Application Security

Hello!

I am Neil Matatall

I do security.

Twitter @ndm

GitHub @oreoshake

1

Why am I here?

Application security is the hottest field in security... and one of the fastest growing industries in the world.

“But there are also unknown unknowns, the ones we don’t know we don’t know.”


SEC. RUMSFELD

You + me

- I can make something secure
- You can make an app
- We can make a secure app

Developers make a bigger impact on security than any security team.

OWASP Top 10

“It skims the surface of the entire lake”

Injection

Yo dawg, I heard you like code in your code.

SQL Injection

```
ActiveRecord::Base.connection.execute("select * from table where id =
#{params[:id]}")

User.destroy_all(["id = ? AND admin = '#{params[:admin]}",  params[:
id]])

User.exists? params[:user]

User.find(:all, :group => params[:group], :conditions => { :admin =>
false })
```

rails-sqli.org

Code != Data

Injection occurs when data crosses a context boundary without consideration. Every time a context boundary is crossed, the data must be bound or escaped.

Often there is a safer, higher level API that handles this for you.

Binding (sort of)

```
User.where(:name => params[:name])

Project.find(:all, :conditions => [ “name
like ?”, “#{params[:name]}”] )
```

Escaping

```
Model.connection.quote(params[:something])
```

Escaping

```
ESCAPES = {
  "\0" => '00', # NUL             = %x00 ; null character
  '*'  => '2A', # ASTERISK        = %x2A ; asterisk ("*")
  '('  => '28', # LPARENS         = %x28 ; left parenthesis ("(")
  ')'  => '29', # RPARENS         = %x29 ; right parenthesis (")")
  '\\' => '5C', # ESC             = %x5C ; esc (or backslash) ("\")
}
# Compiled character class regexp using the keys from the above hash.
ESCAPE_RE = Regexp.new(
  "[" +
  ESCAPES.keys.map { |e| Regexp.escape(e) }.join +
  "]")

##
# Escape a string for use in an LDAP filter
def escape(string)
  string.gsub(ESCAPE_RE) { |char| "\\" + ESCAPES[char] }
end
```

Code != Data

Injection occurs when data crosses a context boundary without consideration. Every time a context boundary is crossed, the data must be bound or escaped.

Often there is a safer, higher level API that handles this for you.

Busted Auth

and session management.

Don’t trust the user

Decisions about identity, authentication, and authorization should never trust input from untrusted systems.

Common Issues

Not being able to log out

Abusing password reset functionality

No brute force protection

Not rotating session IDs on login

Authentication secrets exposed

Passwords stored in recoverable format

Predictable session IDs

Session hijacking

Session fixation

XSS

Hello <script>alert(1)</script>!

Every website in the world has or will have XSS

It's not that complicated, but it is, but it's not really.

Trivia

Rules for avoiding XSS

Never put data inside a CSS or Javascript context.

Escape every single piece of data. Everything. Always.

Input validation is NOT a valid XSS control.

Always use auto-escaping templates to generate html. Do not use raw/html_safe.

Don't forget about <a href="javascript:"! Validate URLs.

Always, always use content security policy.

I.D.O.R.

Insecure direct object reference, or simple “I can guess where things are”

It's really just an auth'z failure

Bad:

```
Invoice.find(params[:id])
```

Good:

```
current_user.invoices.find(params[:id])
Invoice.where(id: params[:id]).where(user:
current_user)
```

“Naked finds on ActiveRecord models considered an anti-pattern.”

Misconfiguration

Because you can’t just throw webrick on the internet

Common Issues

Unpatched systems

Poor access controls

Lack of segmentation

Default error pages

Default passwords

Not requiring TLS

Admin exposed to internet

Not living by “least privilege”

Unmanaged hosts

VPN not required

Former employees have access

Source code served uninterpreted

Required Security Headers

Content-Security-Policy

Strict-Transport-Security

X-Frame-Options

X-Xss-Protection

X-Content-Type-Options

Public-Key-Pins

Feature Flags

“Oh shit” button

Getting hacked? Just turn it off.

staff shipping

Use yourselves as beta testers. Pay attention!

dark shipping

Requests are sent to two systems: the current version, and an idempotent “new” version

targeted rollout

Just like staff shipping, but to public groups (beta testers)

gradual rollout

1% 10% 25% 50% 100% per dc.

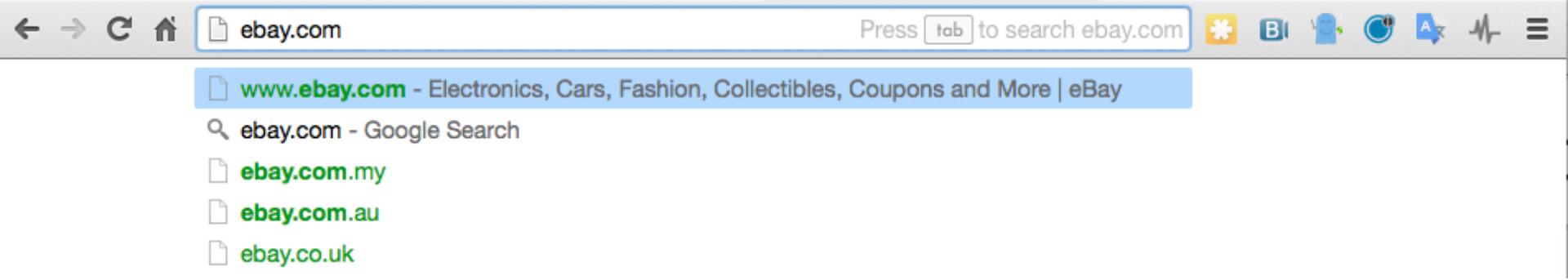
granularity is key

Mo’ switches. Mo’ control.

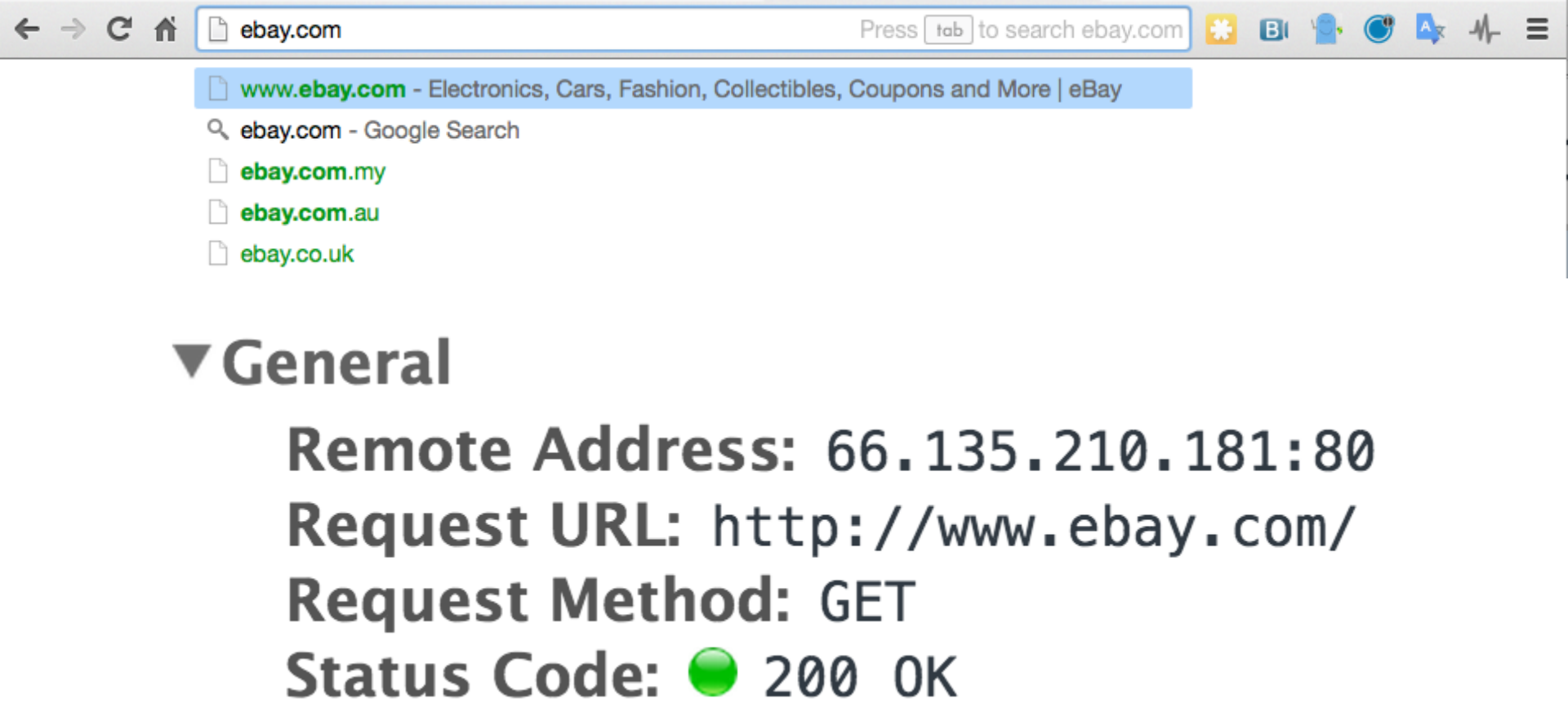
Data Exposure

Private data is Private.

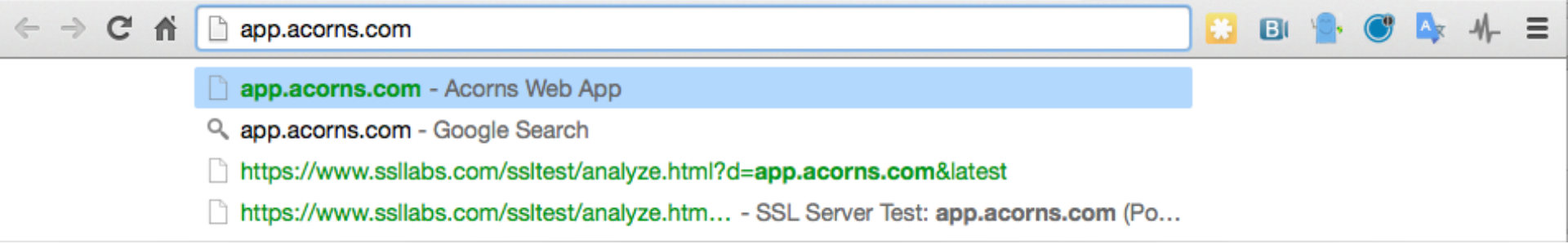
Encrypt all the things



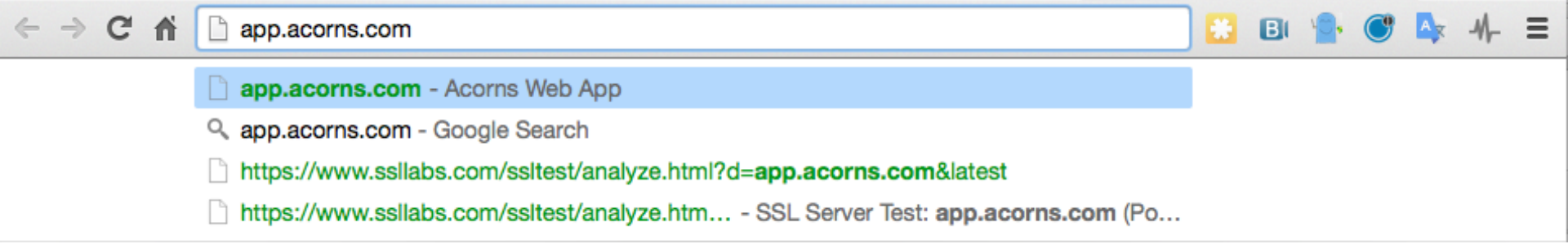
Browsers will expose data by default



Browsers will expose data by default



HSTS to the rescue!



▼ **General**

Request URL: `http://app.acorns.com/`
Request Method: `GET`
Status Code: 307 `Internal Redirect`

▼ **Response Headers**

Location: `https://app.acorns.com/`
Non-Authoritative-Reason: `HSTS`

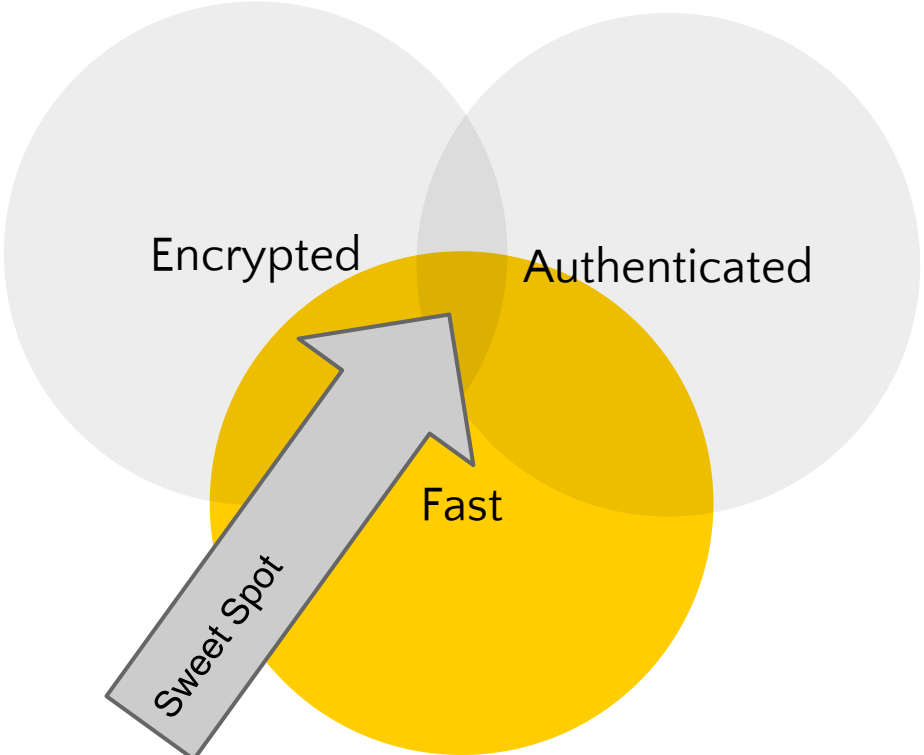
HSTS to the rescue!

State-run SSL certificate authorities make Congress nervous about web security

SSL certificates issued by state-run agencies could be influenced or abused by political motivations. At least, that's what worries the House of Representatives' Energy and Commerce committee, and it's asking Apple, Google, Microsoft and Mozilla for help.

Pinning to the rescue!

Crypto Is Hard



No really, crypto is hard

Padding Oracle Attacks (Unauthenticated AES CBC)

Keys with weak primes (RSA)

Keys generated with weak PRNG (debian)

CRIME, BEAST, POODLE, oh my!

More on TLS

Forward Secrecy

Ensures that even if the keys are compromised, it's impossible to "go back in time".

HSTS / Pinning Preload

Browsers can hard code this information to solve the bootstrap problem.

OSCP

Service run by CAs to revoke certificates (unreliable)

Cipher Suites

RC4 considered bad. 3DES considered OK. AES GCM preferred.

AEAD

Authenticated encryption modes. The new hotness. Thwarts traditional attacks.

OCSP Stapling

Services hit the CA endpoint, sign and cache the result. So much better.

Other exposure vectors

Logs

People sure like to store interesting info in logs.

Malicious Insider

Technical controls and auditing a must!

3rd parties

Integrations can often leak more data than intended.

Access Control

RBAC is dead.

Subject, object, action

Consider every action a subject can perform in an app. Consider the object on which the action is being performed.

action(object) if subject.can?(action, object)

Just like I.D.O.R.

```
current_user.accounts.find(account)

if current_user.owns?(account123)
  deposit(amount)
else
  raise HaxAttemptOrAppBug
end
```

Other examples

Mass assignment

?admin=true

app.acorns.com/admin

Mass assignment

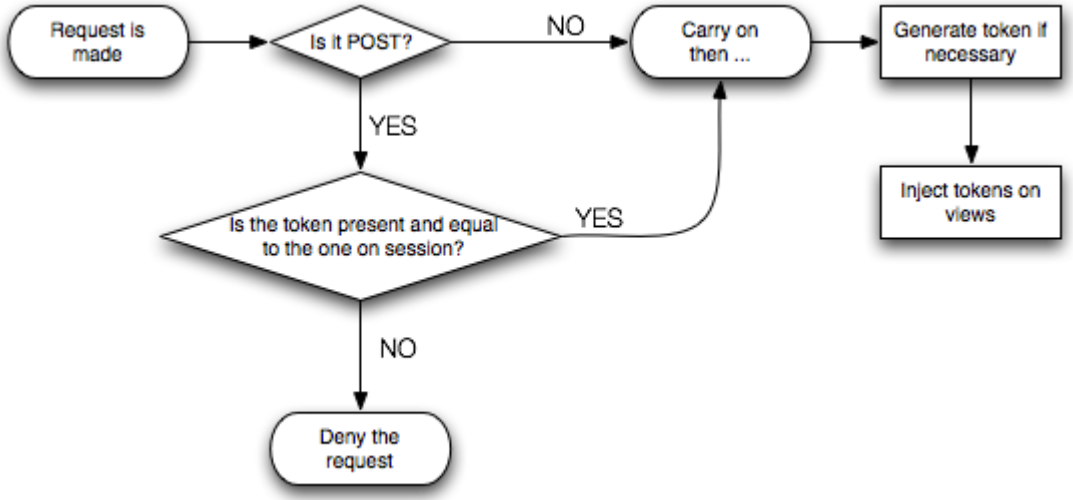
```
class PublicKeyController < ApplicationController
  before_filter :authorize_user
  ...
  def update
    @current_key = PublicKey.find_by_id params[:key]['id']
    @current_key.update_attributes(params[:key])
  end
end
```

CSRF

/si sɜrf/

cross site request forgery


<!DOCTYPE HTML PUBLIC
<html>
<head>
<meta name="TITLE"
<meta name="KEYWORDS"
<meta name="DESCRIPTION"
<link rel="stylesheet"
<script language="javasc
</head>
<body bgcolor="#ffff
width="1


Request is made
Is it POST?
NO
Carry on then ...
Generate token if necessary
YES
Inject tokens on views
Is the token present and equal to the one on session?
YES
NO
Deny the request

Verb-based CSRF

CSRF Protection is only applied to POST requests

match ‘photos’, :to => “photos#show”

“postback” considered an anti-pattern.

Vulnerable Libs

It’s hard enough to write secure code, let alone audit all libraries

What can you do?

bundler audit

Scans your Gemfile.lock for known vulnerable gems

retire.js

Scan directories, grunt, or live apps for vulnerable javascript libs

OWASP Dependency Check

For your java-based projects

CVE database

A central place where vulnerabilities are publicly disclosed

Audit your libraries

Just outsource it or give up.

NIH

Don't use any 3rd party libs. Probably not practical for almost everyone.

Unprotected Redirects

redirect_to params[:return_to] considered harmful yesterday.

Attack Scenarios

A link with your domain is used to distribute malware

acorns.com/ ?
return_to=badsite.com

Attack is chained with CSRF. User logs in, is redirected to GET-based CSRF

acorns.com/login?
return_to=/delete_account

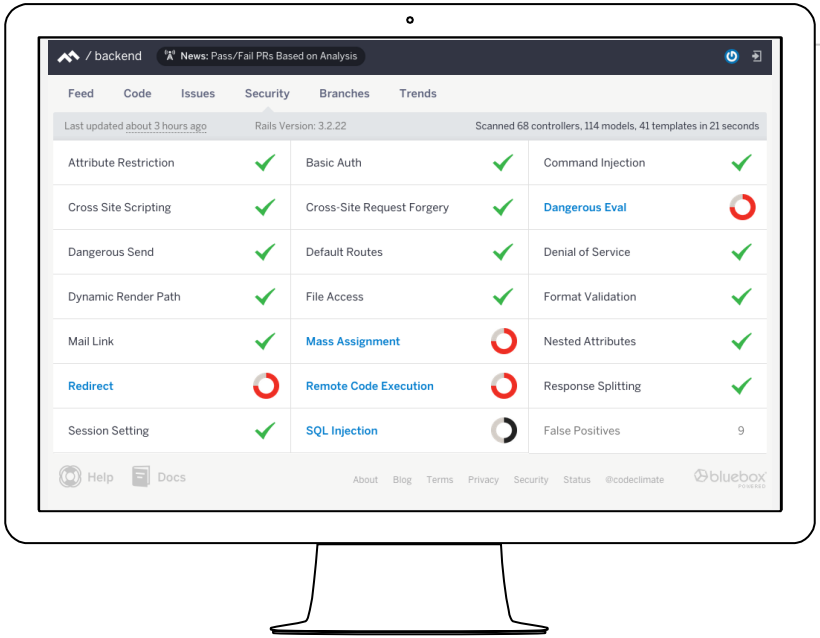
Solutions

```
redirect_to params[:url], :only_path => true

parsed_url = URI.parse(params[:url])
if is_whitelisted_host(parsed_url.host)
  redirect_to parsed_url
end
```

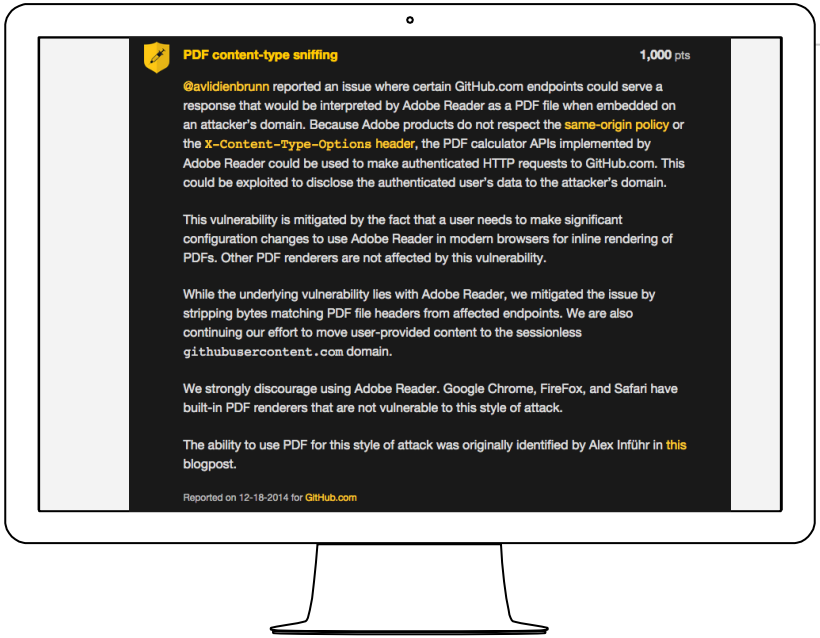
Brakeman + Code Climate

I'd say it's pretty easy to see what should be fixed.



Bugcrowd

Because bug bounties are the most cost effective way to find security issues. Ever.



Thanks!

Any questions ?

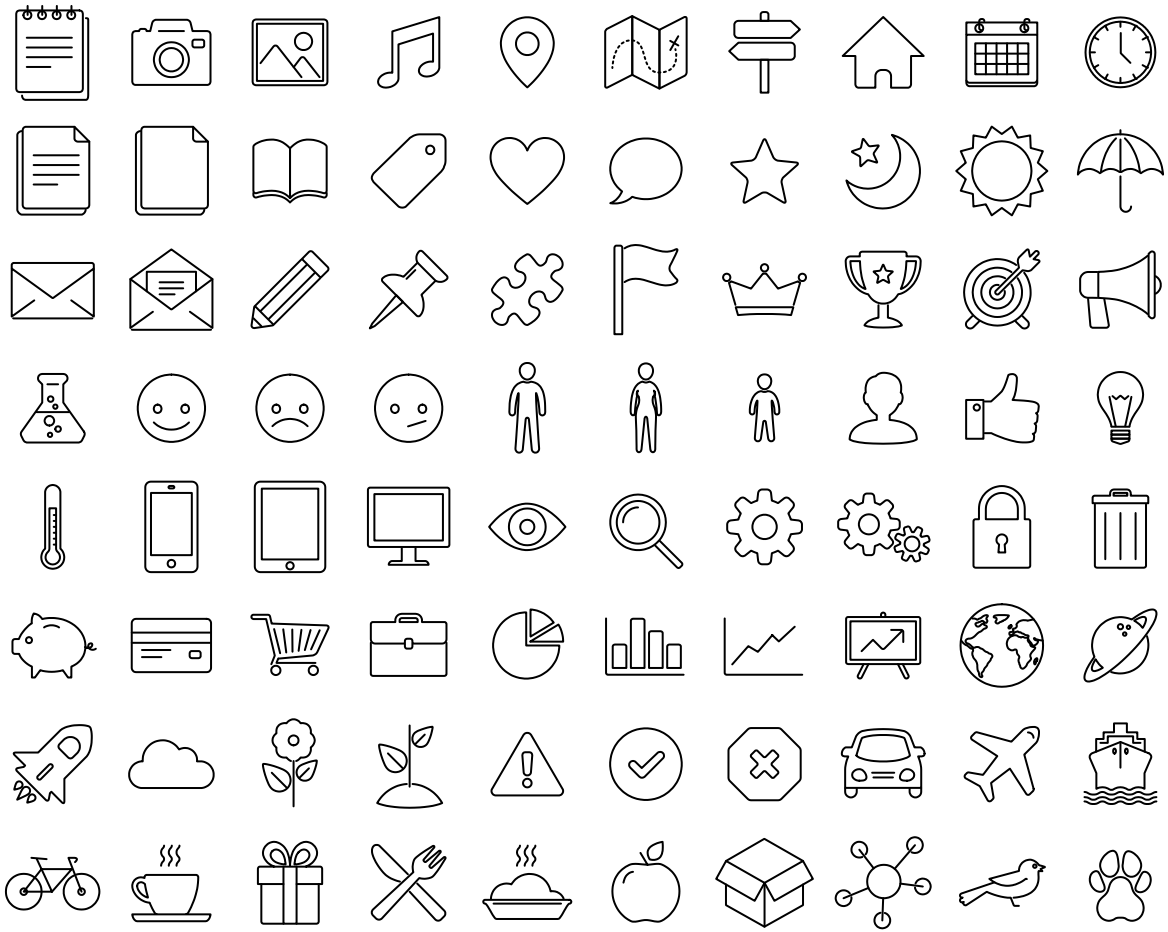
You can find me at

- Twitter @ndm / GitHub @oreoshake
- neil@acorns.com

Resources

Special thanks to all the people who made and released these awesome resources for free:

- Brakeman
- Code Climate
- Railsgoat
- OWASP
- securityheaders.com



SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples: