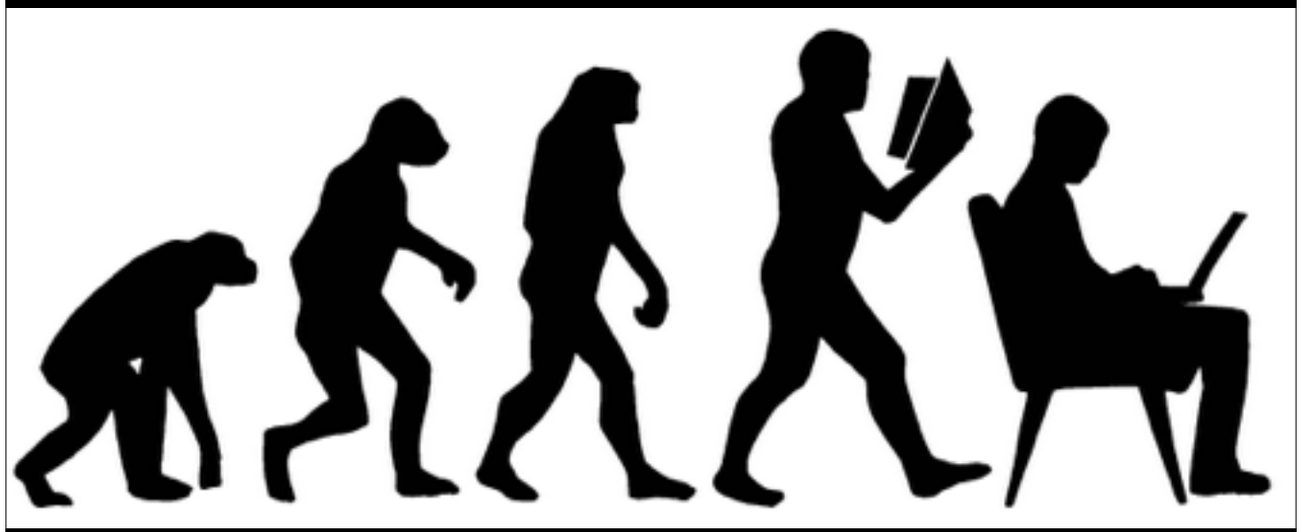
garry.bodsworth@gmail.com @garrybodsworth

WHAT IS SECURITY?

- Protect infrastructure
- Protect business
- MOST IMPORTANTLY **Protect Customers**
- Process not a product
- Privacy

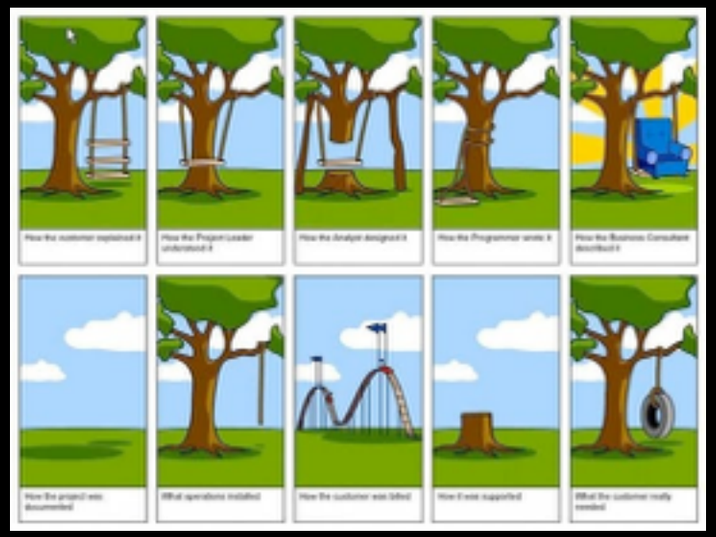
- Process - not a product - it is in the fabric
- When you buy more security - YOU ARE DOING IT WRONG
- Deliberately conflating security and privacy - both are intertwined

SECURITY NEEDS TO BE PART OF THE PROCESS FROM THE START



- Evernote - May 2013 - no security - get compromised - then they start trying to hire a security team.
- As your program evolves, security must always be considered.
- It goes beyond simply product but the whole organisation.

WHY IS THERE NO SECURITY-DRIVEN-DEVELOPMENT?



- Most people will recognise this comic

- This is what a security person sees.
- And this is just scratching the surface

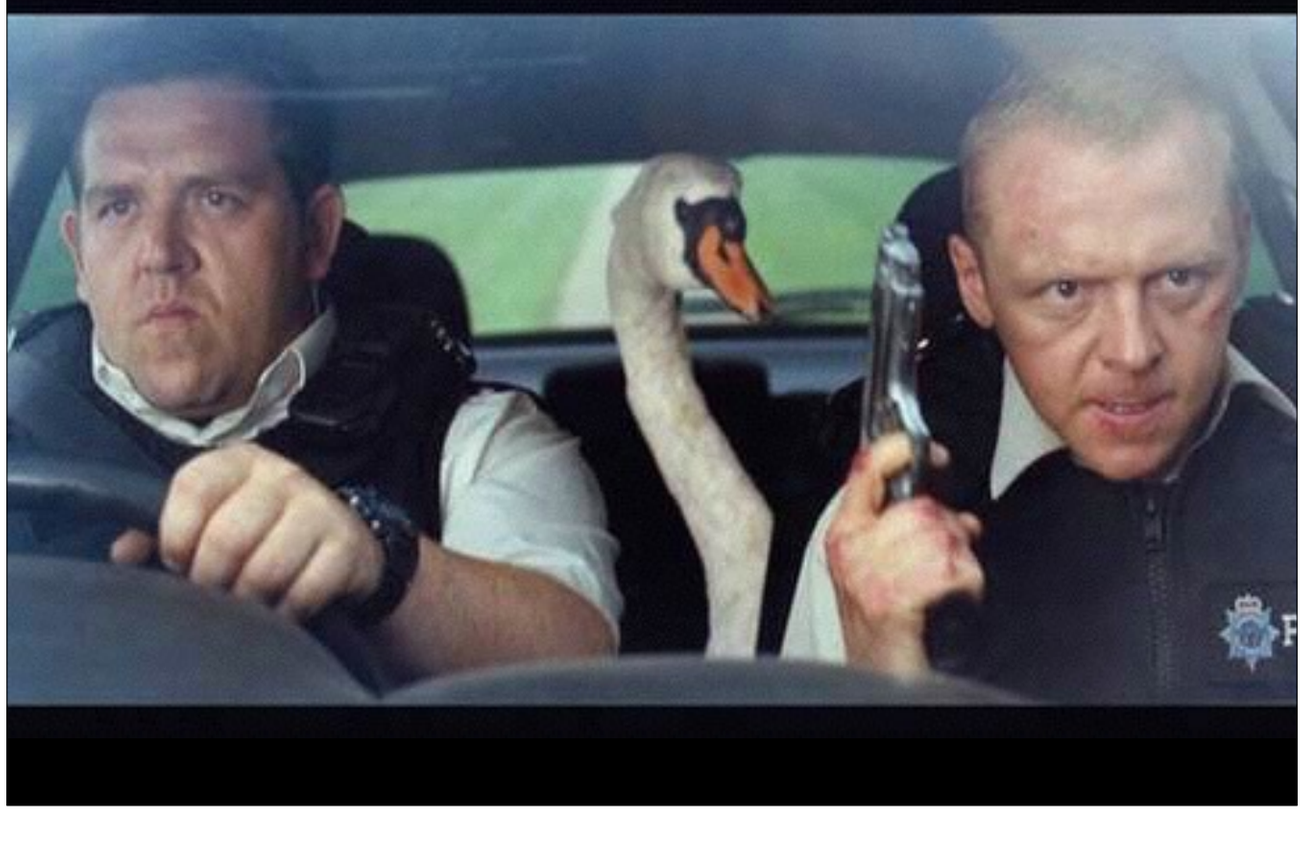
RETROACTIVELY ADD
"SECURITY"?

HAVE YOU TRIED TO ADD TESTS TO A LEGACY SYSTEM???

THIS IS EVEN HARDER

- Retroactively writing tests for a legacy system is hard
- Adding security is an order of magnitude harder
- Evolution is sometimes not possible and involves starting again

FUZZING



- Was: Threat modelling
- Great to find unexpected data input issues.
- Must have a good, reproducible process.

BUT... BUT... THERE ARE NO BUFFER OVERFLOWS

PERFECT CODE STILL EXPLOITABLE

- Perfect code is still exploitable
- Imagine mathematically proved perfect code getting released. Gets owned by a script kiddie in a few hours.
- Need to consider more than simple buffer overflows when designing.

- Unwittingly exposing internal implementation and data relationships - I get quite a lot of push back on this - especially from REST purists.
- Your API should abstract into higher level semantics, because if it looks like the building blocks you can’t protect how those interact easily.

WHY NOT?

- When I first joined MS the API already existed
- You give the mobile client the user ID when it registers then it can use that to identify to the API.
- Can you think of a way to exploit this?
- Change the number!
- OK, it's a GUID - can you think of ways to get the numbers? Get your friends numbers! Go mad!
- This is a massive level of screwed.

- Does the client need to know the user ID?
- Usually login session and user ID should be decoupled.
- For instance, you can use a social login that maps to a particular user, this can be managed completely server side.
- Is a mobile client the user?
- NEVER TRUST THE CLIENT - ASSUME IT IS COMPROMISED

HOW?

TOKENS!

- TOKENS!
- You need something to represent the user allowing a piece of software to do something on their behalf.
- The token should encapsulate
  - a set of permissions
  - a finite lifespan
  - be entirely managed server side.

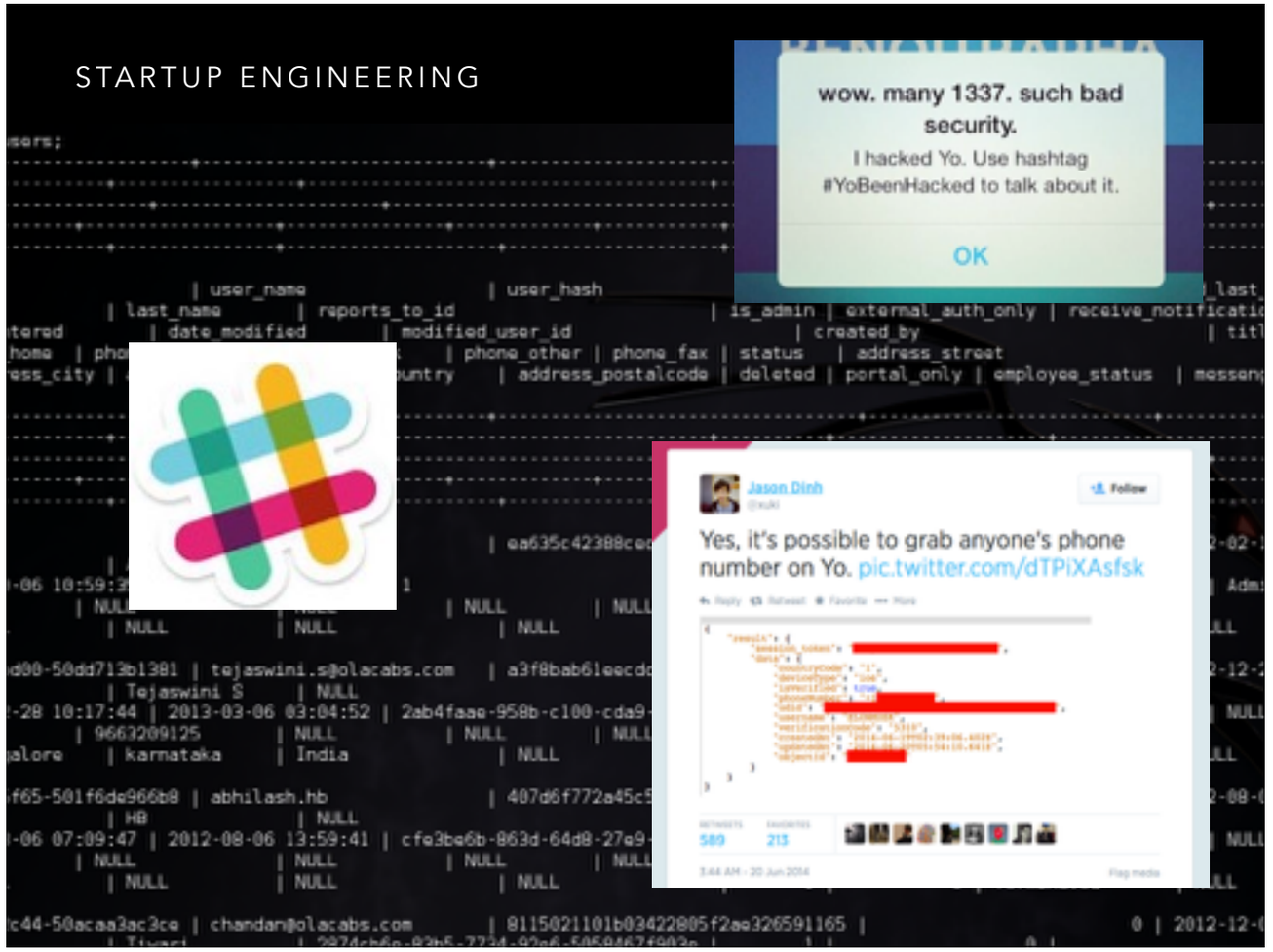
IT'S WHAT EVERYONE USES

OAUTH 2.0

- OAuth 2.0
- Used by Google, Twitter, Facebook…..
- GUTTED OF SECURITY
- WHY? Sacrificed at the later of performance when faced with huge traffic.
- This lead to the lead of the project quitting.
- There are some IETF proposals like the Bearer Token
  - I used a version of this at MS

- BANHAMMER!
- You get granular revocation all the way up to nuking a user from orbit.
- Doesn’t need to compromise the whole account.
- Makes more complex ownership, sharing and workflows possible
- The matchmaking example - list of users or a list of matches identified uniquely.

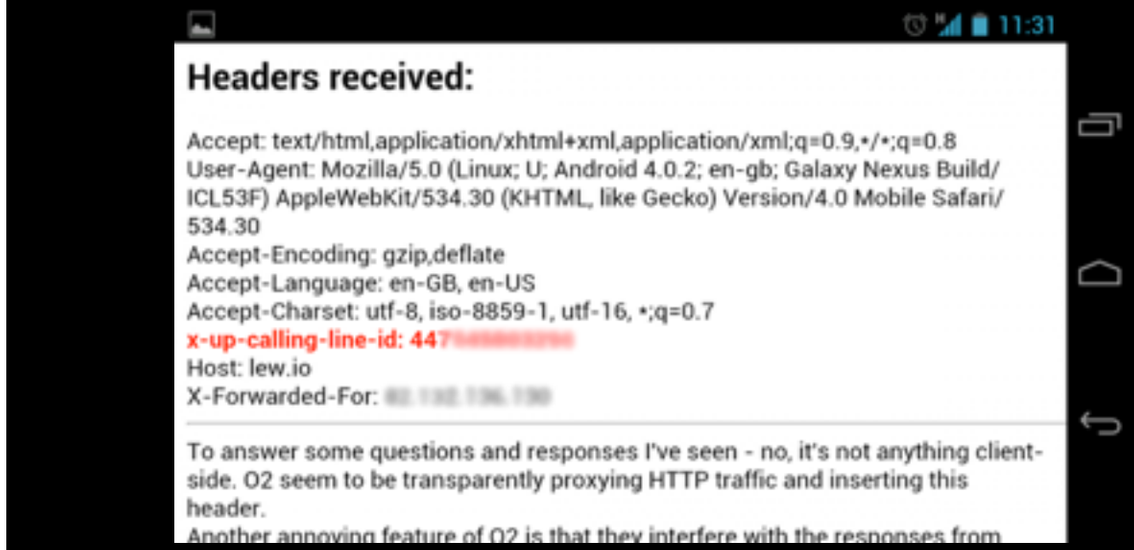
STARTUP ENGINEERING



- Startup engineering
- Injected code
- Bad API
- Bad access controls
- Ola cabs - data breach - credit cards. Team Unknown published on Reddit
- Slack

INFORMATION LEAKAGE

O2



- Information leakage
- 2012 O2 phone numbers in HTTP headers

ENCRYPTION

- Transport
- Data at rest

- Cryptographic hash (one-way)
- Reversible (two-way)

- Symmetric
- Asymmetric

- Transport - SSL, TLS, HTTPS
- Data at rest - your storage - database, filesystem, nosql
- Cryptographic hash - for storing passwords
- Reversible - you eventually need the original data like files on your hard drive
- Symmetric - Both know the same secret - less computationally intensive typically
- Assymetric - Public key encryption - big primes

SSL CERTIFICATES

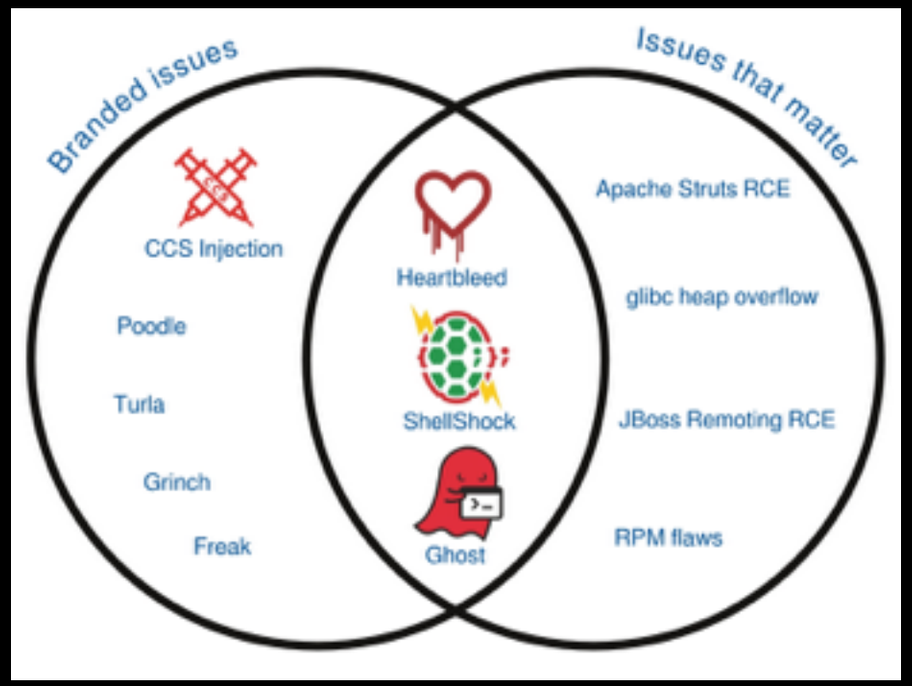
- Who has bought one?
- Who has generated one?
- Who has invalidated one?
- Who can deploy one?

- Ordering one - certificate signing request
- Know how to get one into production? Know about IP/vHost limitations and SNI?
- Have you made your own certificate authority before?
- Managed CRLs before?
- Recent revocations
- Recent applications - Amazon, new free as in beer
- Configuring servers, protocols and ciphers

- We know all your secrets
- Where do you store
  - certificates
  - passwords
  - critical information
- Source control? VERY BAD
- Secret server/service? Allows you to decouple secrets from the service.
- Encrypted file? Not bad.
- Rolling keys - have a plan for keys being changed or rotated. Reasons: compromise, age, management….
- USE PASSWORDS ON YOUR DATABASE

BUGS BUGS BUGS

FUNKY NAMES AND LOGOS



https://securityblog.redhat.com/2015/04/08/dont-judge-the-risk-by-the-logo/

Sometimes you just have honest-to-goodness coding bugs

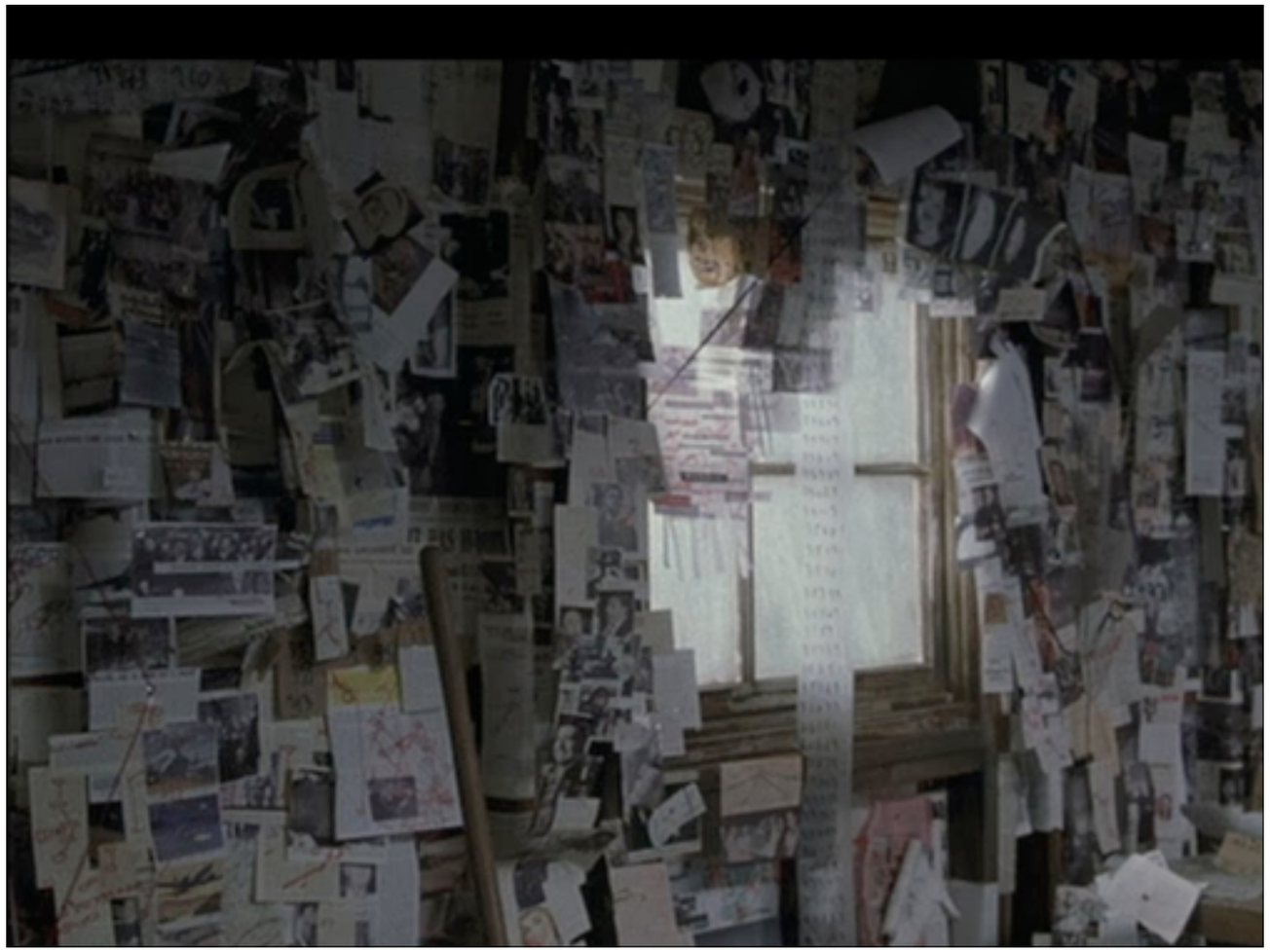
Have a plan for deploying updates QUICKLY

Also evaluate risk not from logos

STORAGE SECURITY

ACCESS CONTROL

- You will find services lacking - e.g., Azure tables and blob storage
- Not just ACLs for end user, but employees, clients as well
- Had to build an Access Control layer on top before. This was to make sure developers knew what resources they were interacting with and how.
- Principle of least privilege - only give the absolute access necessary.
- You will see data relationships that cause you to segment away more critical data from more frequently accessed data.
- You will also discover how tightly coupled your data and code are.



- We’ve only begun to scratch the surface
- How do we go about solving this?
- These will be obvious, but you need solid foundations for good security

- Start with the most secure settings.
- All warnings on, all warnings as errors
- Static analysis - stops the dumbest of errors
- Language specific tools - Python: e.g. PEP8, PyLint, PyFlakes
- Don’t Repeat Yourself - fix a bug once - easier to test
- Code coverage - if the code is never executed except in production how do you know it is good?

TESTING

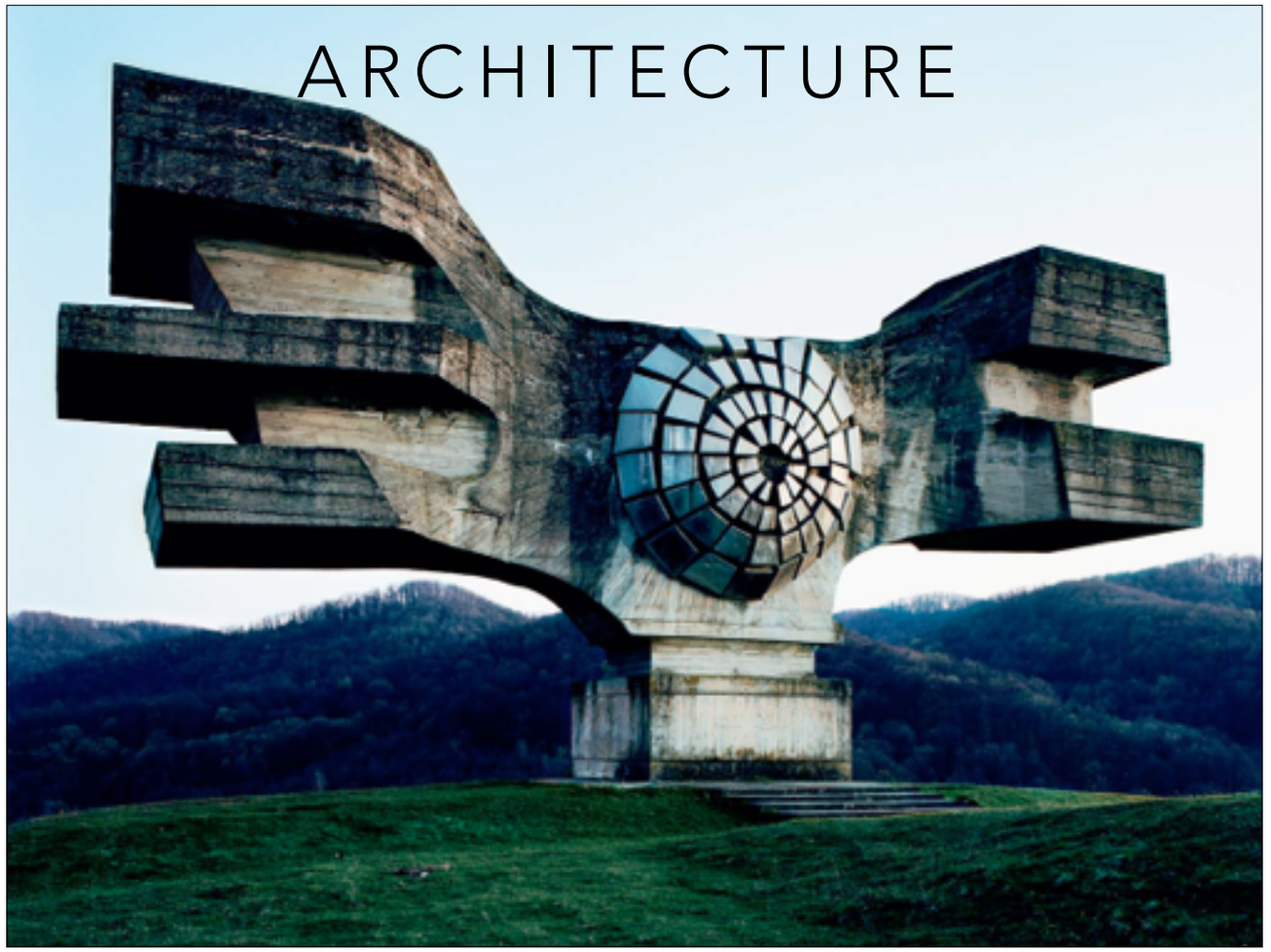
- Load
- Unit
- API/Client
- Browser
- Integration

- Run locally
- Run SSL dev servers
- Know your snake oil

- Get yerself some unit, load, browser, integration (end-to-end), API/Client
- Tests should run easily on developer's machines
- Developers should understand and edit the test code and framework
- Developers should run local SSL server
- Should know how to regenerate snake oil

- Continuous deployment
- 3rd party code drops and updates
- Updating certificates and third party service dependencies
- Be ready to switch providers, the decision may be out of your hands for money, buddies, or plain security disasters (I have an example there…)

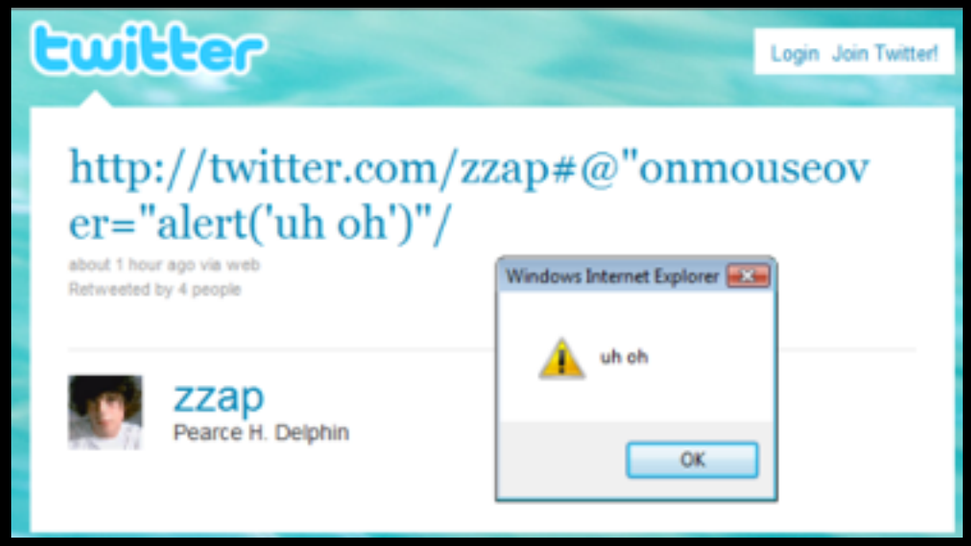
ARCHITECTURE



- Architecture
- Load balancer/reverse-proxy
  - DDOS protection
  - ELB
  - Hiding server layout
  - Limit entry points into the system
  - Secure subnets
  - Use the infrastructure if possible
- CDN
  - Reduce load
  - Increase responsiveness
  - Make sure you trust them!
- Cache
  - Be careful of stale data
  - Hackers could exploit it

XSS - CROSS SITE SCRIPTING ATTACKS

TWITTER



Sanitise your output

- Twitter XSS
- Sanitise your output!
- Bleach
- Escaping
- US National Vulnerability Database had XSS! http://www.theregister.co.uk/2015/06/18/us_vuln_database_contained_xss_vuln/

(D)DOS

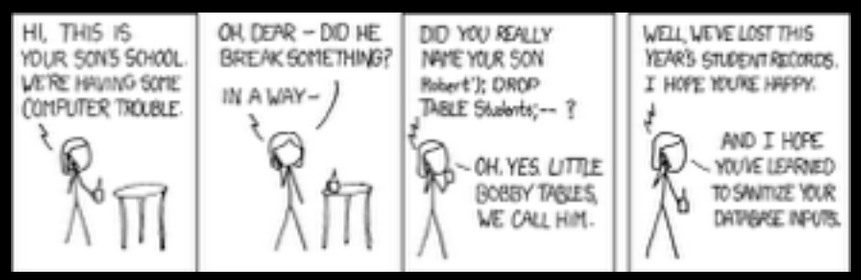
- DDOS
- Load balancers
- Logging
- Closed-source (Azure - going to build into server 2015, Amazon) - so behaviour is not known or understood
- Rolling your own is tricky
  - IP? What about NAT?
  - Other metrics?
  - Constantly need tweaking

SQL INJECTION

- SQL Injection
- Always use the ORM!
- Escaping!
- Django has no known SQL injection attacks open

SQL INJECTION

TWITTER



- Twitter SQL injection
- XKCD example
- Gaana
- Uber petition
- Doing it right LastPass

MIXED CONTENT

- HTTP and HTTPS on one page
- IFRAMEs

- Mixed content
- HTTP can extract HTTPS content on the page - they share the DOM
- IFRAME - cry cry cry - weep weep
- Use XFrame headers - you have to be sure the browsers implement it right
- Understand your cookies, secure, httponly

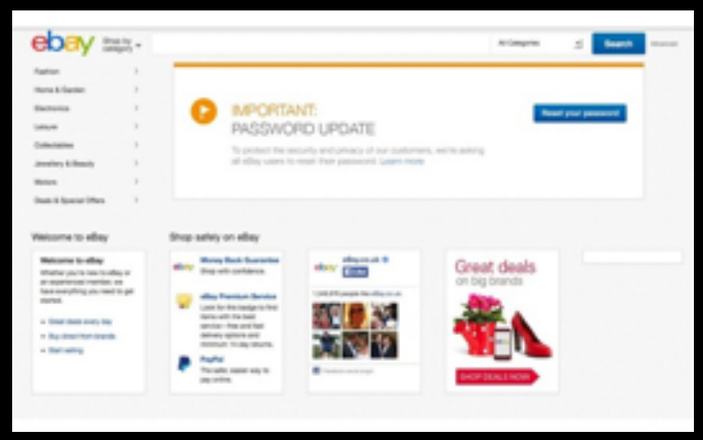
LOGS
GOOD VS BAD

- Intrusion detection
- Error logs
- Easy to accidentally expose
- Contains sensitive information

- Logs are your friend and your enemy
- Good
  - You can find intrusion information
  - Errors could be early warnings of vulnerabilities
- Bad
  - Accidentally publicly readable
  - Contains sensitive information because they are used for debugging
  - Easy attack vector to get that kind of information
- You can make PHP execute your log so if there is some way of a user getting something written there, you could be in trouble.

INTERNAL NETWORK DATA BREACH

EBAY



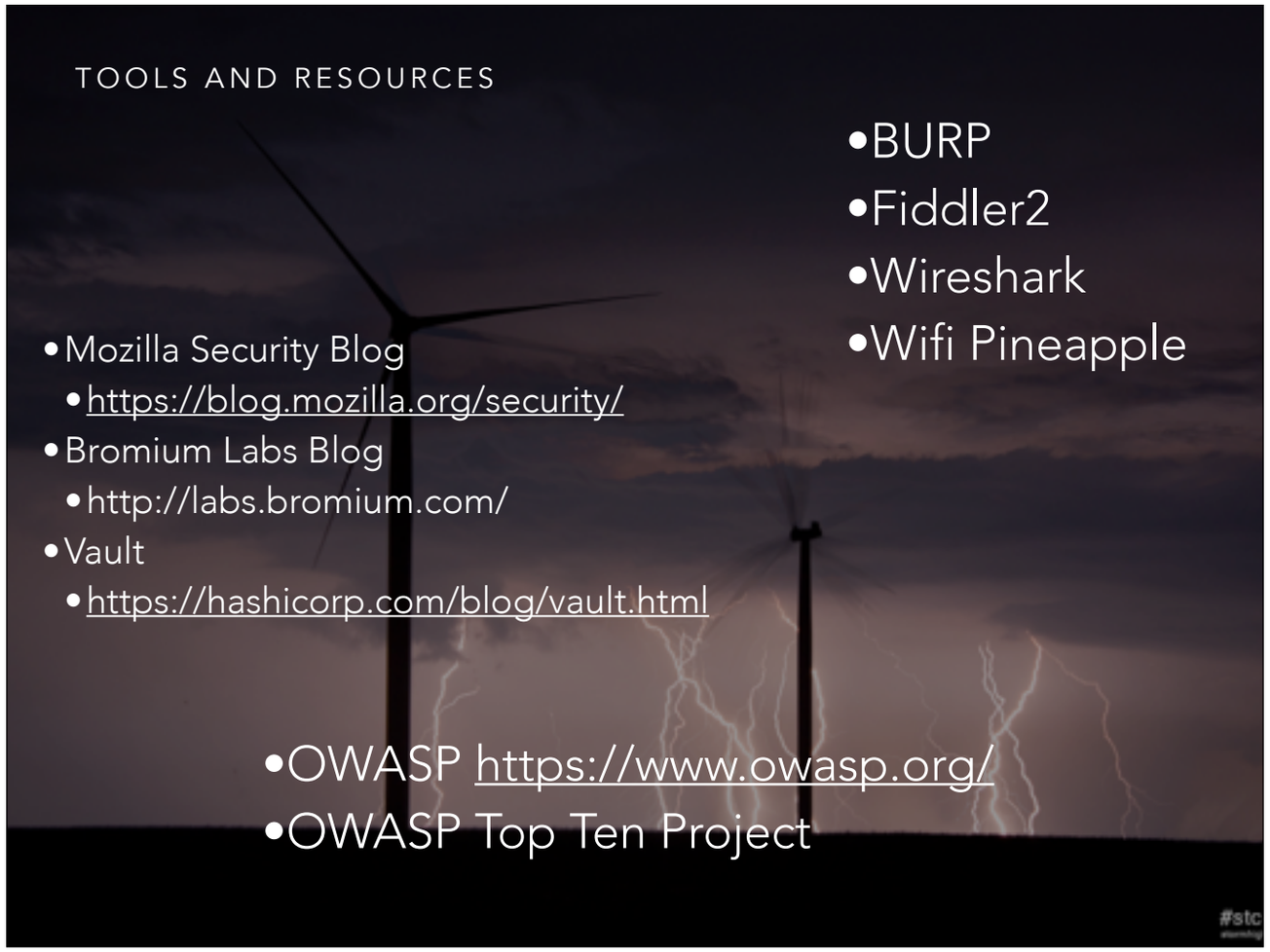
Internal network data breach - May 2014

What have I missed?

- Open relays
- CSRF
- Brute force
- Hashing is not encryption!
- SSL ciphers
- PCI compliance
- CVEs and vulnerability disclosures
- Do not trust the user!
- File format parsing
- Pen tests
- Physical pen tests
- Cookies
- Reproducible builds

- SChannel
- PHP
- RSA4
- goto fail;
- Malicious governments
- Physical intrusion
- More passwords stolen
- Incident response
- Physical security solutions
- Fuzz testing
- Session hijacking
- And sooooooooooo much more

So much, so much, my head, this could take HOURS!



Ch-ch-ch-check it out