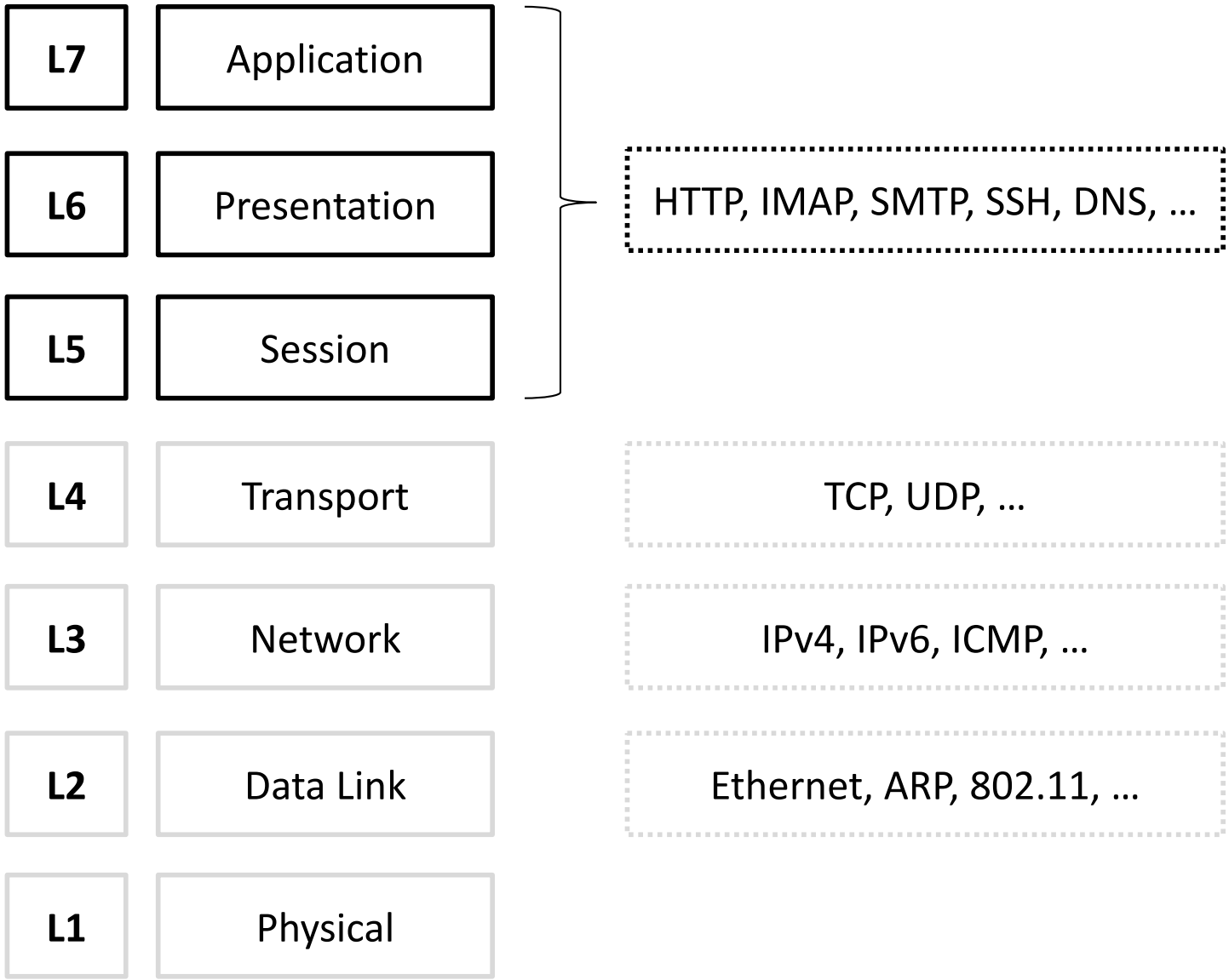
CS326 – Systems Security

Lecture 21

Introduction to Web Security

Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cy

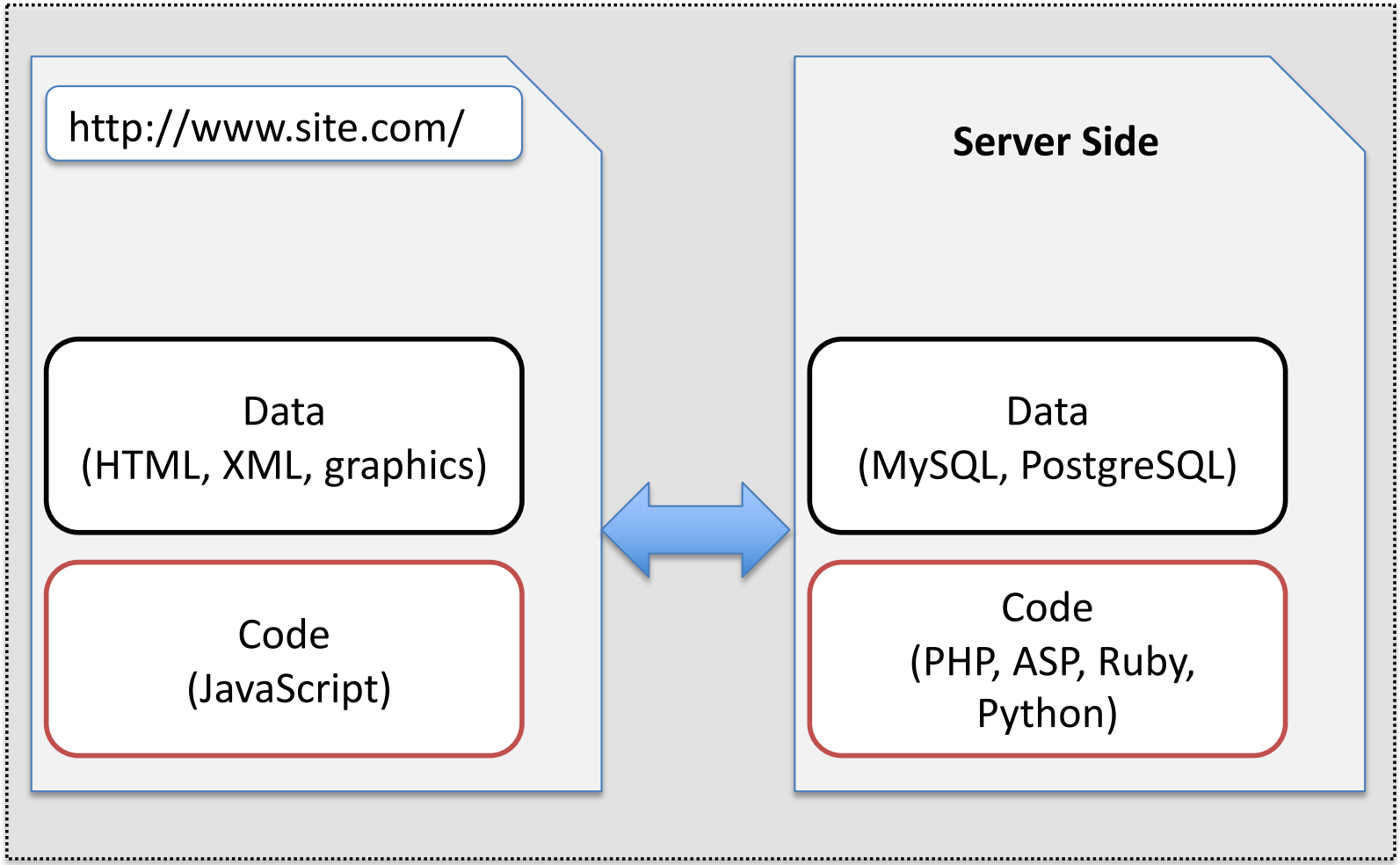
Network Layers (OSI Model)



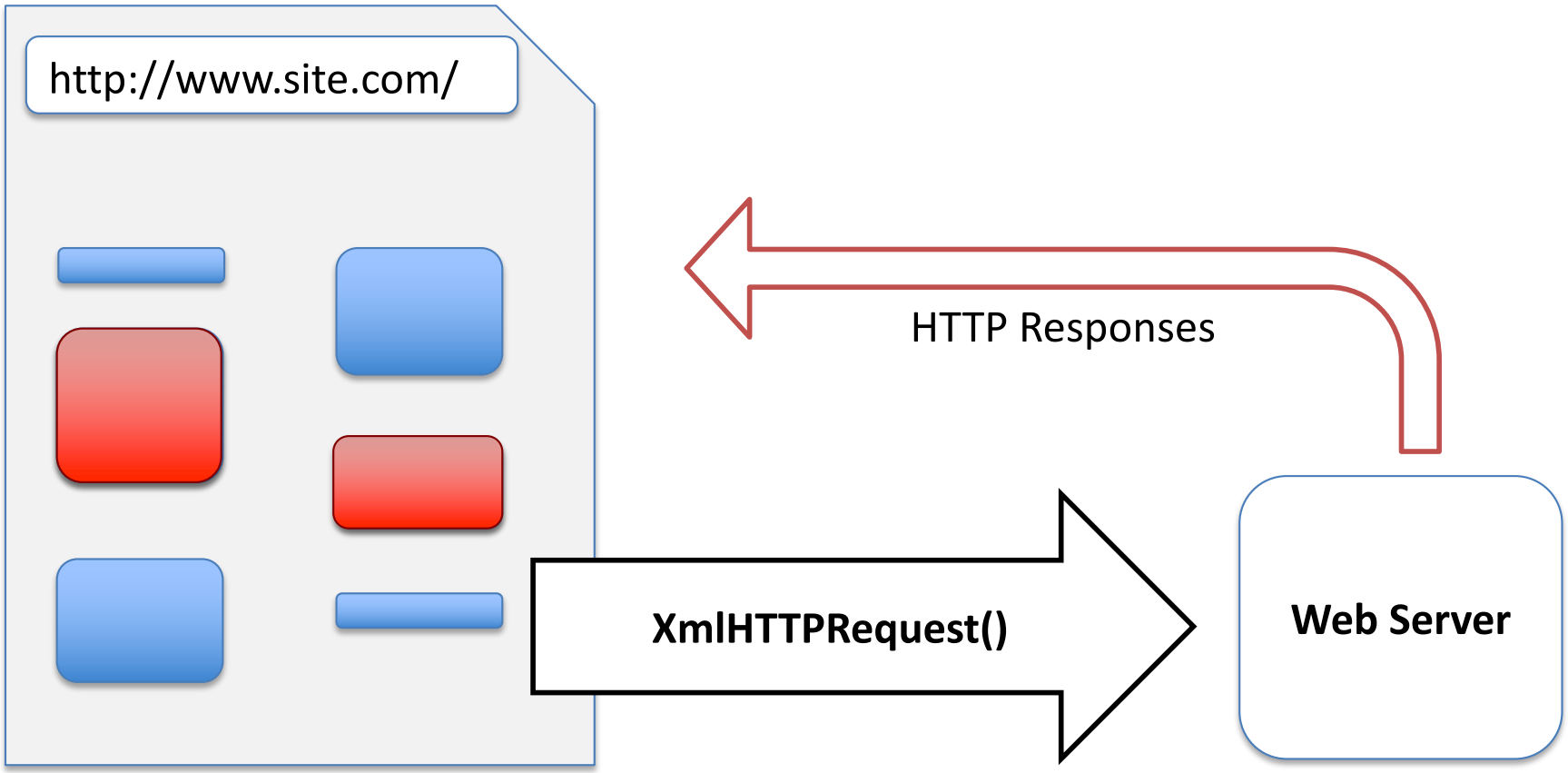
Web applications

- Code executing in the browser environment
  - JavaScript
  - Client-part or client-side
- Code executing in a web server
  - PHP, Ruby, Python, C, etc.
  - Server-part or server-side
- Client-side communicates with the server-side using the HTTP protocol

Web Applications



Interactive Web Apps (Web 2.0)

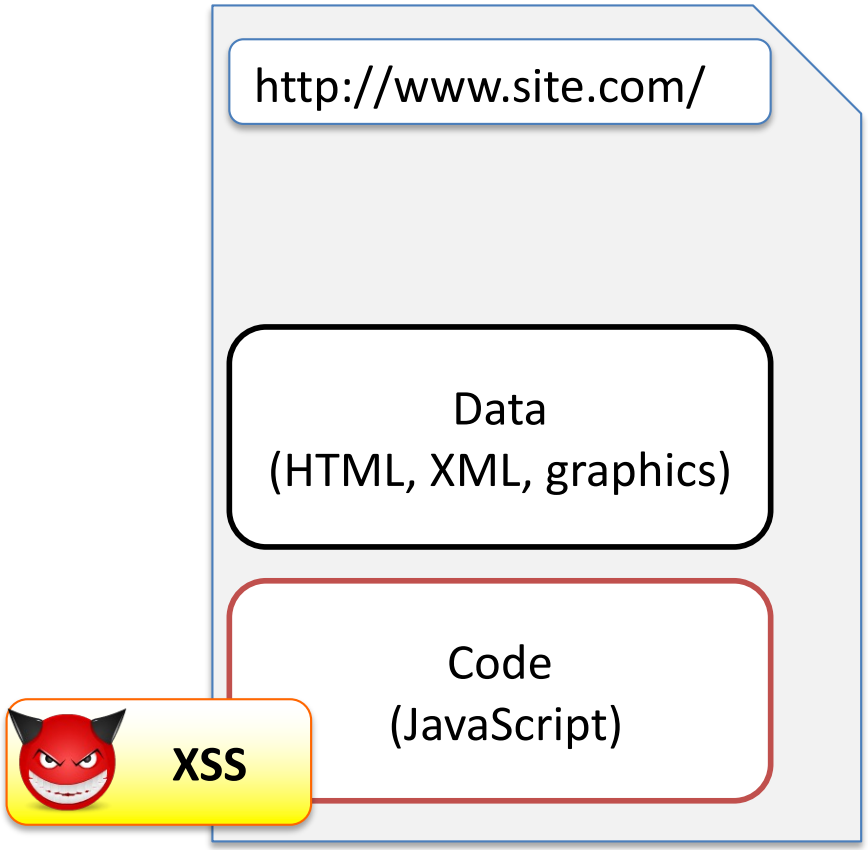


Same Origin Policy

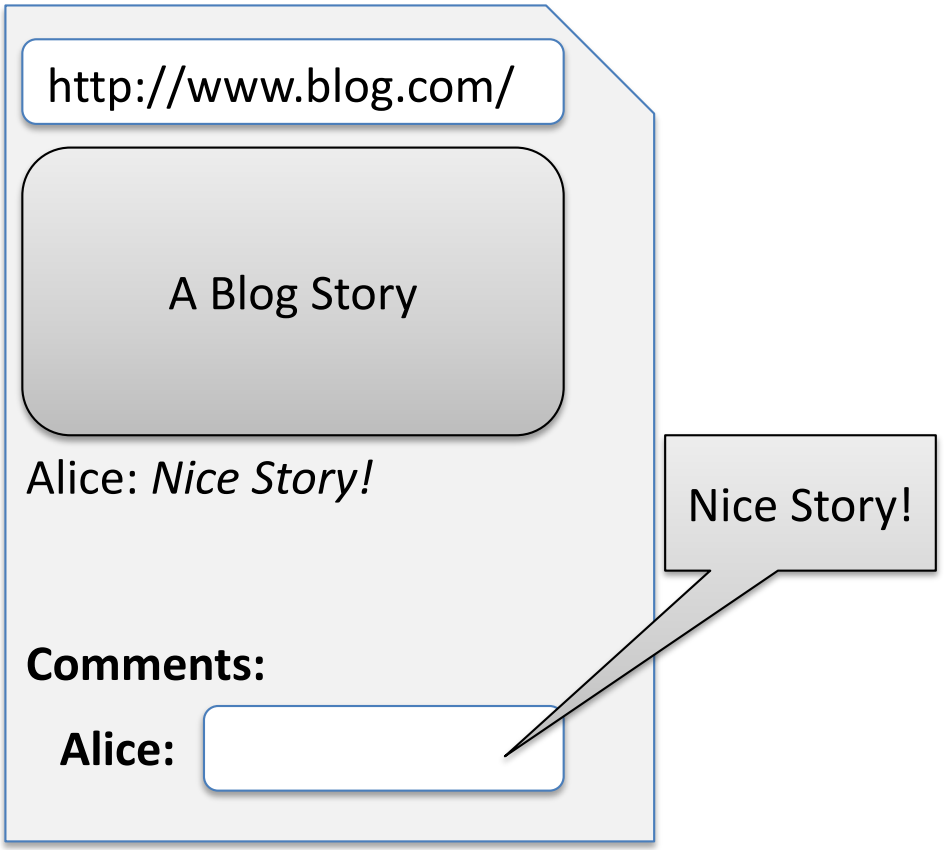
- The scripts allowed to run have the same origin:
  - (**scheme**, **hostname**, and **port number**)

- Example (http://www.example.com/)
  - https://www.example.com/script1
  - http://www.example.com/script2
  - http://www.example.com:81/script3
  - http://www.example.com/script4

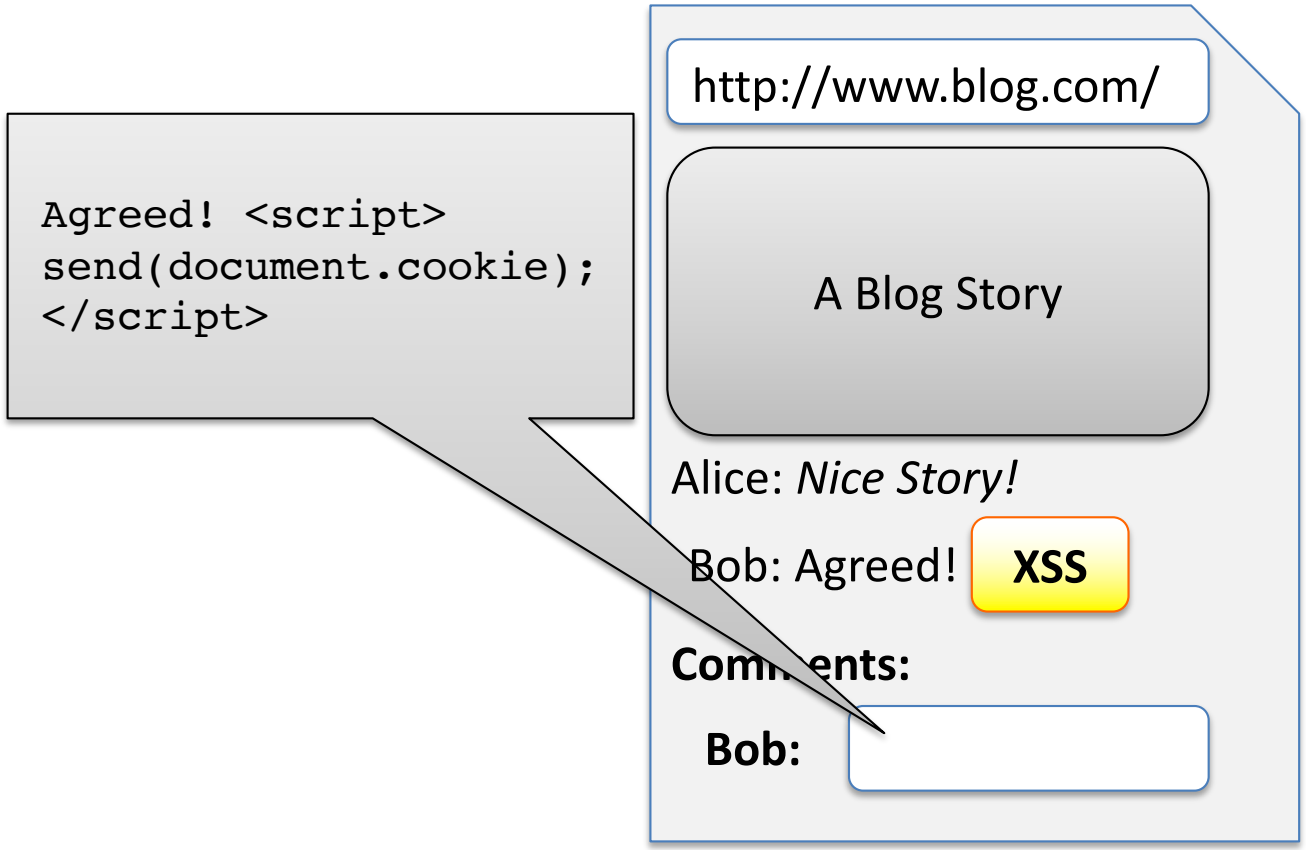
Code Injections



Cross-Site Scripting (XSS)



Cross-Site Scripting (XSS)



Cross-Site Scripting (XSS)

Alice's Cookie

http://www.blog.com/

A Blog Story

Alice: *Nice Story!*

Bob: Agreed! **XSS**

Comments:

Alice:

```
<script>
send(document.cookie);
</script>
```

Reflective XSS

http://loadgamesvf.bet365.com/f1x2games/loadGame.jsp?**gameID**=F1X2_FOOTBALL&**version**=1&**lang**=%22en&**acc_id**=1EC6296318CF49888464BDA22A78EB2C000004&**baseURL**=**%22);%3C/script%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E**

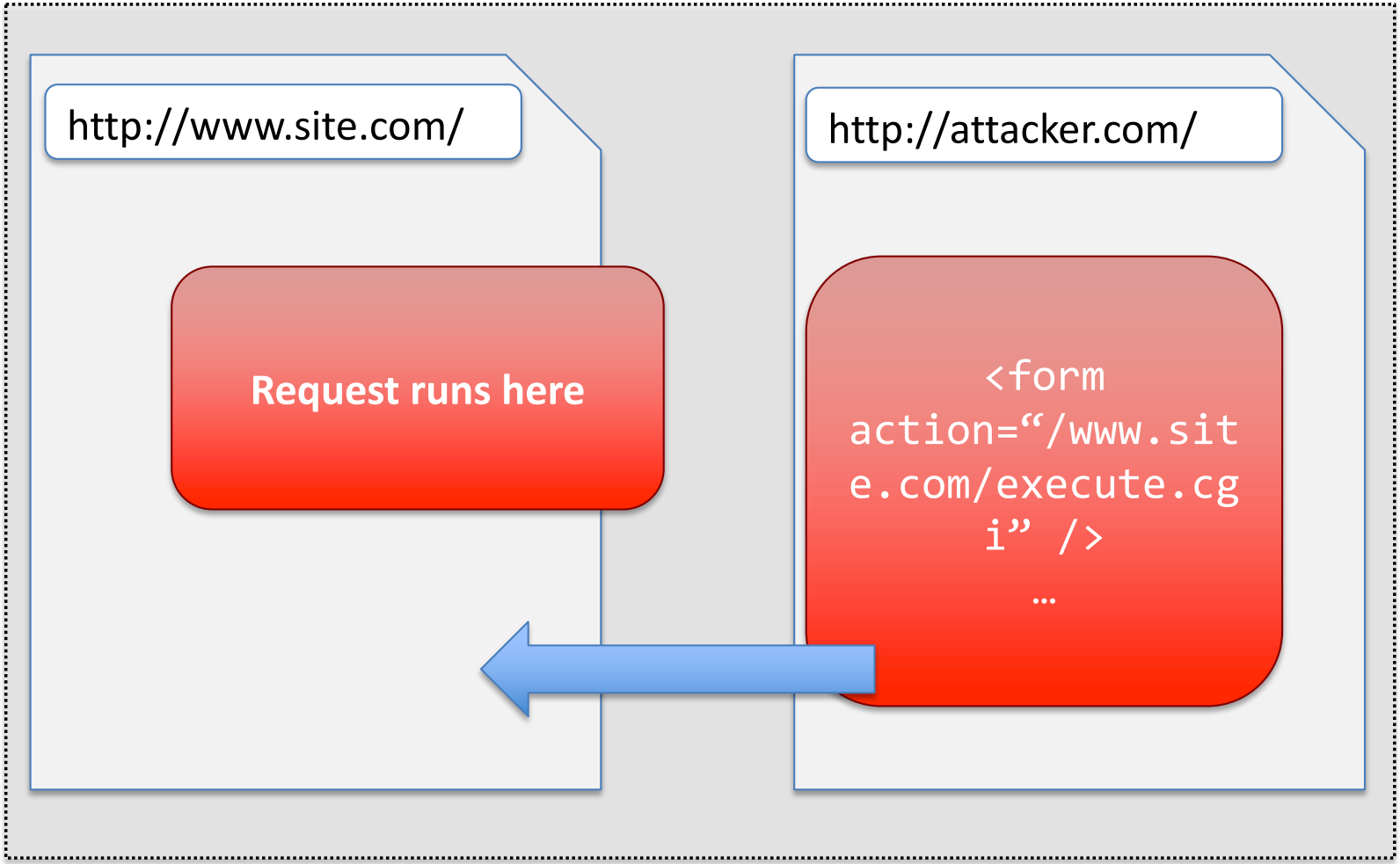
XSS Solutions

- Content Security Policy (CSP)
  - Mozilla
  - Run only white-listed scripts
- Filter our data/code
  - Hard to enforce

Filtering data

```
function xss_clean($data)
{
// Fix &entity\n;
$data = str_replace(array('&amp;','&lt;','&gt;'), array('&amp;amp;','&amp;lt;','&amp;gt;'),
$data = preg_replace('/(&#*\w+)[\x00-\x20]+;/u', '$1;', $data);
$data = preg_replace('/(&#x*[0-9A-F]+);*/iu', '$1;', $data);
$data = html_entity_decode($data, ENT_COMPAT, 'UTF-8');

// Remove any attribute starting with "on" or xmlns
$data = preg_replace('#(<[^>]+?[\x00-\x20"\'])(?:on|xmlns)[^>]*+>#iu', '$1>', $data);

// Remove javascript: and vbscript: protocols
$data = preg_replace('#([a-z]*)[\x00-\x20]*=[\x00-\x20]*([`\'"]*)[\x00-\x20]*j[\x00-\x20]*a[
$data = preg_replace('#([a-z]*)[\x00-\x20]*=([\'"]*)[\x00-\x20]*v[\x00-\x20]*b[\x00-\x20]*s[
$data = preg_replace('#([a-z]*)[\x00-\x20]*=([\'"]*)[\x00-\x20]*-moz-binding[\x00-\x20]*:#u'

// Only works in IE: <span style="width: expression(alert('Ping!'));"></span>
$data = preg_replace('#(<[^>]+?)style[\x00-\x20]*=[\x00-\x20]*[`\'"]*.*?expression[\x00-\x20
$data = preg_replace('#(<[^>]+?)style[\x00-\x20]*=[\x00-\x20]*[`\'"]*.*?behaviour[\x00-\x20]
$data = preg_replace('#(<[^>]+?)style[\x00-\x20]*=[\x00-\x20]*[`\'"]*.*?s[\x00-\x20]*c[\x00-

// Remove namespaced elements (we do not need them)
$data = preg_replace('#</*\w+:\w[^>]*+>#i', '', $data);

do
{
        // Remove really unwanted tags
        $old_data = $data;
        $data = preg_replace('#</*(?:applet|b(?:ase|gsound|link)|embed|frame(?:set)?|i(?:fra
}
while ($old_data !== $data);

// we are done...
return $data;
}
```

Cross-Site Request Forgery (CSRF)



Solutions to CSRF

- Check referrer
- Check Origin header
- Sensitive forms include a hidden random token

Mitigations can be bypassed if CSRF is used in combination with XSS.

```
<form action="/transfer.do" method="post">
  <input type="hidden" name="CSRFToken"
  value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWE...
  wYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZ...
  MGYwMGEwOA==">
  ...
</form>
```