, set using XSS exploits

Session Fixation Attacks

Attack: (say, using URL tokens)

1. Attacker gets anonymous session token for site.com
2. Sends URL to user with attacker's session token
3. User clicks on URL and logs into site.com
4. Attacker uses elevated token to hijack user's session

Session Fixation: Lesson

When elevating user from anonymous to logged-in:

☞ always issue a new session token

After login, token changes to value unknown to attacker

- Attacker's token is not elevated

Session Hijacking Quiz Solution

Check all the statements that are true:

- [✓] Active session hijacking involves disconnecting the user from the server once that user is logged on. Social engineering is required to perform this type of hijacking.
- [ ] In Passive session hijacking the attacker silently captures the credentials of a user. Social engineering is required to perform this type of hijacking.

Session Management Summary

Always assume cookie data retrieved from client is adversarial

Session tokens are split across multiple client state mechanisms.

- Cookies, hidden form fields, URL parameters
- Cookies by themselves are insecure (CSRF, cookie overwrite)
- Session tokens must be unpredictable and resist theft

Ensure logout invalidates session on server