State Management

Christian Wenz

@chwenz

Demo

Know the cookie, hijack the session

Cookies are sent via HTTP headers

Cookies are bound to domain, path

Sessions use cookies for identifier storage

Securing Cookies

Use HTTPS

Set secure cookie flag

Set HttpOnly cookie flag

```
session_start();


$_SESSION['name'] = 'value';


if (isset($_SESSION['name']))
{
  echo htmlspecialchars(
    $_SESSION['name']);
}
```

- Initialize session support, create cookie
- Store data in session
- Read data from session

```
session.save_handler = files
session.save_path = "/tmp"

session.use_cookies = 1

session.name = PHPSESSID

session.auto_start = 0

session.gc_maxlifetime = 1440
```

Basic php.ini Session Settings

Found in [Session] section

Attacks Against Sessions

Session Hijacking

Session Fixation

Prevent Cross-Site Scripting (XSS)!

Store user information in session (headers, IP address)

Change session ID with session_regenerate_id()

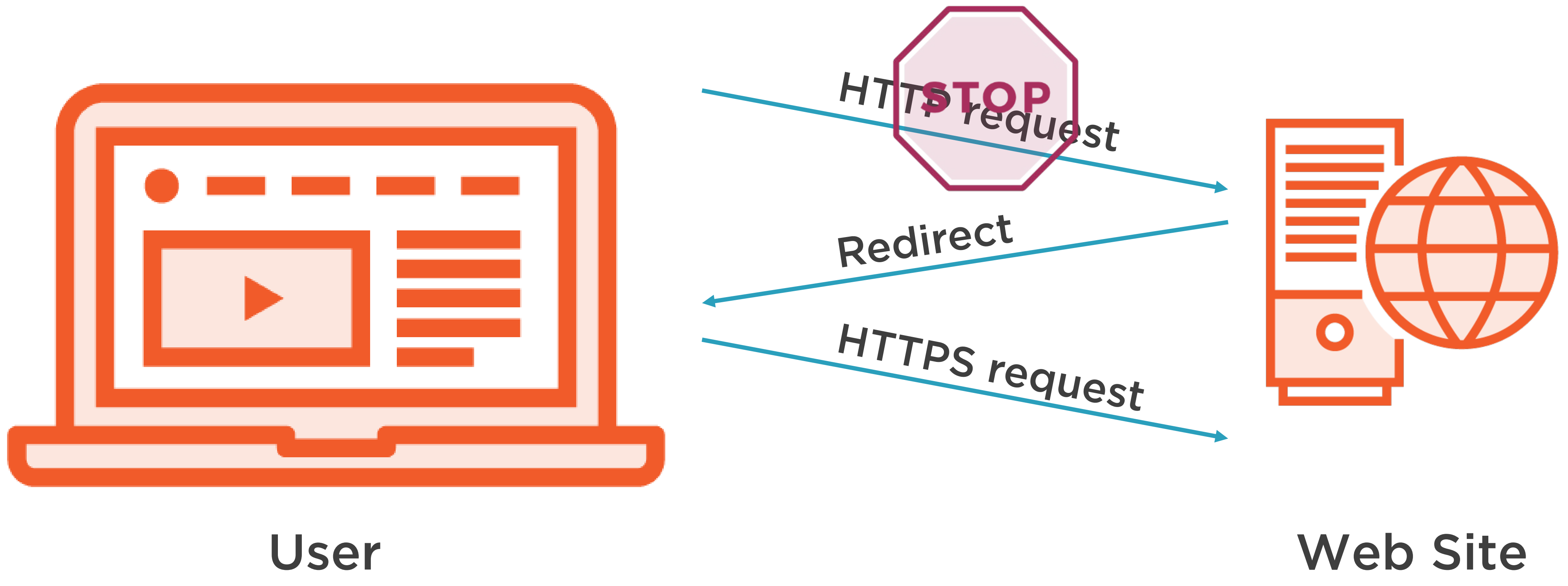
Require login prior to „dangerous“ actions

```
session.use_strict_mode = 1

session.cookie_secure = 1

session.use_only_cookies = 1

session.cookie_httponly = 1

session.hash_function = 1
session.hash_bits_per_character = 6
```

Secure PHP Session Configuration

Set in php.ini

Redirect to HTTP



```
Strict-Transport-Security:
  max-age=31536000; includeSubDomains; preload
```

HTTP Strict Transport Security (HSTS)

Forces browser to only use HTTPS from now on

Google Chrome maintains preload list

Add domain via https://hstspreload.appspot.com/

Summary

Cookies are easy to manipulate

Try to detect session hijacking

Use HSTS to make stealing (session) cookies harder