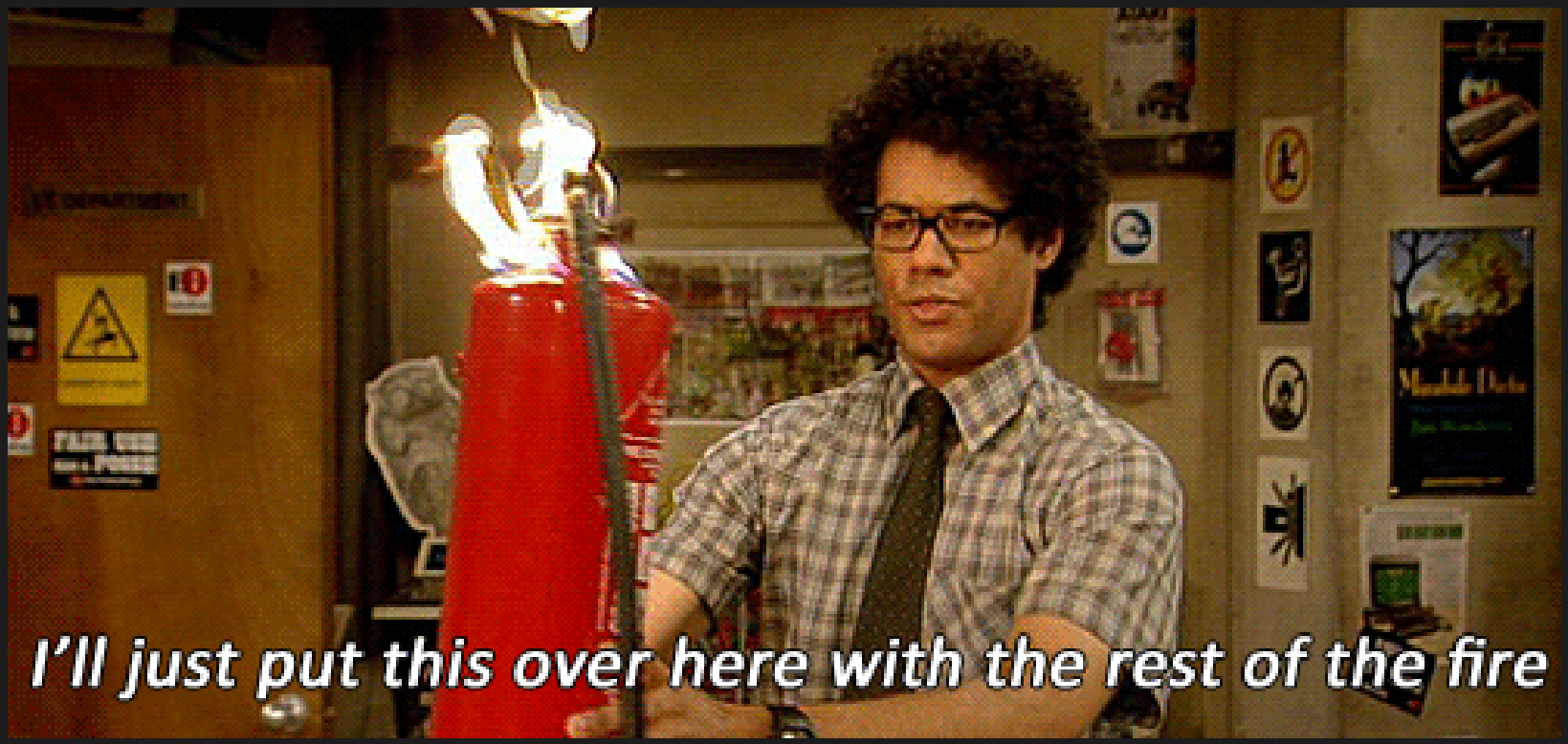
An experiment in Agile Threat Modelling

Fraser Scott
@zeroXten

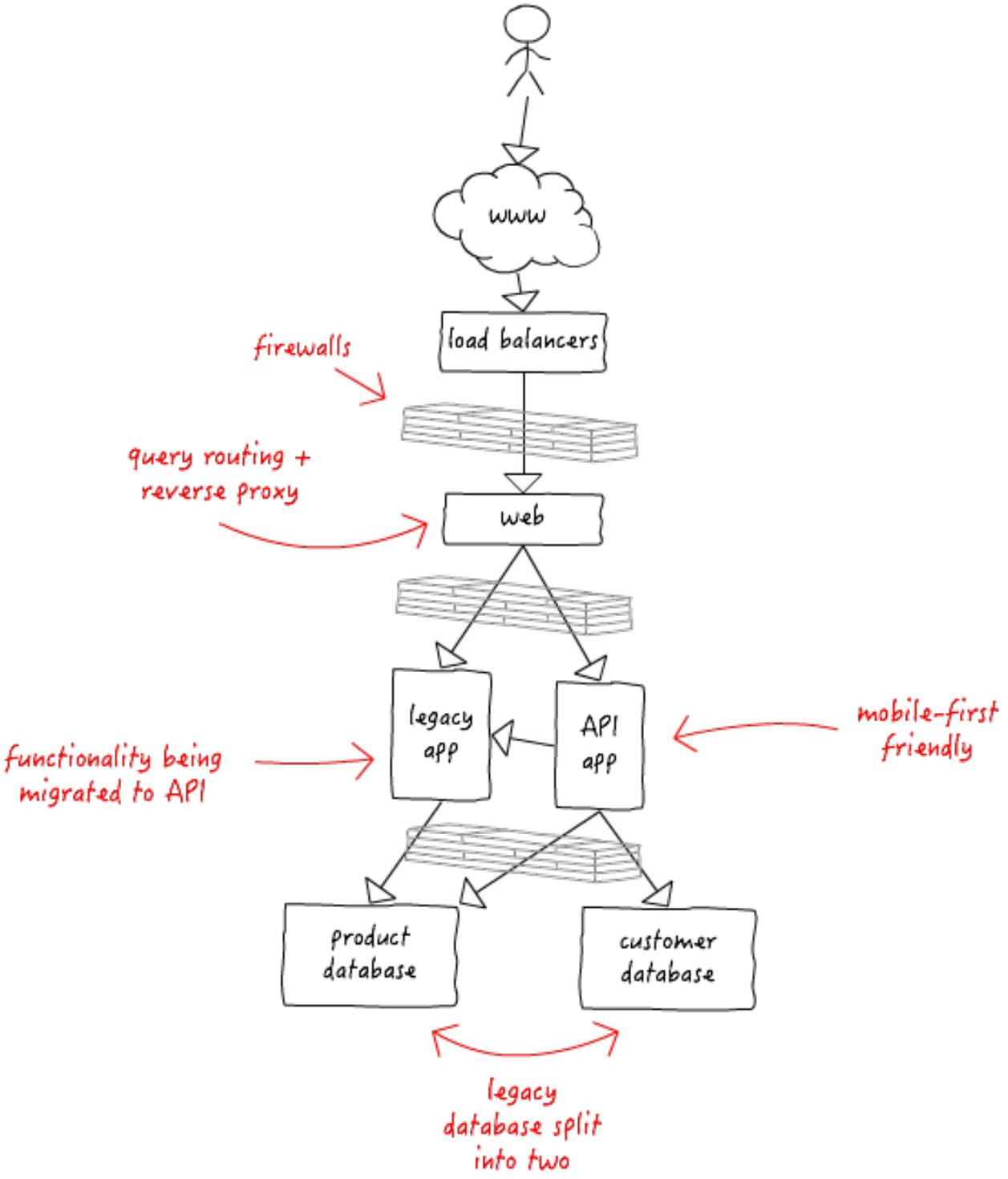

I'll just put this over here with the rest of the fire

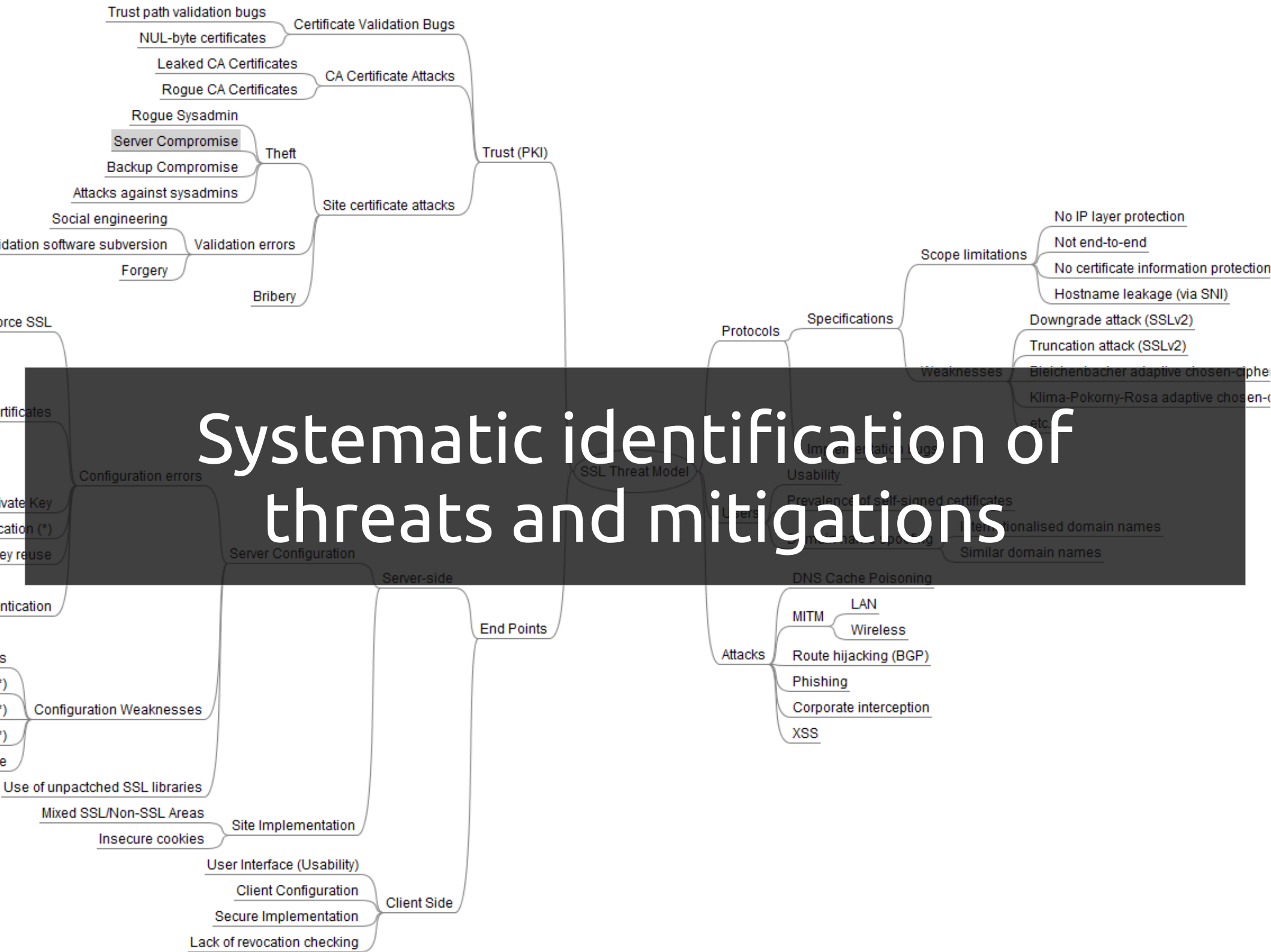
To err is human


To propagate error to all server
in automatic way is #devops
@DEVOPS_BORAT


THE AT&T GLOBAL NETWORK
OPERATIONS CENTER

```
root@server# tail /var/log/slowqueries
# Time: 130320  7:30:26
# User@Host: db_user[db_database] @
localhost []
# Query_time: 41.545309  Lock_time: 0.000069
Rows_sent: 219002  Rows_examined: 254105
SET timestamp=1363779026;
SELECT * FROM products WHERE id='' OR 1;
```

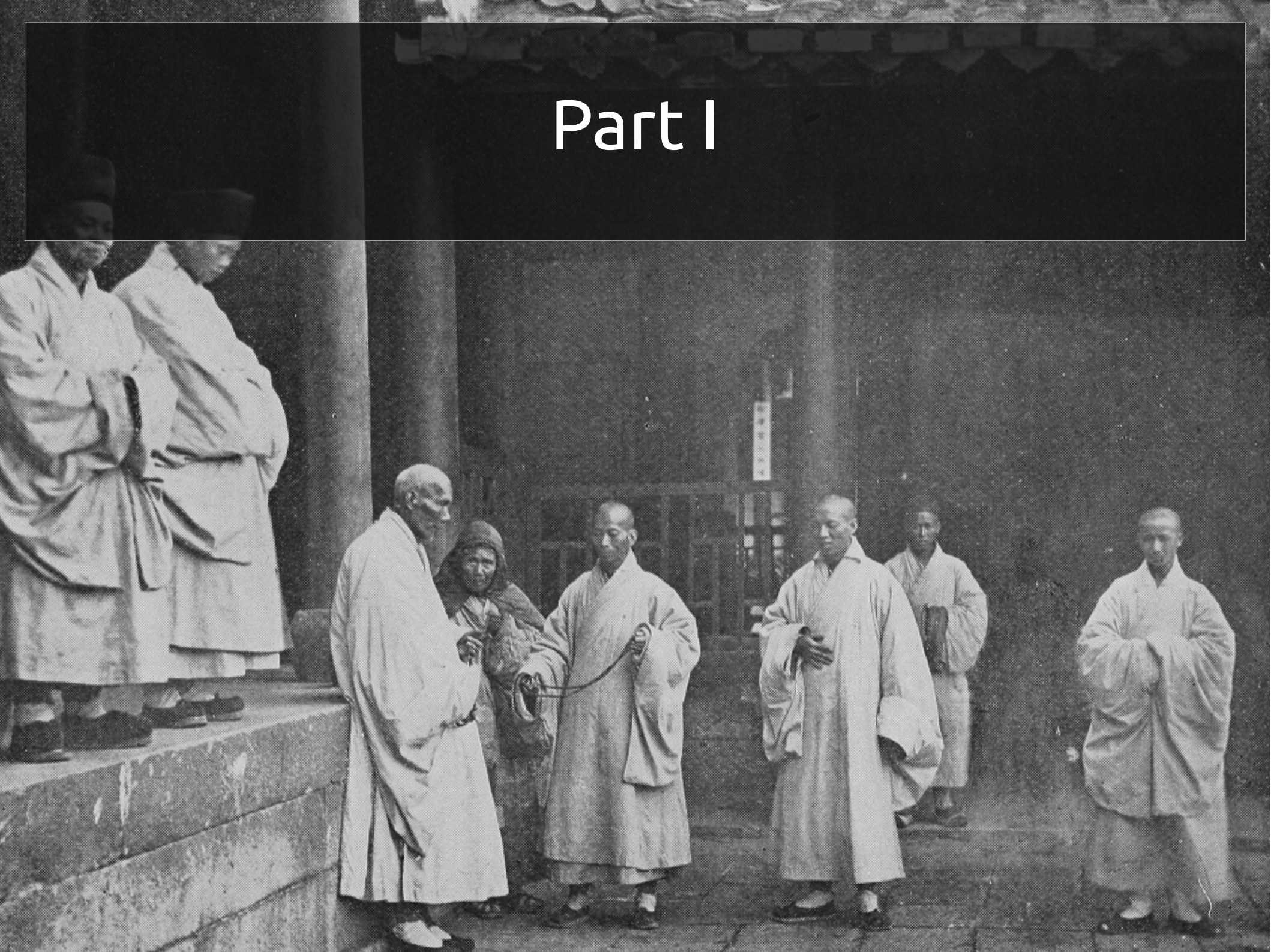

www
load balancers
firewalls
query routing +
reverse proxy
web
legacy
app
API
app
functionality being
migrated to API
mobile-first
friendly
product
database
customer
database
legacy
database split
into two



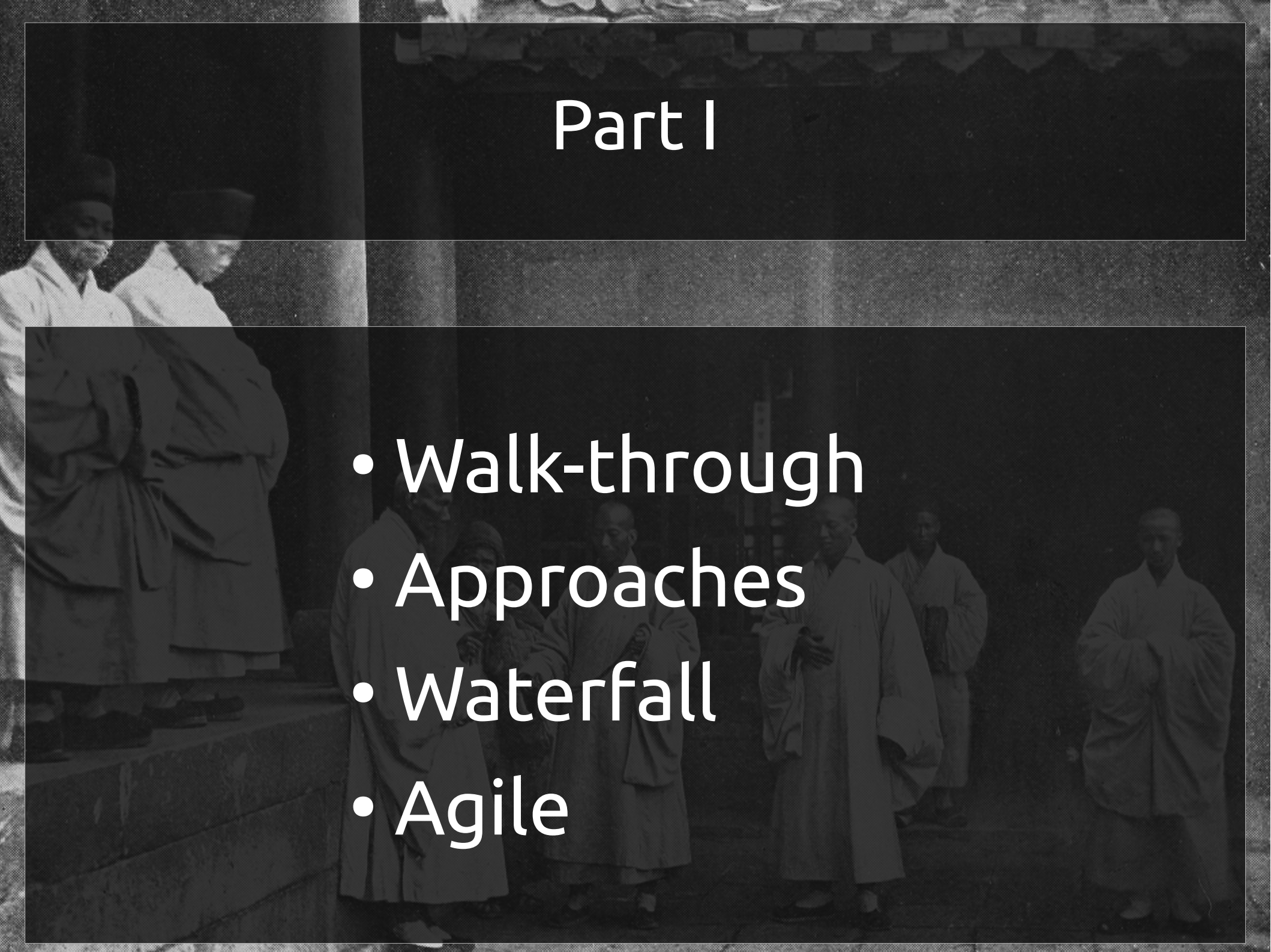
Systematic identification of threats and mitigations


Build better and more robust
systems and services


Road Map



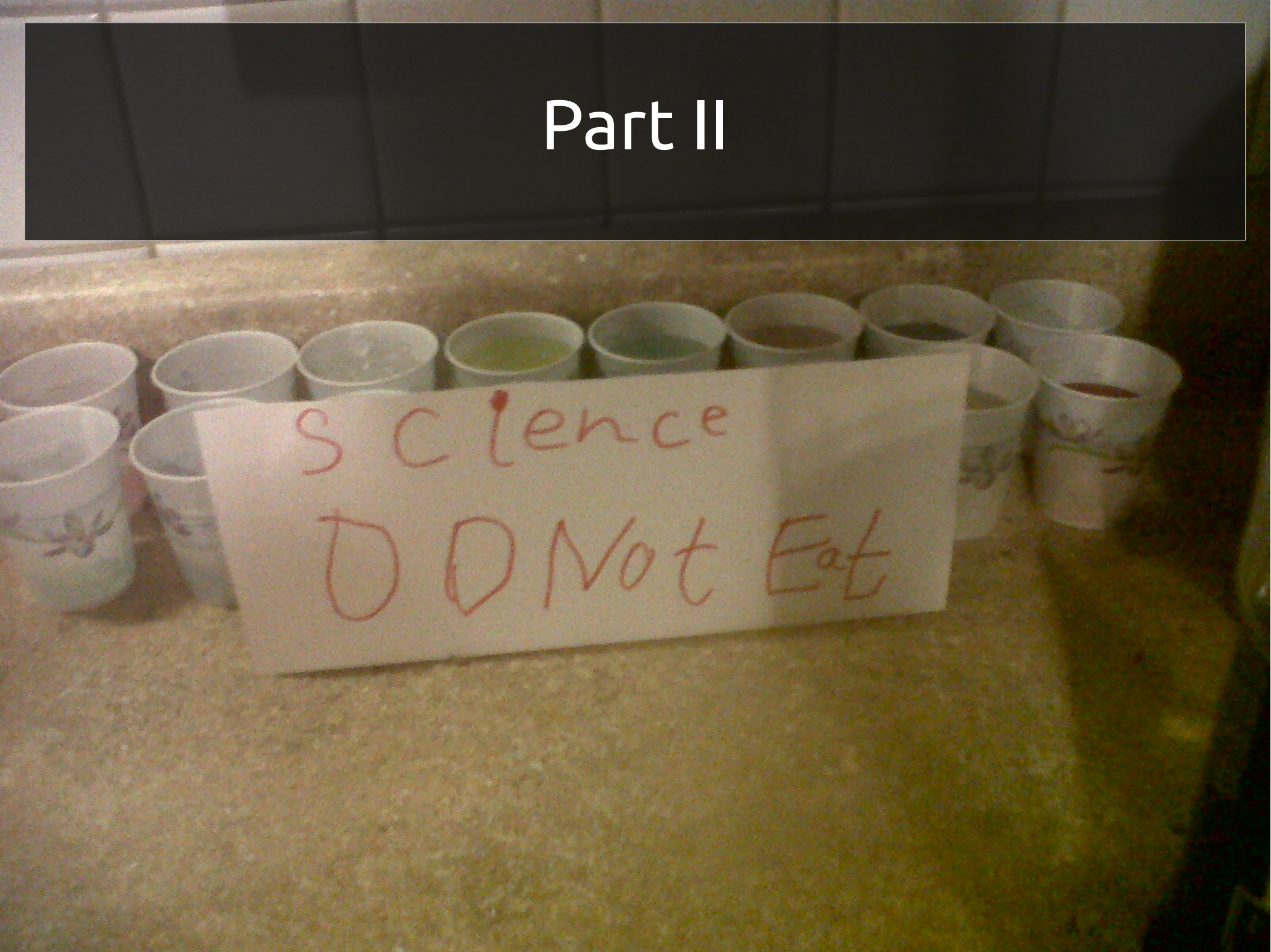
Part I



Part I

- Walk-through
- Approaches
- Waterfall
- Agile

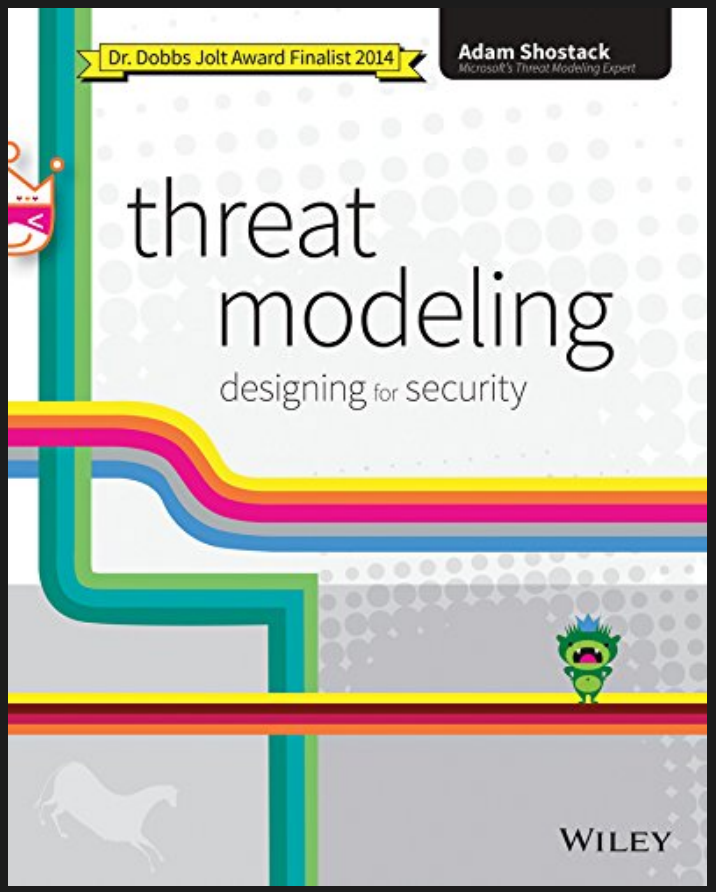
Part II



Part II

The experiment*

* actual science not included



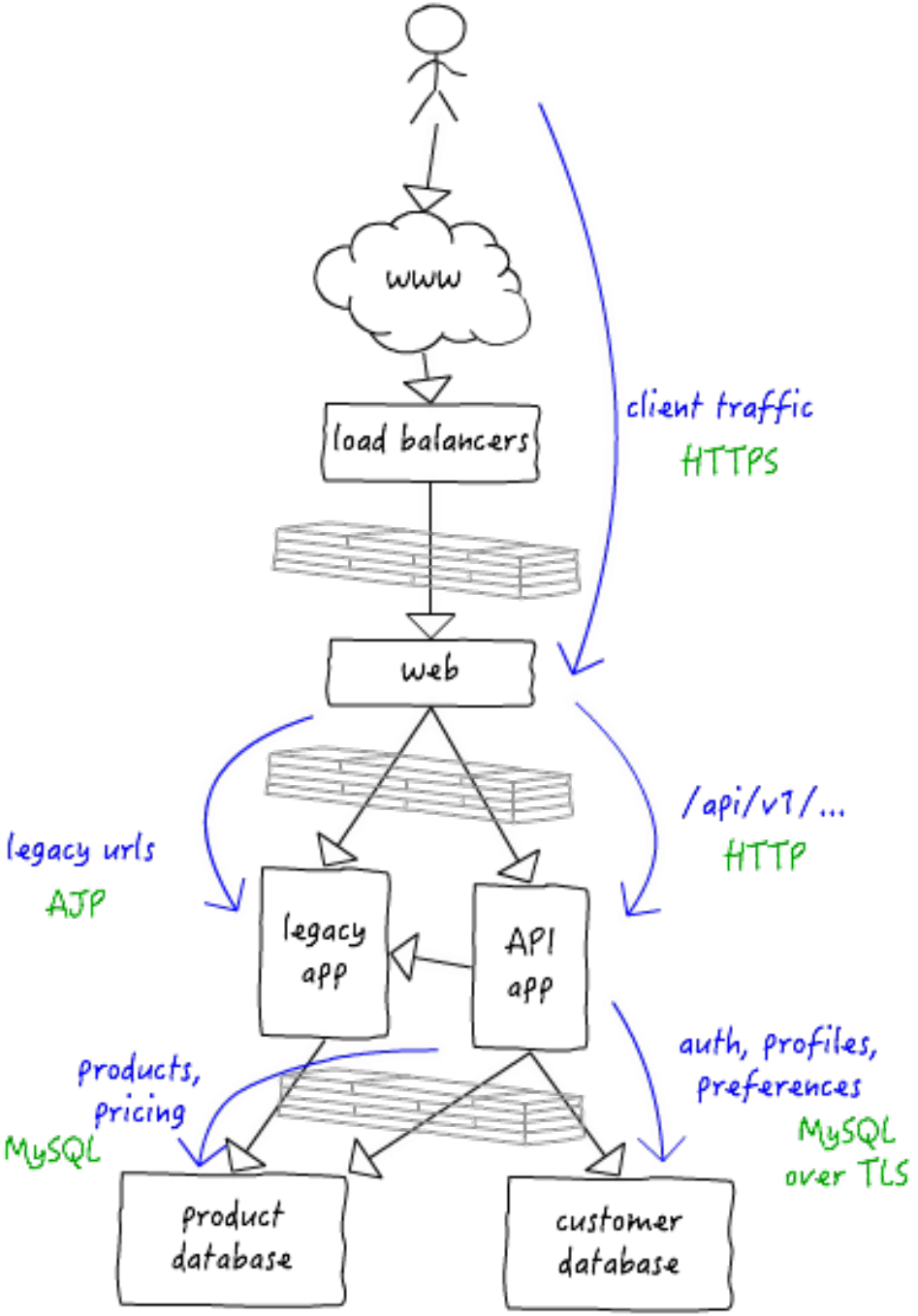
Threat Modeling: Designing for Security

Adam Shostack

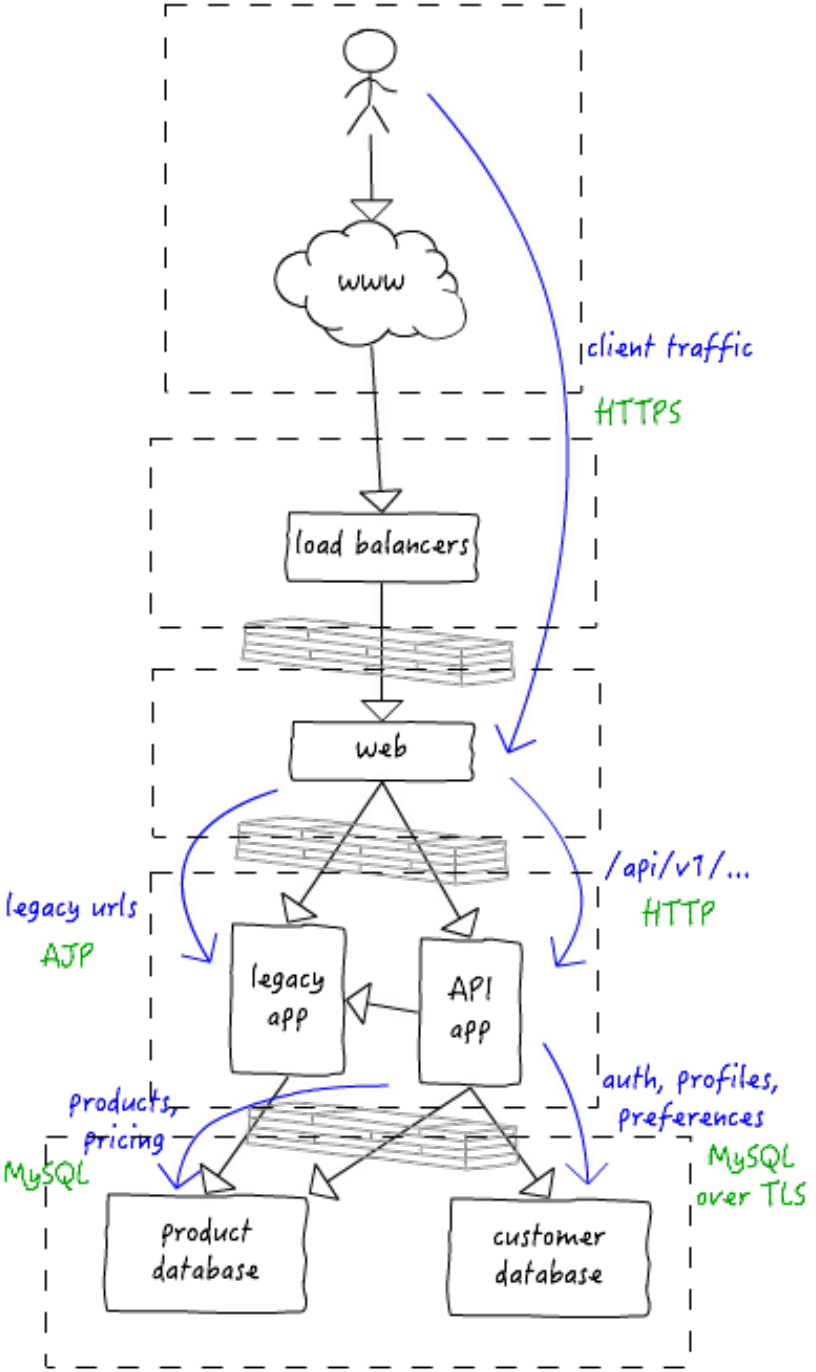
Overview

1. What are you building?
2. What can go wrong?
3. What should you do about the things that can go wrong?
4. Did you do a good job of 1-3?

Data Flow Diagram



Trust boundaries



Assets

- Systems – access to data and pivoting
- Customer records (i.e. PII)
- Product data
- Credentials

Attackers

Motivation and Resources

- Script kiddies
- Hackivists
- Professional criminals
- ~~Chin~~Nation states

Software

The thing that actually delivers value to your organisation

Elevation of Privilege

STRIDE

- Spoofing identity
- Tampering with data
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

STRIDE EXAMPLES

Load balancers / Web

- Bug could lead to a DoS
- 0 day could result in elevation of privilege

Legacy app / API

- Tampering of URL could result in a DoS or elevation of privilege
- Information disclosure of DB creds

Product / customer database

- Elevation of privilege in customer DB would probably result in information disclosure of customer data (PII, credit cards etc)
- Tampering of product could lead to repudiation attack

Mitigate

Eliminate

Transfer

Accept

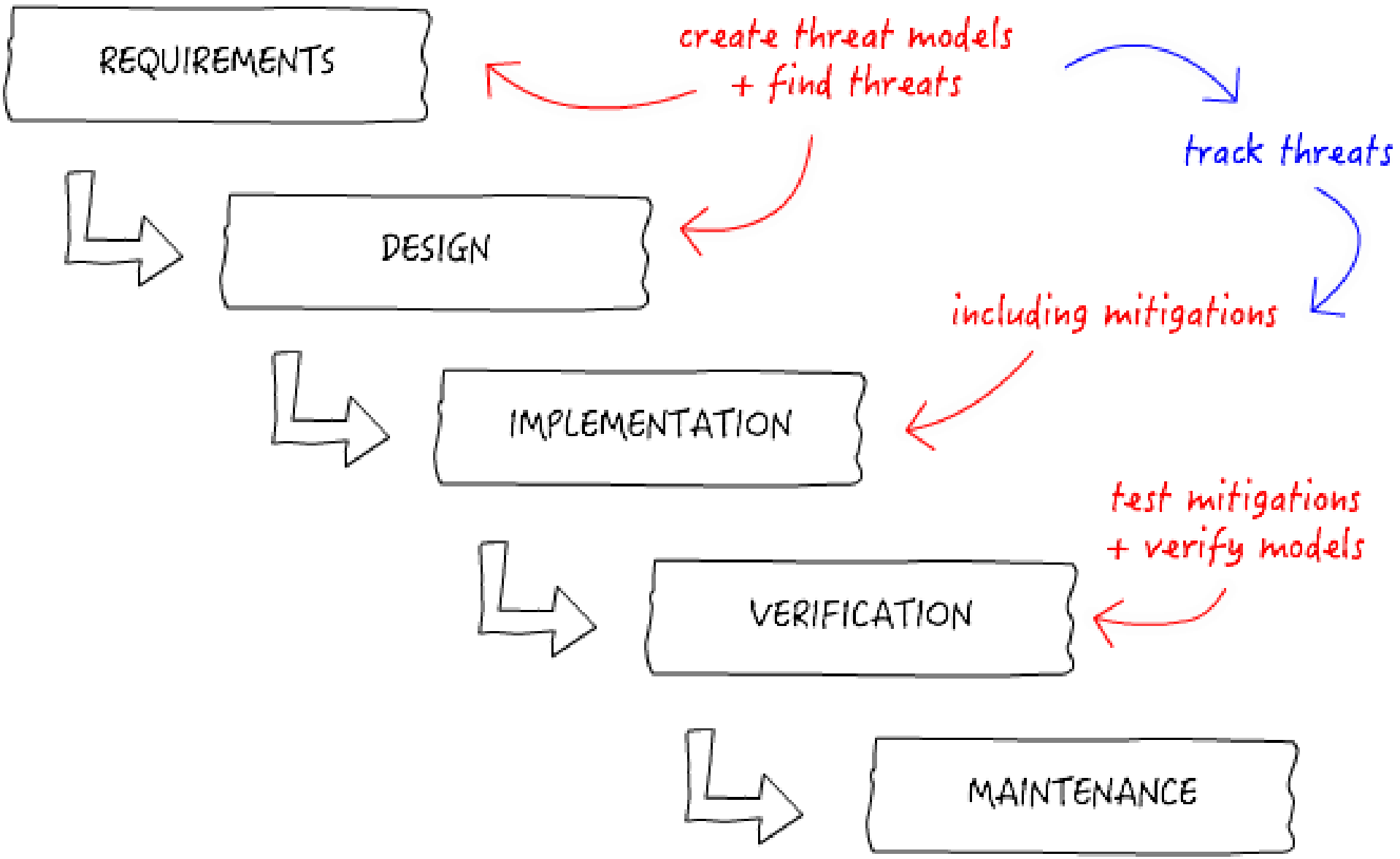
Ignore

Measurement

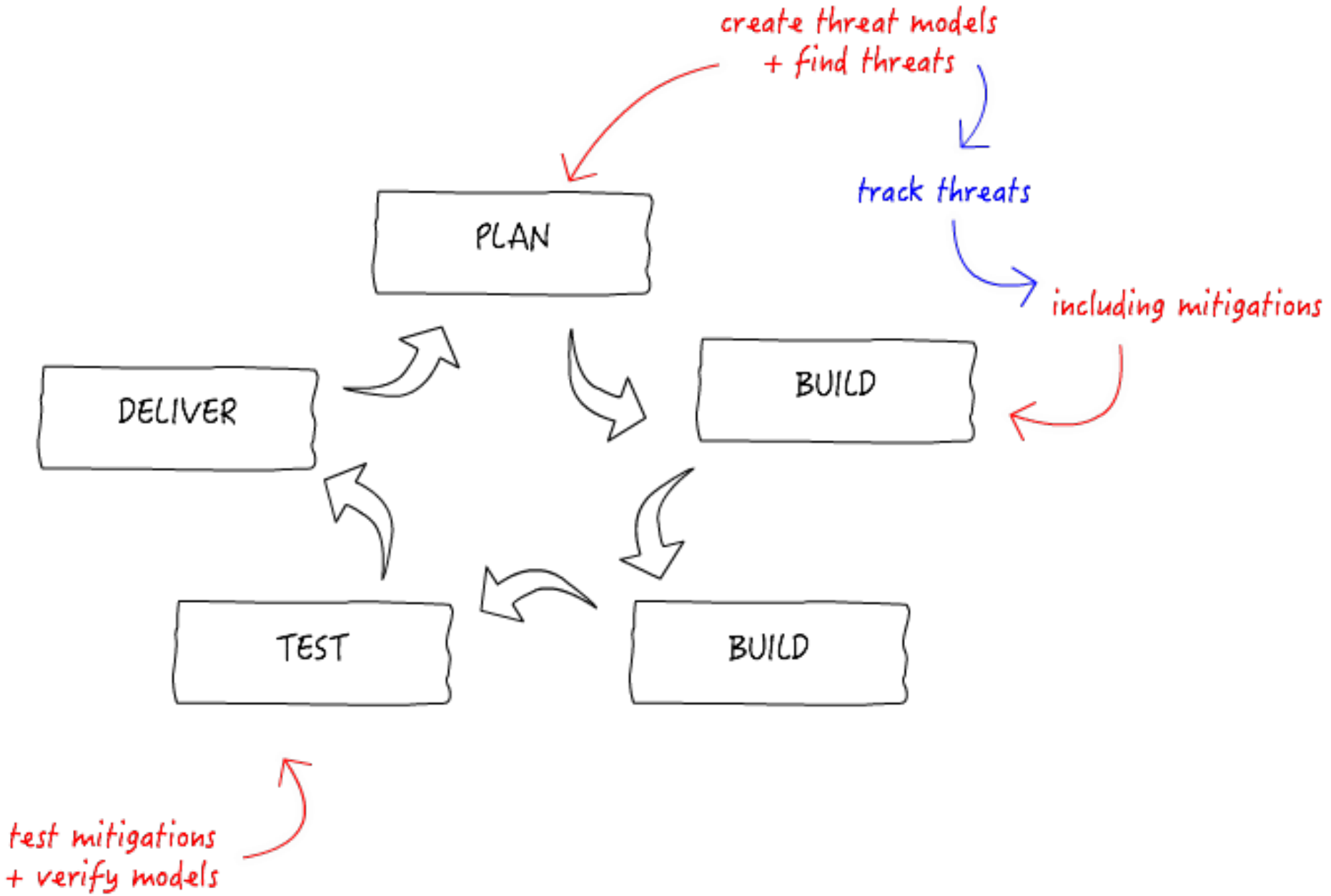
Validation

Keep up to date

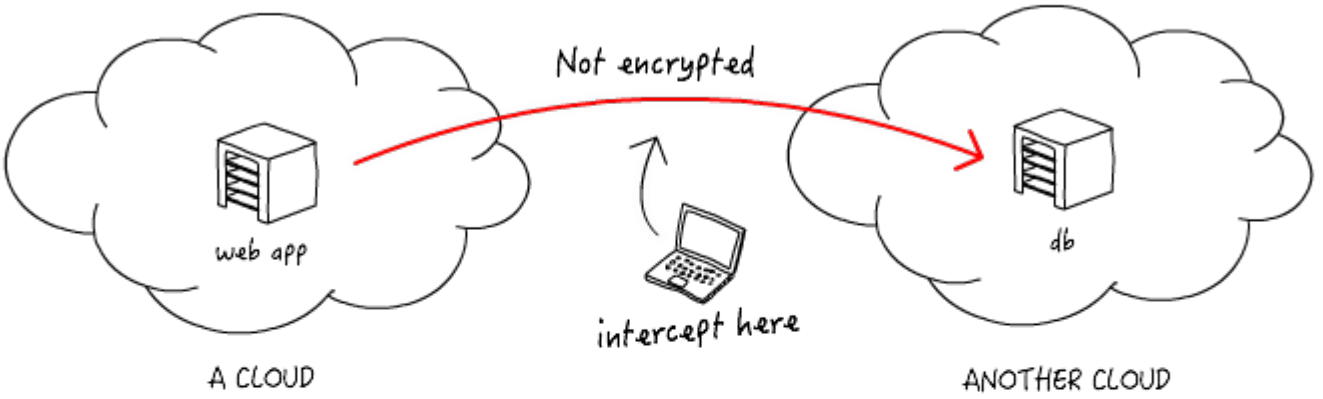
Waterfall


REQUIREMENTS
create threat models
+ find threats
track threats
DESIGN
including mitigations
IMPLEMENTATION
test mitigations
+ verify models
VERIFICATION
MAINTENANCE

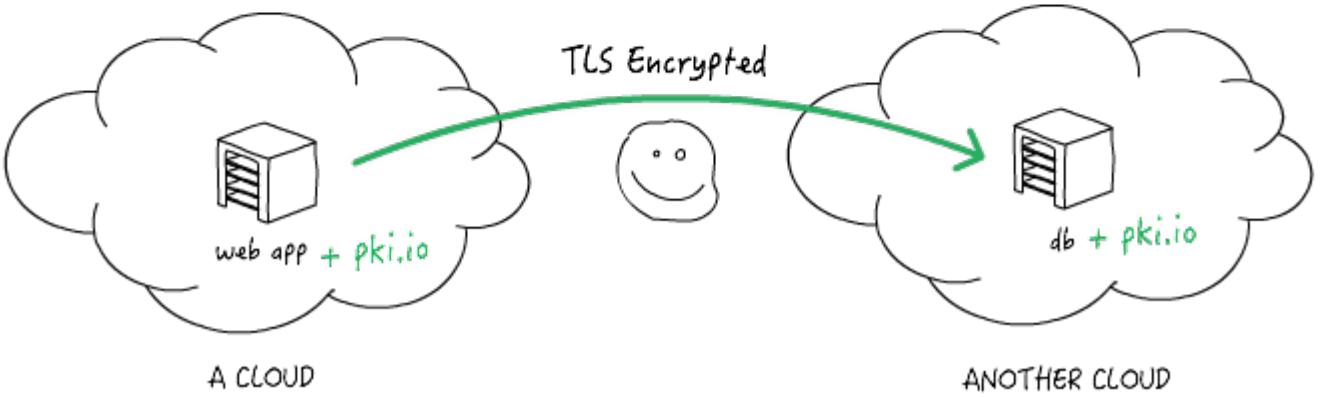
Agile


create threat models
+ find threats
track threats
including mitigations
PLAN
BUILD
BUILD
TEST
DELIVER
test mitigations
+ verify models

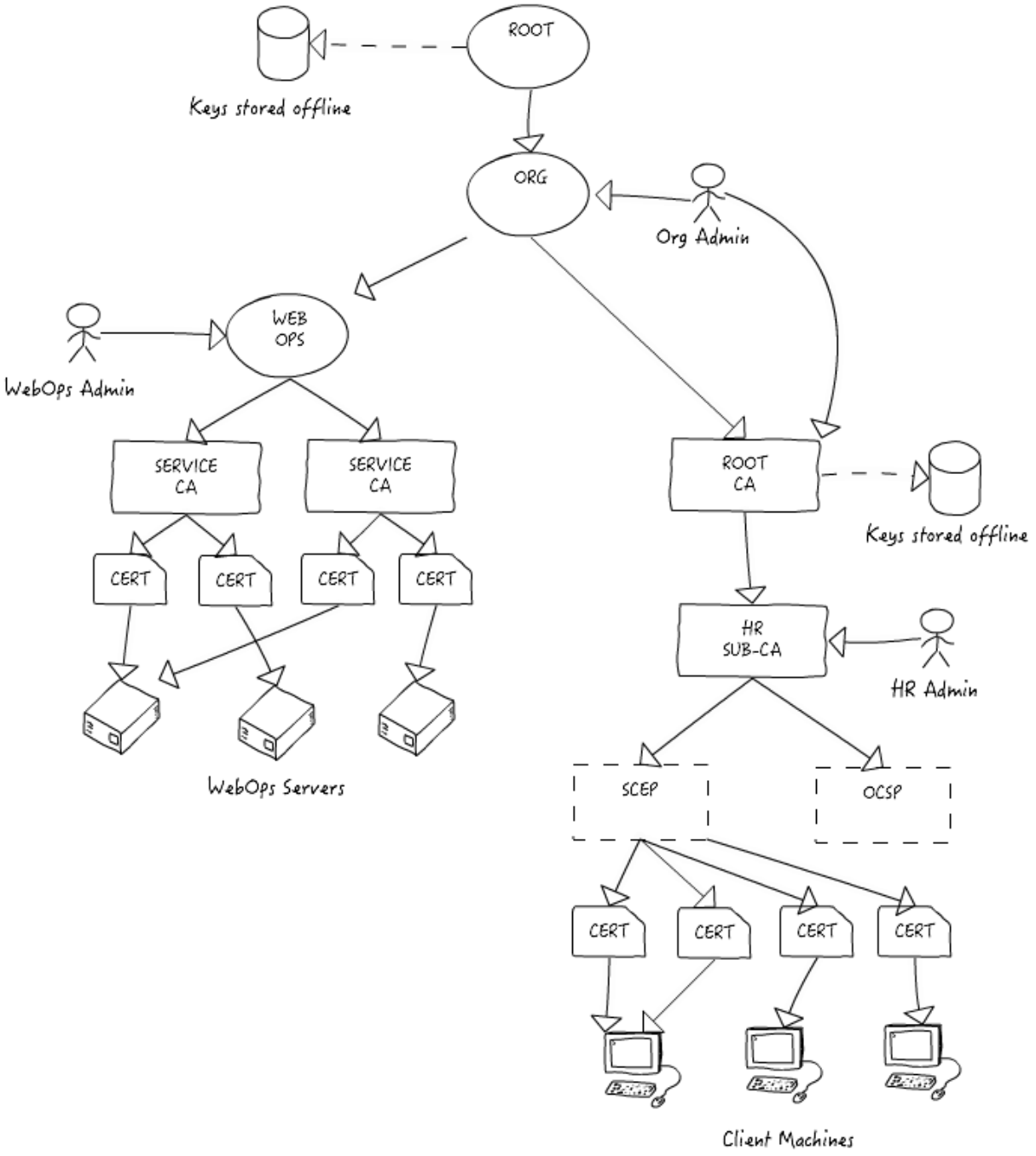
Without pki.io



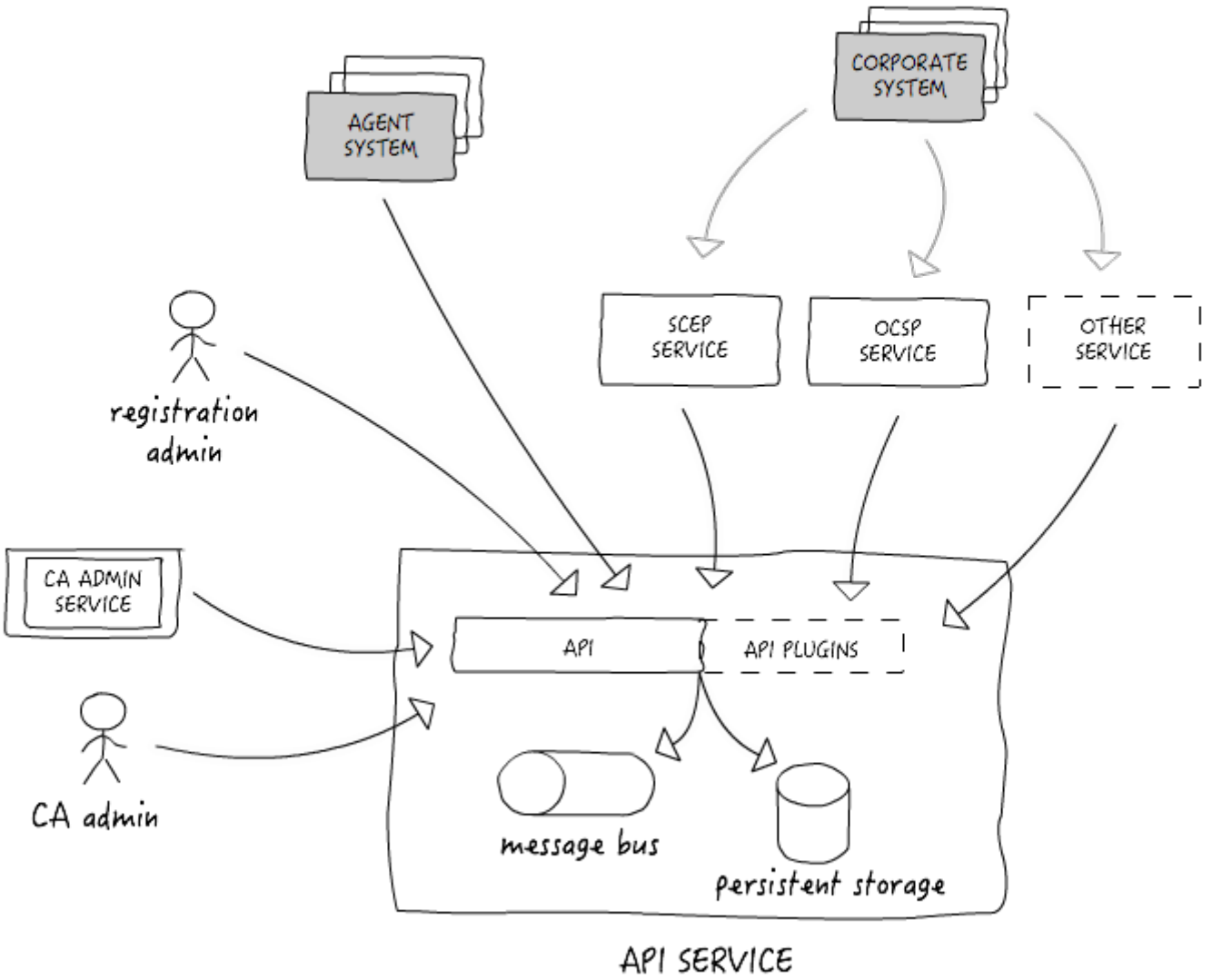
With pki.io



Tiered Architecture



Architecture Overview



Distributed developers

Convinient

Developer-focused


• R-Spec
• Cucumber
• BDD-Security
• Gauntlt

R-Spec

```
# in spec/calculator_spec.rb
RSpec.describe Calculator do
  describe '#add' do
    it 'returns the sum of its arguments' do
      expect(Calculator.new.add(1, 2)).to eq(3)
    end
  end
end
```

Cucumber

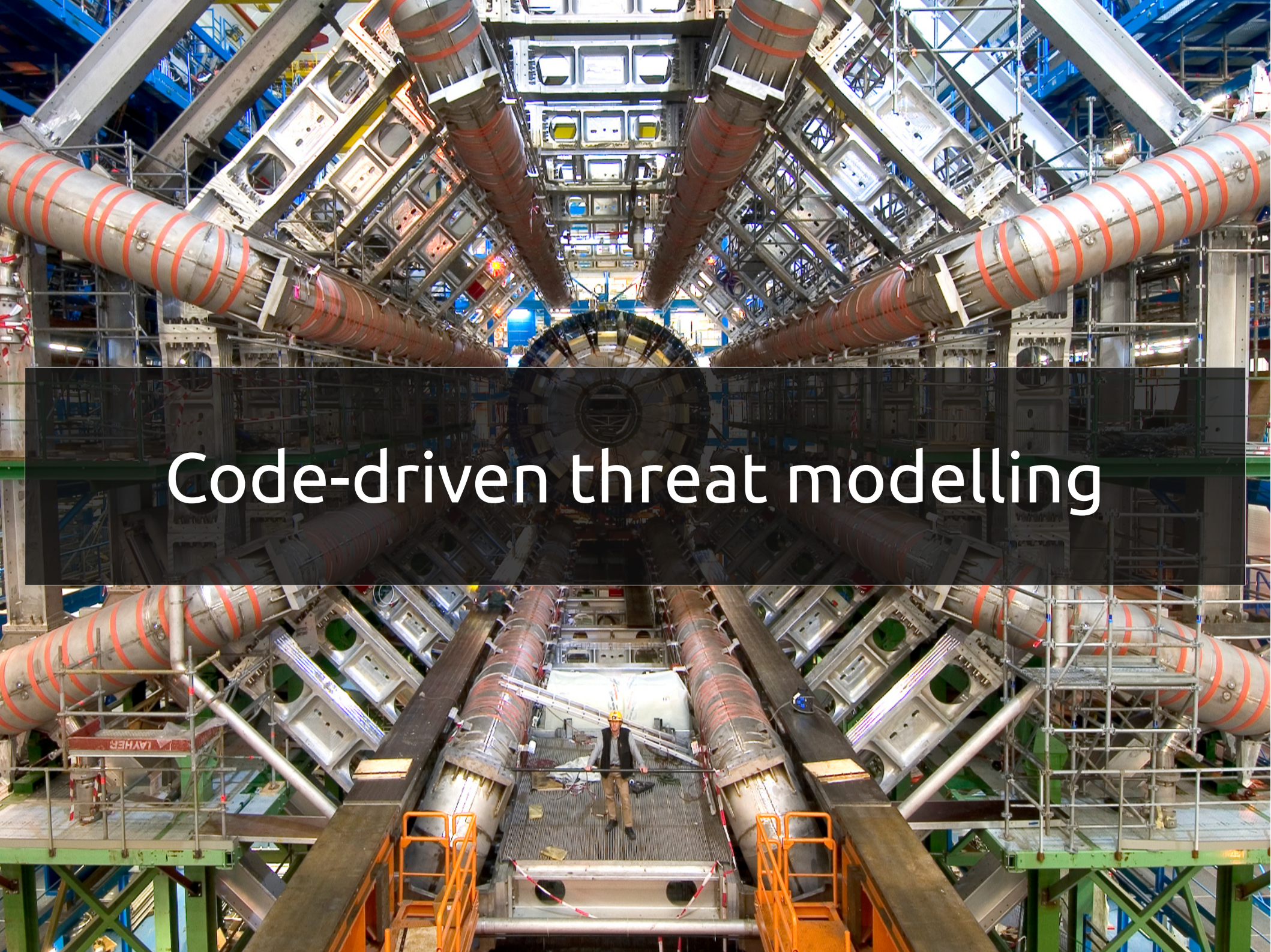
```
Feature: Refund item

  Scenario: Jeff returns a faulty microwave
    Given Jeff has bought a microwave for $100
    And he has a receipt
    When he returns the microwave
    Then Jeff should be refunded $10
```

Bdd-security

```
Scenario: Present the login form itself over an HTTPS connection
Meta: @id auth_login_form_over_ssl @cwe-295-auth @browser_only
Given a new browser instance
And the client/browser is configured to use an intercepting proxy
And the proxy logs are cleared
And the login page
And the HTTP request-response containing the login form
Then the protocol should be HTTPS
```

Gauntlt

```
# nmap-simple.attack
Feature: simple nmap attack to check for open ports

  Background:
    Given "nmap" is installed
    And the following profile:
      | name     | value       |
      | hostname | example.com |

  Scenario: Check standard web ports
    When I launch an "nmap" attack with:
      """
      nmap -F <hostname>
      """
    Then the output should match /80.tcp\s+open/
    Then the output should not match:
      """
      25\/tcp\s+open
      """
```



Code-driven threat modelling

- Components
- Trust boundaries
- Threats
- Mitigations
- Other stuff

Exposes WebApp:FileSystem to arbitrary file writes with insufficient path validation

Mitigates WebApp:FileSystem against unauthorised access with strict file permissions

```
\s*(?:\/\/|\#)\s*Mitigates
(?<component>.+?) against (?
<threat>.+?) with (?
<mitigation>.+?)\s*(?:\((?
<ref>.*?)\))?\s*$
```

```
var (
	addr = flag.Bool("addr", false, "find open address and print to final-port.txt")
)

type Page struct {
	Title string
	Body  []byte
}

// ThreatSpec SimpleV1 for (*main.Page).save
// Does page saving for WebApp:FileSystem
// Exposes WebApp:FileSystem to arbitrary file writes with insufficient path validation
// Mitigates WebApp:FileSystem against unauthorised access with strict file permissions
// Sends notification email from WebApp:App to User:Mail Client

func (p *Page) save() error {
	filename := p.Title + ".txt"
	return ioutil.WriteFile(filename, p.Body, 0600)
}

// ThreatSpec SimpleV1 for main.loadPage
// Does page loading for WebApp:FileSystem
// Exposes WebApp:FileSystem to arbitrary file reads with insufficient path validation

func loadPage(title string) (*Page, error) {
	filename := title + ".txt"
```

ThreatSpec Report for ...

Analysis
* Functions found: 9
* Functions covered: 88.89% (8)
* Functions tested: 12.5% (1)

Components
WebApp FileSystem
Threat: arbitrary file writes
* Exposure: insufficient path validation ((*main.Page).save in simple.go:31)

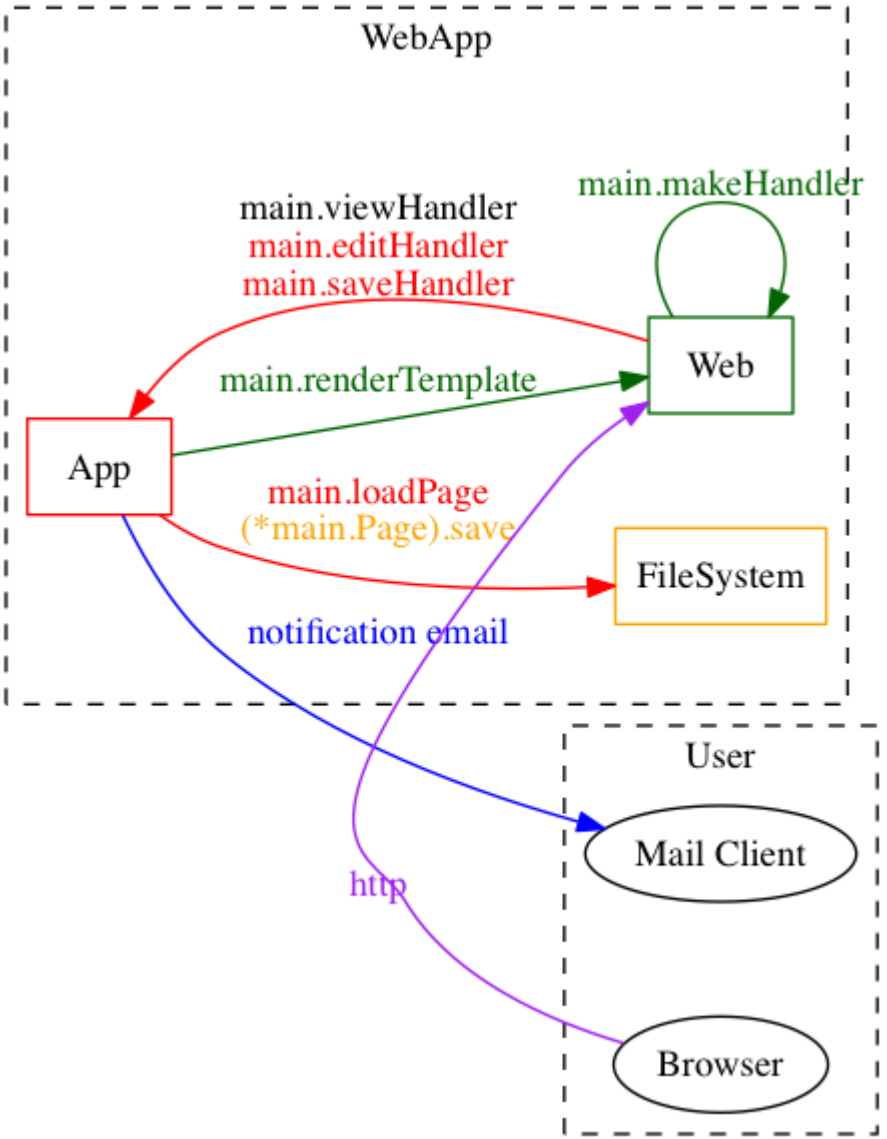
Threat: arbitrary file reads
* Exposure: insufficient path validation (main.loadPage in simple.go:40)

WebApp App
Threat: XSS injection
* Exposure: insufficient input validation (main.editHandler in simple.go:65)

Threat: content injection
* Exposure: insufficient input validation (main.saveHandler in simple.go:77)

Rendered report

```
$ callgraph *.go | ./threatspec.rb *.go
```


WebApp
main.makeHandler
main.viewHandler
main.editHandler
main.saveHandler
Web
main.renderTemplate
App
main.loadPage
(*main.Page).save
FileSystem
notification email
User
Mail Client
http
Browser

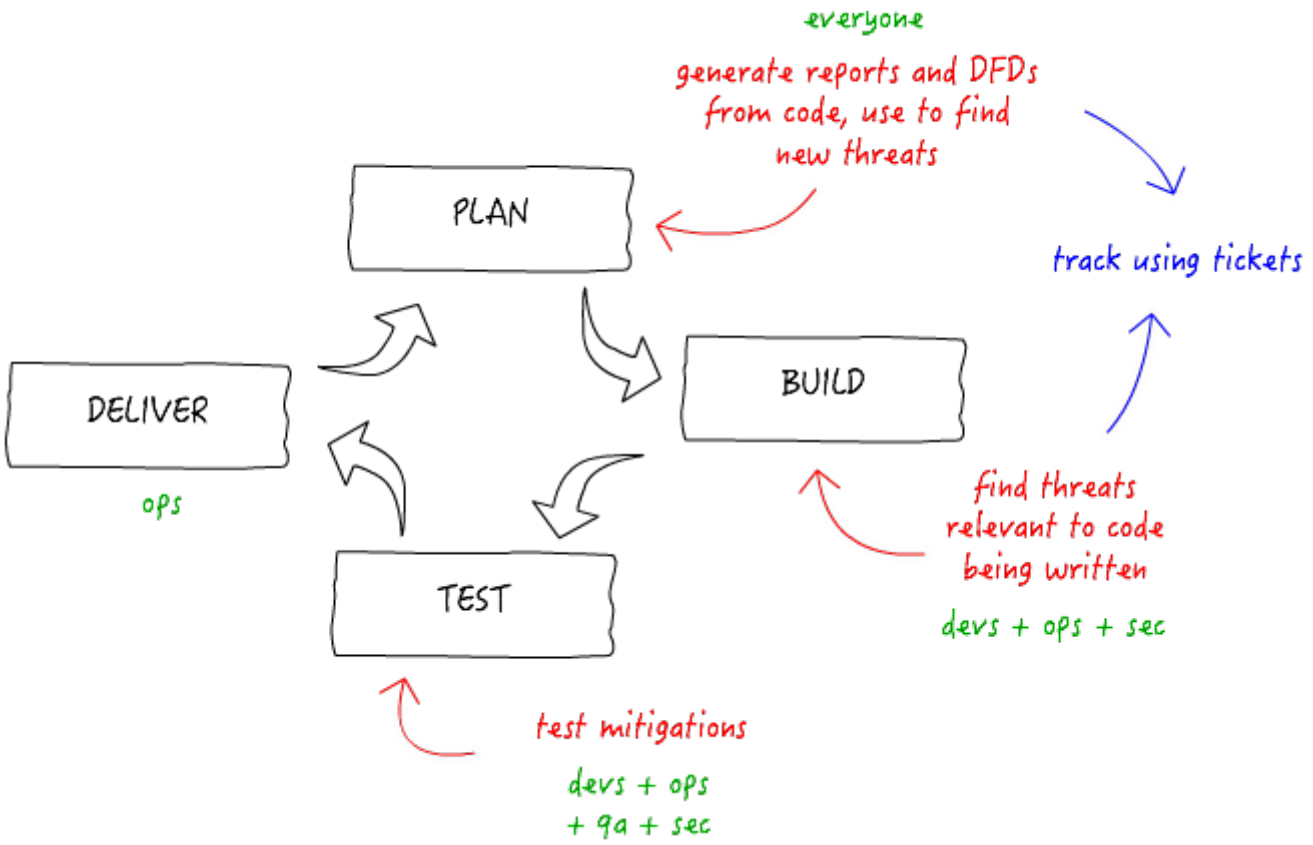
```
// ThreatSpec SimpleV1 for main.TestListenHandler
// Tests (*main.Page).ListenHandler() for TLS listener
func TestListenHandler(t *testing.T) {
  ...
```

Workflow

Devs write threatspec as they write new functions and tests

Review by security or senior devs

Periodic review of generated reports and DFDs


everyone
generate reports and DFDs from code, use to find new threats
PLAN
track using tickets
BUILD
DELIVER
ops
find threats relevant to code being written
devs + ops + sec
TEST
test mitigations
devs + ops + qa + sec

Problems?

- Starting point – rough DFD
- Complexity of generated DFD
- Missing functions etc
- Typos breaking stuff

The good stuff

Dev and Sec working together

Bigger picture

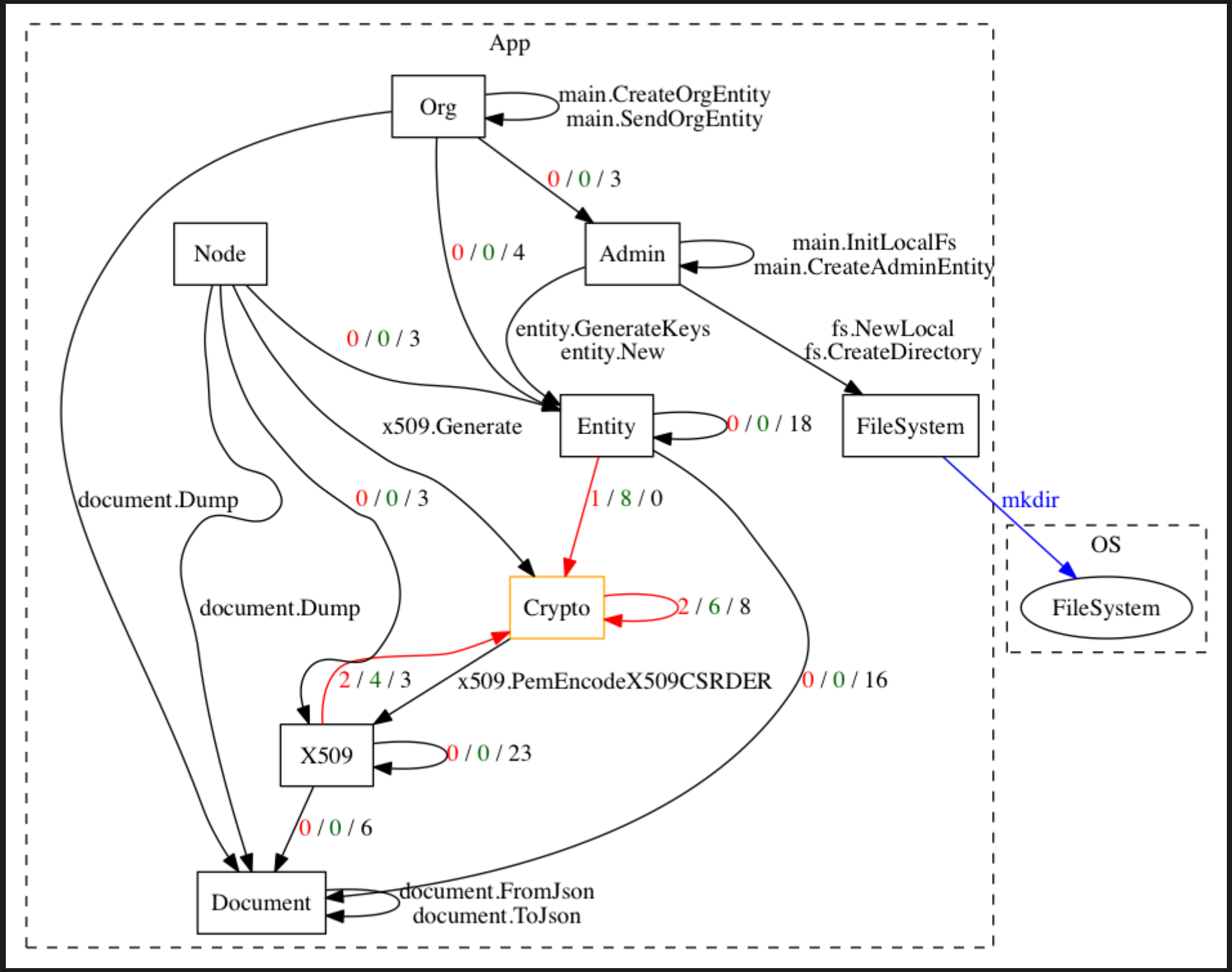
Model and code in sync

Pki.io example

- Admin and core
- Number of functions
- Several attempts
- Retro-fitting threatspec
- Started off with just

Example spec

Generated report as html


App
Org
main.CreateOrgEntity
main.SendOrgEntity
0 / 0 / 3
Node
0 / 0 / 4
Admin
main.InitLocalFs
main.CreateAdminEntity
0 / 0 / 3
entity.GenerateKeys
entity.New
fs.NewLocal
fs.CreateDirectory
x509.Generate
Entity
0 / 0 / 18
FileSystem
document.Dump
0 / 0 / 3
1 / 8 / 0
mkdir
OS
FileSystem
document.Dump
Crypto
2 / 6 / 8
2 / 4 / 3
x509.PemEncodeX509CSRDER
0 / 0 / 16
X509
0 / 0 / 23
0 / 0 / 6
Document
document.FromJson
document.ToJson

In conclusion...

- Threat modelling is awesome
- You should probably be doing it
- Get people involved
- Find an approach that works for you
- Code-driven threat modelling may even work

The future?

The future?

- Improvements
- Ceremony
- Infrastructure as code

Fraunhofer AISEC / SSE in partnership with TU Munich

Translate From: German To: English View: Translation Original

Chair for IT Security

TECHNISCHE UNIVERSITÄT MÜNCHEN

Home Teaching Research Publications People Projects Student Work Jobs Malware Zoo

Student Work "Open Topic" code close protection concepts

Code-close protection concepts

Type of work: Bachelor's thesis, Master's Thesis

Supervisor: Stephan Renatus

Posted: 06/24/2014

Tender in cooperation with the Fraunhofer Institute for Applied and Integrated Security AISEC, Garching

Fraunhofer AISEC

Background

The protection concept of an application reflects what threats is exposed to the application and what measures responded. In software systems implemented protection mechanisms are partly realized in the source code, so that in addition to a follow - What measures have been implemented? - Characteristics of the code (changes or metrics) can be returned based on the protection concept. For this purpose, a combination of code and protection concept is required.

Task

In this work a concept for language-independent, lightweight specification of threats and mechanisms should be worked out in the Code. Given initially to existing approaches such as ThreatSpec be researched and evaluated for the connection of code and protection concept. The concept is then implemented in a tool and evaluated on an application example. Here are both questions of creation and the necessary care of such specified protection concept and adjoining facilities, such as the parameterization of vulnerability scanners, part of the consideration.

Student Work

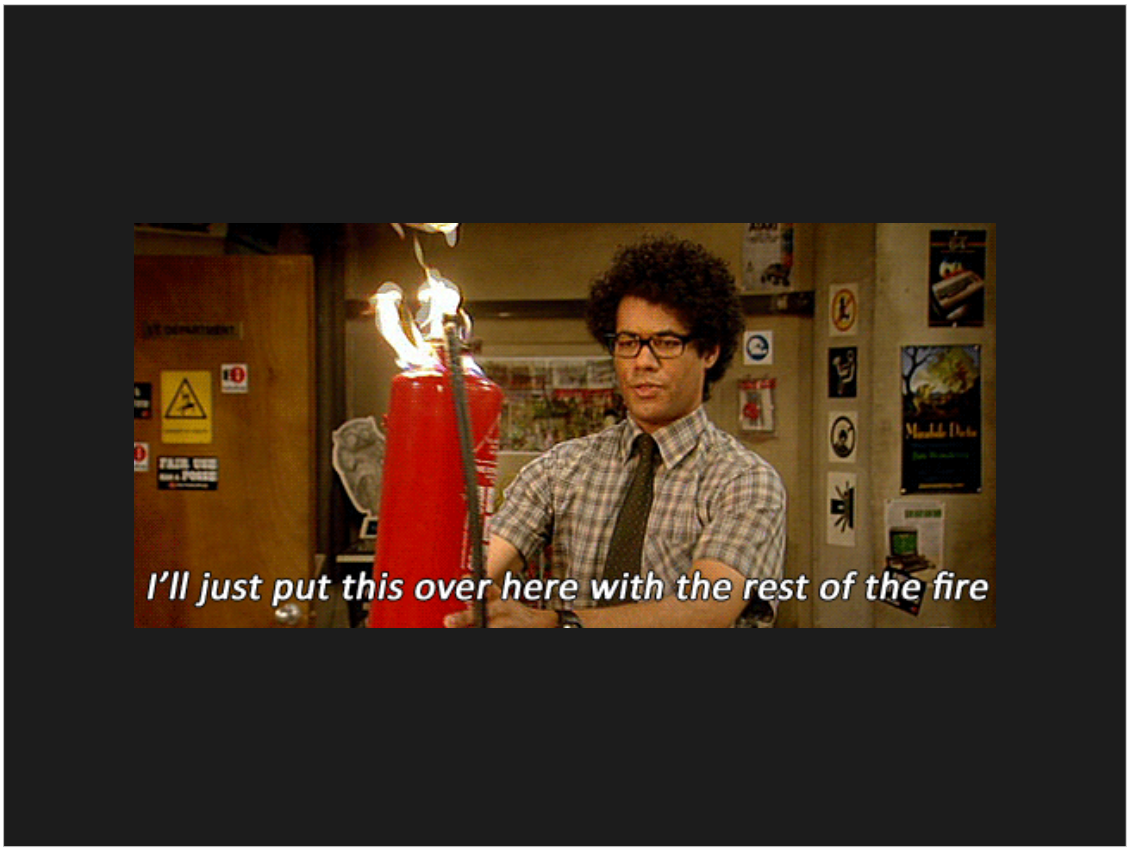
- Open Topics
  - Android Security
  - Code-close protection concepts
  - Development of a Simulat ...
  - Development of a secure ...
  - Static Detection of Inte ...
  - Static Vulnerability Det ...
  - Reducing the Attack Surf ...
  - Reverse Engineering the ...
  - Master's Thesis: Self-ad ...
  - Tool support for ...
  - Vetrauenswurdige PaaS Sy ...
- In Progress
- Finished Work

threatspec.org

An experiment in Agile Threat Modelling

Fraser Scott
@zeroXten

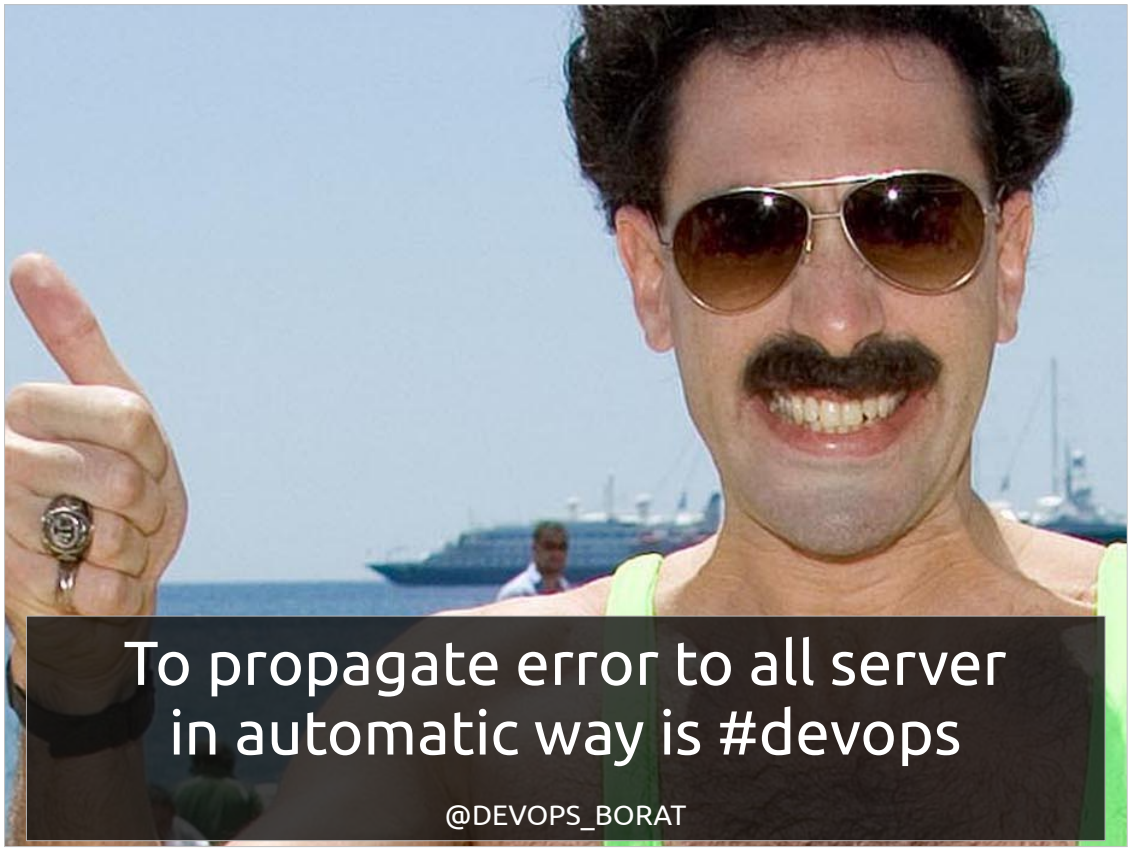
How many of you – after an outage, security incident or close-call – have thought ...



"hmm, should probably have thought that through a little bit more..."

Everyone right? These things happen...

It's natural. To err is human...



To some extent, devops brings with it a new set of challenges.

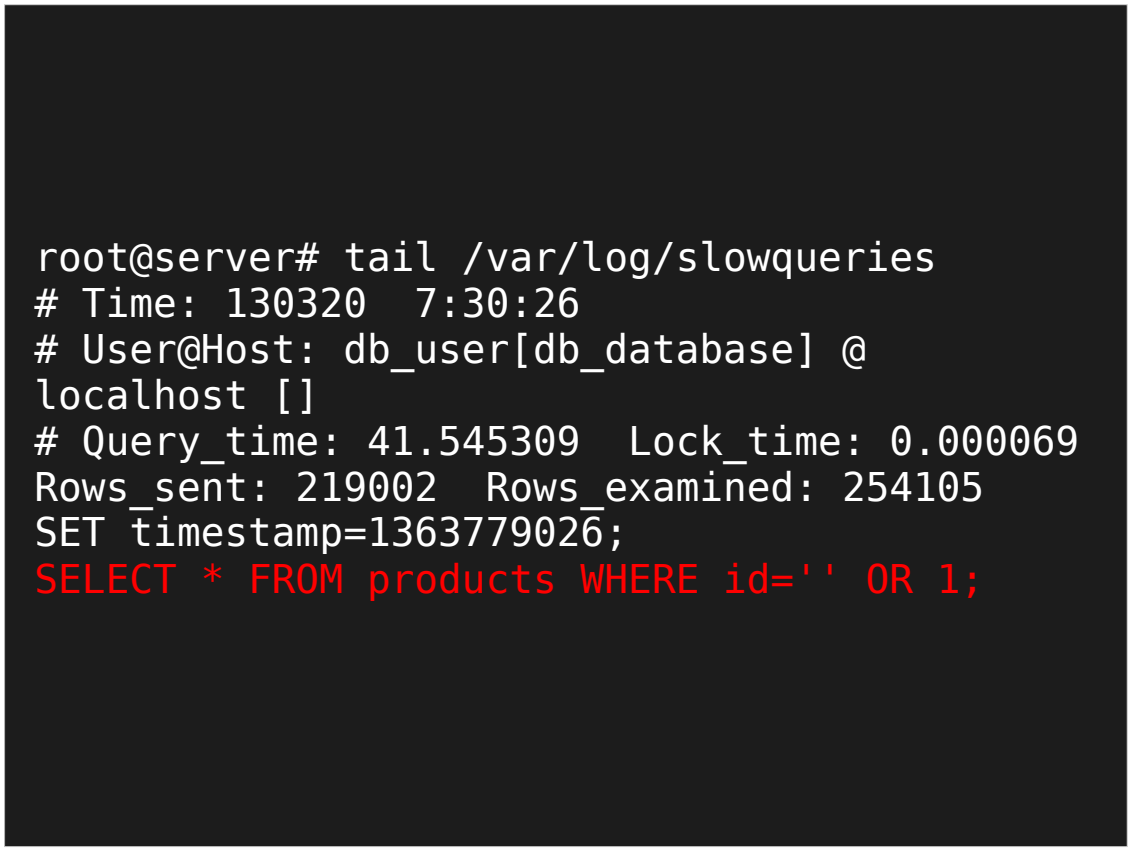
I'm going to start off with a little story.

So you've got all your shiny dashboards. You're collecting all the stats into graphite or influxdb, alerts are generated by Nagios, Zabbix or Sensu, and your ElasticSearch cluster is hoovering up more data than the NSA.

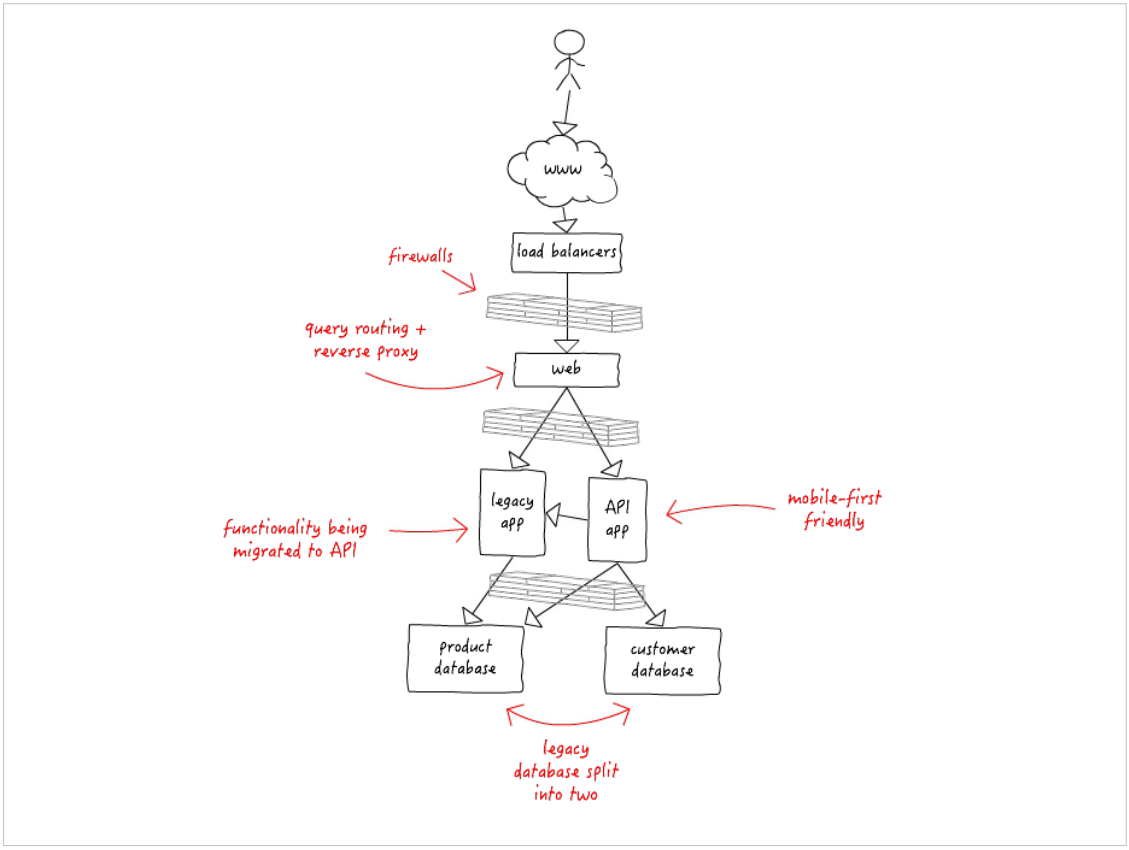
Your monitoring screens probably look something like...

… this.

One day you get an alert for a load spike on one of the legacy application servers, immediately followed by a load spike on one of the product database servers. Then pop! - the application process crashes. Traffic levels look average, but you notice a spike in error logs from the app server. You dig around the raw logs in ELK, suddenly your jaw drops...

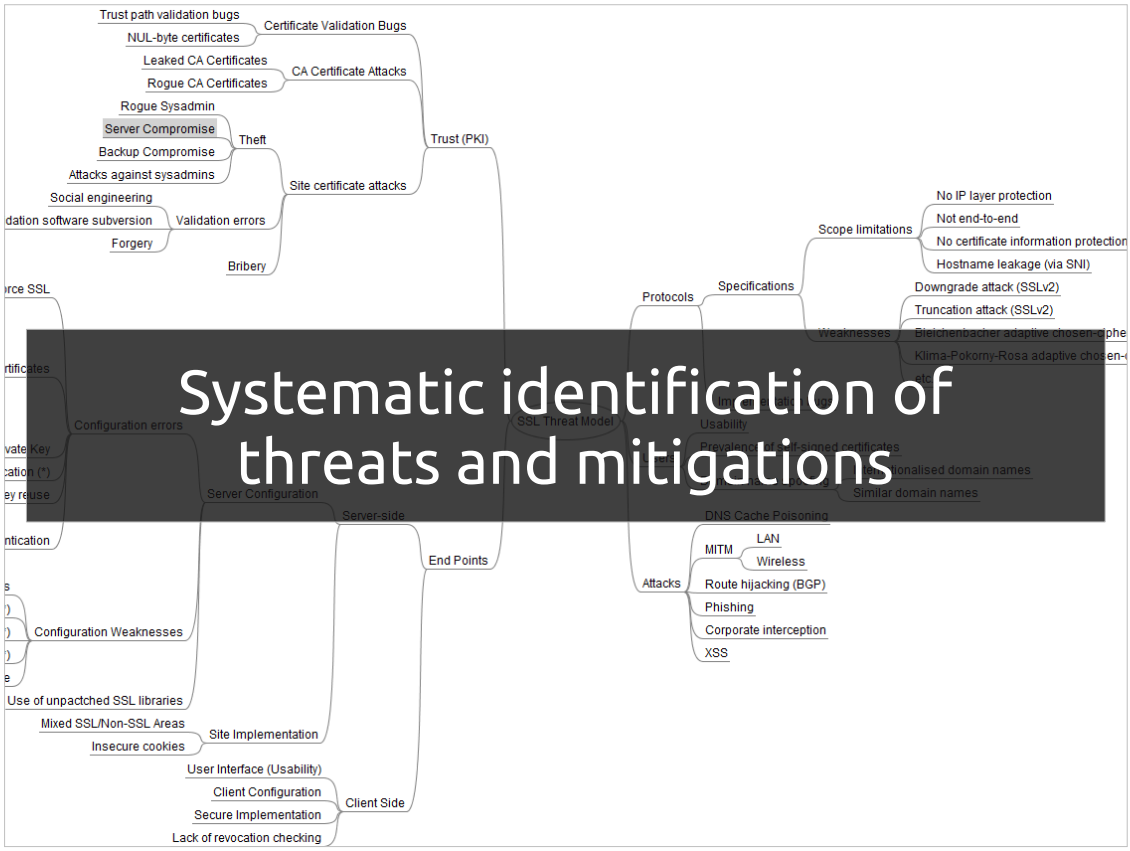


.. an SQLi attack. Did they succeed? Does that explain the database load spike? How did this happen? Don't we mitigate that sort of thing? You quickly get your manager, a DBA and the security guy involed. Before you know it you're in a war room pouring over graphs and logs. They managed to pull a load of data out of the product database which trigged a cache bug in the application – hence it crashed. Still, it doesn't look like they got into the customer database...



The post mortem lasts 4 hours. Lucky they didn't get customer data, otherwise you'd be there for 4 weeks. At least it's behind a firewall and only accessible from the API servers. Wait, same subnet as the product databases? Same admin user accesses both databases? Shit. The SQLi? New component wedged in to the legacy app last week. Didn't they test it?

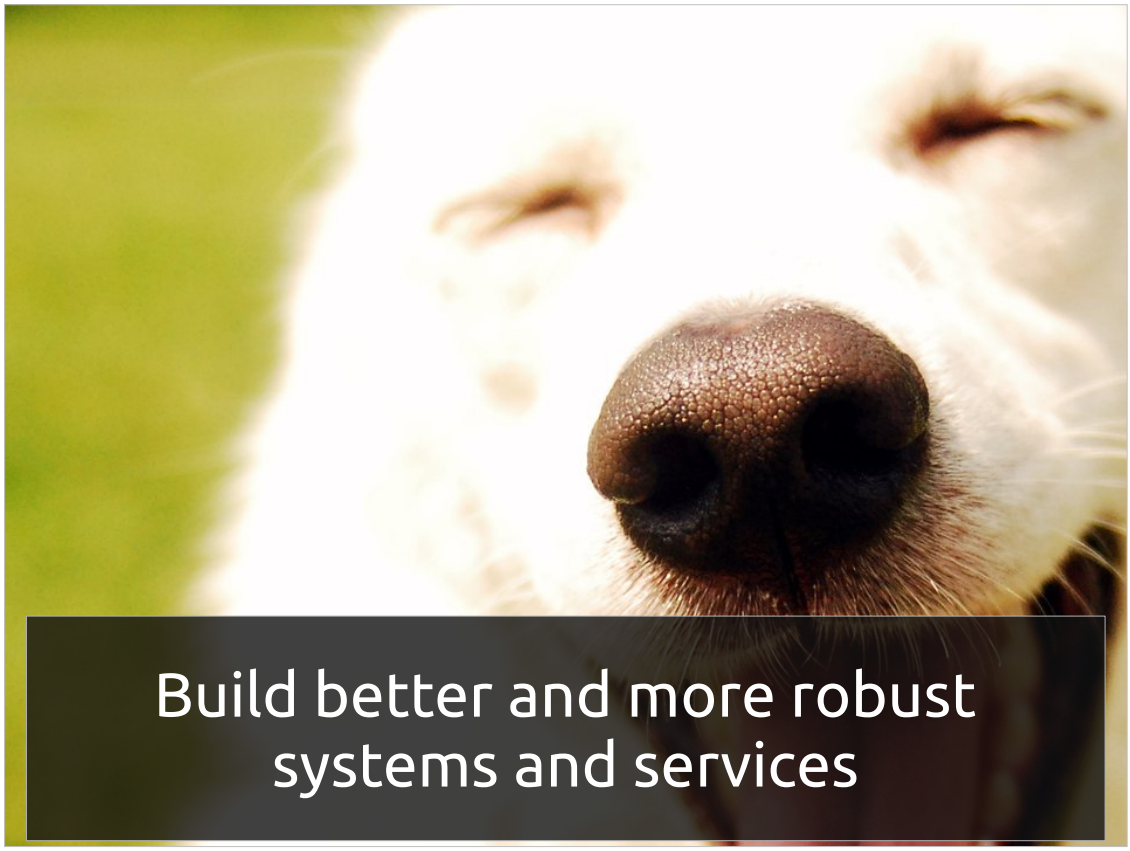
You spend the last hour discussing how the components connect, where the strengths and weaknesses are, creating tickets as you go. A follow up meeting is arranged. An informal threat modelling session...



Think of threat modelling as a post-mortem, *before* the incident happens.

Background image is TM of SSL by Ivan Ristic

So, why would we do formal threat modelling?...

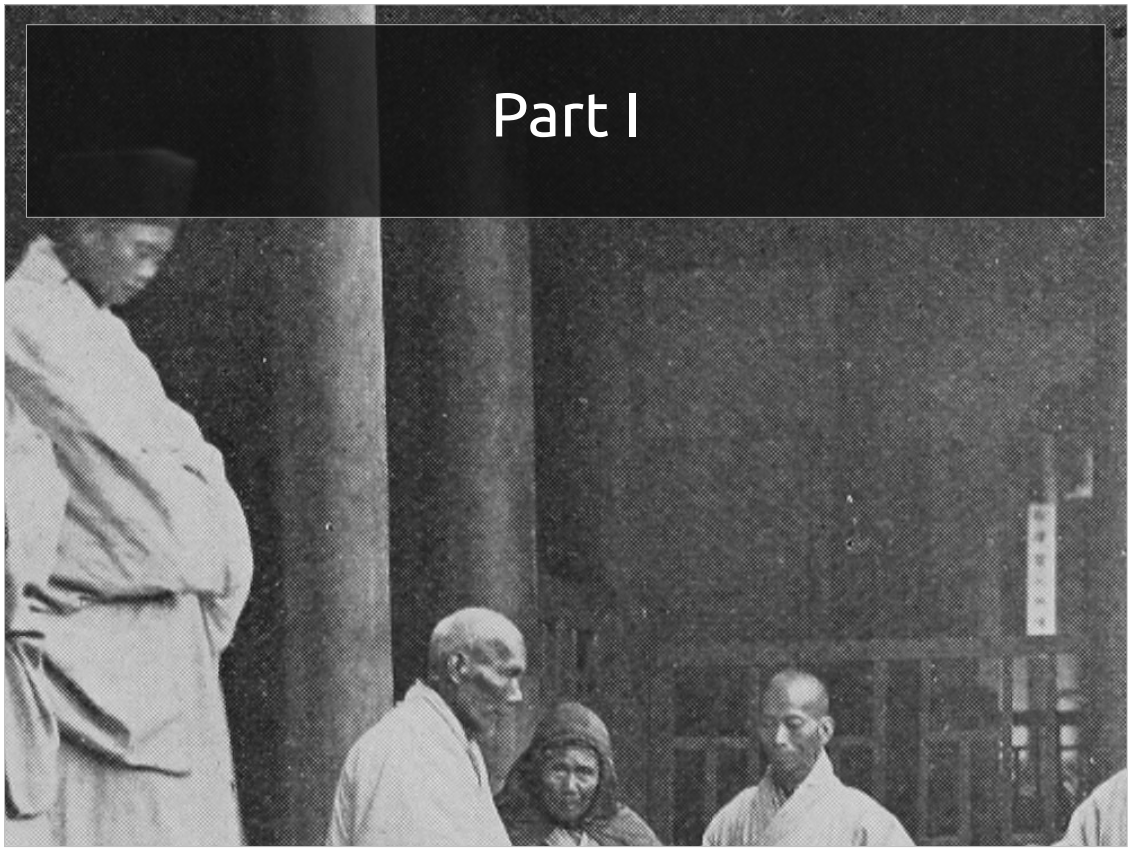


To make things even more awesome and to help keep them awesome

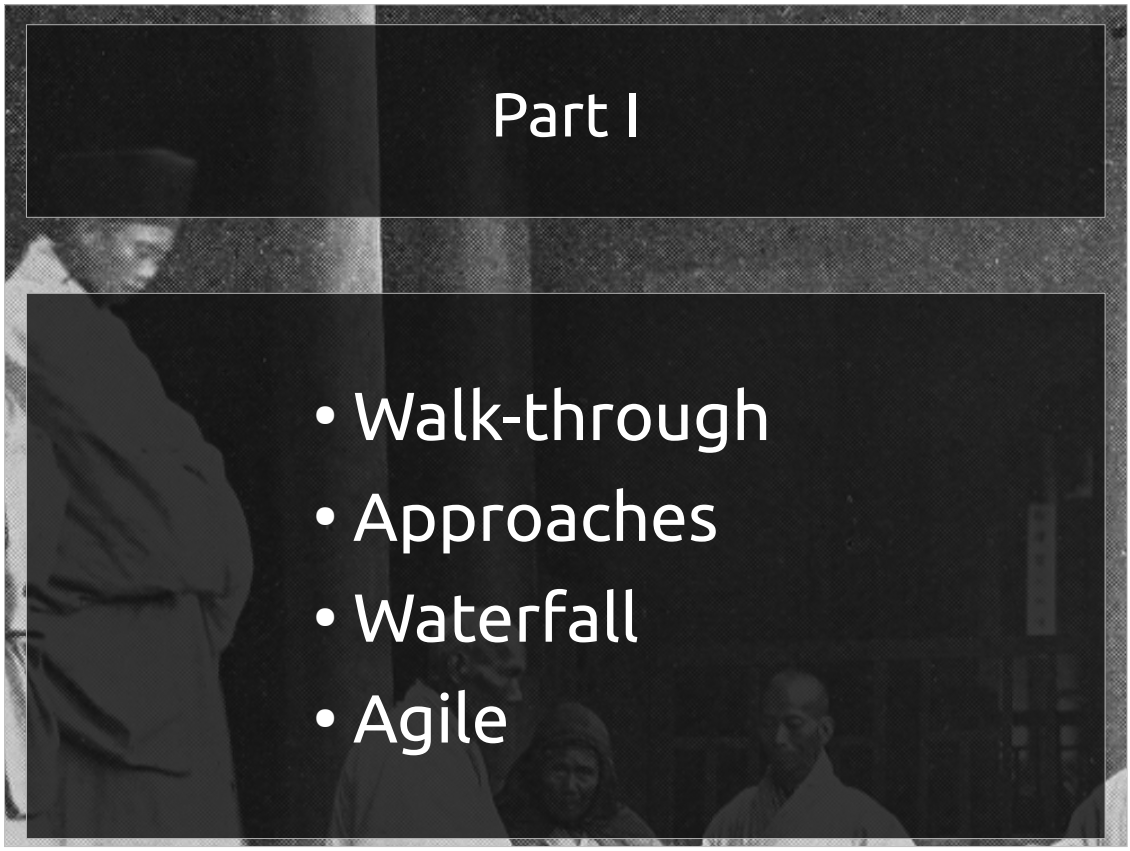
Hopefully you have operational controls like firewalls.
Hopefully you already use secure development practices like OWASP
Hopefully you already do decent automated testing, perhaps even security testing

Baking security in to the design rather than bolting it on..

... As you can tell from the title, this talk is about threat modelling...



We'll start off looking at traditional threat modelling...

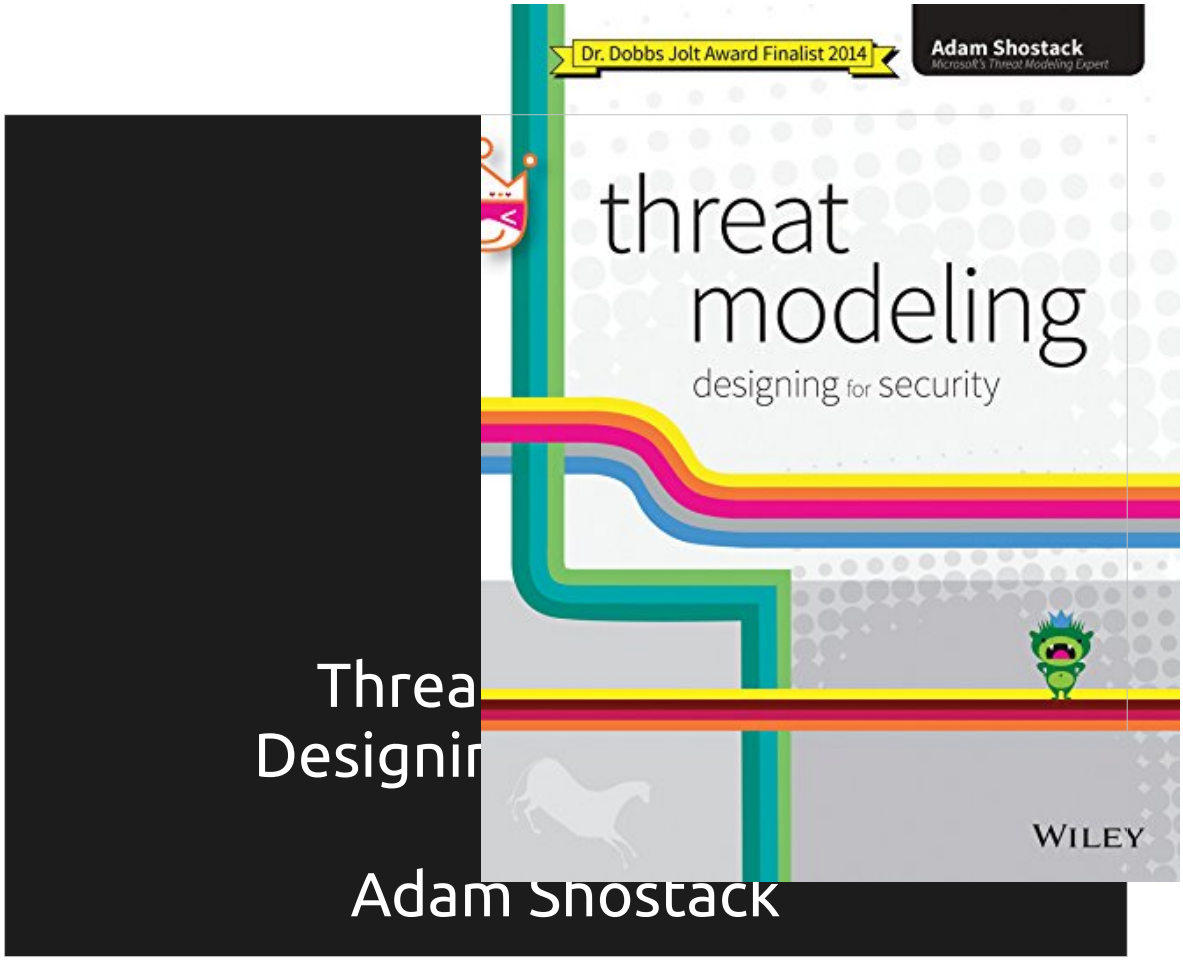

Part I
• Walk-through
• Approaches
• Waterfall
• Agile

Then we'll move on to....

Part II

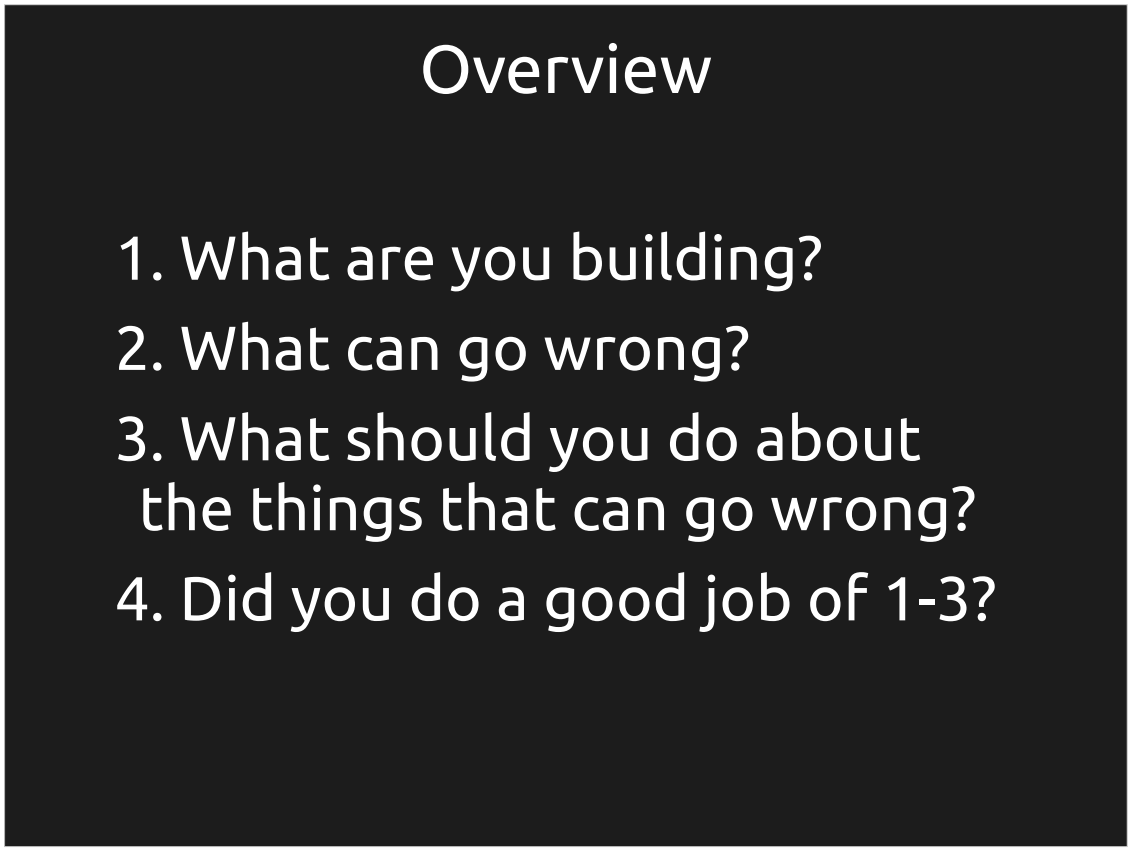
The experiment*

* actual science not included



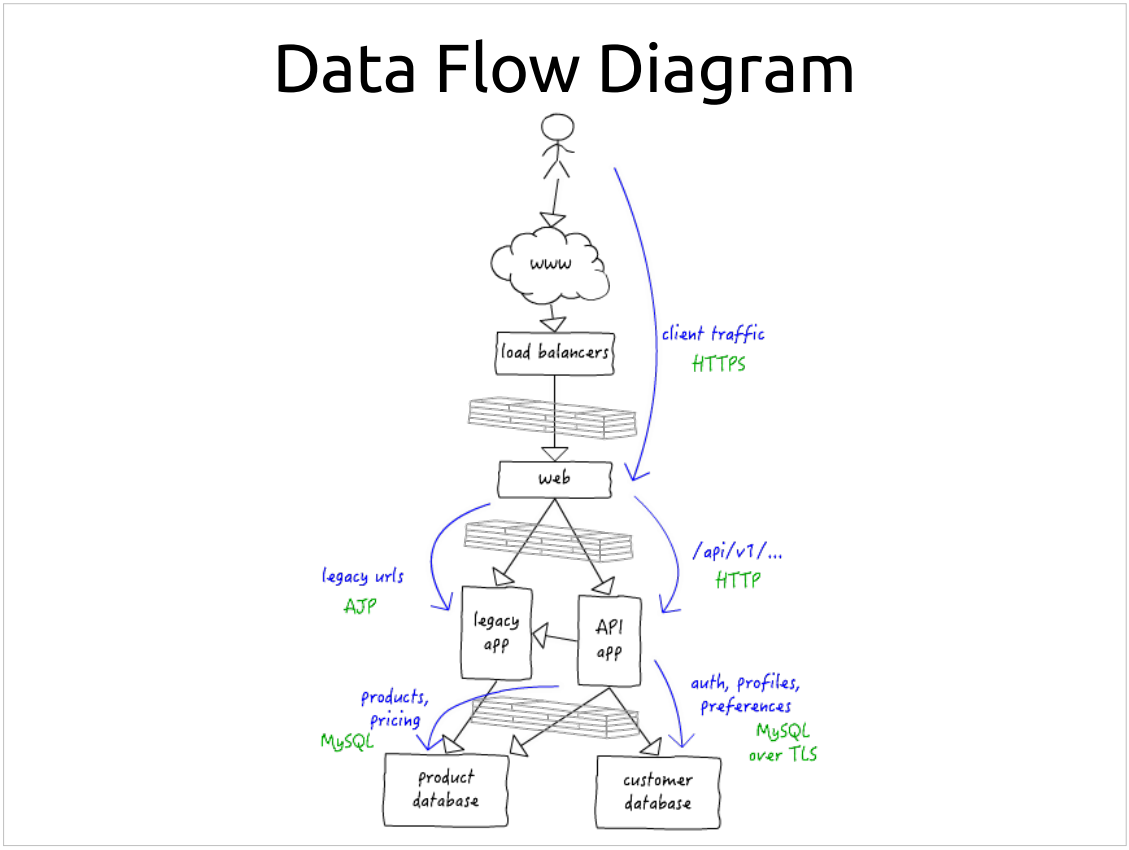
For a fully comprehensive introduction to threat modelling, go read Adam Shostack's book.

The first part of this talk will give a summary of threat modelling, but I urge everyone to read the book.



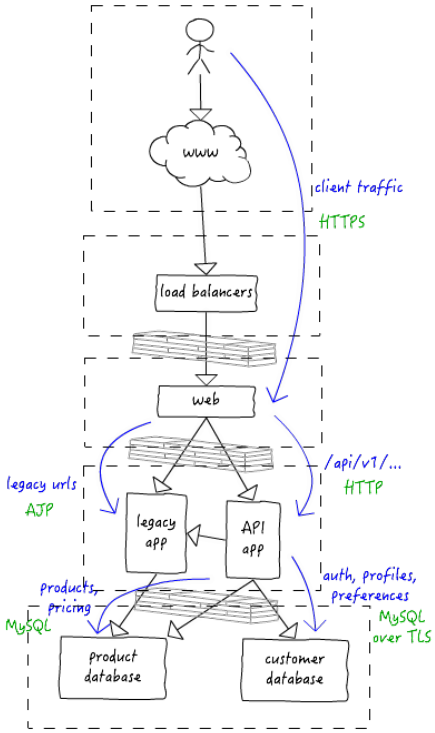
Threat modelling involves asking a lot of important questions.

Now, back to our story.



For the follow-up meeting you decide to do some research into formal threat modelling. Having started on Adam Shostack's book, you use the sketch from the post-mortem to help you answer the question “What are you building?”. You turn the sketch into a data-flow diagram. You needed some help from a senior developer and the networks team, but you now have a pretty good idea of what components there are and roughly how they connect. It's just a model, but it will do.

Trust boundaries



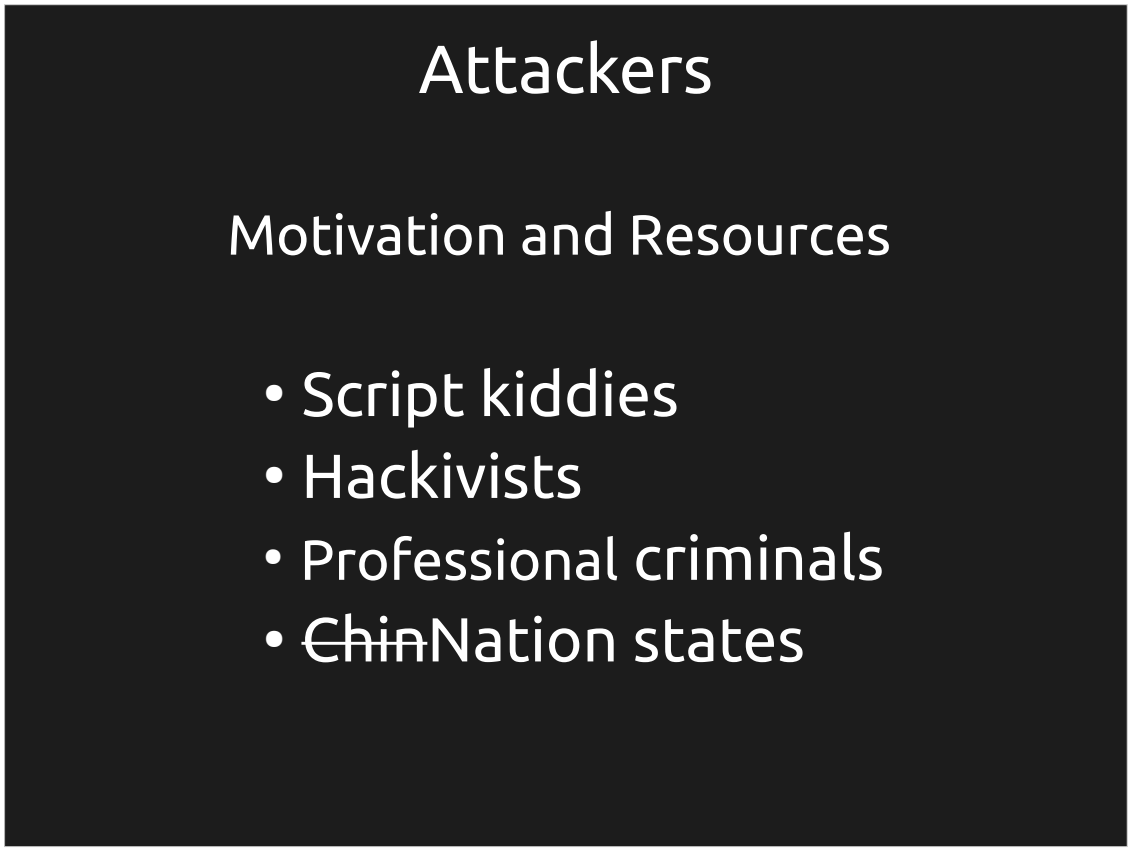
You add some trust boundaries to the data-flow diagram, to show where the different areas of responsibilty and control start and end.

You now have to decide how you're going to approach identifying the threats to answer the “What can go wrong” question.

Assets

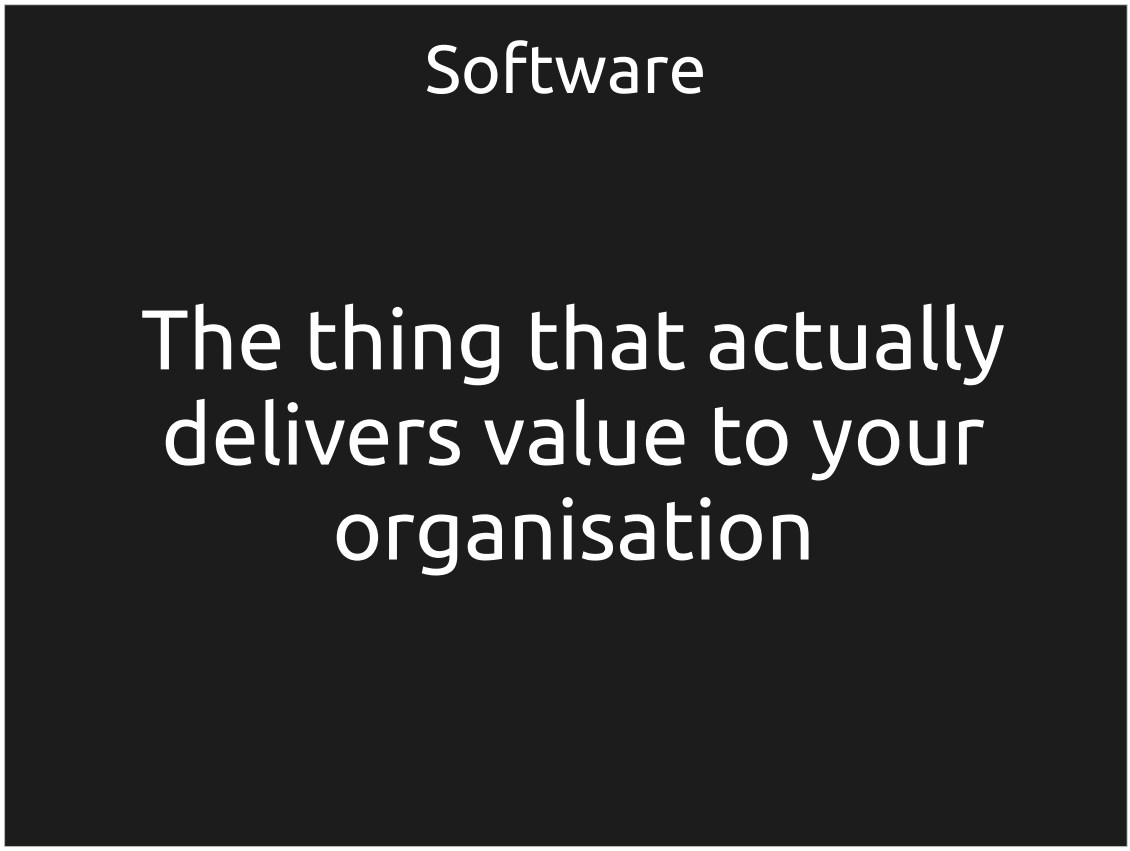
- Systems – access to data and pivoting
- Customer records (i.e. PII)
- Product data
- Credentials

Firstly you consider your assets, thinking about what assets are most critical and how they're exposed… disadvantages of this approach..



Then you start thinking abou the capability and motivations of your attackers. What sort of techniques do they use? What are their goals? How would you go about attacking the system?

… problems with this approach...



Finally, you think about what software components are involved. The data-flow diagram already maps out the high-level components, so you start think about how different threats apply to different software components.

Is that connection encrypted? Is there authentication in place? Does that component handle untrusted user input? Does it write to the file system?

To help you focus on the treats, you get a few of your colleagues together one friday afternoon and have a go at playing ...

Elevation of Privilege

Summary of EoP goes here....

STRIDE

- Spoofing identity
- Tampering with data
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

In the follow up meeting you document STRIDE per component. For each of the identified threats you need to decide on what action to take. Your options are ...

STRIDE EXAMPLES

Load balancers / Web

- Bug could lead to a DoS
- 0 day could result in elevation of privilege

Legacy app / API

- Tampering of URL could result in a DoS or elevation of privilege
- Information disclosure of DB creds

Product / customer database

- Elevation of privilege in customer DB would probably result in information disclosure of customer data (PII, credit cards etc)
- Tampering of product could lead to repudiation attack

Mitigate means introducing a fix or workaround . Adding authentication could mitigate a confideniatitly threat.

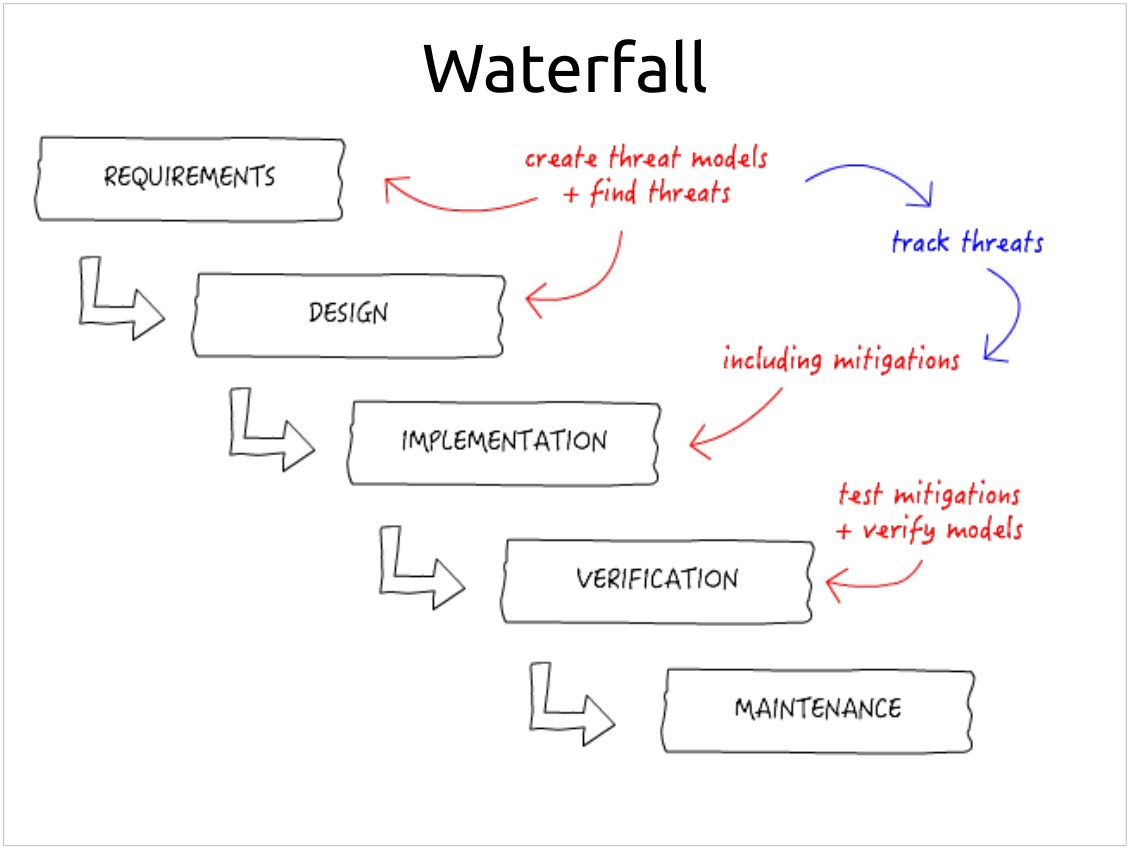
Eliminate would involve something like removing functionality. Remove the code and you remove the threat.

Transfer makes another party responsible for the risk. Perhaps through insurance, or a technical solution.

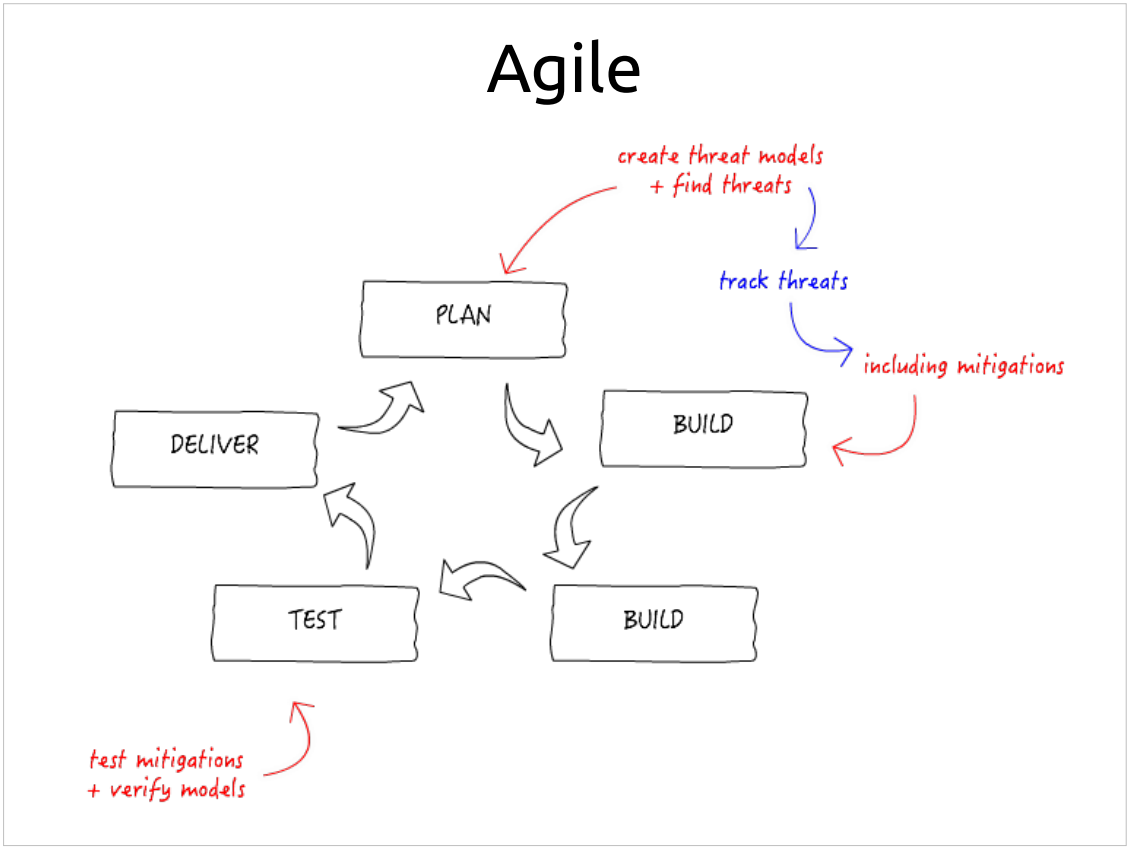
To accept a threat is to accept the risks and probably just hope for the best. This is the default for inaction, whether you realise it or not. Not threat modelling means to accept all the unidentified risks.

Measurement
Validation
Keep up to date

You use your existing ticketing system to track the threats and their mitigations ... plus other measurements...



So that's a quick overview of TM. The trick is of course to actually apply it within your organisation. If you have long release cycles you may be using the traditional “waterfall” methodology. TM can easily fit into the initial design and requirements phase with sessions that include the PM, architect, developers, ops, testers and security. Threats and mitigations can be tracked like any other work...



...

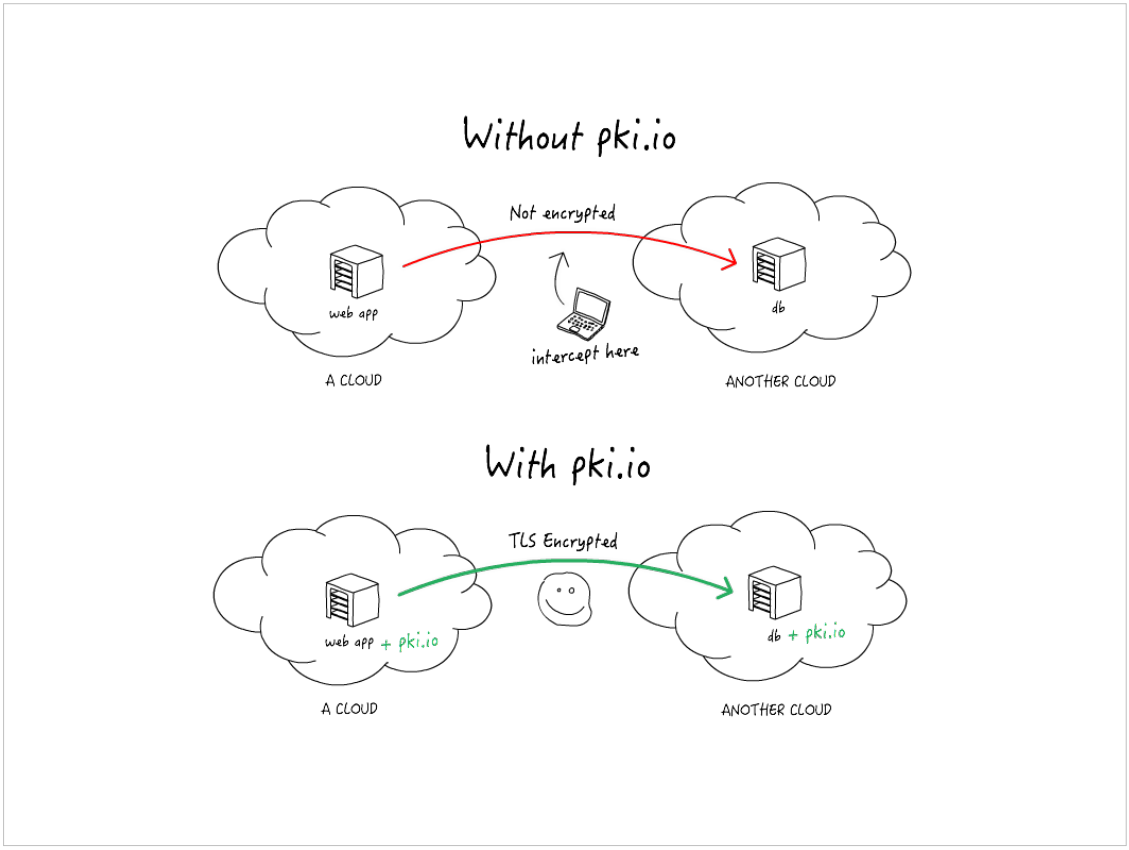
In summary, threat modelling is ...

What makes me qualified to talk about TM?

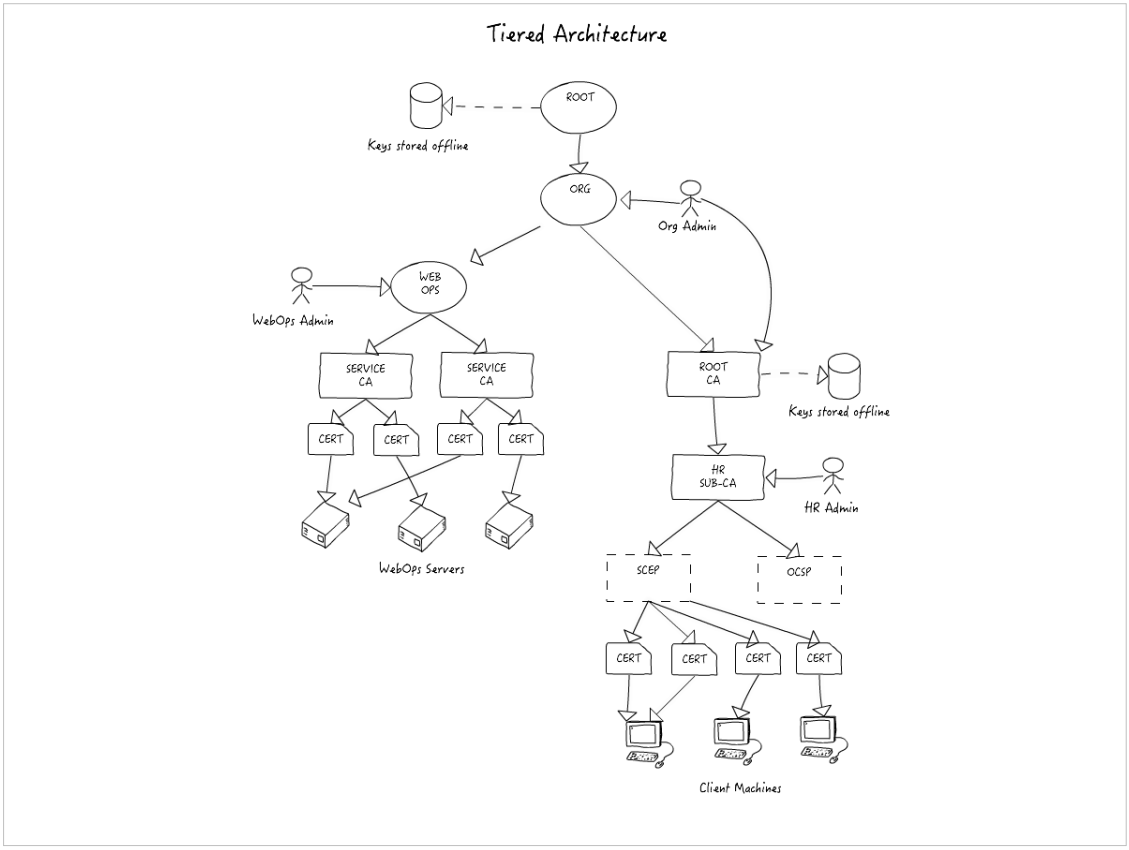
Nothing really.. I'm just a devopsy sysadmin-that-codes. I build and architect systems. I look after some security platforms. I write tools and Infrastructure as Code...

Exactly the type of person that should be involved in TM

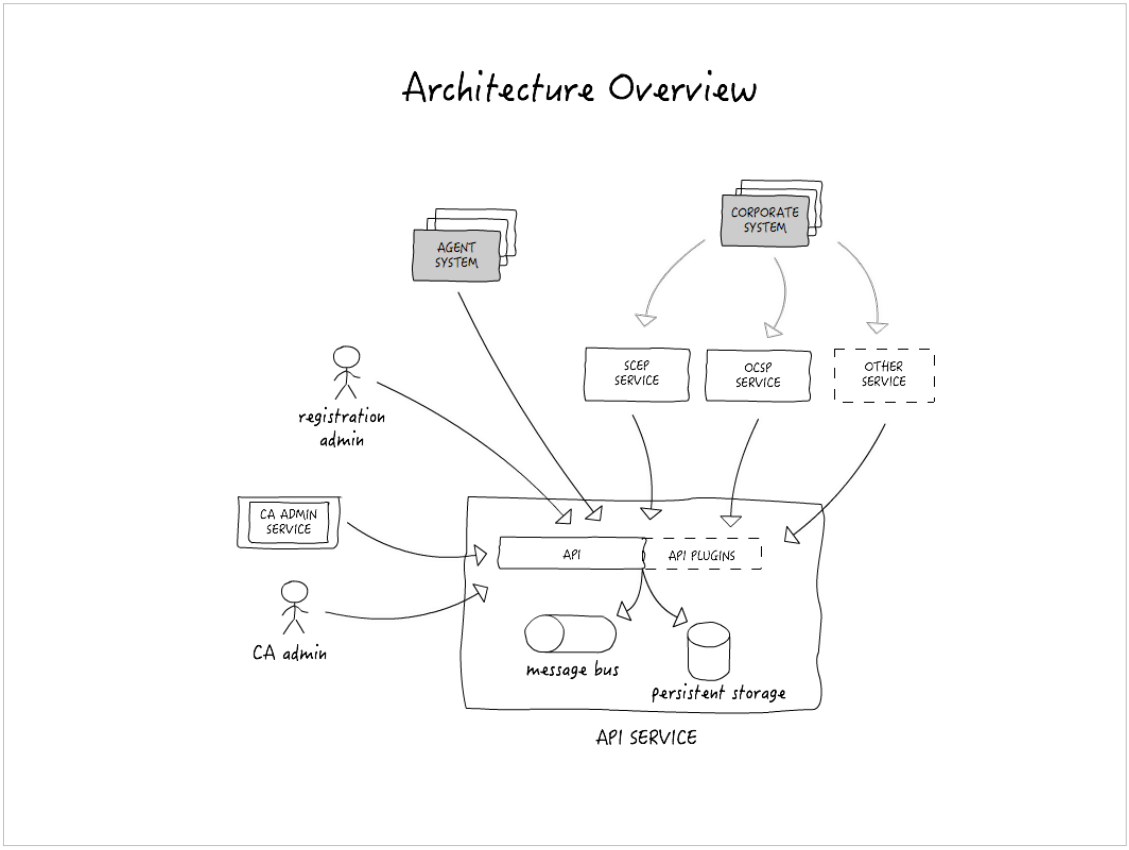
About a year ago I started an open source project called pki.io – got tired of dealing with internal CAs, wanted something that could grow...



So I started pki.io. The idea is to make running internal CAs easy, secure and devops friendly.



The first release allows admins to manage CAs and certs in an offline mode. Data is encrypted with admin keys and can be synchronised using git etc. The second release will allow admins to create CSRs on remote servers and easily sign them over SSH. The 3rd release will have a central API service and allows admins and an admin service to manage everything automatically and asynchronously.



Security tool so security is a primary concern..
so aims are

* clear and comprehensive security model
* secure coding practices
* security focused unit tests
* disclosure policy and security contact
* further security testing with bdd-security or gauntlt

... threat modelling?...

Pki.io is an open source project living on GitHub. No project manager, no security team.. everything revolves around code.

* Distributed developers across different timezones taking on different roles
* People working in the spare time – things need to be convinient or they won't happen
* Different devlopers from different backgrounds with different amounts of free time

Look at something like unit testing, tdd and bdd. Brings testing into the real of developers..

Let's look at some examples.

R-Spec

```
# in spec/calculator_spec.rb
RSpec.describe Calculator do
  describe '#add' do
    it 'returns the sum of its arguments' do
      expect(Calculator.new.add(1, 2)).to eq(3)
    end
  end
end
```

Looks like code

Cucumber

```
Feature: Refund item

  Scenario: Jeff returns a faulty microwave
    Given Jeff has bought a microwave for $100
    And he has a receipt
    When he returns the microwave
    Then Jeff should be refunded $10
```

More natural language that allows non-developers to participate. Even better, get the product team to write the requirements as a spec.

Bdd-security

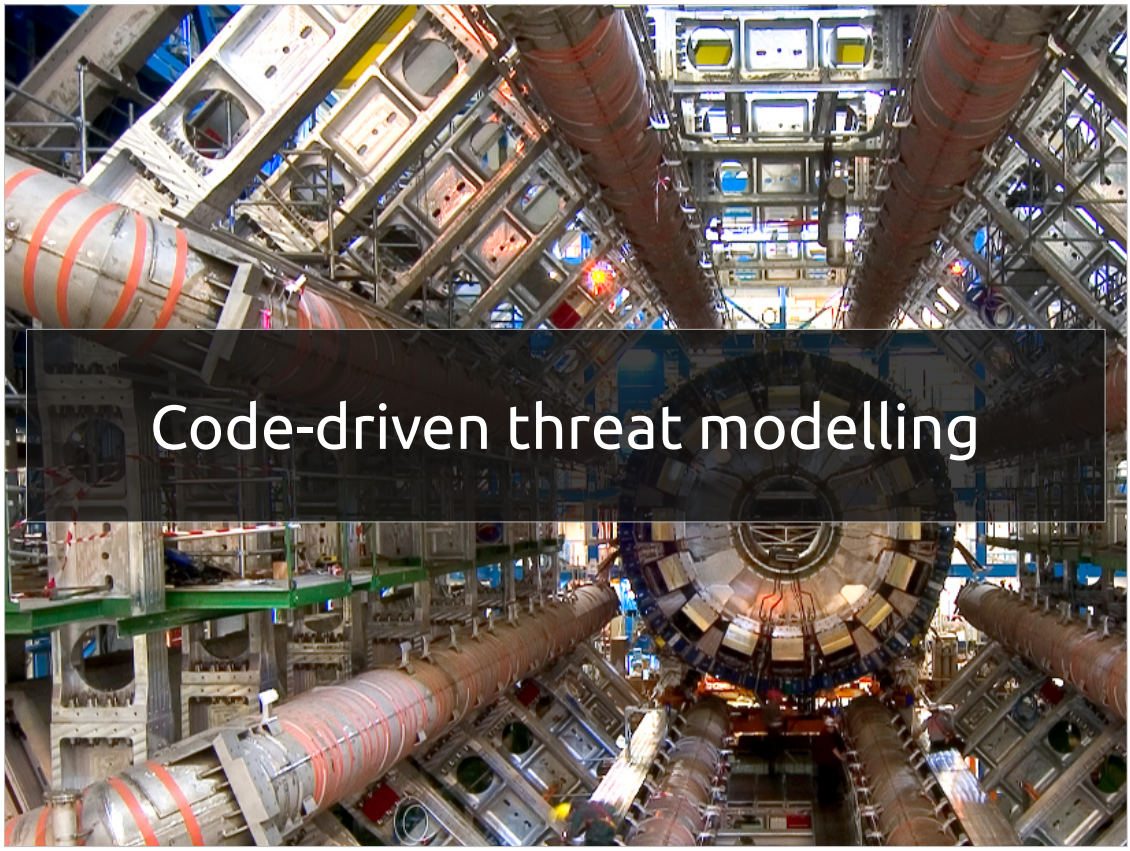
```
Scenario: Present the login form itself over an HTTPS connection
Meta: @id auth_login_form_over_ssl @cwe-295-auth @browser_only
Given a new browser instance
And the client/browser is configured to use an intercepting proxy
And the proxy logs are cleared
And the login page
And the HTTP request-response containing the login form
Then the protocol should be HTTPS
```

Behaviour-driven development is being applied to security.

Gauntlt

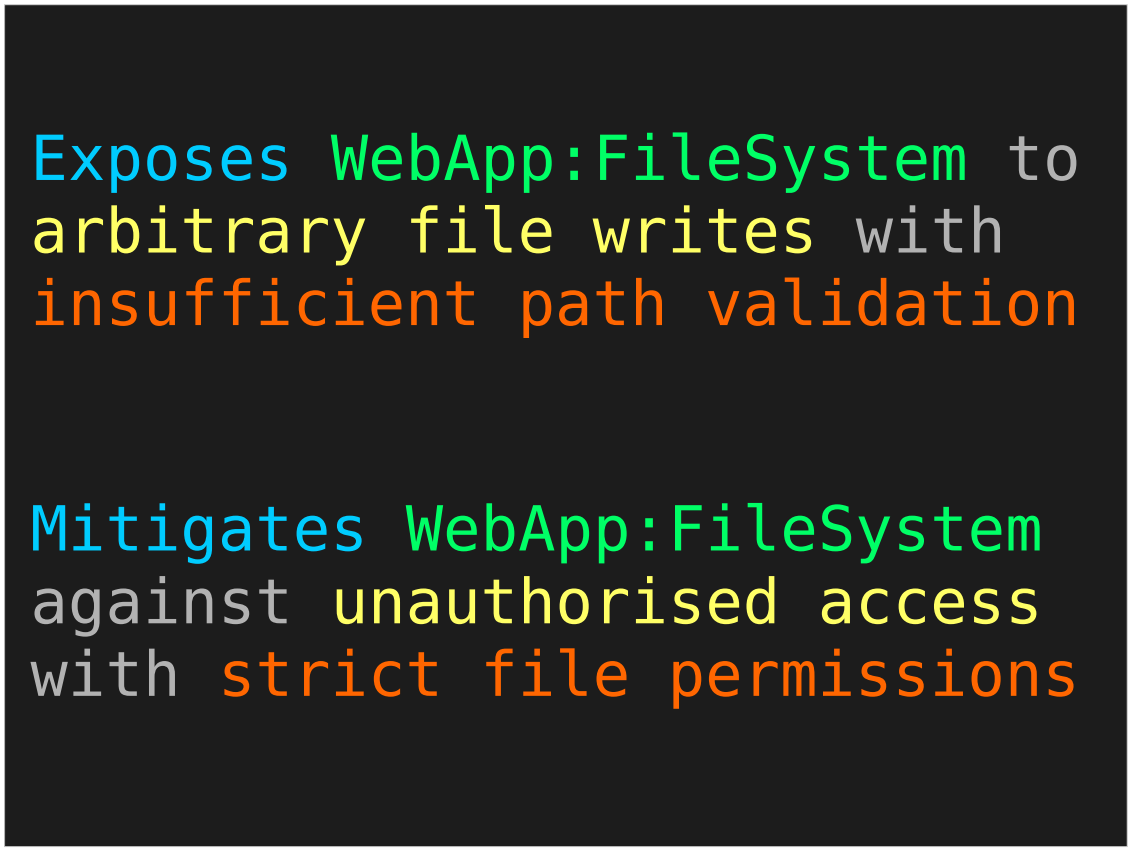
```
# nmap-simple.attack
Feature: simple nmap attack to check for open ports

  Background:
    Given "nmap" is installed
    And the following profile:
      | name     | value       |
      | hostname | example.com |

  Scenario: Check standard web ports
    When I launch an "nmap" attack with:
      """
      nmap -F <hostname>
      """
    Then the output should match /80.tcp\s+open/
    Then the output should not match:
      """
      25\/tcp\s+open
      """
```

… can we apply a similar approach to threat modelling?



Would this work?

Things we would need to be able to express. Other stuff might include things like tracking references.



These examples express a unhandled threat and a mitigated threat. The green states the trust boundary and component, separated by a colon. The threat itself is shown in yellow and orange shows either the cause or solution.

The specification could live in its own spec file or along side code as comments.

As this is relatively fixed, we can easily parse it with a regular expression, like ...

```
\s*(?:\/\/|\#)\s*Mitigates
(?<component>.+?) against (?
<threat>.+?) with (?
<mitigation>.+?)\s*(?:\((?
<ref>.*?)\))?\s*$
```

This is for a mitigation.

The first few characters handle the fact that we're embedding it in a comment. Blue matches our mitigation or exposure. The green component holds the zone and component combination. Then we match the threat and then mitigation. We've also allowed an optional reference.

```
var (
	addr = flag.Bool("addr", false, "find open address and print to final-port.txt")
)

type Page struct {
	Title string
	Body  []byte
}

// ThreatSpec SimpleV1 for (*main.Page).save
// Does page saving for WebApp:FileSystem
// Exposes WebApp:FileSystem to arbitrary file writes with insufficient path validation
// Mitigates WebApp:FileSystem against unauthorised access with strict file permissions
// Sends notification email from WebApp:App to User:Mail Client

func (p *Page) save() error {
	filename := p.Title + ".txt"
	return ioutil.WriteFile(filename, p.Body, 0600)
}

// ThreatSpec SimpleV1 for main.loadPage
// Does page loading for WebApp:FileSystem
// Exposes WebApp:FileSystem to arbitrary file reads with insufficient path validation

func loadPage(title string) (*Page, error) {
	filename := title + ".txt"
```

Applying it to code could look something like this.

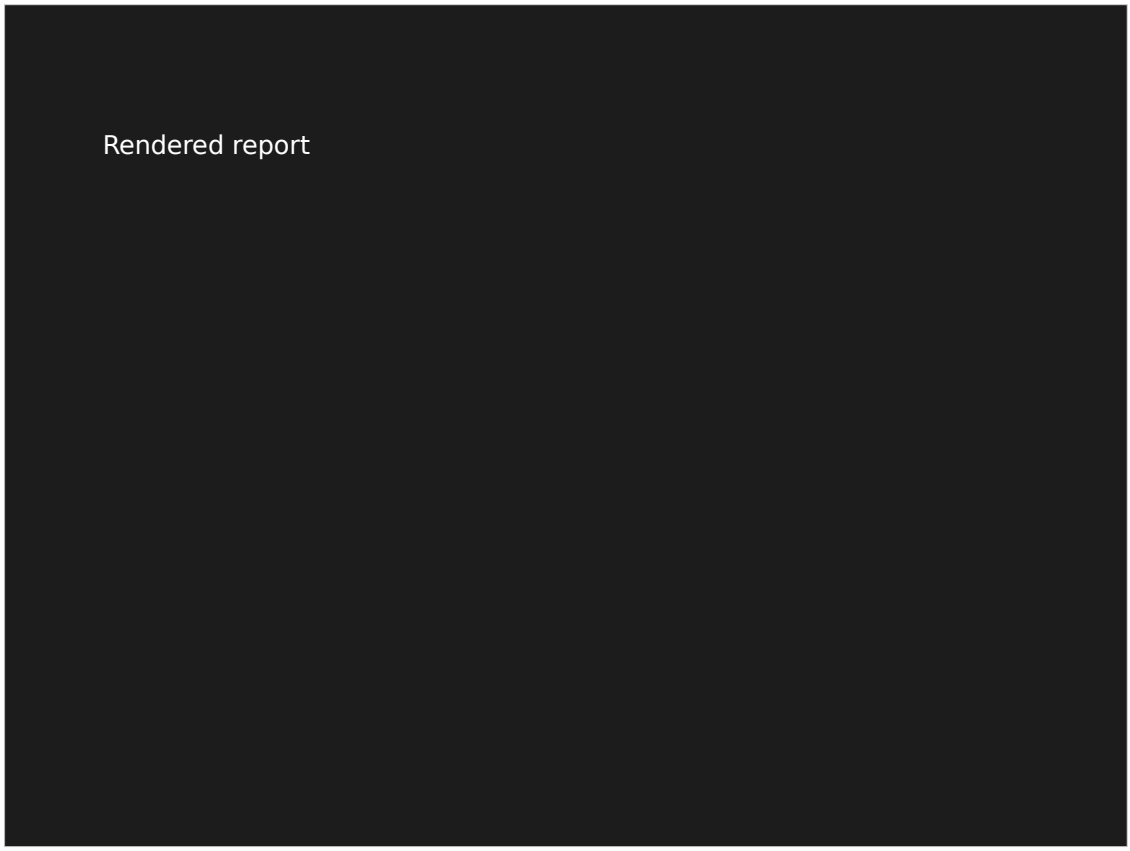
The threat specification is written above the function it's about. It would also be near any other function documentation. Reason for this approach:

* Keep relevant things together
* Easy overview of function behavior
* Threat modelling also adds to general documentation about the function.
* Developers can contribute in code as can the security team

```
# ThreatSpec Report for ...

# Analysis
* Functions found: 9
* Functions covered: 88.89% (8)
* Functions tested: 12.5% (1)

# Components
## WebApp FileSystem
### Threat: arbitrary file writes
* Exposure: insufficient path validation ((*main.Page).save in simple.go:31)

### Threat: arbitrary file reads
* Exposure: insufficient path validation (main.loadPage in simple.go:40)

## WebApp App
### Threat: XSS injection
* Exposure: insufficient input validation (main.editHandler in simple.go:65)

### Threat: content injection
* Exposure: insufficient input validation (main.saveHandler in simple.go:77)
```

The current ruby script used for pki.io generates a simple markdown report that would look a little bit like the above.

Some overall analysis can be done by also matching function definitions. We can then arrange the threats/mitigations by component and include as much detail as needed.

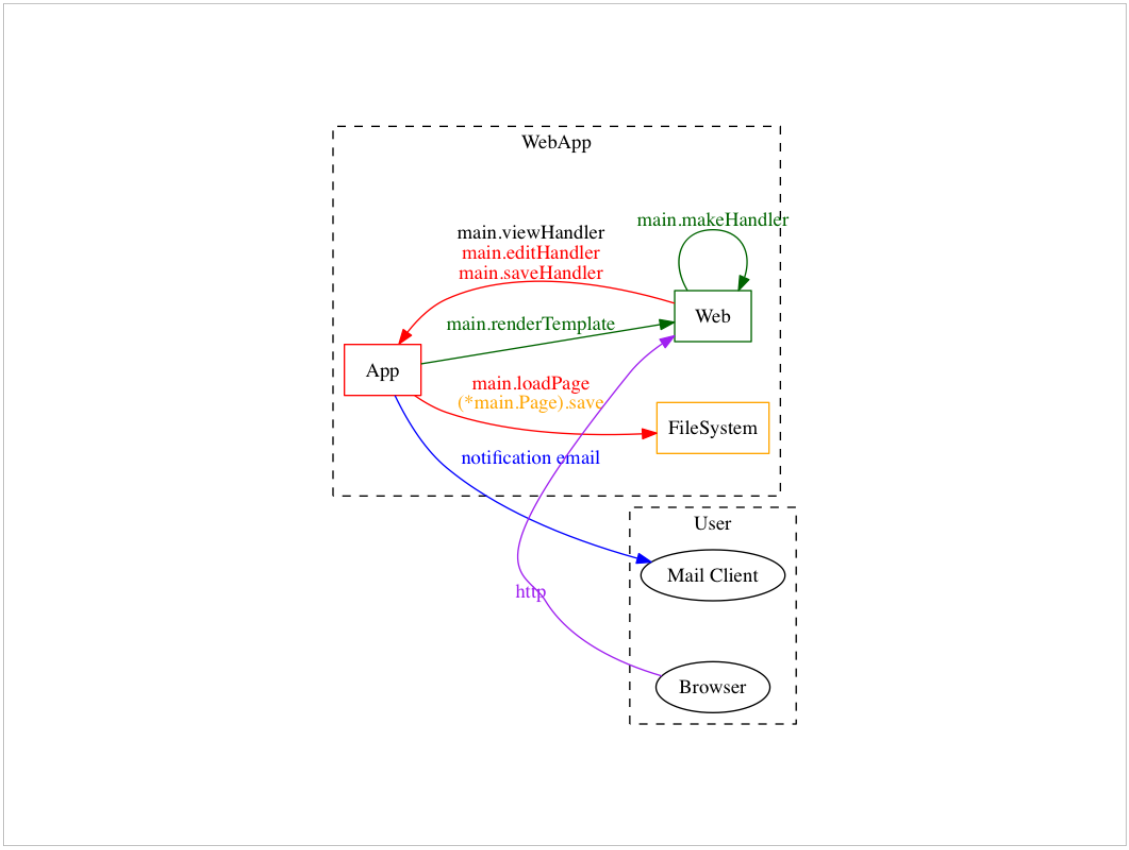


Because it is generating markdown, the report can easily be pasted into a wiki or SECURITY.md file.

This is of course especially useful for a GitHub-based project.

A decent tool would probably want to render PDFs or HTML directly. Maybe.

Pki.io is written in go, so we can use Go's callgraph command to generate a call graph of the code. By passing this in to the ruby script, we can see how the different functions covered by the threat spec call eachother, essentially generating...



A data-flow diagram. With trust zones. And some sort of threat / mitigation scoring.

In this simple example, red means the “called” function only has exposures, orange means a mix of mitigations and exposures, and green means it has no exposures and only mitigations. Black means it has neither.

The blue and purple lines are refences to external components that live outside of our code.

```
// ThreatSpec SimpleV1 for main.TestListenHandler
// Tests (*main.Page).ListenHandler() for TLS listener
func TestListenHandler(t *testing.T) {
  ...
```

It has probably occurred to you that this threat modelling stuff is, or should be, in some way related to the security testing we mentioned earlier. And that's true. Your threats should be tracked, and your mitigations should be tested. So if you mitigate eavesdropping with TLS, then you should be testing that your system implements strong TLS.

You could reflect this in your threat model by documenting the test code. Notice how we refer to the function that it's testing.

Workflow

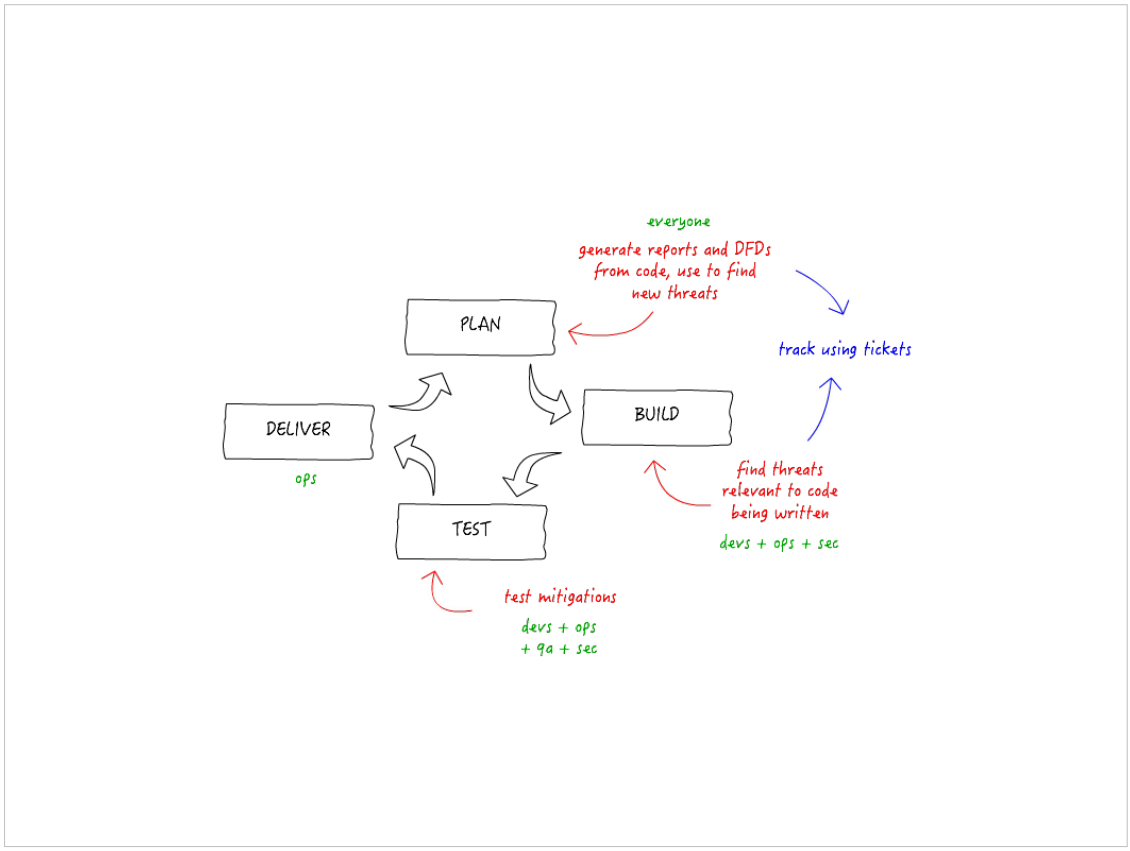
Devs write threatspec as they write new functions and tests

Review by security or senior devs

Periodic review of generated reports and DFDs

So, this idea is just an experiment. I'm not aware of anyone doing something like this in the real world. I imagine that applying this to a real-world workflow could look something like this.

Writing the threat spec makes the devs more aware of the security implications of what they're doing. If the security team review the code and update the specs, then the next time the dev reviews the code, they'll be able to learn from the updates ...



Not very happy about this image. Doesn't really show the dev-sec loop being tightened.

Problems?

- Starting point – rough DFD
- Complexity of generated DFD
- Missing functions etc
- Typos breaking stuff

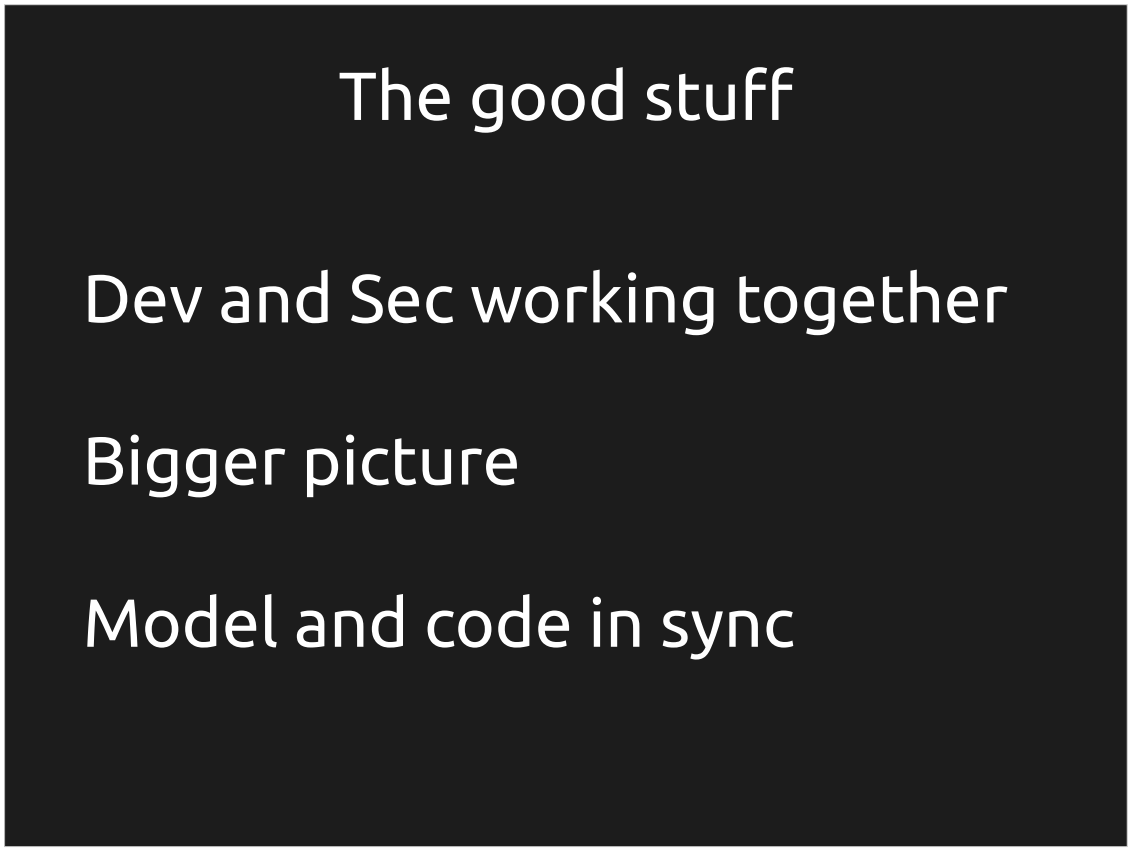
From trying to apply this to pki.io, I came across a few annoyances.

Firstly, you've got to have a rough idea of the components and zones before you start writing the specs. Otherwise you'll do a lot of find/replace.

Large projects will probably generate large DFDs. Summarizing the edges. Use starting point with degrees of separation.

A lot of work to retro-fit this to an existing project.

Typos in components end up with broken DFDs



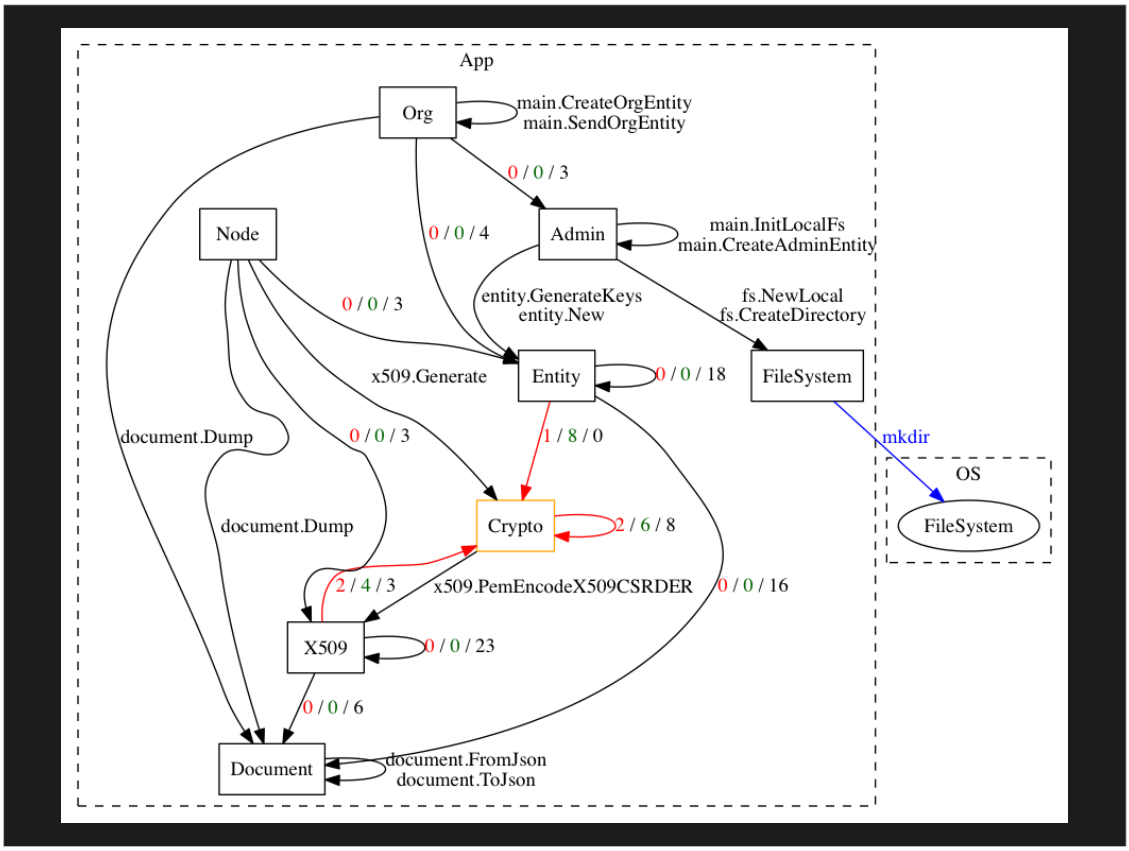
* devs learn from sec
* sec contributes directly – no more “No”
*

Pki.io example

- Admin and core
- Number of functions
- Several attempts
- Retro-fitting threatspec
- Started off with just

Example spec

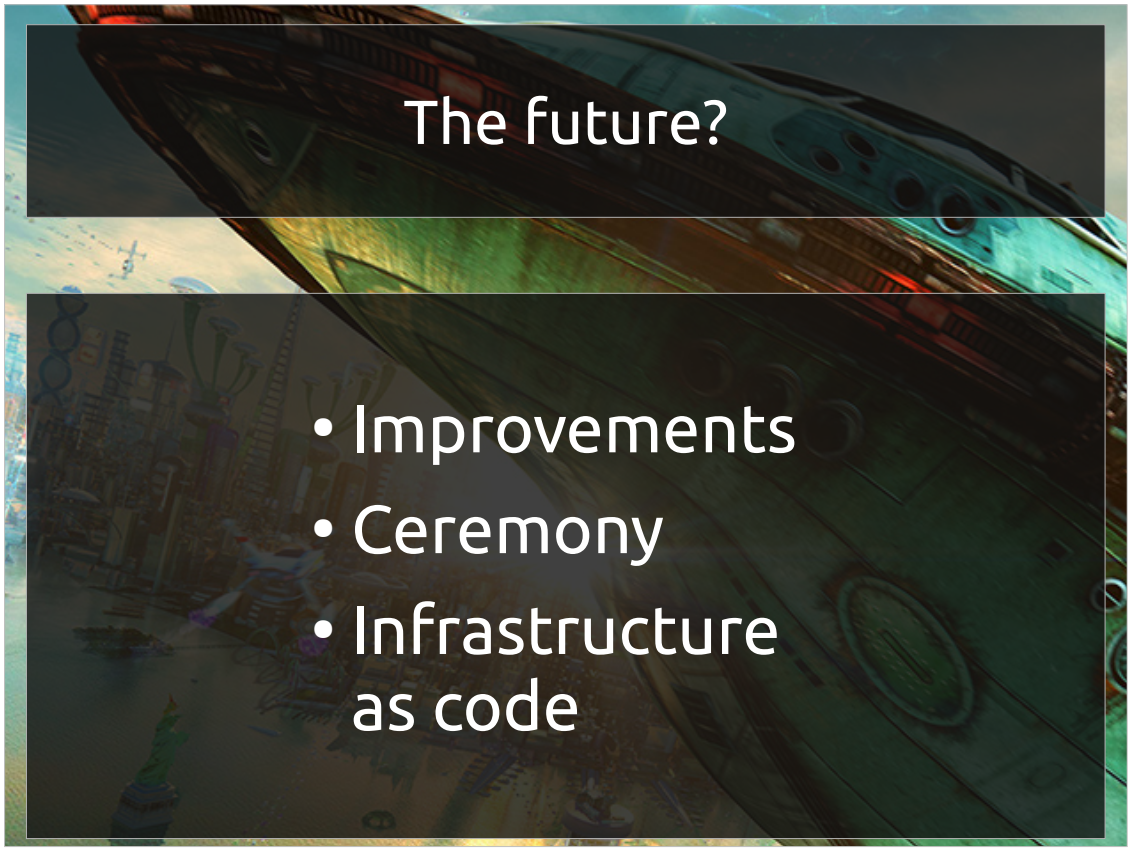
Generated report as html


App
Org
main.CreateOrgEntity
main.SendOrgEntity
0 / 0 / 3
Node
0 / 0 / 4
Admin
main.InitLocalFs
main.CreateAdminEntity
0 / 0 / 3
entity.GenerateKeys
entity.New
fs.NewLocal
fs.CreateDirectory
x509.Generate
Entity
0 / 0 / 18
FileSystem
document.Dump
0 / 0 / 3
1 / 8 / 0
mkdir
OS
FileSystem
document.Dump
Crypto
2 / 6 / 8
2 / 4 / 3
x509.PemEncodeX509CSRDER
0 / 0 / 16
X509
0 / 0 / 23
0 / 0 / 6
Document
document.FromJson
document.ToJson


In conclusion...

- Threat modelling is awesome
- You should probably be doing it
- Get people involved
- Find an approach that works for you
- Code-driven threat modelling may even work

The future?


The future?
• Improvements
• Ceremony
• Infrastructure as code

The stuff I've been talking about today is really just an idea. I've written a hacky ruby script for pki.io based on the specification ideas we looked.

There are probably better approaches to code-driven threat modelling. Spec should be language agnostic, tools could be too or specific to a langauge.

A German security company that partners with various universities, including TU Munich, has added a code-driven threat modelling as one of their research topics. Students can choose to do the research and get mentoring etc.

Slides, github repos, credits, links are available at threatspec.org

Thank you very much.