Glen Campbell | Yahoo! Inc.

FAIL

The best ways to bring down your website (and How to avoid them)

About You

- You run a website
- The website has visitors
- You care what the visitors see when they visit the site

About Yahoo!

- Yahoo! runs websites.
- If the websites aren't running, Yahoo! doesn't make any money.

About Me

- Built Yahoo! Tech, Yahoo! Shine, Yahoo! News.
- Failed with all of them.
- You can consider me an expert on website failure.

Let’s get started

What is FAIL?

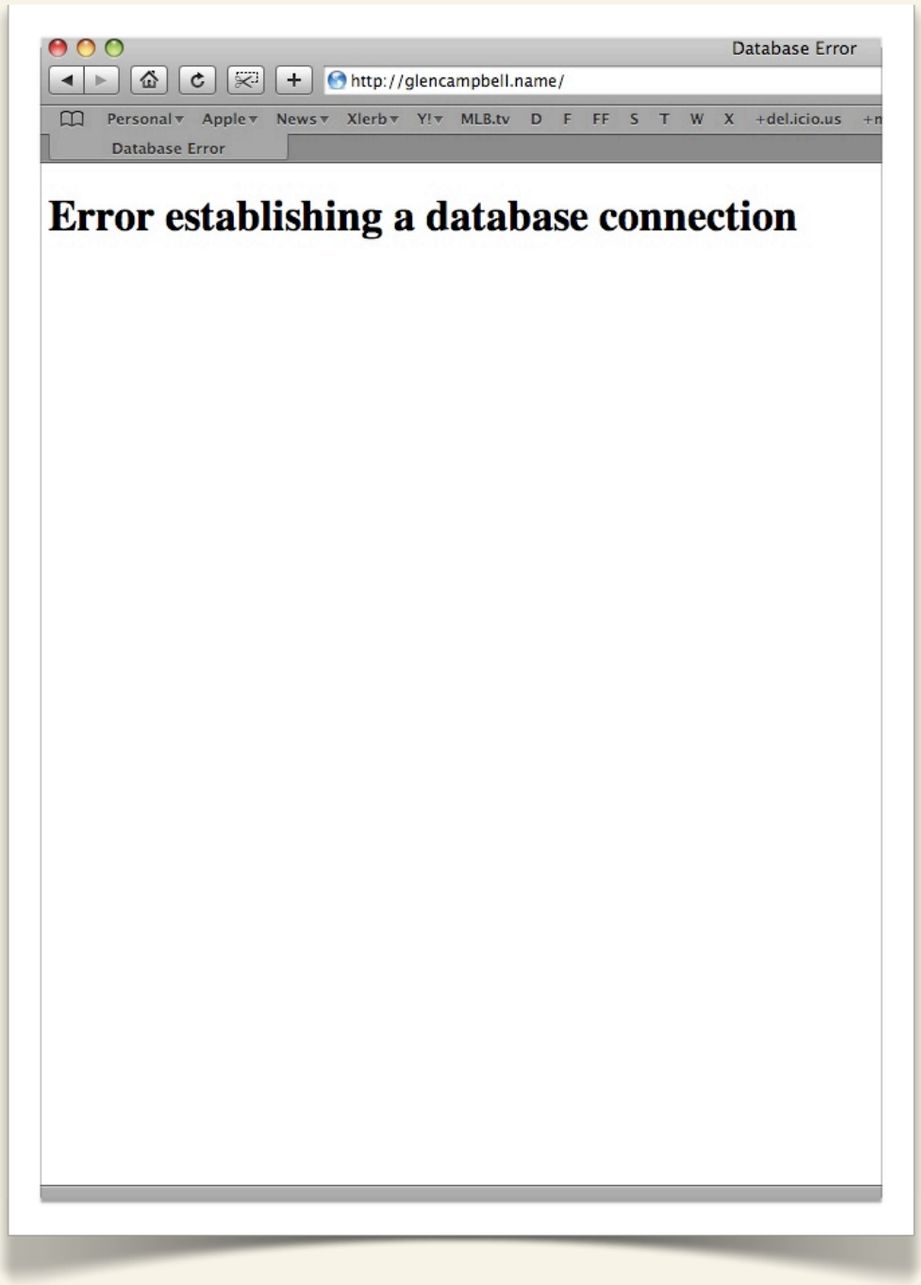
Someone clicks on a link to the site.

They see something other than the site.

Photo by Jeremy Zawodny

Why is FAIL?

- Out of bandwidth
- Out of Apache connections
- Out of sockets
- Out of disk
- Out of CPU
- Out of luck



Fundamentally, failure is because of a lack of resources.

Some Terms

- SPOF: Single Point Of Failure
- BCP: Business Continuity Planning
- XSS: Cross-Site Scripting
- SLA: Service-Level Agreement

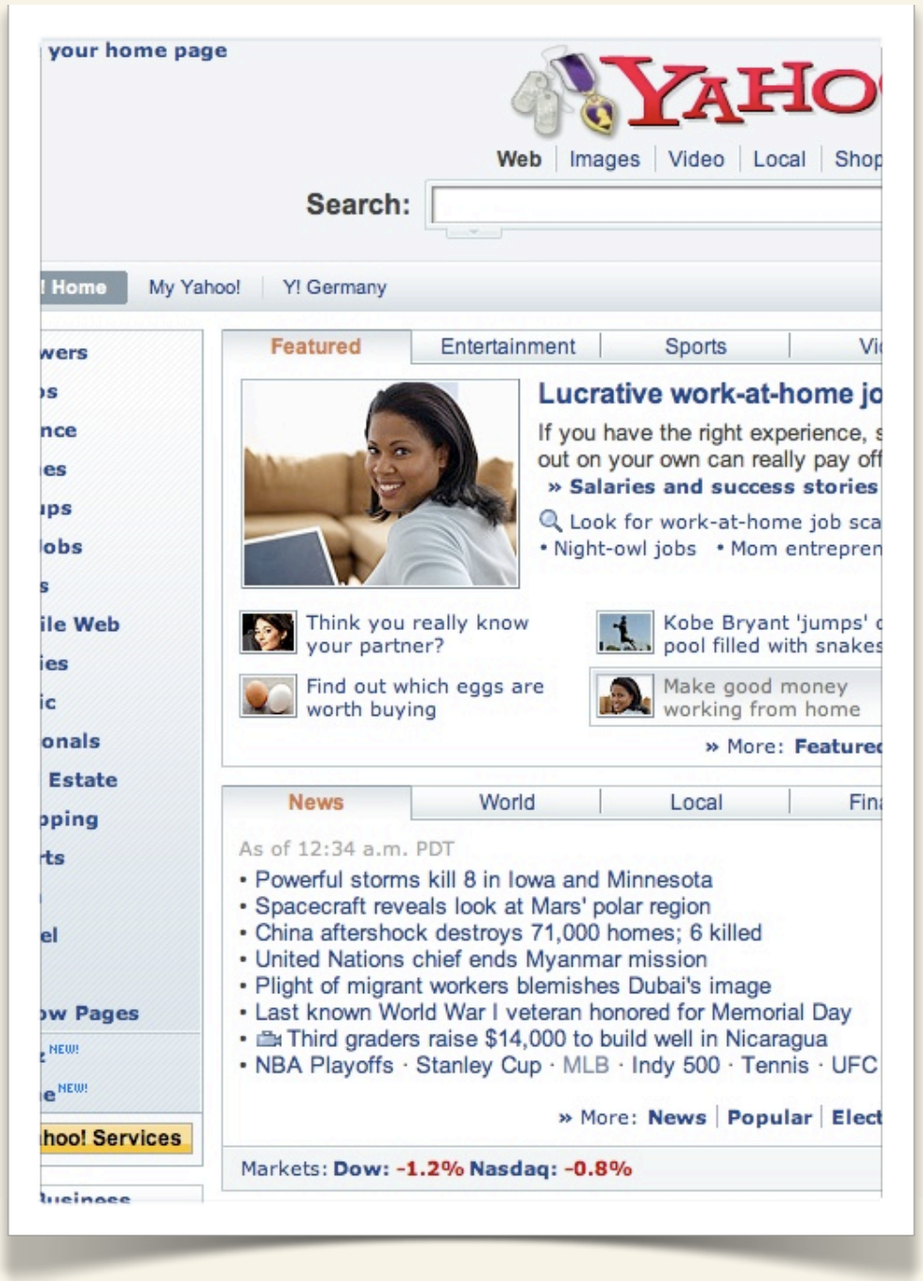
How to avoid FAIL

- Multiple techniques to avoid FAIL
- This presentation selects the best per incident
- Always remove SPOF
- Note that “avoidance” is better than “recovery”

Be popular

Example

- Get linked to from Yahoo!
- 10-100x Digg, Slashdot
- This is a “high quality problem”



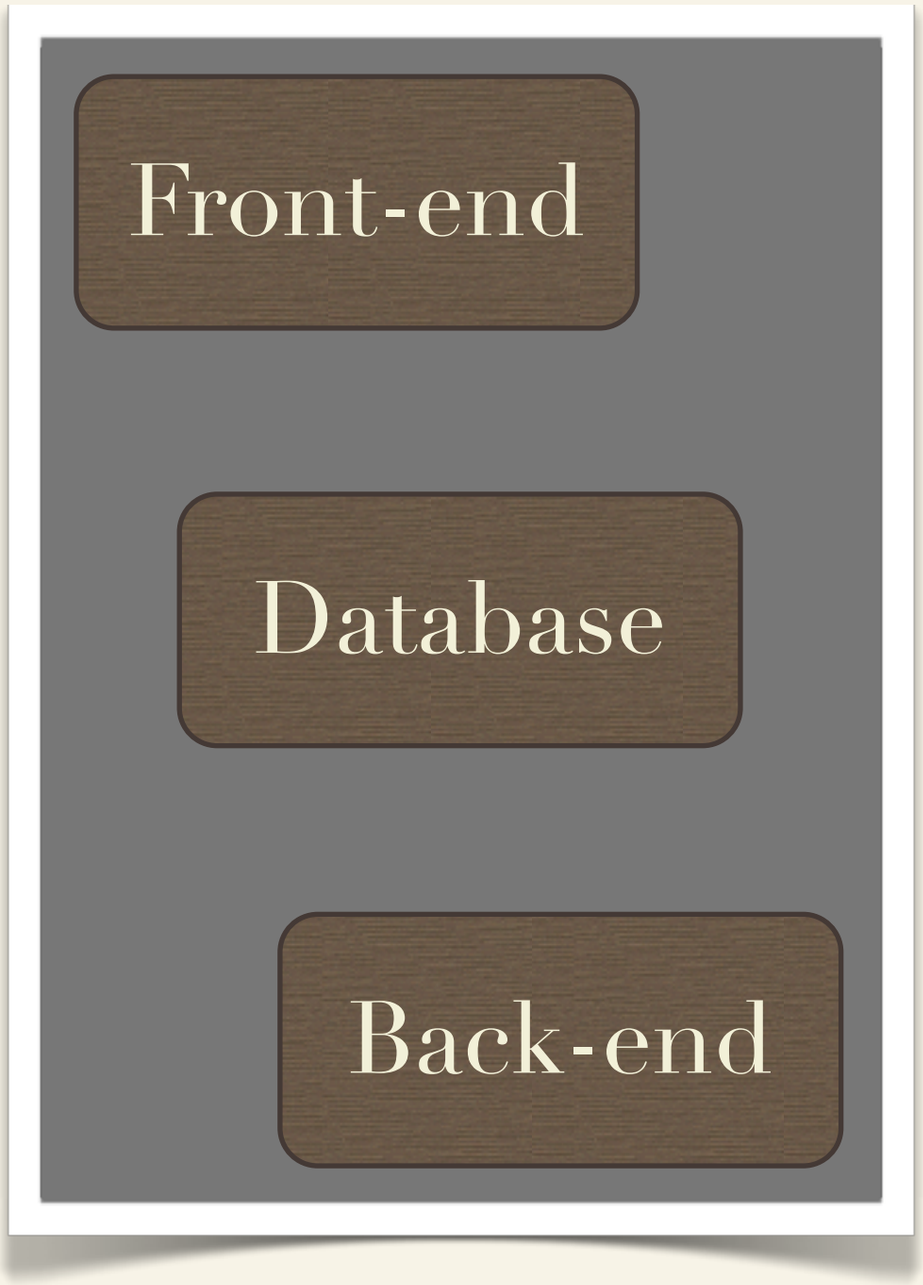
How to avoid

- Be Prepared—Scale Everything:
  - Servers
  - Database
  - Images
- Test, test, and test again:
  - HTTPLoad

Trust your hardware

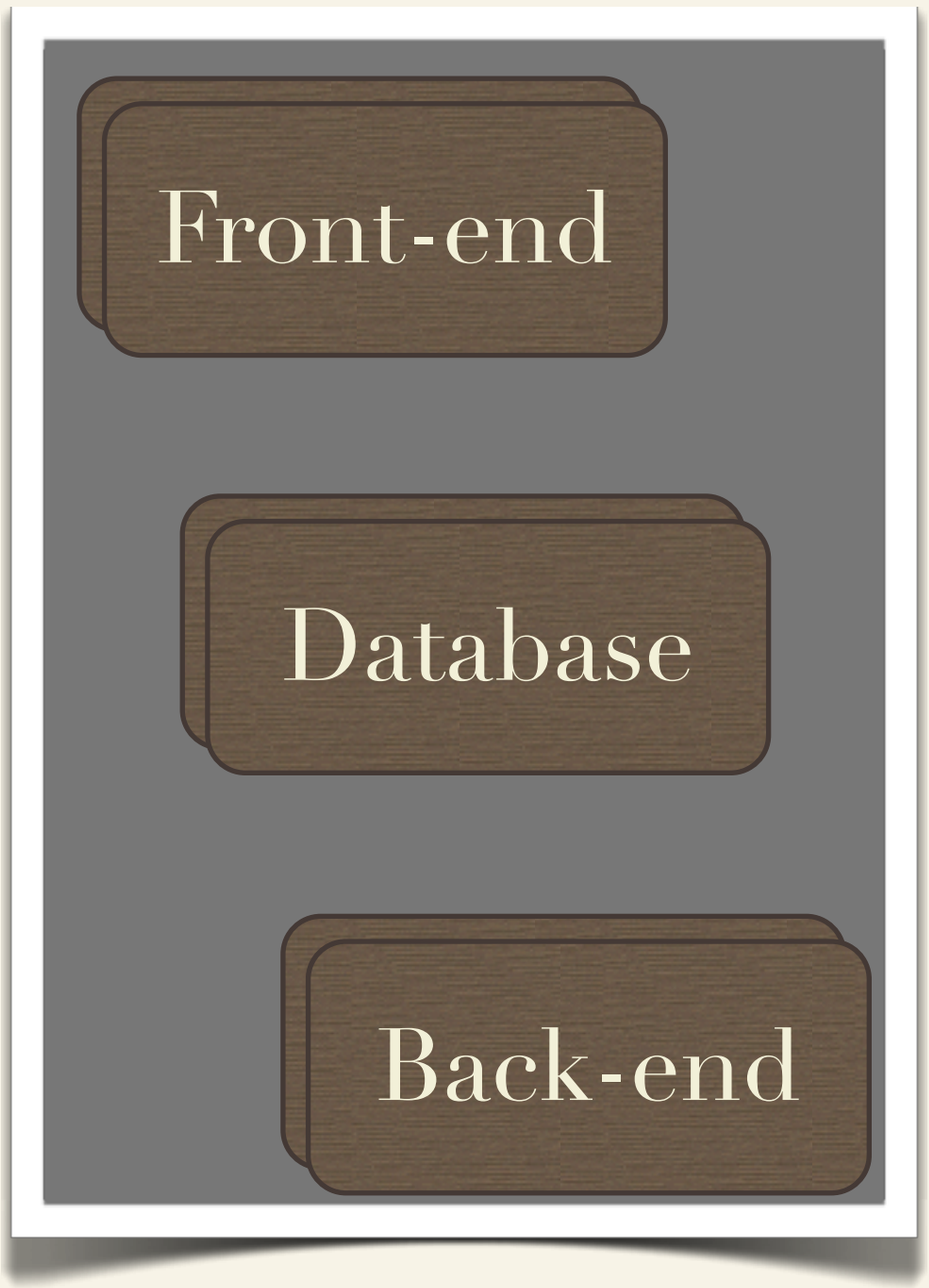
Website Architecture

- Front-end server
- Database server
- Feed processor
- Ask yourself: what happens if one of them goes down?



Linear Scaling

- The ability to increase capacity merely by adding additional hardware.



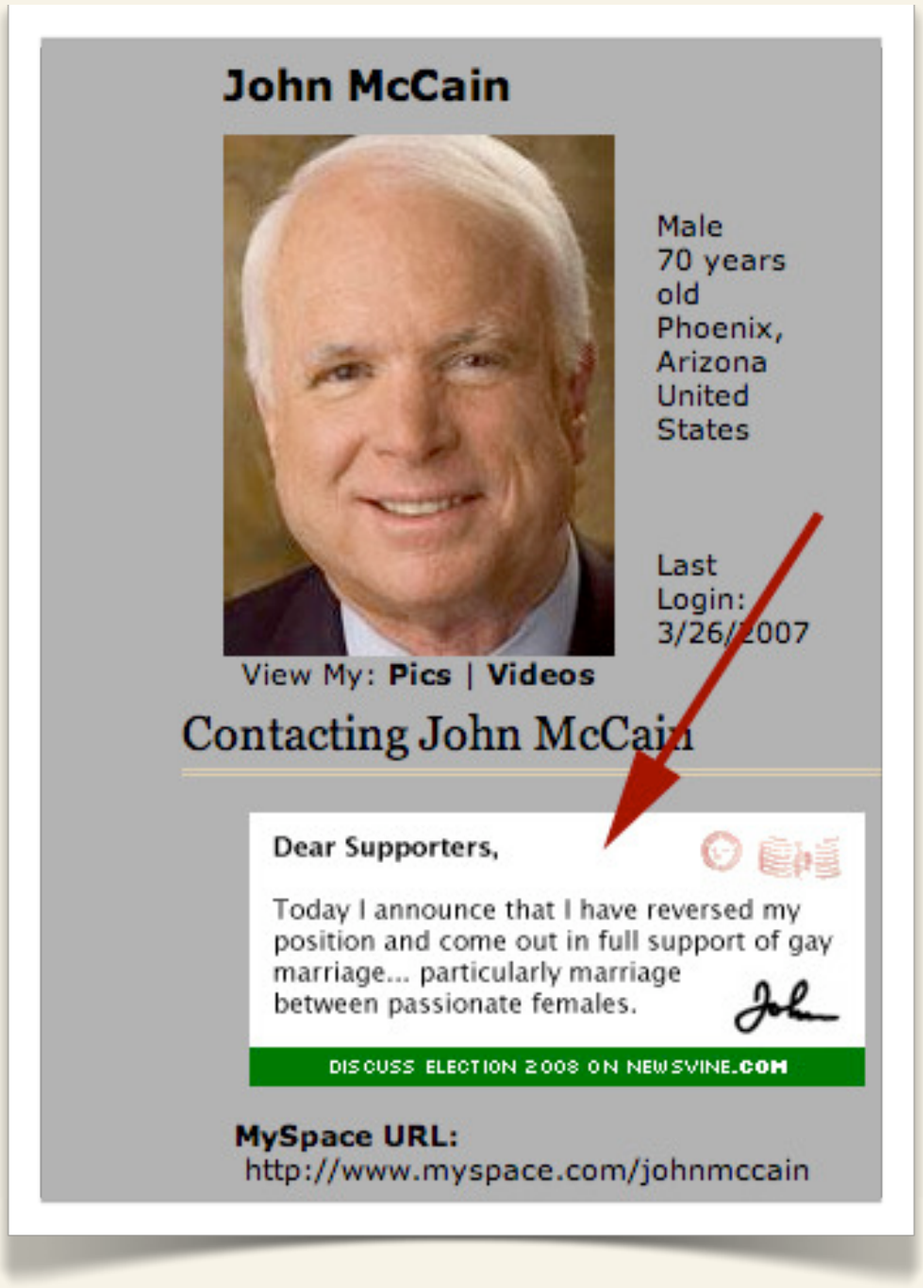
Scalability

- Depends on *architecture*, not programming language.
- Easier to build in than retrofit.
- Need multiple layers of redundancy at *all* levels.
- Single point of failure = total failure.

Don’t serve your own images

Example

- Linking to images on other sites gives those sites control of your images
- Results can be funny, however



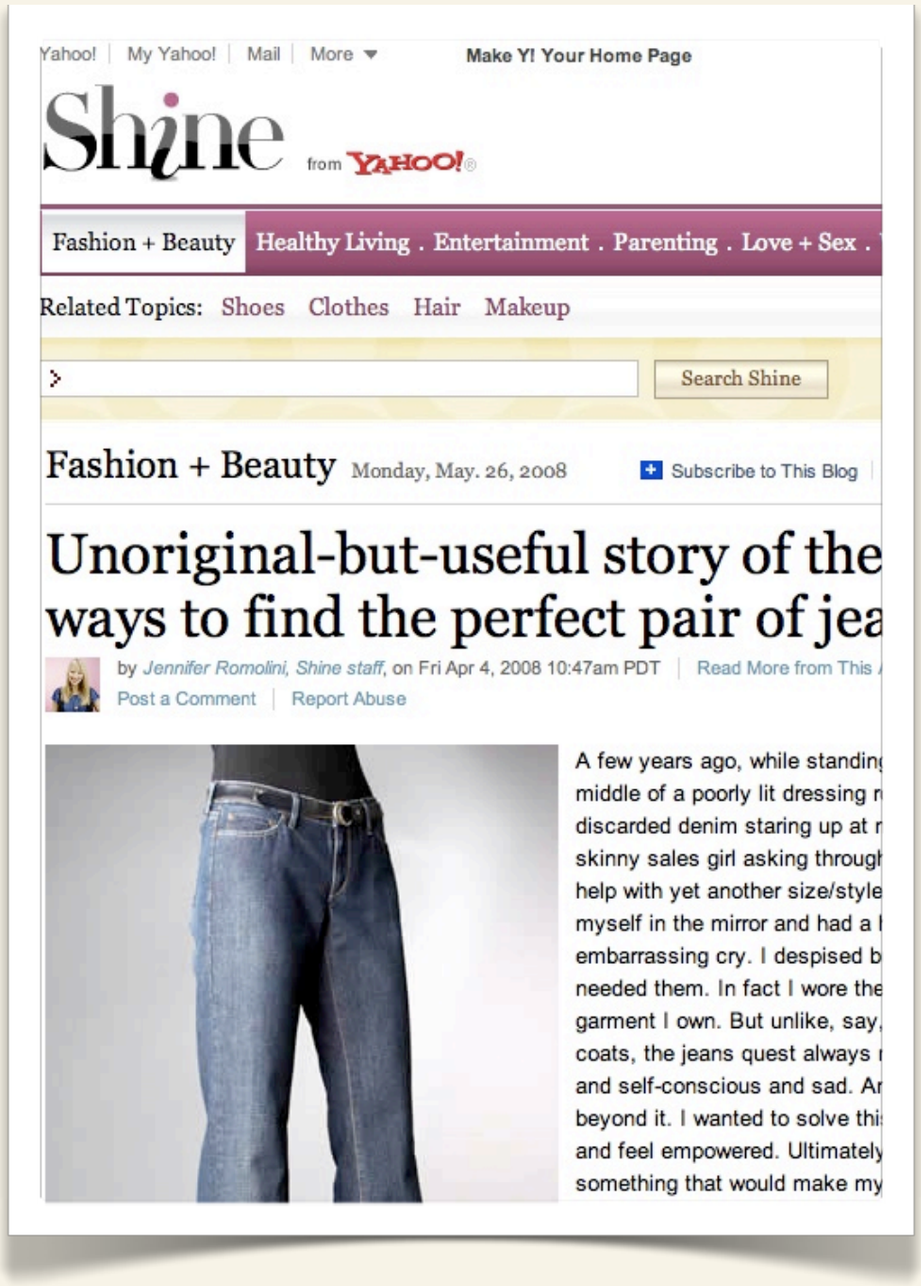
How to avoid

- Um, don't hot-link images
- For user-generated content, put controls in place that restrict the image source
- Validate `<img src="">` in feeds

Serve someone else's images

Example

- Yahoo! Shine used image from clothing distributor
- Our traffic brought down their site



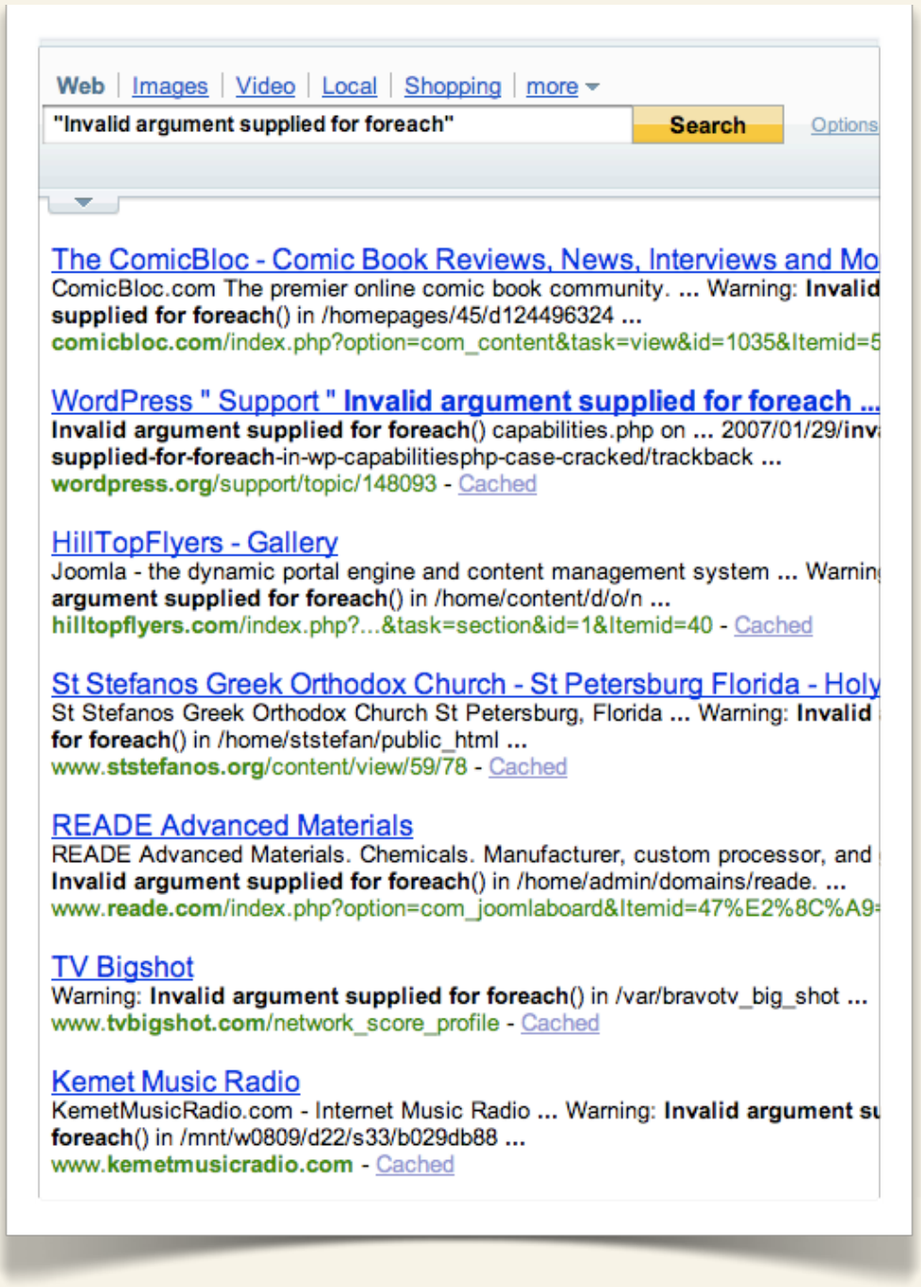
How to avoid

- Check your referrer logs
- Use scripts to block images to non-local referrers
- Use edge-caching (Akamai, for example)

Put bugs in your code

Example

- There are 5.9+ million pages with “Invalid argument supplied for foreach” errors on them



Search the web for your favorite error message.

Next, search the web for your favorite error message ON YOUR SITE.

How to avoid

- Review your engineering processes
- Time to start programming like grown-ups
- Bugs should not be caught by testing; 85% of all bugs should be caught prior to testing in the development process.

Review fastcompany.com article on the Shuttle flight control systems programming team.

Delete all your files

Example

```
Open ETR: Unknown-filer-flickr03 - volumes 1 - 6
accidentally deleted. Storage Ops working to
recover volumes. Outage page has been put up on
flickr.com.
Srv. Impact: 15 Mins
Evnt Dur: 15 Mins
Dispatched: Storage Ops, Flickr Devel -YNOC,x5555
```

How to avoid

- No manual commands on production: only pre-tested scripts
- All scripts have to go through QA process and testing, even in “urgent” situations
- Scripts have known, understood effects

Trust your enemies

Security is a huge issue, and could probably deserve a conference by itself. This specific issue is security risks caused by user-submitted content, not by a direct, sophisticated attack.

By “enemies,” I mean “enemies.” They are among your valued users, and you can’t tell them apart.

Example

- April 21, 2008—The campaign site of presidential candidate Barack Obama redirected visitors to the site of Hillary Clinton (see video at link).
  http://tinyurl.com/4am929
- Note that sites with “social” features are usually the most vulnerable.

A Warning

- Your site is not secure.
- Your site will never be secure.
- Your only choice risk versus ignorance.
- Banks have begun to disclaim *any* responsibility for online transactions.

Are you smarter than the millions of people who are out there spending 18 hours per day trying to devise new ways of breaking your security?

Risk vs. Ignorance: in essence, you can choose to assume some reasonable level of risk and plan for contingencies; or, you can pretend like there's no risk and be surprised.

Banks: new scheme: XSS downloads a program; the program waits until you connect to your bank; as you're connected (as you), it initiates a transfer of $4.00 to another account, then rewrites the HTML to ensure that you don't ever see the transaction.

How to avoid

- Input scrubbing and validation
- NEVER permit `<script>`, `<embed>`, or `<object>` tags in content.
- NEVER trust query strings, POST values, or image file uploads.
- Don't trust anyone.

<embed> and <object> tags are useful for embedding video; however, they can also (very easily) be used to embed malicious code. If you're going to permit these types of tags, then you MUST scrub them to ensure that the source is from a trusted domain.

Tools

- Eternal vigilance: threats are ever-changing, require constant monitoring
- Microsoft's XSS Detect (Beta) plugin
- scanmus

Don’t trust your friends

Examples

- Yahoo! property requires corporate login to access public site
- Another property breaks other sites by restricting access to (internal-only) RSS feeds

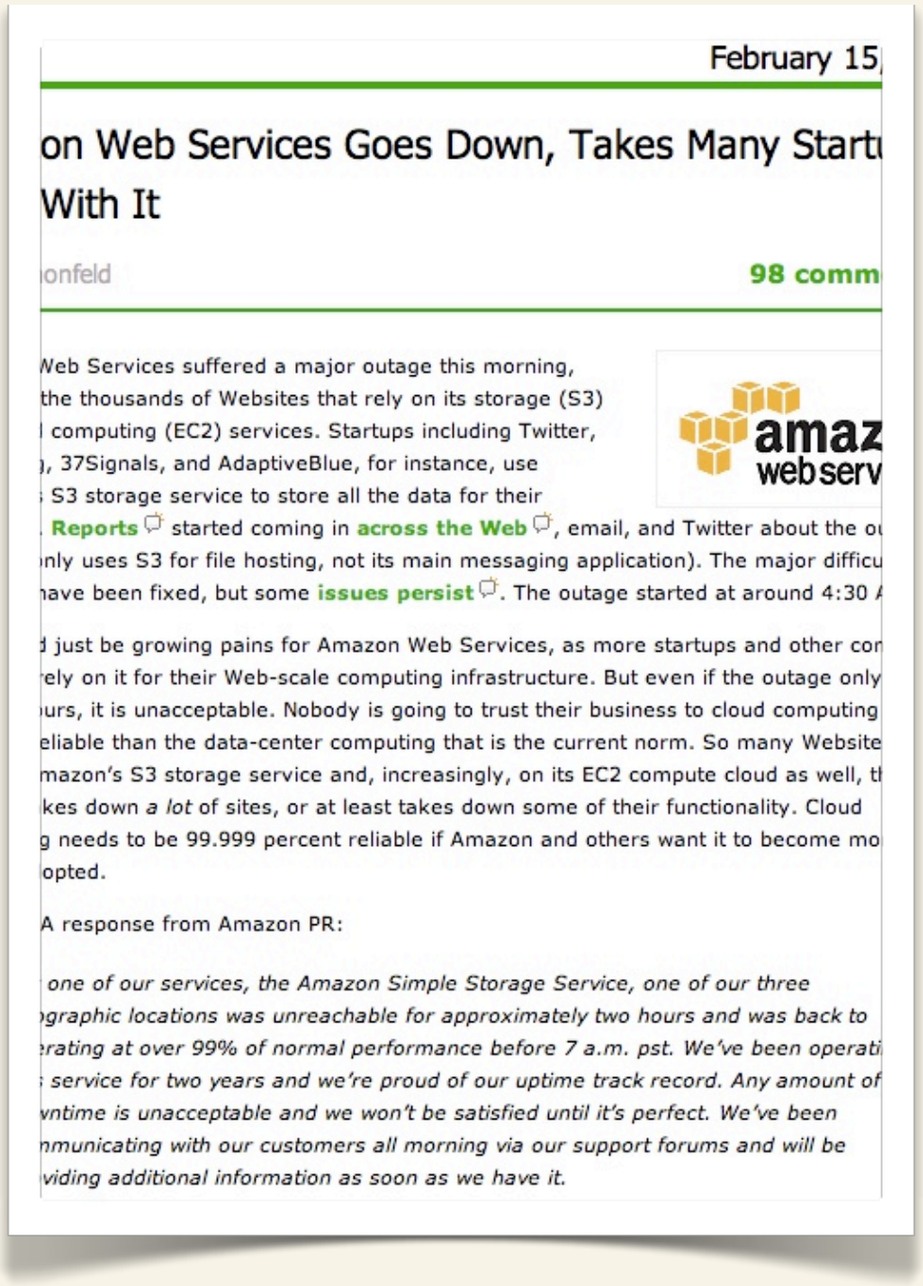
How to avoid

- Create a staging environment
- Duplicate of production so that you can test installations

Trust outsiders with your data

Example

- Amazon S3 goes down on February 15, 2008
- Affects Twitter, 37signals, AdaptiveBlue, many others.



February 15,

on Web Services Goes Down, Takes Many Startu
With It

onfeld 98 comm

Web Services suffered a major outage this morning,
the thousands of Websites that rely on its storage (S3)
computing (EC2) services. Startups including Twitter,
, 37Signals, and AdaptiveBlue, for instance, use
S3 storage service to store all the data for their
Reports started coming in across the Web, email, and Twitter about the ou
nly uses S3 for file hosting, not its main messaging application). The major difficu
have been fixed, but some issues persist. The outage started at around 4:30 A

just be growing pains for Amazon Web Services, as more startups and other cor
ely on it for their Web-scale computing infrastructure. But even if the outage only
urs, it is unacceptable. Nobody is going to trust their business to cloud computing
eliable than the data-center computing that is the current norm. So many Website
mazon's S3 storage service and, increasingly, on its EC2 compute cloud as well, th
kes down *a lot* of sites, or at least takes down some of their functionality. Cloud
g needs to be 99.999 percent reliable if Amazon and others want it to become mo
opted.

A response from Amazon PR:

one of our services, the Amazon Simple Storage Service, one of our three
graphic locations was unreachable for approximately two hours and was back to
erating at over 99% of normal performance before 7 a.m. pst. We've been operati
service for two years and we're proud of our uptime track record. Any amount of
ntime is unacceptable and we won't be satisfied until it's perfect. We've been
nmunicating with our customers all morning via our support forums and will be
viding additional information as soon as we have it.

How to avoid

- Don't rely on external services for critical functions, if possible (for example, SmugMug uses S3 for backup, not live storage).
- If you must, then plan in advance for how you'll handle failure.
- FAIL is a matter of *when*, not *if*.

SmugMug is a photo-sharing site geared towards professional photographers; one of their touted benefits is that they keep multiple backup copies (in S3) of all your photos so that they can be recovered if necessary.

More

- Failure of outside services (outside the property, not of Yahoo!) is a major cause of outages at Yahoo!
- Require SLA and notifications for problems from vendors.

SLA = Service Level Agreement.

Let humans work for you

Examples

- Yahoo! Tech
  - 10-minute outage when the operator took all the servers offline.
  - 2-hour outage when someone at the data center turned off the database servers and removed them from the racks.
- “Human error” is a top cause of failures.

How to avoid

- Technique developed in the State of Michigan hospital system.
- In two years, has caused a 98% reduction in intensive-care unit human errors.
- Length of hospital stays have declined 20% since implementing it.

Checklists

- Yes.
- Printed checklists that are completed with a pencil.
- Seriously.

Do you FAIL?

How To Tell?

- Relying upon users is risky.
- Need to put monitoring in place.
- Simple monitor retrieves a page, checks for a specific string.

SLA

- You need to have an SLA for yourself as well as your vendors.
- You need to be able to monitor and enforce the SLA.
- When there's a problem, don't just fix the problem; fix the process that allowed the problem.

SLA's need to be reasonable; Yahoo!'s is 99.85%. The popular "six-sigma" (99.9999%) is not feasible; it implies that your site sill only be down 32 seconds per year. A 15-minute outage means that you've used up your SLA for the next century.

Q&A / More

Some Resources

- Medical checklists:
  http://www.newyorker.com/reporting/2007/12/10/071210fa_fact_gawande
- Grow-up programming:
  http://www.fastcompany.com/node/28121/print?donttaseme

I'd like to hear from you

- Email: gecampbell@yahoo.com
- Twitter: glenc
- http://glencampbell.name
- http://files.glencampbell.name/webinale2008
- Ask me about GarageBand Ping-Pong