Rails Derailed

Presenter: Tim Goddard (pruby)

Who is this guy?

Tim Goddard (a.k.a. pruby)

- Pentester @ Insomnia Security
- Ex Rails Dev
- Focus on white box web application review
- … in my own time dance West Coast Swing.

Conventional Testing

We test most apps the same way, regardless of how they're written:

- Find the endpoints / entry points.
- Capture samples of how to invoke these.
- Manipulate network traffic, look for technical vulns (XSS, SQLi, etc).

Rails et al.

Opinionated Development Frameworks

- One way to do things.
- Many security benefits from the common method.
- … but also add their own attack surface.

Spectrum of Convention

The Manual, the Mainstream, and the Magical

Developer Understands

Consistent Implementation

The Mainstream

e.g. Authorization

- Things like cancan and cancancan exist.
- … but every app is different.
- You still have to remember to specify and check rules.

Business Logic

- No framework can prevent business logic bugs.

- Different for every application. For example, in an online store can I apply a discount code twice?

Magic can protect…

```
…
<div class=“alert”>
        <%= @warning %>
</div>
…
```

… but never perfectly

```
…
<script>
        var message = <%= raw JSON.dump(@message) %>
</script>
…
```

The Magical: Additional Attack Surface

Complicated Input Model

- Automatic coercion of parameters to objects:
  - ?foo=bar — params[“foo”] == “bar”
  - ?foo[]=zig&foo[]=zag — params[“foo”] == [“zig”, “zag”]
  - ?foo[zig]=1&foo[zag]=2 — params[“foo”] == {“zig” => 1, “zag” => 2}
- HTTP-Method-Override

Type Confusion Attacks

- Ruby is a Dynamically Typed Language.
- Any variable could be of any supported type (e.g. string, number, array).
- No automatic type checks, but methods often vary behaviour.

Type Confusion Attacks

Example: phonebook with anti-scraping requirement

Entry.find(surname: params[:surname], city: params[:city])

Type Confusion Attacks

… WHERE city IN (‘Auckland’, ‘Wellington’, ‘Christchurch’) AND surname IN (‘SMITH’, ‘WRIGHT’, ‘CARTER’)

Imperfect Components

Common Components, Common Flaws

- Devise framework used for authentication, contains well-known user enumeration.

- Paperclip framework containing SSRF flaw – to my embarrassment never reported.

Mass Assignment

```
@account.update_attributes(params[“account”])
```

Mass Assignment

account[**name**]=Tim%20Goddard&account[**telephone**]=0211234567

Mass Assignment

account[**balance**]=1000000

Mass Assignment

```
@account.update_attributes(balance: 1000000)
```

Info Disclosure

/account/1/profile

Info Disclosure

/account/1/profile.json

Info Disclosure

{…, password: “Winter17”, …}

Race Conditions

Convention: All Checks in Ruby

TOC-TOU

Race Conditions

Person:

validates :card_number, uniqueness: true

Company:

has_many :people

accept_nested_attributes_for :people

Rare but Severe

Surprising amount of rope…

Rare but Severe

RCE via

YAML.load(params[:data])

Rare but Severe

@record.send(params[:type])

Rare but Severe

@record.send(‘delete’)

Rare but Severe

RCE via

render(inline: @page.content)

Rare but Severe

RCE via

render(inline: ‘<% `rm –rf /` %>’)

Rare but Severe

RCE via

File.read(params[:filename])

Rare but Severe

RCE via

File.read(‘|whoami’)

Rare but Severe

RCE via

eval(params[:data])

Security over Support

Rails gets better via breaking changes…

What can I do?

If you're a developer, run these tools yourself:

- Brakeman finds lots of flaws by static analysis, great hit rate.
  - **Don't** ignore the findings – treat every one as an opportunity to improve.
- bundler-audit finds out of date libraries, good hygiene.
- Beyond that, focus on authorisation and business rules.

If you're a tester, try the following:

- Insist on white-box code review.
  - config/routes.rb lists every endpoint in the app…
- Hunt the manual and the magical.

Insomnia Security Group Limited

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

www.insomniasec.com

For sales enquiries: sales@insomniasec.com

All other enquiries: enquiries@insomniasec.com

Auckland office: +64 (0)9 972 3432

Wellington office: +64 (0)4 974 6654