Chapter

06

How To Fix The Web

Obscure Back-End Techniques And Terminal Secrets

Written by Paul Tero

CHAPTER SIX · BY PAUL TERO

HOW TO FIX THE WEB

IMAGINE THAT YOU WAKE UP ONE MORNING, reach groggily for your laptop and fire it up. You've just finished developing a brand new website and last night you were proudly clicking through the product list. The browser window is still open, the Widget 3000 is still sparkling in its AJAXy newness. You grin like a new parent and expectantly click on "More details". And nothing happens. You click again, still nothing. You press Refresh and get that annoying swirling icon and then the page goes blank. Help! The Internet is gone!

This chapter starts with the worst case scenario and works inwards, exploring the infrastructure of the Internet and the make-up of a Web server, imparting lots of little tips and commands along the way, opening up a new perspective on how websites can stop working — and be fixed.

The End Of The World

It is unlikely that civilization has collapsed overnight, especially if you are light sleeper. You can verify this with a battery-powered radio. An apocalypse would certainly make the news and perhaps qualify for a full-blown government warning. All stations should be reporting it, assuming there are any left; it would be really, really bad news if not.

In the US, many broadcasters participate in the Emergency Alert System, which theoretically allows the President to address the nation within 10 minutes, though it might be vulnerable to hackers.[1] France uses air raid sirens for its Signal National d'Alerte[2] and Japan's J-Alert uses loudspeakers.[3] All are part of the United Nation's International Early Warning Program. This is important, especially now that the International Decade for Natural Disaster Reduction (the 1990s) is long over.[4]

You should be able to ascertain pretty quickly if your website woes are related to this. If not, move on to the next section.

Infrastructure

The Internet depends on electricity. Your hosting company probably has an uninterruptible power supply (UPS) which will take over instantly in the event of power failure. It can provide power to your Web server for a few minutes, long enough to have a diesel generator ready to take over.[5] The major networking equipment connecting your Web server to the Internet is probably protected with UPSes and generators as well. And your laptop should survive for a few more hours if fully charged.

Your wireless router, however, will cease to function. It is the weakest link. You can get around it by checking the Internet via your smartphone, which should work as long as your nearest tower has backup power, and there is a route to the Internet through other working towers. It might be very slow though, especially if everyone in your neighborhood has also had

1 Lucian Constantin, "Emergency Alert System devices vulnerable to hacker attacks, researchers say", Computer World, Feb 13 , 2013. http://smashed.by/emergency

2 Le Signal National d'Alerte, Institut Français des Formateurs, Risques Majeurs et protection de l'Environment, Mar 28, 2007. http://smashed.by/national-alert

3 "Japan Launches Alert System For Tsunamis And Missiles", Terra Daily, Feb 9, 2007. http://smashed.by/alert-system

4 A-RES-44-236 General Assembly Resolution 44/236, United Nations General Assembly, Dec 22 , 1989. http://smashed.by/un-solution

5 "UPS and Generators - How long can they run for?", Power Continuity. http://smashed.by/power-continuity

the same idea. If your website is still gone, then the problem is a bit more personal.

Networking

Power outages aren't the only things which upset broadband routers. They have many inventive ways of failing, as does all the other networking equipment between you and your website, and all the copper and fiber-optic cable in between.

To explore networking issues, you'll need to run some commands. Much of this information is also available through various menus, but the command line gives you more data more quickly. On Mac OS X, go to Applications → Utilities and run Terminal. In Ubuntu Linux, the terminal is under Applications → Accessories. In Windows, go to Start → All Programs → Accessories and choose Command Prompt.

YOUR IP ADDRESS

Every computer connected to the Internet has a numerical IP (Internet Protocol) address. To find out yours, run the command `ifconfig` on Mac and Linux and `ipconfig /all` on Windows. The result will look something like this:

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:10:dc:75:d9:5b
          inet addr:192.168.0.11  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::210:dcff:fe75:d95b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1...

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1...
```

This says that the computer (Linux in this case) has two network interfaces. The `eth0` is the one which communicates with the Internet via a cable. If this computer had wireless there would also be an `eth1` or `wlan0` interface. The loopback interface `lo` is used as a shortcut for communicating with itself. Each interface has an IP address on the local network. The important line here is `inet addr:192.168.0.11`. This gives the computer's IP address. If you have a cable attached and wireless turned on, you may have two active interfaces and two IP addresses, but it's usually just the one. Macs tend to call these interfaces `en0` and `en1`. Windows is more verbose and uses sexy names like "Ethernet adapter Local Area Connection".

DHCP

How does your computer know its IP address? Especially on a home or wireless network, you do not need to enter this information yourself. When your computer first connects to your home network, it sends out a request to every other device on the network, something like: "Can someone please give me an IP address?"

Your broadband router should dutifully respond and assign your computer an IP address. As you probably know, routers are the devices that hold the Internet together. Unlike laptops and desktops, routers have more than one network interface, more than one cable (or wireless point) attached to them and so more than one IP address. In your home or office, the router is the little box which provides your connection to the Internet via your broadband service.

The method used to assign an IP address is Dynamic Host Configuration Protocol (DHCP). If there is no IP address when you run `ifconfig` or `ipconfig`, you can force your computer to retrieve new DHCP settings. On Windows, run `ipconfig /release` followed by `ipconfig /renew`. On a Mac, run sudo `ipconfig set en0 DHCP`, and on Linux use `sudo dhclient eth0`. On Mac and Linux, the actual command is prefaced by `sudo` which forces you to put in the root password for the computer. You must also specify which interface to renew, usually `eth0` on Linux and `en0` for Mac.

If this was successful, then voilà! You're a tiny bit closer to being back on the Internet. If not, check your broadband router. It may be off, disconnected or broken, or it may just need resetting.

DEFAULT GATEWAY

You know your broadband router was alive at some point in the recent past because it gave you an IP address. But that could have been up to three days ago: is it still there now?

Every computer on the Internet also has a default gateway. This is basically the IP address of the piece of networking equipment at the other end of your network cable or wireless connection. It is your computer's gateway to the Internet. Every time you request anything from the Internet, it goes via this gateway. To find our your default gateway, run `netstat -nr` on Mac and Linux, and `route print` (or `ipconfig` again) on Windows. You will get something a bit like:

```
Destination    Gateway      Genmask        Flags  Metric   Ref    Use Iface
192.168.0.0    0.0.0.0      255.255.255.0  U      0        0      0 eth0
0.0.0.0        192.168.0.1  0.0.0.0        UG     100      0      0 eth0
```

In the Destination column above, the 0.0.0.0 (or the word "default") means anywhere and the G in the Flags column stands for "Gateway". The default gateway of this computer is therefore 192.168.0.1. For people at home or in a small office this is probably the internal IP address of the broadband router.

You can use the `ping` command to see if it is up and available. Type `ping 192.168.0.1`.

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=1.31 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.561 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=12.6 ms
```

The “64 bytes from 192.168.0.1” represents a successful reply. If you get a reply, then your broadband router is reachable. If not, then check it again. On Mac and Linux the replies will go on forever until you press Control + C. On Windows, it will quit after the fourth ping.

BEYOND THE ROUTER

To go beyond your router, you will need the `traceroute` command on Mac and Linux, and `tracert` on Windows. This command traces a route through the Internet, reporting each networking device (router) it comes across. IP addresses are formed of four numbers from 0 to 255. Pick an IP address out of a hat and try it:

```
$ traceroute -q 1 -n 1.2.3.4
traceroute to 1.2.3.4 (1.2.3.4), 30 hops max, 60 byte packets
 1  *
 2  80.3.65.217  9.163 ms
 3  213.105.159.157  11.158 ms
 4  213.105.159.190  11.215 ms
...
13  72.14.236.149  98.484 ms
14  209.85.252.47  104.071 ms
15  *
16  *...
```

The `-q 1` option on Mac and Linux tells the command to try each router only once. The `-n` tells it not to bother looking up the human-readable name for each router, which makes the command much slower. This option is `-d` on Windows, so use `tracert -d 1.2.3.4`.

Each step above is known as a hop in networking jargon. The first hop is the broadband router. It is probably configured not to provide any information about itself, so `traceroute` just shows an asterisk. The second hop takes it outside the local network to the other side of the broadband router.

At each subsequent hop sits another router, probably with many network interfaces and many IP addresses. Each router has a routing table like the one above. Its table contains rules like: if the destination starts with 0 to 91, then send the packet down interface `eth1` (the Use Iface column); if it starts with `92` to `128`, use `eth2`.

This example goes 14 hops before reaching a dead end, either because the IP address is blocked or not in use.

That is about as far as numbers alone can take us. Hopefully you've established that civilization is still going at least a couple networking hops beyond your front door. You've also learned how to use the useful networking commands `ifconfig`, `ping` and `traceroute`. To explore further, you'll need DNS.

The Domain Name System

Smashing Magazine could have gone with http://80.72.139.101 as its main website address rather than http://www.smashingmagazine.com. It would have had two advantages: it would have used less space on business cards; and it would still have worked when DNS was down. However, Smashing's marketing people may have objected, and their customer base would have been limited to people with extremely good memories.

The domain name system makes the Internet more human-friendly by translating between domain names like www.smashingmagazine.com and IP addresses like 80.72.139.101. DNS is a big hierarchical distributed database. You are probably aware that there is no single computer which knows all the translations and which everybody else consults. Rather, it is a huge network of computers each of which know a few translations, and know who else to ask if they don't.

YOUR LOCAL DNS SERVER

Every computer has a local DNS server. It is one of the crucial bits of information your broadband router provides via DHCP: your IP address; your default gateway's IP address; and your local DNS server's IP address.

When you type a website address into your browser, your computer first asks its local DNS server to translate it into an IP address. To find out your DNS server, run the command `cat /etc/resolv.conf` on Mac and Linux, or `ipconfig /all` on Windows. On Mac and Linux, the `cat` command displays a file, and the file `/etc/resolv.conf` contains your domain name servers. The file looks like:

```
$ cat /etc/resolv.conf
nameserver 194.168.4.100
nameserver 194.168.8.100
```

NSLOOKUP

To diagnose DNS problems, first check that your local DNS server is alive with `ping`. If it is, then you can use the `nslookup` command to see if it's responding correctly. `nslookup` stands for name server lookup and is used like this:

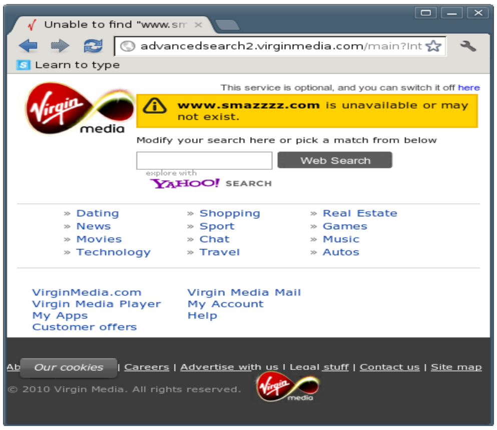
```
$ nslookup www.smashingmagazine.com
Server:    194.168.4.100
Address:   194.168.4.100#53
Non-authoritative answer:
Name:      www.smashingmagazine.com
Address:   80.72.139.101
```

This command tells you the DNS server used (194.168.4.100) and the IP address you are looking for (80.72.139.101). If `nslookup` didn't respond, then your Internet woes lie with your local DNS server. If you are at home or in a small office, your DNS server is probably provided by your broadband company.

They generally provide two or more of them, so it's unlikely that they will all fail. If you happen to know the IP address of a different name server, you could query that instead (`nslookup www.smashingmagazine.com 194.168.8.100`), but it may well refuse to talk to a stranger. You'll probably need to complain to your broadband company about the problem instead.

FREE ADVERTISING

Have you ever typed in a website address incorrectly and come up with a branded page from your broadband provider? Instead of admitting "I don't know", your local name server is sneakily replying with an alternative IP address of its choice, promoting the broadband company it is owned by. It's nice to see that the marketing people are into DNS.



This service is optional, and you can switch it off here

www.smazzzz.com is unavailable or may not exist.

Modify your search here or pick a match from below

© 2010 Virgin Media. All rights reserved.

Broadband company intercepting a non-existent website.

A FULL JOURNEY

Having confirmed that your local DNS server is working, you can tentatively try to establish human-friendly contact with the Internet using `traceroute` with a domain name instead of an IP address, and leaving out the `-n` option. This will report a readable name for each router.

```
$ traceroute -q 1 www.smashingmagazine.com
traceroute to www.smashingmagazine.com (80.72.139.101), 30 hops max, 60
byte packets
 1  *
 2  brig-core-2b-ae6-725.network.virginmedia.net (80.3.65.177)  10.542 ms
 3  popl-bb-1b-ae14-0.network.virginmedia.net (213.105.159.157) 13.934 ms
 4  popl-bb-1c-ae1-0.network.virginmedia.net (213.105.159.190) 14.454 ms
 5  nrth-bb-1c-ae7-0.network.virginmedia.net (62.253.174.137) 15.982 ms
 6  nrth-tmr-1-ae1-0.network.virginmedia.net (213.105.159.30) 16.215 ms
 7  fran-ic-1-as0-0.network.virginmedia.net (62.253.185.81)  36.090 ms
 8  FFMGW4.arcor-ip.net (80.81.193.117)  39.064 ms
 9  92.79.213.133 (92.79.213.133)  47.404 ms
10  92.79.200.190 (92.79.200.190)  45.385 ms
11  kar-145-254-15-178.arcor-ip.net (145.254.15.178)  40.421 ms
```

```
12  145.253.159.106 (145.253.159.106)  46.436 ms
13  1-3-frr02.continum.net (80.72.130.170)  49.321 ms
14  1-6-frr04.continum.net (80.72.130.158)  47.194 ms
15  www.smashingmagazine.com (80.72.139.101)  48.081 ms
```

This reveals much more about the journey packets of data take. The first few hops in any traceroute are probably owned by the local broadband company, Virgin Media in this case. If the traceroute stopped here, then it would be an issue for them resolve. You could phone them for more information.

Once the traceroute leaves your broadband provider, it enters a no man's land of big inscrutable networking devices. In this case they are owned by Arcor, a subsidiary of Vodafone. If the traceroute fails here, it may represent a fairly major networking problem and there's not much you can do.

Eventually, it will reach the hosting company for the website in question (Continum.net in this case). If it fails there, then the fault may lie with your hosting company. Or it may simply be that the traceroute is blocked by a firewall. Or that your website has moved.

MOVING WEBSITES

It's unlikely that your website has moved to a different server without you knowing, especially as you were just working on it last night, but you can double-check this.

Every DNS server keeps a cache of every domain name it has been asked for. This saves clogging up the Internet with requests for things that rarely change. The downside is that if someone changes the IP address for a domain like www.smashingmagazine.com, it can take 24 to 48 hours for all the caches to clear so that everyone in the world knows the new IP address.

To ascertain that you have the latest information, you first need to find out the local name server for the domain name you are querying. To do this, give `nslookup` the option `-type=ns`, like this on Mac, Linux and Windows:

```
$ nslookup -type=ns www.smashingmagazine.com
Server:         194.168.4.100
Address:        194.168.4.100#53
Authoritative answers can be found from:
smashingmagazine.com
        origin = a.regfish-ns.net
        mail addr = postmaster.regfish.com...
```

The origin (or sometimes primary name server) is the local DNS server for www.smashingmagazine.com. You can use `nslookup` again to query this server directly:

```
$ nslookup www.smashingmagazine.com a.regfish-ns.net
Server:         a.regfish-ns.net
Address:        79.140.49.11#53
Name:   www.smashingmagazine.com
Address: 80.72.139.101
```

Compare this to the `nslookup` on your local DNS server. It no longer says "non-authoritative". This is now the authoritative reply. It's the same IP address, so we know that www.smashingmagazine.com didn't suddenly move last night.

On Mac and Linux, you can use the `dig` command to find out exactly how long your local DNS server has cached this translation for. It stands for domain information groper. Windows users will need to search for an online `dig` tool as Windows doesn't natively support this command:

```
$ dig www.smashingmagzine.com
...
;; ANSWER SECTION:
www.smashingmagzine.com. 246    IN      A       80.72.139.101...
```

The 246 is the number of seconds before the local DNS server's cache expires and it has to rediscover the IP address for smashingmagazine.com.

YOUR BROADBAND ROUTER REVISITED

Now that DNS is working, you can find out what the world thinks of you. You have already discovered your computer's own IP address above. But that may not be the one that it uses on the Internet. If it starts with 192.168 or 10, then it is definitely not a public address. Those IP address ranges signify local IP addresses for use within your internal network only.

When your computer sends a request to the Internet, it first goes to your default gateway (your broadband router), which also has a local internal IP address such as 192.168.0.1. But your router has another public IP address as well. Your router wraps up your request and resends it from this public IP address instead.

Therefore, your broadband router's public IP address is basically your public IP address. This is how the rest of the Internet sees you as you browse. This is the IP address that will show up in log files in any of the websites you visit. Consequently, anybody else using the same broadband router will have the same public IP address as you. Your router handles all of this using a process called network address translation, making sure requests and information go out and come back to the right local IP address.

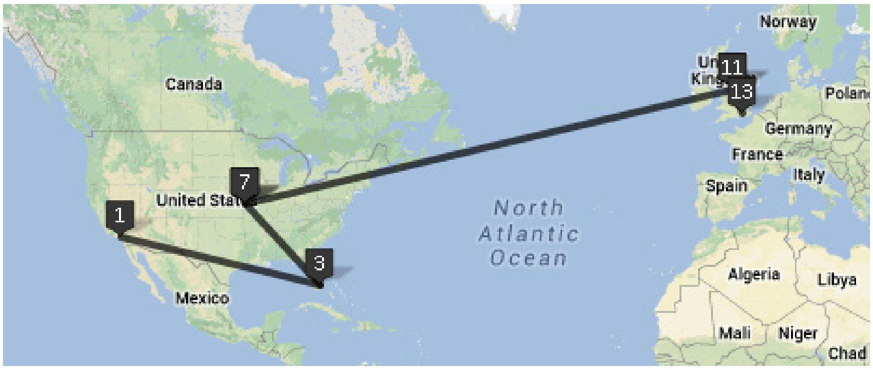
You can find out your broadband router's public IP address by visiting a website like whatismyipaddress.com. Alternatively, you can run the command `curl ipinfo.io/ip` on Mac or Linux, or `wget -O- -q ipinfo.io/ip` on Linux. Both `curl` and `wget` retrieve a Web page (http://ipinfo.io/ip) from the Internet. The `-O-` option (that's a letter O, not zero) tells `wget` to output the result to the screen (signified by a single hyphen) rather than save it to a file, and `-q` tells it to do it quietly. `curl` automatically outputs to the screen. To use `curl` on Windows you have to download and install it first. All these methods go outside your local network and look back. There is no way to find out your router's public IP address from the inside. The output is quite simple:

```
$ curl ipinfo.io/ip
85.106.118.199
```

WHERE ARE THEY?

Websites like whatismyipaddress.com and ipinfo.io do more than just tell you your public IP address. They also provide geolocation services. It is interesting to take the IP addresses from the traceroute above and copy and paste them in. Geolocation guesses at the physical location of the router, and can also tell you who owns the IP address. The address 62.253.185.81 above is in southern England but the next one crosses the Channel to 80.81.193.117 in Frankfurt, Germany. This type of geolocation relies on databases of IP address ownership.

In fact, there are websites which can map this all out for you, such as DNStools.ch[6] and YouGetSignal[7]. The starting points for these traceroutes will be the Web server hosting the tool, rather than your own computer. Below is an example from a website in Los Angeles to the BBC website in London.



Visual traceroute from Los Angeles to London covering 7,904 miles in 4.8 seconds.

Connecting To Your Server

So, civilization and its Internet are both up and running. What's gone wrong? Your website lives on a computer somewhere out there, probably in a big air-conditioned room full of other computers, with multiple fire

6 http://smashed.by/visual-traceroute
7 http://smashed.by/visual-tracert

doors and an awful lot of colorful cabling. This computer is colloquially known as a Web server.

Imagine for a moment that your Web server is the nation of France. If you want to send a large item of furniture to somewhere in France, it will be wrapped up tight on a container ship and sent off across the sea. It will arrive in one of France's major ports, maybe Marseille or Bordeaux or Le Havre. It doesn't really matter to you which port it goes through, but it does matter to the shipping company. Computers are similar, except they are a bit smaller and have 65,535 ports.

On computers, some ports are assigned specific functions. On a Web server, port 80 receives and replies to Web browsing requests. Ports 25 and 110 deal with email. A typical Web request could involve port 50133 on your computer (the sending port is often chosen randomly) sending a request to port 80 at 80.72.139.101, something like "Hey you, send me the Web page /index.html".

TELNET AND NETCAT

The `telnet` command allows you to mimic a container ship and connect to a specific port on a server. Windows does not have `telnet` by default, but you can enable it on Windows 7 by going to Start → Control Panel → Programs → Turn Windows features on or off → Telnet Client.

Since we're dealing with a website problem, and since the Web server is almost always on port 80, try telnetting to port 80:

```
$ telnet www.smashingmagazine.com 80
Trying 80.72.139.101...
telnet: Unable to connect to remote host: Connection refused
```

Mac and Linux support an alternative command: `netcat`. It is more specifically suited for networking tasks and supports additional features like proxies. This chapter will focus on `telnet`, however, as it also works on Windows. Add `-v` to `netcat` to make it verbosely tell you what it's doing.

```
$ netcat -v www.smashingmagazine.com 80
netcat: connect to www.smashingmagazine.com port 80 (tcp) failed: Connec-
tion refused
```

Uh oh.

Except, not really. I faked the issue above. Smashing Magazine wasn't really down. But I will use www.smashingmagazine.com as an example domain throughout this chapter. Suspend your disbelief and pretend that Smashing has moved into the Widget 3000 market and has sequentially fallen victim to just about every networking and website problem imaginable, and subsequently overcome them.

CONTROL PANEL

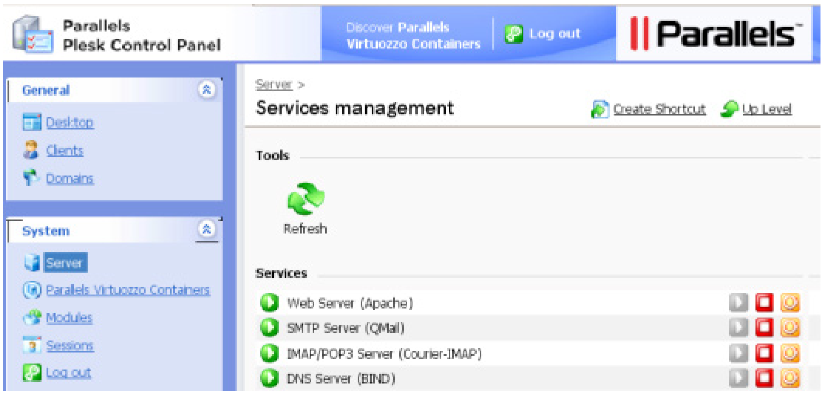
Whenever your Web server receives data on port 80, it sends it to a piece of software for processing. Confusingly, that software is also called a Web server. By far the most common type of Web server software is Apache. According to W3Techs in June 2013, it had a market share of 65.2%.[8]Microsoft's Internet Information Server (IIS) is second with 15.7%, just in front of nginx at 14.3%.

Web server software shouldn't ever stop working. But if it does you can, hopefully, just restart it again. The quickest way to do this is using a control panel provided by your server. Windows servers (34.3% market share in June 2013[9]) are often managed by Remote Desktop or VNC which allow you to take control of the server's screen, keyboard and mouse, and change settings directly on the server.

The rest of this chapter, however, will focus on Linux and UNIX servers (65.7%), which are usually managed via a Web interface such as Plesk, CPanel or Webmin. These management tools are really just websites, but running on a different port. Plesk, for example, usually runs on port 8443, CPanel on 2082 or 2083 and Webmin on 10000.

8 „Usage of Web servers for websites", W3Techs. http://smashed.by/web-servers

9 „Usage of operating systems for websites", W3Techs. http://smashed.by/operating-sys



Restarting a Web server with Plesk.

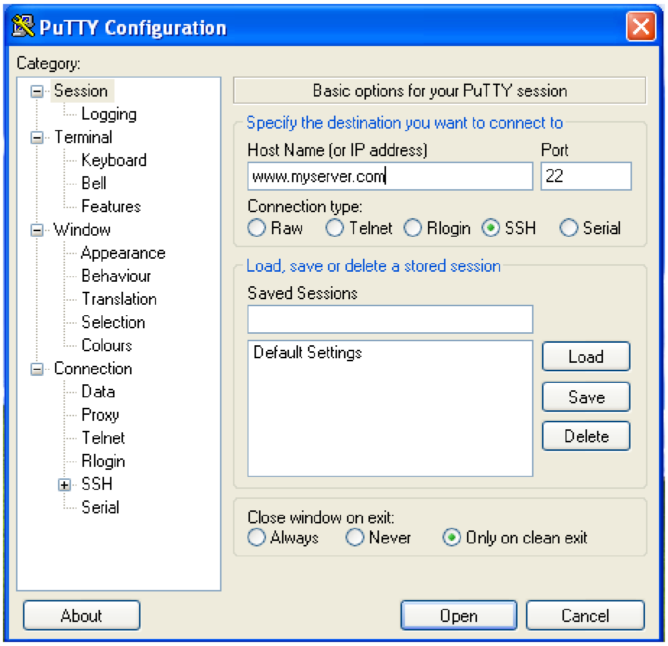
Dig deep into your email folders and look for the URL, username and password for your control panel. Log in and find the screen which allows you to restart your Web server software. In Plesk, look for "Services management" (in Server or Tools & Settings) and press the Play button next to "Web Server (Apache)".

SSH

If port 80 is down, there's a good chance that the control panel won't be available either. You will need to log in to the server and issue commands directly. For this there is Secure Shell (SSH). SSH is like the text-only version of Remote Desktop. It allows you to open a terminal window on the server. From a Linux or Mac desktop or laptop, use the command `ssh`. From a Windows computer, download and run PuTTY.

You'll need the username and password for your server, contained in the same email as above. On a Linux server, root is the most powerful administrative user. For security reasons, the SSH user from your email will often be something less privileged like admin. When you run SSH, you have to provide the username as part of the command. It will ask you to accept a security fingerprint and enter a password:

```
$ ssh root@www.smashingmagazine.com
```



Host Name (or IP address): www.myserver.com
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

Using PuTTY for SSH from a Windows computer.

```
The authenticity of host 'www.smashingmagazine.com
(80.72.139.101)' can't be established.
RSA key fingerprint is 00:5a:cf:51:83:46:0b:91:29:ef:2e:1d:c9:59:e9:ab.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'www.smashingmagazine.com,80.72.139.101' (RSA)
to the list of known hosts.
root@www.smashingmagazine.com's password: ...
```

If successful, you'll end up with a welcome message and a terminal prompt:

```
Linux dedivps-13236 2.6.10-091stab048.3 #1 SMP Fri Dec 7 17:06:14 GMT
2012 x86_64
Last login: Thu May  2 07:20:11 2013 from cpc1-brig18-2-0-cust123.3-3.
cable.virginmedia.com
root@dedivps-13236:~#
```

Note that this will only work on Linux or UNIX servers that have an SSH server which accepts connections, and the rare Windows servers that have opted to install it. For most other Windows servers, you'll need Remote Desktop instead. If you can't get at your server via a control panel or SSH, your options are very limited. There's a slim chance that an overenthusiastic firewall is getting in the way, or that you're experiencing a denial of service attack. Or else you'll need to contact your hosting company and ask for help.

FIREWALLS

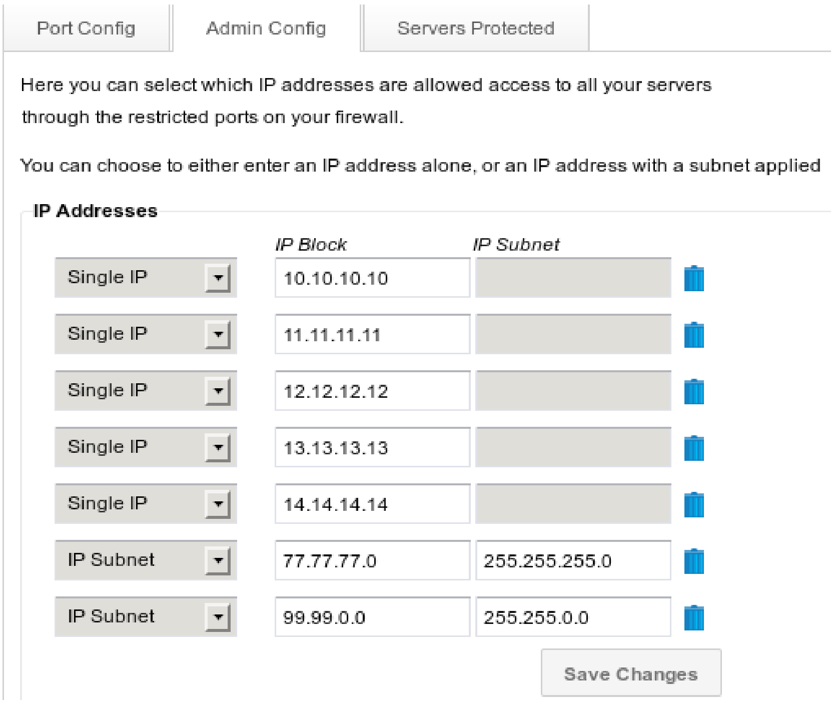
Firewalls are bits of hardware or software that filter incoming and outgoing data. These filters are applied according to the source and destination IP address and port. So, for example, a firewall should allow all requests going to the server's port 80 or else nobody will be able to get to the website. But it may block all requests to port 8443 (Plesk), port 22 (SSH) or port 3389 (remote desktop) except from a few known and trusted IP addresses.

You can sort of tell if there's a firewall in your way depending on how the connection fails. To test if SSH is being blocked, you can run the command `ssh` or use `telnet` as above to port 22:

```
$ telnet www.smashingmagazine.com 22
Trying 80.72.139.101...
telnet: Unable to connect to remote host: Connection refused
```

"Connection refused" means that your data reached the server but was probably refused entry for non-firewall reasons. For example, SSH may be turned off or running on a different port. The message "Connection timed out" or no message more strongly indicates a firewall block. If it does connect, press Control +] to get to the `telnet>` prompt and then type "quit" to quit.

So if you have a firewall (that email should tell you), you need to make sure that SSH is allowed and that your public IP address is in the list of trusted ones. Even if you know your IP address was in the list yesterday, it



Here you can select which IP addresses are allowed access to all your servers through the restricted ports on your firewall.

You can choose to either enter an IP address alone, or an IP address with a subnet applied

IP Addresses

	IP Block	*IP Subnet*
Single IP	10.10.10.10	
Single IP	11.11.11.11	
Single IP	12.12.12.12	
Single IP	13.13.13.13	
Single IP	14.14.14.14	
IP Subnet	77.77.77.0	255.255.255.0
IP Subnet	99.99.0.0	255.255.0.0

Many firewalls maintain a list of trusted IP addresses.

may have changed today, particularly if you have had broadband issues recently. The public IP addresses assigned to home routers can stay the same for months on end, and then suddenly change. How and why and when depends on your broadband company and your neighbors. The only way to avoid this is to pay extra for a static or fixed IP address.

DENIAL OF SERVICE

Imagine that the Widget 3000 suddenly goes viral. The Queen of England is filmed throwing one at the winner of X Factor and suddenly everybody in the world wants to own one. You might think “Fantastic!” But unless your server infrastructure is prepared to go from 100 visits an hour to 100 million, you probably won’t actually sell very many. All those visitors accessing your website at once will grind the network and your server to a halt. The first few thousand visitors may receive half a Web page, the rest will be staring at blank browsers.

And when you try to telnet to your server as above, it will also sit there waiting — no refusal but no entry either. This is roughly what happens in a distributed denial of service (DDoS) attack. All those hackers who have spent the last 15 years finding holes in Internet Explorer were not working in vain. They have managed to plant Trojan horses on millions of computers worldwide. When they issue the command, all those computers suddenly try to send data to and request data from your Web server, overwhelming it and making it unreachable.

Unless you are running a bank or a bomb factory, or have managed to make some clever and determined enemies, it is unlikely to happen to you. Let's assume telnet has instead connected successfully.

Checking Your Server

Now you're in business. You've got a terminal window open on your server waiting for your every command. From now on, all the commands are being issued on your Linux server, not your laptop.

LISTENING TO PORT 80

The first step is to figure out which software should have responded when you tried to telnet to port 80. For that, you can use the `netstat` command to display all the networking information about your server. Add `-tlpn` to the command to make it show only TCP connections that it is listening for, along with the numeric port and the program they belong to.

```
$ netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address State   PID/Program name
tcp        0      0 127.0.0.1:53   0.0.0.0:*       LISTEN  4290/named
tcp        0      0 127.0.0.1:5432 0.0.0.0:*       LISTEN  3507/postmaster
tcp        0      0 0.0.0.0:3306   0.0.0.0:*       LISTEN  7117/mysqld...
```

This shows only an abbreviated output. It normally shows all the ports which the server is listening to, and there can be between 10 and 100 of them. When running any command, you can whittle down its output using the `grep` command. `grep` filters the output and only shows lines containing a word or phrase you have specified. The vertical bar | makes a pipe, piping the output of one command (`netstat`) into the input of another (`grep`). Try this instead:

```
netstat -tlpn | grep :80
tcp6   0    0 127.0.0.1:8005   :::*   LISTEN   22885/java
```

This runs `netstat` and shows only results containing `:80`. This server has a `java` process listening to port 8005 but no Web server running.

Which Web Server

When a Linux server starts up it looks in the directory `/etc/init.d` and runs the scripts there to launch its software. On other UNIX flavors like BSD this might be `/etc/rc.d` or `/etc/rc.d/init.d`. This is similar to the Startup menu folder in Windows.

You can see what your server starts using the `ls` command which lists the files in a directory. The `-l` shows a long format with permissions, owners, size and date created.

```
$ ls -l /etc/init.d
total 368
-rwxr-xr-x 1 root root  7621 Sep 13  2012 apache2
-rwxr-xr-x 1 root root  3281 Oct  5  2012 bind9
-rwxr-xr-x 1 root root  2444 Jan  1  2011 bootlogd
-rwxr-xr-x 1 root root  5364 Nov  1  2011 courier-imap
-rwxr-xr-x 1 root root  3753 Dec 19  2010 cron...
```

You are looking for a Web server software package such as `apache2`, `httpd` (the old name for Apache) or `nginx`. You can use `grep` again with the

`-e` option which tells it to use a regular expression. In regular expressions, the vertical bar means "or" so this displays any files containing "apache" or "http" or "nginx". The bar must be escaped with a backslash:

```
$ ls -l /etc/init.d | grep -e "apache\|http\|nginx"
-rwxr-xr-x 1 root root  7621 Sep 13  2012 apache2
```

RESTARTING THE WEB SERVER SOFTWARE

The files in `/etc/init.d` are called shell scripts. The commands you've been using on Mac and Linux up till now form part of a C-type language which also includes setting variables and running loops. As in JavaScript, the semicolon at the end of each line is optional, but if you use it, you can cram several commands onto a single line. Here is a simple `for` loop on the command line:

```
$ for i in 1 2 3; do echo Line $i; done
Line 1
Line 2
Line 3
```

To see a complex shell script, take a look at some of the startup scripts in `/etc/init.d`. Use the `less` command to view a file. Press space to view the next page or `q` to quit from `less`.

```
$ less /etc/init.d/apache2
#!/bin/sh
set -e
SCRIPTNAME="${0##*/}"...
```

The reason we're really here, though, is to restart the Web server software. If you logged into `ssh` as the administrative user root, you can run the restart command directly. In this case, you will have been typing your commands after a # instead of a $. If not, you'll need to prefix it with the `sudo` command, which says do some command as the super user. You'll need the root password to hand for `sudo`. To tell the Web server software to start, run:

```
$ sudo /etc/init.d/apache2 start
Password:
Starting apache2...
```

Hopefully this will fail with a useful error message. Hopefully, because then you will know why it crashed in the first place, and you can plug the error message into Google to find out how to fix it.

If it was unlucky enough to work, then your Web server is now running. All the scripts in `/etc/init.d` run as background processes, or daemons in UNIX parlance. This means that you can start them and they will stay running even after you exit from `ssh`, turn off you computer and go to the beach. This is unlike commands like `traceroute` and `ls` which do their thing and finish.

You can run `netstat` again to double-check the Web server is now running. Notice the `d` at the end of `apached`. It stands for daemon.

```
netstat -tlpn | grep :80
tcp        0      0 :::80            :::*    LISTEN      18201/apached
tcp6       0      0 127.0.0.1:8005   :::*    LISTEN      22885/java
```

WEB SERVER ERROR LOGS

If it failed to start and didn't give a friendly error message, the next place to look is the error logs. These are usually in the `/var/log` directory named after the software. Run `ls /var/log` to double check. For example, Apache's logs are in `/var/log/apache2`.

```
$ ls -l /var/log/apache2/*log*
-rw-r----- 1 root adm  1944899 May  5 09:59 /var/log/nginx/access.log
-rw-r----- 1 root adm   538152 May  4 02:40 /var/log/nginx/access.
log.2.gz
-rw-r----- 1 root adm    28647 May  5 08:18 /var/log/nginx/error.log
-rw-r----- 1 root adm     5701 May  4 02:35 /var/log/nginx/error.log.2.gz
```

This shows that Apache has an access log and an error log. Both logs are zipped up around 02:30 each morning. The next command first changes into the `/var/log/apache2` directory with the `cd` command, and then uses tail to view the last few entries in a log file.

```
$ cd /var/log/apache2; tail error.log
```

To look at a gzipped log file, use the `zcat` command to output it and pipe through `tail`. The `-20` shows the last 20 lines.

```
$ zcat error.log.2.gz | tail -20
```

Or better yet, just look for errors using `grep`. Use `zcat` with the `-f` option to display both normal and zipped log files. Then pipe the output through `grep` to search for the word “error” case insensitively:

```
$ zcat -f error.log* | grep -i error
```

This command may produce a lot of output. If a Matrix fan happens to walk past your computer right now, they’ll be impressed to see all that raw data whizzing by. It won’t help you much, though, so pipe it through `less`:

```
$ zcat -f error.log* | grep -i error | less
```

`less` is powerful. You can press arrow keys or `j` to go down, `k` to go up, `/something` to search for “something” and `h` to see a helpful list of all the commands. If you can narrow down the moment of failure of your Web server to a few hours, you can use `less` to navigate to that part of the log file.

SYSTEM LOGS

There are several other useful log files in `/var/log` such as syslog (the system logger) and dmesg (bootup messages). They all use a similar date format so if you can narrow down the time when you suspect something went wrong, you can search them all at once. This command changes to the `/var/log` directory and then outputs all the files using `zcat -f`. The `[234]` in `grep` is borrowed from regular expressions and matches the numbers 2 or 3 or 4. So this will display any error messages in any of the logs that took place on May 5 between 02:00 and 04:00 in the morning:

```
$ cd /var/log; zcat -f * | grep "May  5 0[234]:" | less
```

OUT OF SPACE

If your Web server software still won't start and the error remains elusive, there are a couple common problems you can explicitly check for. Your server could have run out of hard drive space. Use the command `df` to show disk file usage. The `-h` shows things in human-friendly form (with M for megabyte and G for gigabyte instead of really big numbers):

```
$ df -h
Filesystem          1M-blocks      Used  Available  Use% Mounted on
/dev/simfs              20.4G      9.8G      10.6G   49% /
tmpfs                    1.6G         0       1.6G    0% /lib/init/rw
tmpfs                    1.6G         0       1.6G    0% /dev/shm
```

If that was a problem, then a quick solution is to find and delete really big files. The find command is very powerful. The `-size` option tells it to look for files of at least the given size, and the `-printf` option tells it to print the size (make sure that "%12s" and the first (not second) instance of "12" are formatted as code.), last modification time (`%t`), directory (`%h`), file name (`%f`) and then a new line (`\n`). To view all files over 10 megabytes try:

```
$ find / -size +10M -printf "%12s %t %h/%f\n"
445631888 Mon Mar 18 13:38:07.0380913017 2013 /var/www/huge-file.zip
```

To delete the file and free up space quickly: `$ rm /var/www/huge-file.zip`

OUT OF MEMORY

To check your server's RAM usage, run the `free` command, again with `-m` to show in megabytes:

```
$ free -m
             total       used       free     shared    buffers     cached
Mem:          3067       2673        393          0          0        819
-/+ buffers/cache:       1853       1213
Swap:            0          0          0
```

This server has 3GB of total memory and 393MB free. This is fine as Linux likes to use a lot of its memory. It's the `buffers/cache` line which you should focus on. If this is nearly all used, then your system may be struggling to cope.

To find out what is hogging all the memory, use the `top` command. It shows a real-time display of system CPU and memory usage. Unfortunately this will also run very slowly if your server is under strain, but it may tell you what's causing the problem.

```
$ top
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
22885 tomcat6   20   0 2061m 159m 2644 S    1  5.2 780:41.85 java
    1 root      20   0  8360  568  536 S    0  0.0   0:52.30 init
    2 root      20   0     0    0    0 S    0  0.0   0:00.00 kthreadd/10086
    3 root      20   0     0    0    0 S    0  0.0   0:00.00 khelper/10086
14579 root      20   0 40900 3124 1668 S    0  0.1   1:30.27 nginx...
```

If something is being a memory hog, you can kill it. However, be sure you know what it is first. You may crash the server completely, or lock yourself out, or stop an important database amalgamation which your efficiency-unaware colleague started three days ago. To kill a process, press `k` and type in the number from the process ID (PID) column. It will ask for confirmation and then try to kill the process. If you are not root, it may say "Operation not permitted", in which case you'll need to run `sudo top` instead.

The PID is used to identify a piece of software running on a computer. Each instance of an application, program or software has a unique PID. If the process refuses to go away, you can press `q` to leave `top` and try the `kill` command instead. Give it the more extreme `-9` option. `top` sends the friendly signal 15 (termination). Signal 9 goes straight for the kill.

```
$ sudo kill -9 22885
```

Run `top` again. If some other similar process has taken over the memory-eating honors, then you have only killed a child process. You will need to find out the parent which spawned the misbehaving child in the first place, because killing the parent will also stop all the children. The `ps` command can be used to display information about a specific process. Normally it does not show the parent process ID, so you need to add the `-o` option and specify that the output should show the parent process ID `ppid` and the full command that started it:

```
$ ps -o ppid,command 14579
 PPID COMMAND
 6950 nginx: worker process
```

This `nginx` process is not the main one.

```
$ ps -o ppid,command 6950
 PPID COMMAND
    1 nginx: master process /usr/sbin/nginx
```

A very low parent PID means that this process is the daddy. Killing process 6950 will kill the main `nginx` process and all its children.

There is an easier way to do this. You can search for processes using `pgrep` and kill them with `pkill`. The `-l` tells `pgrep` to list the process name as well. For example:

```
$ pgrep -l nginx
6950  nginx
14579 nginx...
```

And then go in for the kill with `sudo pkill nginx`. A further way to search for processes is using `ps` with the `aux` option as in `ps aux | grep nginx`. Easier, but you wouldn't have learned about the wonder of `top`.

Speaking HTTP

At this stage, your Web server software is hopefully up and running. If it did crash, you've restarted it, found out the reason and taken steps to prevent it from happening again.

You can now double-check your Web server is up and running by telnetting to port 80 from your laptop again. This time it should say "Connected" and then wait for your request. Web servers understand HTTP (hypertext transfer protocol). After a connection is established type `GET / HTTP/1.1` to tell the server you would like to `GET` (as opposed to `POST`) the home page `/` and that you speak version 1.1 of the protocol.

Press Enter and then type `Host`: followed by the host name. This line is only necessary on servers which host more than one website. HTTP does not know that you telnetted to www.smashingmagazine.com. As far as it is concerned, you telnetted to 80.72.139.101 and it needs to know which of its many websites you are interested in. Press Enter twice to make the request. You should get back a long stream of text and HTML:

```
$ telnet www.smashingmagazine.com 80
Trying 80.72.139.101...
Connected to www.smashingmagazine.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.smashingmagazine.com

HTTP/1.1 200 OK
Date: Thu, 09 May 2013 13:25:52 GMT
```

```
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.2.17
Content-Length: 25023
Content-Type: text/html

<html>
<head><...
```

The lines above are mostly HTTP headers. The `HTTP/1.1 200 OK` says that the server also speaks version 1.1 of HTTP and gives the successful HTTP response code 200. Other common responses are 500 for Internal Server Error and 404 for File Not Found. It then continues with the HTML. If the Connection header specified "keep-alive" then telnet will wait for your next request and you'll need to type `Control + ]` and then "quit" to exit. If the Connection header said "close" then it will finish by itself and say "Connection closed by foreign host" at the bottom.

FINDING YOUR WEBSITE

The 200 code means that your home page is okay, and you should be able to visit it in your browser. However, it may not show what you expected, and your fabulous Widget 3000 page may still be absent.

VIRTUAL HOSTS AND STREAMS

As mentioned above, many servers host multiple websites. One of these is the default website. It is the website you get when you visit the server by IP address http://80.72.139.101/ instead of by name, or when you leave off the `Host:` line in the HTTP request while telnetting. The rest of the websites are known as virtual hosts. Every one of these websites has a physical location on the server known as its document root. To further investigate your website woes, you need to discover its document root.

Fortunately and sensibly, most server management packages like Plesk store their virtually hosted websites according to their domain name, so you can usually just `find` directly on the domain name. The `/` in the command below tells `find` to search the whole file system, the `-type d` looks only for directories, and the `-name` part searches for any directories containing "smashingmagazine". The asterisks are wild cards. You'll need to either escape them `\*smashingmagazine\*` or put them in quotes `"*smashingmagazine*"`:

```
$ find / -type d -name "*smashingmagazine*"
find: '/var/run/cups/certs': Permission denied
find: '/var/run/PolicyKit': Permission denied
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

If you run this command as a normal unprivileged user, you will probably see lots of "Permission denied" as find tries to explore forbidden places. You are actually seeing two types of output here: `stdout` for standard output and `stderr` for standard error. They are called output streams and are confusingly mixed together.

You have already encountered the pipe symbol | for piping the output stream (`stdout`) of one command into the input stream (`stdin`) of another. The symbol > can redirect that output into a file. Try this command to send all the matches into a file called matches.txt:

```
$ find / -type d -name "*smashingmagazine*" > matches.txt
find: '/var/run/cups/certs': Permission denied
find: '/var/run/PolicyKit': Permission denied...
```

In this case, all the `stdout` is redirected into the file matches.txt and only the error output stream `stderr` is displayed on the screen. By adding the number 2 you can instead redirect `stderr` into a file and just display `stdout`:

```
$ find / -type d -name "*smashingmagazine*" 2> matcherrors.txt
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

There is a special file on Linux, UNIX and Mac computers which is basically a black hole where stuff gets sent and disappears. It's called `/dev/null`, so to only see `stdout` and ignore all errors:

```
$ find / -type d -name "*smashingmagazine*" 2> /dev/null
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

The end result is that this `find` command tells you roughly where your document root is. In Plesk, all the virtual hosts are generally stored within the `/var/www/vhosts` directory, with the document roots in `/var/www/vhosts/domain.com/httpdocs`.

THE LONG WAY

You can find the document root more accurately by looking through the configuration files. For Apache servers, you can find the default website's document root by looking through the main configuration file which is usually `/etc/apache2/apache2.conf` or `/etc/httpd/conf/httpd.conf`.

```
$ grep DocumentRoot /etc/httpd/conf/httpd.conf
DocumentRoot "/var/www/html"
```

Somewhere inside this conf file will also be an `Include` line which references other conf files, which may themselves include further conf files. To find the DocumentRoot for your virtual host, you'll need to search through them all. You can do this using `grep` and `find` but its a long command, so we will build it up gradually.

First, we will find all the files (because of the `-type f`) on the whole server (the `/`) whose names end in "conf" or "include". The `-type f` finds only files and the `-o` lets us look for files ending in "conf" or "include", with surrounding escaped parentheses. As above, the errors are banished into the ether:

```
$ find / -type f \( -name \*conf -o -name \*include \) 2> /dev/null
/var/spool/postfix/etc/resolv.conf
/var/some file with spaces.conf
/var/www/vhosts/myserv.com/conf/last_httpd.include...
```

This is not quite complete as any files with spaces will confuse the `grep` command we are about to attempt. To fix that you can pipe the output of the `find` command through the `sed` command which allows you to specify a regular expression. Regular expressions are a huge topic in their own right. In the command below, the `s/ /\\ /g` will replace all spaces with a slash followed by a space:

```
$ find / -type f \( -name \*conf -o -name \*include \) 2>/dev/null | sed
‘s/ /\\ /g’
/var/spool/postfix/etc/resolv.conf
/var/some\ file\ with\ spaces.conf
/var/www/vhosts/myserv.com/conf/last_httpd.include...
```

Now you can use a backtick to embed the results of that `find` command into a `grep` command. Using ` is different than | as it actually helps to build a command, rather than just manipulating its input. The `-H` option to `grep` tells it so show file names as well. So, now we will look for any reference to “smashingmagazine” in any conf files.

```
$ grep -H smashingmagazine `find / -type f \( -name \*conf -o -name \*in-
clude \) 2> /dev/null | sed ‘s/ /\\ /g’`
/var/www/vhosts/smashingmagazine.com/conf/last_httpd.include: ServerName
“smashingmagazine.com”...
```

This may take a few seconds to run. It is finding every conf file on the server and searching through all of them for “smashingmagazine”. It may reveal the DocumentRoot directly. If not, it will at least reveal the file where the ServerName or VirtualHost is defined. You can then use `grep` or `less` to look through that file for the DocumentRoot.

You can also use the `xargs` command to do the same thing. It also allows the output from one command to be embedded into another:

```
$ find / -type f \( -name \*conf -o -name \*include \) 2> /dev/null | sed
's/ /\\ /g' | xargs grep -H smashingmagazine
/var/www/vhosts/smashingmagazine.com/conf/last_httpd.include: ServerName
"smashingmagazine.com"...
$ grep DocumentRoot /var/www/vhosts/smashingmagazine.com/conf/last_httpd.
include
DocumentRoot "/var/www/vhosts/smashingmagazine.com/httpdocs"
```

The end result, hopefully, is that you've definitively found the document root for your website.

You can use a similar technique for nginx. It also has a main conf file, usually in `/etc/nginx/nginx.conf`, and it can also include other conf files. However its document root is just called "root".

APACHE CONTROL INTERFACE

With Apache, there is yet another way to find the right conf file, using the `apachectl` or newer `apache2ctl` command with the `-S` option.

```
$ apachectl -S
VirtualHost configuration:
80.72.139.101:80       is a NameVirtualHost
         default server default (/usr/local/psa/admin/conf/genera-
ted/13656495120.10089200_server.include:87)
         port 80 namevhost default (/usr/local/psa/admin/conf/genera-
ted/13656495120.10089200_server.include:87)
         port 80 namevhost www.smashingmagazine.com (/var/www/vhosts/
smashingmagazine.com/conf/last_httpd.include:10)...
```

If this whizzes by too fast, you can try piping the results through `grep`. It won't work, however, because `grep` only operates on `stdout` and for some reason `apachectl` outputs its information to `stderr`. So, you have to first direct `stderr` into `stdout` and then send it through `grep`. This is done by redirecting the error stream 2 into the output stream 1 with `2>&1`, like this:

```
$ apachectl -S 2>&1 | grep smashingmagazine
         port 80 namevhost smashingmagazine.com (/var/www/vhosts/
smashingmagazine.com/conf/13656495330.08077300_httpd.include:10)
```

This also reveals the conf file which contains the DocumentRoot for this website. As above further `grep` or `less` will reveal the DocumentRoot.

CHECKING THE DOCUMENT ROOT

Now that you've found the document root, you can snoop around to make sure it's alright. Change to the directory with `cd`:

```
$ cd /var/www/vhosts/smashingmagazine.com/httpdocs
bash: cd: /var/www/vhosts/smashingmagazine.com/httpdocs: No such file or
directory
```

If you get the error message "No such file or directory", that is bad news. Either the DocumentRoot has been incorrectly set or your whole website has been deleted. If it is there, you can list the files with `ls`. The `-a` also shows hidden files which start with a dot, and `-l` displays them in long format with permissions and dates:

```
$ ls -al
drwxrwxrwx  8 nobody  nogroup  4096 May  9 14:03 .
drwxr-xr-x 14 root    root     4096 Oct 13  2012 ..
```

Every folder will at least show these two entries. The single "." is for the current directory and ".." is for the parent directory. If that's all there is, then the directory is empty. While you're there, you can double-check you are in the correct place. Create a new file using `echo` and again using the > symbol to send the output to a file.

```
$ echo "<h1>My test file</h1>" > testfile.html
```

This will create a file called testfile.html containing a bit of HTML. You can use your browser or `telnet` or `curl` or `wget` to see if the file is where it should be.

```
$ curl http://www.smashingmagazine.com/testfile.html
<h1>My test file</h1>
```

If that worked, then well done, you have found your website! Remove that test file to clean up after yourself with `rm testfile.html` and keep going.

BACK UP AND RESTORE

The `tar` and `zip` commands can be used to back up and restore. If your website is missing, then restoring won't help you much unless you have previously backed up. So go back in time and backup your data with one of the commands below. To go back a whole day:

```
$ gobackintime 86400
It is now Sat May 10 20:30:57 BST 2013
```

Just kidding — but it would be nice! The `tar` command stands for tape archive and comes from the days when data was backed up on magnetic tapes. To create an archive of a directory, pass the `cfz` options to `tar` which will create a new archive in a file and then zip it in the gzip format.

```
$ tar cfz backupfile.tgz /var/www/vhosts/smashingmagazine.com/httpdocs
tar: Removing leading `/' from member names
```

All Mac and Linux computers support the `tar` command and most also have `zip`. To do the same with `zip`:

```
$ zip -r backupfile.zip /directory/to/backup
```

To see what an archive contains, run:

```
tar tfz backupfile.tgz
var/www/vhosts/smashingmagazine.com/httpdocs/
var/www/vhosts/smashingmagazine.com/httpdocs/.htaccess...
```

Or for `zip` format:

```
unzip -l backupfile.zip
Archive:  test.zip
  Length      Date    Time   Name
---------  ---------- -----  ----
        0  2012-05-28 00:33 var/www/vhosts/smashingmagazine.com/httpdocs
      234  2012-05-28 00:33 var/www/vhosts/smashingmagazine.com/httpdocs/.htaccess
```

Both `tar` and `zip` strip the leading slashes when they backup. So when you restore the files, they will be restored within the current directory. To restore them in the same location they were backed up from, first `cd to /`.

```
$ tar xfzv backupfile.tgz
var/www/vhosts/smashingmagazine.com/httpdocs/...
```

The "v" above stands for verbose and causes `tar` to show what it's doing. `zip` has a similar option:

```
$ unzip -v backupfile.zip
Archive:  backupfile.zip
 Length   Method    Size  Cmpr    Date    Time   CRC-32   Name
--------  ------  ------- ---- ---------- ----- --------  ----
       0  Stored        0   0% 2012-05-28 00:33 00000000  var/www/vhosts/
smashingmagazine.com/httpdocs/...
```

Website Errors

Let's assume your website hasn't actually disappeared. The next place to look is the error log file.

FINDING THE LOG FILE

When using a server management package like Plesk, each website probably has its own log file. You can find it by grepping for the word "log" in the conf file you identified above. The `-i` means case-insensitive.

```
$ grep -i log /var/www/vhosts/smashingmagazine.com/conf/last_httpd.include
    CustomLog /var/www/vhosts/smashingmagazine.com/statistics/logs/ac-
cess_log plesklog
    ErrorLog  "/var/www/vhosts/smashingmagazine.com/statistics/logs/er-
ror_log"...
```

There is also a server-wide log where any non-website-specific errors go. You can find this in the main conf file:

```
$ grep -i log /etc/apache2/apache2.conf
ErrorLog /var/log/apache2/error.log...
```

HTACCESS ERRORS

It is very easy to screw up a website. You can quite readily bring down a very big website by removing a single character from the .htaccess file. Apache uses the file .htaccess to provide last-minute configuration options for a website. It is most often used for URL rewriting rules. They look like this:

```
RewriteRule   ^products/.*/([0-9]+)$   products/view.php?id=$1   [L,QSA]
```

This rule says to rewrite any URL in the form "products/widget-3000/123" to the actual URL "products/view.php?id=123". The `L` means that this is the last rule to be applied and `QSA` means that Apache should attach any query string to the new URL. URL rewriting is often used for search engine optimization so that Web managers can get the name of the product into the URL without actually having to create a directory called "widget-3000". However, make a single typo and your whole website will give a 500 Internal Server Error.

The `tail` command will display the last 10 lines of a log file. Give it a `-1` to display the single last line instead. An .htaccess problem will look like this:

```
$ tail -1 /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Thu May 06 11:04:00 2013] [alert] [client 81.106.118.59] /var/www/
vhosts/smashingmagazine.com/httpdocs/.htaccess: Invalid command 'Rewi-
teRule', perhaps misspelled or defined by a module not included in the
server configuration.
```

You can `grep` for all of these types of errors:

```
$ grep alert /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Thu May 06 11:04:00 2013] [alert] [client 81.106.118.59]...
```

PHP PARSE AND RUNTIME ERRORS

Many websites use the LAMP combination: Linux, Apache, MySQL and PHP. A common reason for Web pages not showing up is that they contain a PHP error. Fortunately, these are quite easy to discover and pinpoint.

There are two broad classes of PHP errors: parse errors and runtime errors. Parse errors are syntax errors and include leaving off a semicolon or forgetting the `$` in front of a variable name. Running errors include undefined functions or referencing objects which don't exist.

Like .htaccess errors, parse errors will cause an HTML response code 500 for Internal Server Error, often with a completely blank HTML page. Runtime errors will give a successful HTML response of 200 and will show as much HTML as they have processed (and flushed) before the error happened. You can use `telnet` or `wget -S` or `curl -i` to get only the headers from a URL. So now, copy and paste your erroneous page into a command:

```
$ curl -i http://www.smashingmagazine.com/products/widget-3000/123
HTTP/1.0 500 Internal Server Error
Date: Sun, 12 May 2013 17:44:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 0
```

```
Connection: close
Content-Type: text/html
```

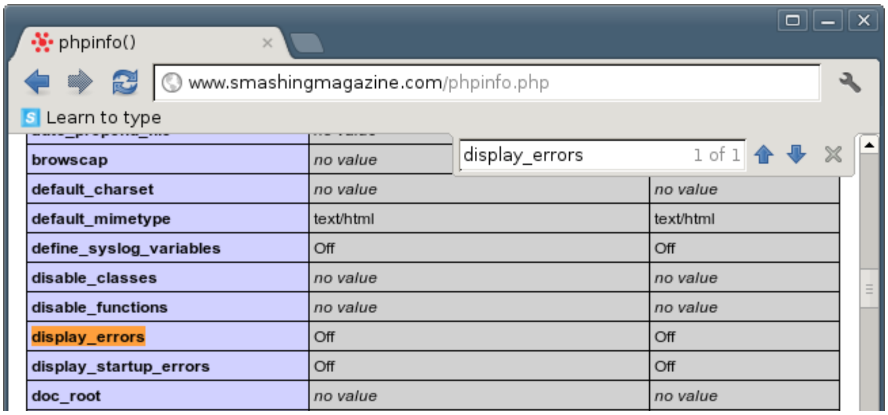
PHP ERROR SETTINGS

To find the exact error, you need to make sure errors are being reported in the log file.

There are several PHP settings which cover errors. `display_errors` determines if errors are shown to the website visitor or not, and `log_errors` says whether they will appear in log files. `error_reporting` specifies the types of errors that are reported: only fatal errors, for example, or warnings and notices as well. All of these can be set in a configuration file, in .htaccess or within the PHP script itself.

You can find out your current settings by running the PHP function `phpinfo`. Create a PHP file which calls the function and visit it in your browser:

```
$ echo “<?php phpinfo()?>” > /var/www/vhosts/smashingmagazine.com/http-
docs/phpinfo.php
```



browscap	no value	
default_charset	no value	no value
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_classes	no value	no value
disable_functions	no value	no value
display_errors	Off	Off
display_startup_errors	Off	Off
doc_root	no value	no value

`phpinfo` function showing configuration settings.

The two columns show the website and server-wide settings. This shows that `display_errors` is off, which is good, because it should be off on live websites. It means that no PHP errors will ever be seen by the casual visitor. `log_errors` on the other hand should be on. It is very handy for debugging PHP issues.

The `error_reporting` value is 30719. This number represents bit flags or bit fields. This is a technique for storing multiple yes/no values in a single number. In PHP there are a series of constants representing different types of errors.[10] For example, the constant `E_ERROR` is for fatal errors and has the value 1; `E_WARNING` is for warnings and equals 2; `E_PARSE` is for parsing or syntax errors and has the value 4. These values are all powers of two and can be safely added together. So the number 7 means that all three types of errors should be reported, as `E_ERROR` + `E_WARNING` + `E_PARSE` = 7. A value of 5 will only report `E_ERROR` + `E_PARSE`.

In reality, there are 16 types of errors from 1 for `E_ERROR` to 16384 for `E_USER_DEPRECATED`. You can type "30719 in binary" into Google and it will give you the binary equivalent: 0b111011111111111. This means that all errors are switched on except the twelfth, which is `E_STRICT`. This particular setup has also been given a constant `E_ALL` = `E_ERROR` + `E_WARNING` + `E_PARSE` + etc = 30719. From PHP version 5.4.0, `E_ALL` is actually 32767 which includes all the errors include `E_STRICT`.

If your `error_reporting` setting is 0, then no errors will show up in the log file. You can change this setting in the file php.ini, but then you have to restart Apache to make it have an effect. An easier way to change this setting in Apache is to add a line in a file called .htaccess in your document root: `php_value error_reporting 30719`.

Or you can do that on the command line, using the double arrow which appends to an existing file or creates the file if it doesn't exist:

```
$ echo "php_value error_reporting 30719" >> .htaccess
$ echo "php_value log_errors On" >> .htaccess
```

10 "Predefined Constants", PHP.net. http://smashed.by/errorfunc

Refresh your erroneous Web page. If there is a PHP error in your page it should now show up in the error log. You can grep the log for all PHP errors:

```
grep PHP /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Sun May 12 18:19:09 2013] [error] [client 81.106.118.59] PHP Notice:
Undefined variable: total in /var/www/vhosts/smashingmagazine.com/http-
docs/products/view.php on line 10...
```

If you have referenced variables or array indices before assigning them values, you may see thousands of PHP notices like the one above. It happens when you do things like `<? $total = $total + 1 ?>` without initially setting `$total` to 0. They are useful for finding potential bugs, but they are not show stoppers. Your website should work anyway.

You may have so many notices and warnings like this that the real errors get lost. You can change your `error_reporting` to 5 to only show `E_ERROR` and `E_PARSE` or you can `grep` specifically for those types of errors. It is very common to chain `grep` commands together like this when you want to filter by multiple things. The `-e` option below tells the second `grep` to use a regular expression. This command finds all log entries containing "PHP" and either "Parse" or "Fatal".

```
$ grep PHP /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
| grep -e "Parse\|Fatal"
[Thu Jul 19 12:26:23 2012] [error] [client 81.106.118.59] PHP Fatal er-
ror:  Class 'Product' not found in /var/www/vhosts/smashingmagazine.com/
httpdocs/library/class.product.php on line 698
[Sun May 12 18:16:21 2013] [error] [client 81.106.118.59] PHP Parse er-
ror:  syntax error, unexpected T_STRING in /var/www/vhosts/smashingmaga-
zine.com/httpdocs/products/view.php on line 100...
```

SEEING ERRORS IN THE BROWSER

If you are tracing a runtime error rather than a parse error, you can also change the `error_reporting` setting directly in PHP. And you can quickly turn display_errors on, so you will see the error directly in your browser. This makes debugging quicker, but means everyone else can see the error too.

Add this line to the top of your PHP page:

```
<? ini_set ('display_errors', 1); error_reporting (E_ERROR | E_WARNING); ?>
```

These two functions change the two PHP settings. The | in the `error_reporting` call is a bit OR operator. It effectively does the same as the + above but operates on bits, so is the correct operator to use with bit flags.

Any fatal errors or warnings later in the PHP page will now be shown directly in the browser. This technique won't work for parse errors as none of the page will run if there's a parse error.

BIT FLAGS

Using bit flags for `error_reporting` avoids having 15 separate arguments to the function for each type of error. Bit flags can also be useful in your own code. To use them, you need to define some constants, use the bit OR operator | when calling the function and the bit AND operator `&` within the function. Here's a simple PHP example using bit flags to tell a function called showproduct which product properties to display:

```
<?
define ('PRODUCT_NAME', 1);
define ('PRODUCT_PRICE', 2);
function showproduct ($product, $flags) {
  if ($flags & PRODUCT_NAME) echo $product['name'];
  if ($flags & PRODUCT_PRICE) echo ': $' . $product['price'];
}
$product = array ('name'=>'Widget 3000', 'price'=>10);
showproduct ($product, PRODUCT_NAME | PRODUCT_PRICE);
?>
```

This will display "Widget 3000: $10" in the browser.

INFINITE LOOPS

PHP's error reporting may struggle with one class of error: an infinite loop. A loop may just keep executing until it hits PHP's time limit, which is usually

30 seconds (PHP's `max_execution_time` setting), causing a fatal error. Or if the loop allocates new variables or calls functions, it may keep going until PHP runs out of workable memory (PHP's `memory_limit` setting).

It may, however, cause the Apache child process to crash, which means nothing will get reported, and you'll just see a blank or partial page. This type of error is increasingly rare, as PHP and Apache are now very mature and can detect and handle runaway problems like this. But if you are about to bang your head against the wall in frustration because none of the above has worked, then give it some consideration. Deep within your code, you may have a function which calls some other function, which calls the original function in an infinite recursion.

DEBUGGERS

If you've gotten this far, and your page is still not showing up, then you're entering more difficult territory. Your PHP may be executing validly and doing everything it should, but there's some logical error in your programming. For quick debugging you can `var_dump` variables to the browser, perhaps wrapping them in an `if` statement so that only your IP address sees them:

```
<? if ($_SERVER['REMOTE_ADDR'] == '85.106.118.199') var_dump ($product); ?>
```

This method will narrow down an error but it is ungraceful and error-prone, so you might consider a debugging tool such as Xdebug or FirePHP. They can provide masses of information, and can also run invisibly to the user, saving their output to a log file. Xdebug can be used like this:

```
<?
ini_set ('xdebug.collect_params', 1);
xdebug_start_trace ('/tmp/xdebugtrace');
echo "This will get traced.";
xdebug_stop_trace();
?>
```

This bit of code logs all function calls and arguments to the file `/tmp/xdebugtrace.txt`. It displays even more information when there is a PHP notice or error. However, the overhead may not be suitable for a live environment, and it needs to be installed on the server, so it's probably not available in most hosting environments.

FirePHP, on the other hand, is a PHP library that interacts with an add-on to Firebug, a plugin for Firefox. You can output debugging information and stack traces from PHP to the Firebug console.

Security Issues

By this point, you should have some HTML reaching your browser. If it's not what you expect, then there's a chance that your website has been compromised. Don't take it personally (at first). There are many types of hacks and most of them are automated. Someone clever but unscrupulous has written a program which detects vulnerabilities and exploits them. The purpose of the exploit may simply be to send spam, or to use your server as part of a larger attack on a more specific target (a DDoS).

SERVER HACKS

Operating systems are very complex pieces of software. They may be built from millions of lines of programming code. They are quite likely to have loopholes where sending the wrong message at just the wrong time will cause some kind of blip which allows someone or something to gain entry. That's why Microsoft, Apple, Ubuntu and others are constantly releasing updates.

Similarly, Apache, nginx, IIS and all the other software on a typical server is complicated. The best thing you can do is keep it up to date with the latest patches. Most good hosts will do this for you.

A hacker can use these flaws to log in to your server and engineer themselves a terminal session. They may initially gain access as an unprivileged user and then try a further hack to become the root user. You should make this as hard as possible by using good passwords, restrictive permissions, and being careful to run software (like Apache) as an unprivileged user.

If someone does gain access, they may leave behind a bit of software which they can later use to take control of your server. This may be detectable by an antivirus scanner or something like the Rootkit Hunter, which looks for anomalies like unexpected hidden files. But there are also a few things you can do if you suspect an intrusion.

The `w` command shows who is currently logged in to a server and what they are doing:

```
$ w
 20:44:32 up 44 days,  7:51,  2 users,  load average: 0.07, 0.03, 0.05
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    cpc1-brig17-2-0- 17:54    1:02m  0.15s  0.13s -bash
root     pts/1    cpc1-brig17-2-0- 20:44    0.00s  0.02s  0.00s w...
```

The `last` command shows who has logged in recently in date order. Pipe it through `head` to show only the first 10 lines.

```
$ last
paul     pts/0        :0.0             Sun May 12 17:21   still logged in
paul     tty7         :0               Sun May 12 17:20   still logged in
reboot   system boot  2.6.32-41-386    Sun May 12 17:18 - 20:48  (03:29)
fred     tty7         :0               Sat May 11 10:10 - down   (01:12)
```

It tells you who has logged in and for how long, plus any terminal session they have open. `down` means until the server shut down. Look for unexpected entries and consult your host or a security expert if you are in doubt.

PHP HACKS

More common are hackers who gain entry though vulnerabilities in PHP scripts, especially popular content management systems like WordPress. Anybody can write a plugin for WordPress and, if it's useful, people will install it. When writing a plugin, most developers think primarily about the functionality and little about security. And because WordPress allows file uploading, hackers who find vulnerabilities can use them to upload their own PHP scripts and later take control of a computer.

These PHP scripts can use the PHP `mail` function to send out spam on demand, but they can also try to execute commands in much the same way as you can via a terminal session. PHP can execute commands with its `exec` or `system` functions. If you do not need to use these functions, it is advisable to disable them. You can do this by adding the `disable_functions` directive to your server's php.ini file (or php5.ini for PHP 5) or to the file php.ini within your document root. If you search for "php disable functions" in Google, you will find a whole list of functions which should be disabled in this way:

```
disable_functions=fpassthru,crack_check,crack_close...
```

A quick check you can make for this type of hack is to look for all PHP files modified recently and make sure there are no anomalies. The `-mtime -1` option tells `find` to only consider files modified within the last day. There's also `-mmin` for minutes. This command searches all websites within /var/www/vhosts for recently modified files ending in "php" or "inc":

```
$ find /var/www/vhosts -mtime -1 \( -name \*php -o -name \*inc \) -printf
"%t %h/%f\n"
Sun May 12 21:20:17.0000000000 2013 /var/www/vhosts/smashingmagazine.com/
httpdocs/products/view.php
```

PHP hacks are difficult to detect because they are designed to not stick out. One method hackers use is to gzip their PHP and then encode it as base64. In that case, you may have a PHP file on your system with something like this in it:

```
eval(gzinflate(base64_decode('HJ3HkqNQEkU/ZzqCBd4t8V4YAQI2E3jvPV8...
```

Another method is to encode text within variables and then combine them and evaluate them:

```
$unywlbxc = “ uwzsebpgi840hk2a jf”;
$hivjytmne = “  jqs9m4y 1znp0  “;
eval ( “m”.”i”. “croti”...
```

Both these methods use the PHP `eval` function, so you can use `grep` to look for `eval`. Using a regular expression with `\beval\b` means that the word “eval” must have a word boundary before and after it, which prevents it being found in the middle of words. You can combine this with the `find` command above and pipe through `less` for easy reading:

```
$ find /var/www/vhosts -mtime -1 \( -name \*php -o -name \*inc \) | sed
‘s/ /\\ /g’ | xargs grep -H -e “\beval\b” | less
/var/www/vhosts/config.php:eval(gzinflate(base64_decode(’HJ3HkqNQE...
```

If you do find this type of hack in your website, try to discover how they got in before completely removing all the tainted files.

ACCESS LOGS

Along with error logs, Apache also keeps access logs. You can browse these for suspicious activity. For example, if you found a PHP hack inside an innocuous looking file called test.php, you can look for all activity related to that file. The access log usually sits alongside the error log and is specified with the `CustomLog` directive in Apache configuration files. It contains the IP address, date and file requested. Search through it with `grep`:

```
$ grep -e “\(GET\|POST\) /test.php” /var/www/vhosts/smashingmagazine.com/
statistics/logs/error_log
70.1.5.12 - - [12/May/2013:20:10:49 +0100] “GET /test.php HTTP/1.1” 200
1707 “-” “Mozilla/5.0 (X11; Ubuntu; Linux i686;...
```

This looks for GET and POST requests for the file test.php. It provides you with an IP address, so you can now look for all other access by this address, and also look for a specific date:

```
$ grep 70.1.5.12 /var/www/vhosts/smashingmagazine.com/statistics/logs/er-
ror_log | grep "12/May/2013"
70.1.5.12 - - [12/May/2013:20:10:49 +0100] "GET /products/view.
php?something HTTP/1.1" 200 1707 "-"...
70.1.5.12 - - [12/May/2013:20:10:49 +0100] "GET /test.php HTTP/1.1" 200
1707 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686;...
```

This kind of debugging can be very useful for normal website errors too. If you have a feedback form on your website, add the user's IP address to the message. If someone reports an error, you can later look through the logs to see what they have been up to. This is far better than relying on vague secondhand information about reported problems.

It can also be useful for detecting SQL injection attacks, whereby hackers try to extract details from your database by fooling your database retrieval functions. This often involves a lot of trial and error. You could send yourself an email whenever a database query goes wrong and include the user's IP address. You can then cross-reference with the logs to see what else they have tried.

LAST RESORTS

William Edward Hickson is credited with popularizing the saying: "If at first you don't succeed, try, try, try again."[11] Hickson was a British educational writer living in early Victorian times. His advice is not appropriate for the modern Web developer, lying in bed on a Saturday morning, drowning in frustration, staring at a blank Web page, preparing to chuck an expensive laptop against a brick wall.

11 Oxford Dictionary of Quotations (3rd edition), Oxford University Press, 1979

You've now been through all the advice above. You've checked that the world hasn't ended, verified your broadband box, tested the Internet and reached your server. You've looked for hardware problems and software problems, and delved into the PHP code. But somehow or other, your Widget 3000 is still not there. The next thing to do is...

HAVE BREAKFAST

Get out of bed and take your mind off the problem for a little while. Have some toast, a bowl of cereal, something to drink. Maybe even indulge in a shower. Try that new lavender and citrus shampoo you bought by mistake. While you're doing all this, your subconscious is busily working on the website issue, and may unexpectedly pop a solution into your thoughts. If so, give it a try. If not...

ASK FOR HELP

Check the level of support that you are entitled to by your hosting company. If you are paying $10 per month, it's probably not much. You may be able to get them to cast a vague glance in your direction within the next 72 hours. If it's substantially more, they may log in and have a look within the next few minutes. They should be able to help with hardware or software issues. They won't help with Web programming issues. Alternatively, ring a colleague or freelancer. If you are still stuck...

PREPARE

...to release some nervous energy. Find one of those squidgy balls that you can squeeze mercilessly in your hands, or a couple pencils to use as drumsticks, or a pack of cigarettes and a pot full of coffee. And then try the last resort to any computing problem...

REBOOT

When your laptop or desktop goes wrong, a common solution is to reboot it. You can try the same trick on your Web server. This is a quite risky. Firstly, it may not solve the problem. If it's a PHP error, then nothing will change. If, however, your issue is caused by some obscure piece of software becoming unresponsive, then it may well help, though it may not fix the problem permanently. The same thing may happen next week.

Secondly, if the reboot fails then you will be really stuck. If the server shuts down but fails to start back up again, then someone may have to go and press the power button on the physical machine. That someone is an employee of your hosting company, and they may be enjoying their breakfast too, in a nice comfortable office somewhere. They may have left their jumper at home. They may not want to enter the air-conditioned bunker where all the servers are kept. You will be thoroughly dependent on their response time.

Given all the risks, the command is:

```
$ sudo /sbin/reboot
Broadcast message from admin@thisserver.com (/dev/pts/1) at 13:21 ...
The system is going down for reboot now.
```

The `reboot` command is really just a wrapper for `/sbin/shutdown -r now`. It causes the server to shut down and then restart. That may take a few minutes. Soon after issuing the command above your SSH session will come to an abrupt end. You will then be left for a few nervous minutes wondering if it will come back up again. Use the tools you prepared above.

While you are waiting, you can issue a `ping` to see if and when your server comes back. On Windows use `ping -t` for an indefinite ping:

```
$ ping www.smashingmagazine.com
PING www.smashingmagazine.com (80.72.139.101) 56(84) bytes of data.
Request timeout for icmp_seq 0
Request timeout for icmp_seq 0
Request timeout for icmp_seq 0...
64 bytes from www.smashingmagazine.com (80.72.139.101): icmp_seq=1 ttl=52
time=39.4 ms
64 bytes from www.smashingmagazine.com (80.72.139.101): icmp_seq=1 ttl=52
time=32.4 ms...
```

You can breathe a sigh of relief when `ping` finally responds. Wait a couple more minutes and you'll be able to use SSH again and then try to view the Widget 3000 in your Web browser.

Conclusion

This has been an epic journey, from the end of the world to a single misplaced character in a file. Hopefully, it will help you through the initial few minutes of panic when you wake up one morning and the beautiful product page you created last night is gone.

Some of the reasons and solutions above are very rare. The most likely cause is simply a slight malfunction in your broadband box. Running out of disk space or getting hacked are the only other things that are in any way likely to happen in the middle of the night when nobody else is working on the website. But throw in other developers, server administrators and enthusiastic clients — and anything is possible. Good luck!

ABOUT THE AUTHOR

Paul Tero is a website and computer programmer living with his family in Brighton, England. He grew up in California and studied computer science at UC Berkeley, before moving to Brighton in 1997. He currently mostly works for Existor, the company which makes Cleverbot, a quirky artificial entity which talks back. And he writes occasionally and enjoyably for Smashing Magazine.

ABOUT THE REVIEWER

Ben Dowling is a British software engineer who lives in Mountain View, California. He loves writing code, learning, and launching new products. He is currently a software engineer at Facebook. Prior to that he was lead server engineer at Lightbox.com, co-founder of Geomium, and prior to that a founding engineer at Mendeley. He blogs at coderholic.com and also tweets as @coderholic.

ABOUT THE REVIEWER

Sergey Chikuyonok is a Russian front-end Web developer and writer with a big passion for optimization: from images and JavaScript to working processes and time-savings on coding. He is the developer of the Emmet (ex-Zen Coding) tool.