Lessons from DevOps: Taking DevOps practices into your AppSec Life

Matt Tesauro

OWASP AppSecEU 15
Amsterdam, The Netherlands

Who am I?

5 months with Pearson

Application Security Lead Engineer

Prior to Pearson

- Rackspace - Lead Engineer, Product Security
- AppSec consulting
  - VP Services, Praetorian
  - Consultant Trustwave's Spiderlabs
- TEA - Senior Security Engineer
- DIR - Penetration Tester
- Texas A&M University
  - Systems Analyst, Sys Admin, Developer, DBA
  - Lecturer in MIS department
- Viatel - Internet App Developer

Who am I?

Other professional experience

- **OWASP Live CD / OWASP WTE**
  - **Project lead 2008 to present, over 300K downloads**
  - **http://appseclive.org**
- **OWASP Foundation Board of Directors**
  - **International charity focused on improving software security**
- **Multiple speaking engagements internationally at AppSec, DHS, ISC2, SANS... conferences**
- **Application Security Training internationally**
- **B.S. Economics, M.S. in MIS**
  - **Strong believer in the value of cross-discipline study**

The Problem

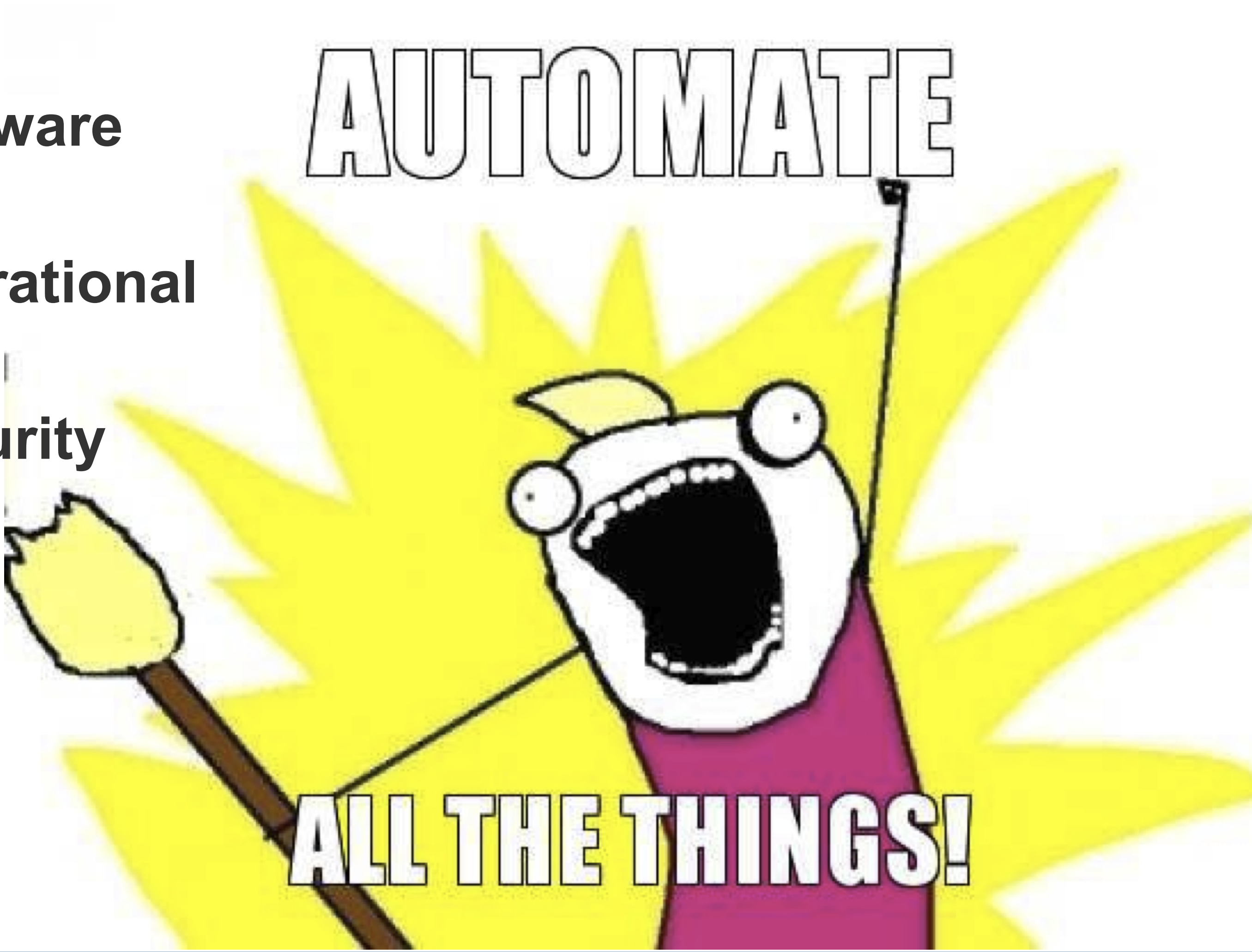
- Cycle time for software is getting shorter
- Continuous delivery is a goal
- Scanning windows are not viable
- First mover / first to market advantage

The Problem – or at least more problems

- Traditional software development left little time to test
- DevOps, Agile and Continuous Delivery squeeze those windows even more
- New languages and programming methods aren't making this better
  - Growth of interpreted languages with loose typing hurts static analysis efforts
  - Few automated tools to test APIs especially RESTful APIs
- Little time for any testing, manual testing is doomed

THE SOLUTION

- Automated software testing
- Automated operational infrastructure
- Automated security testing



"Don't get set into one form, adapt it and build your own, and let it grow, be like water".

A time to morn...

Traditional Software Dev & Ops

The old way...

Very early and prescriptive requirements and design

Long development cycles

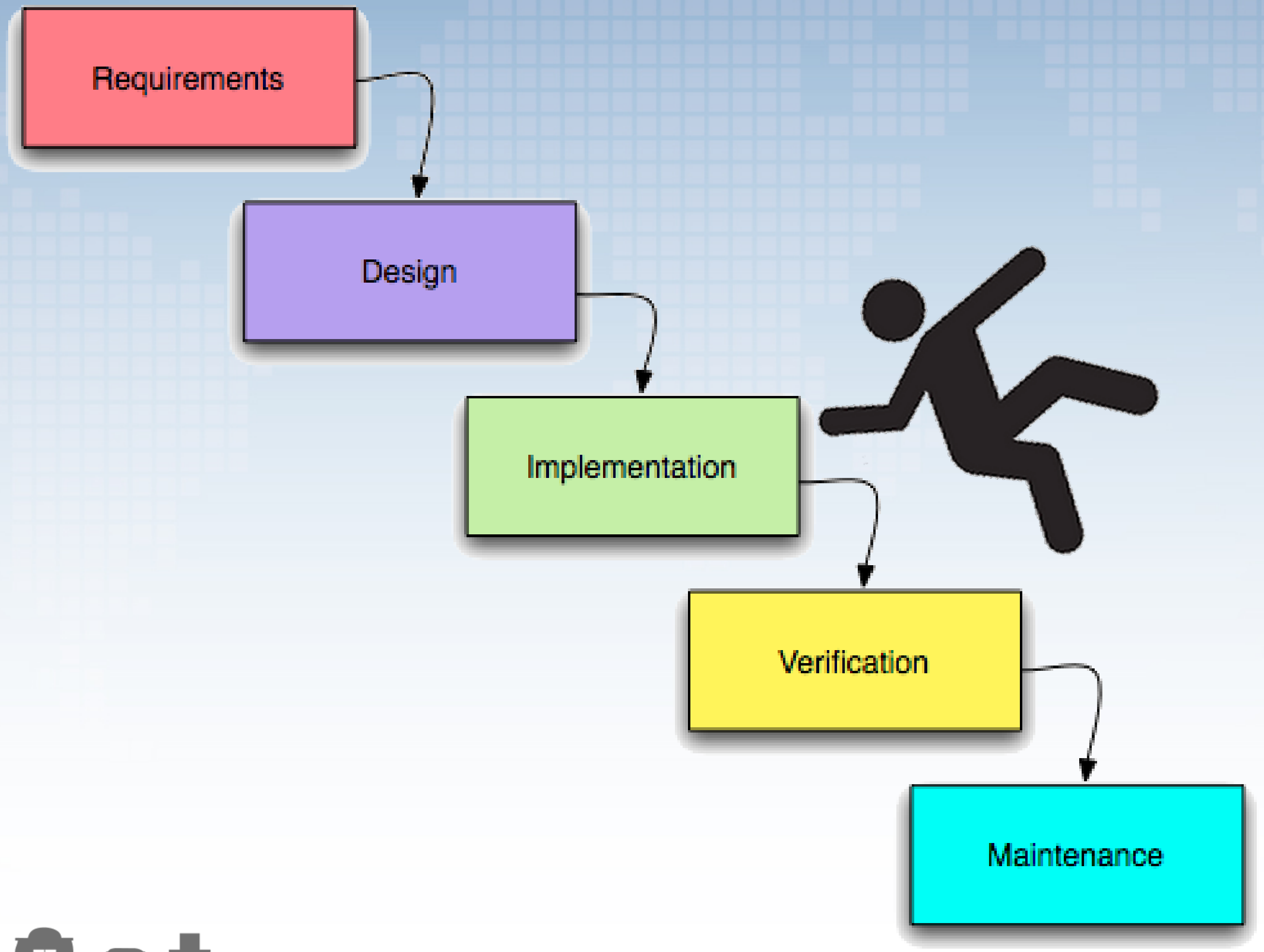
Waterfall Approach

Groups work in Silos - Dev, SysAdmin, QA, Security

Possible feedback from bug reports but little else

Throwing code over the wall

Waterfall Development



The DevOps Answer

Why DevOps came to be

Web/Cloud companies needed

- high availability
- fast introduction of new features

Easy for users to switch to a competing service + fist mover advantage

No media to ship with SaaS models

What's different about DevOps

Cultural change – not just new cool tech aka CI/CD, Docker...

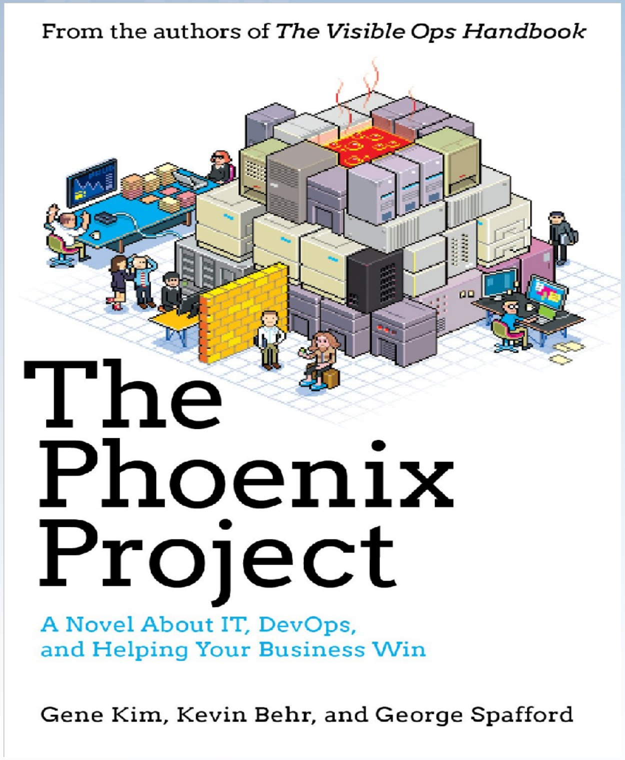
Focus on clear business objectives

Dev and SysAdmins share responsibility for uptime, deploys, downtime

Emphasize people and process, repeatability

Goal is better uptime and lower operational costs

"Notice that the stiffest tree is most easily cracked, while the bamboo or willow survives by bending with the wind."


From the authors of The Visible Ops Handbook
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford

The Phoenix Project
3 Ways of DevOps

Strategies for Improving Operations

The 3 Ways of DevOps

1 Workflow

2

3

OWASP AppSecEU 15
Amsterdam, The Netherlands

#1 - Workflow

Look at your purpose and those process which aid it

- Make sure the process is correct from beginning to the end

Then look at ways to speed up that process

- Value Stream – the name a the process which provides value to the business
- Working from left to right – think of a time line:

  business / development => customer / operations

- Flow [rate] – the speed work goes through the process

OWASP AppSecEU 15
Amsterdam, The Netherlands

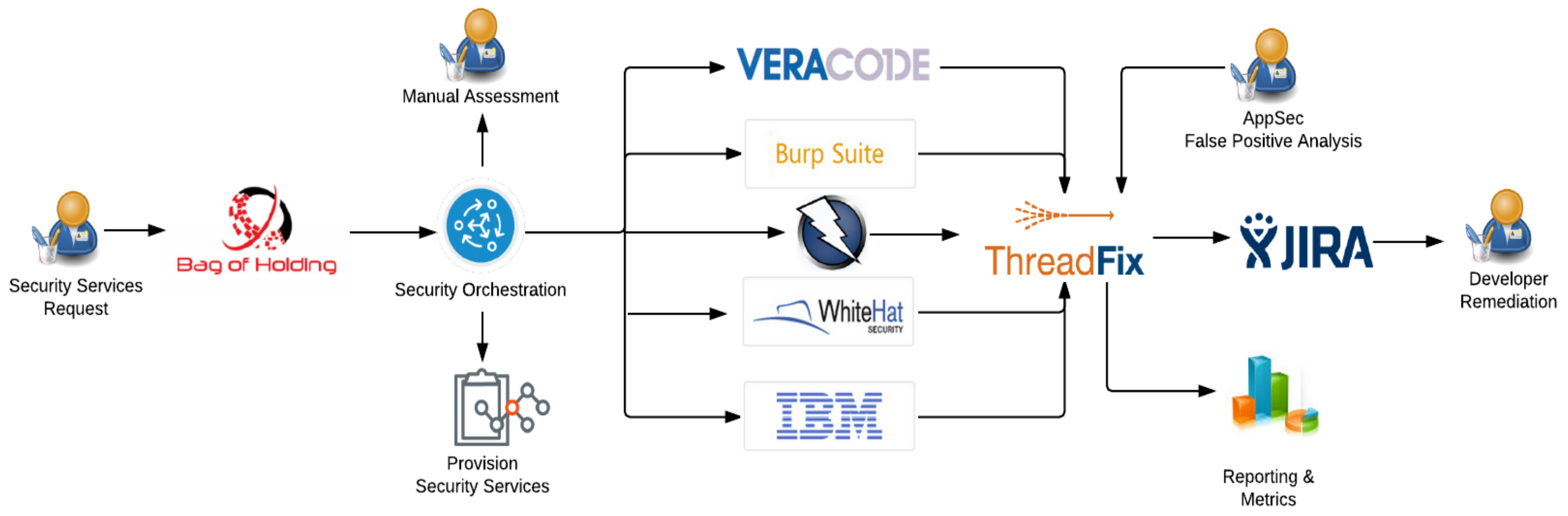
#1 - Workflow

An example workflow

Software release process

- Code written
- Code committed to a code repository
- Unit test the code
- Package the code for deployment
- Integration testing
- Deploy code to production

The AppSec Pipeline



Key Features of AppSec Pipelines

- Designed for iterative improvement
- Provides a reusable path for AppSec activities to follow
- Provides a consistent process for both the team and our constituency
    - One way flow with well-defined states
- Relies heavily on automation
- Has the ability to grow in function organically over time
- Gracefully interconnects with the development process

Spending time optimizing anything other than the critical resource is an illusion.

Key Goals of AppSec Pipelines

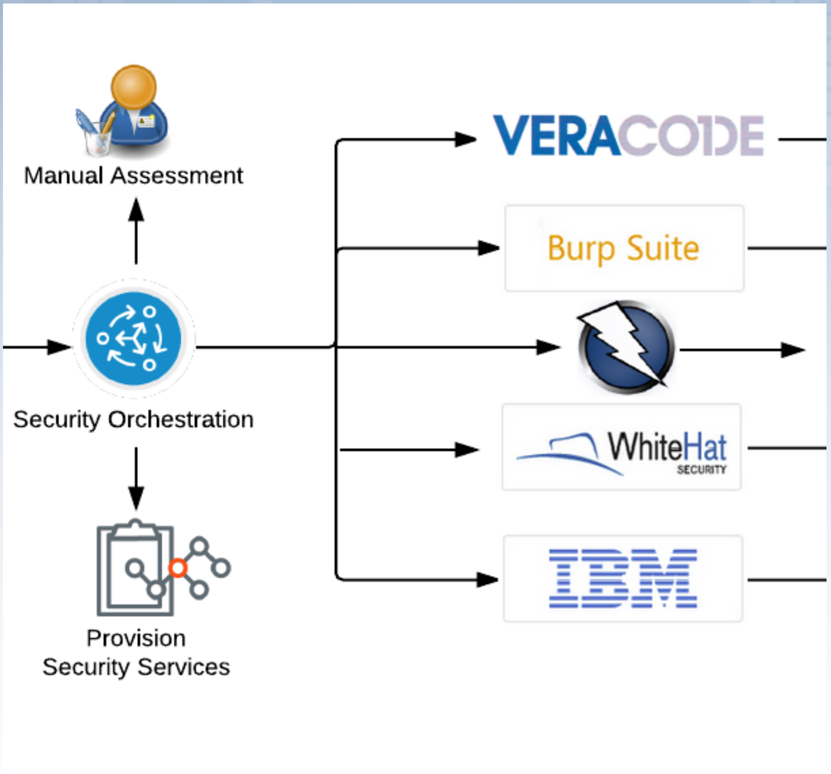
- Optimize the critical resource – *App Sec personnel*
- Automate all the things that don't require a human brain
- Drive up consistency
- Increase tracking of work status
- Increase flow through the system
- Increase visibility and metrics
- Reduce any dev team friction with application security

Security Services Request
Bag of Holding

Pipeline - Intake

- "First Impression"
- Major categories of Intake
    - Existing App
    - New App
    - Previously tested App
    - App to re-test findings
- Key Concepts
    - Ask for data about Apps only once
    - Have data reviewed when an App returns
    - Adapt data collected based on broad categories of Apps

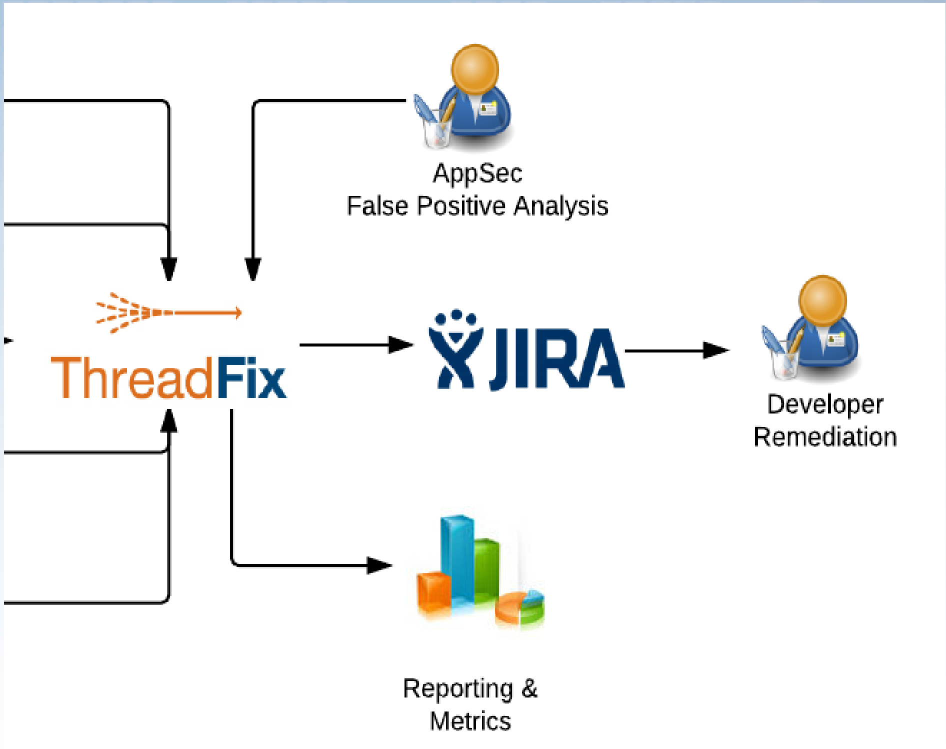
Pipeline – the Middle



- Inbound request triage
- Ala Carte App Sec
    - Dynamic Testing
    - Static Testing
    - Re-Testing mitigated findings
    - Mix and match based on risk
- Key Concepts
    - Activities can be run in parallel
    - Automation on setup, configuration, data export
    - Focus people on customization rather than setup

Pipeline – the End

- Source of truth for all AppSec activities
- ThreadFix is used to
  - Dedup / Consolidate findings
  - Normalize scanner data
  - Generate Metrics
  - Push issues to bug trackers
- Report and metrics automation
  - REST + tfclient
- Source of many touch points with external teams

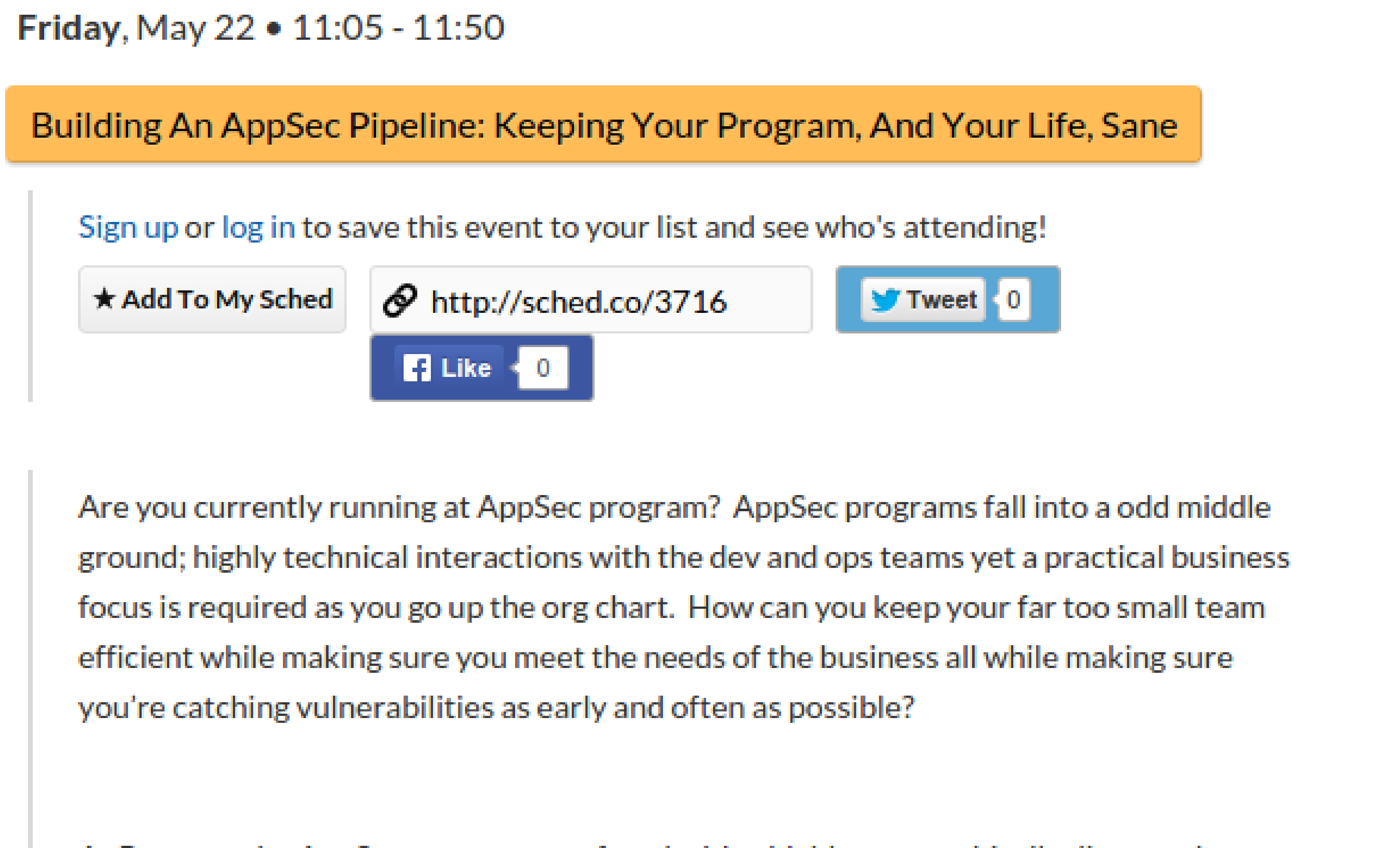


OWASP AppSecEU 15
Amsterdam, The Netherlands

Why we like AppSec Pipelines

- Allow us to have visibility into WIP
- Better understand/track/optimize flow of engagements
  - Average static test takes ...
- Great increase in consistency
- Easier re-allocation of engagements between staff
- Each step has a well defined interface
- Knowing who has what allows for more informed "cost of switching" conversations
- Flexible enough for a range of skills and app maturity

If you want to hear more...



Friday, May 22 • 11:05 - 11:50

Building An AppSec Pipeline: Keeping Your Program, And Your Life, Sane

Sign up or log in to save this event to your list and see who's attending!

Are you currently running at AppSec program? AppSec programs fall into a odd middle ground; highly technical interactions with the dev and ops teams yet a practical business focus is required as you go up the org chart. How can you keep your far too small team efficient while making sure you meet the needs of the business all while making sure you're catching vulnerabilities as early and often as possible?

#1 - Workflow

Each Step Repeatable

Making things repeatable

Remove all haphazard and ad hoc work from the process

Repeat until stable, I like doing the first couple times manually
 with a 'run book'

Scripting languages are your friends

Config Mgmt – Puppet, Chef, Salt, Ansible, Jenkins, CFEngine, …

Creating deployable artifacts from a branch/release aka .rpm / .deb / .msi

Make sure what you do can be done on 1 server or 10,000 servers

#1 - Workflow

Each Step Repeatable

Making things repeatable in AppSec

- Make tests easily repeatable
  - You will be re-testing after dev fixes so repeatable tests help retesting
  - You can hand them to devs to test as they write mitigation
- Make tests easy to understand
  - You will likely be handing work off between App Sec staff or to devs
- Make tests abstract and combine-able
  - Ala carte tests for mixing and matching
  - Think about the Unix pipe | and its power

"I fear not the man who has practiced ten thousand kicks once,
but I fear the man who has practiced one kick ten thousand times."

#1 - Workflow

Never Pass on Defects

Work left to right but don't pass on failures

Test early and often

Increase the rigor of testing as you work left to right

When a failure occurs, end that flow and start a new one after corrections

The further right you are, the more expensive failure is

For AppSec, Defects == False Positives

If you can automate code review, you still must triage
  1 false positive == 100 valid bugs
If results aren't actionable, you've failed
Best security ROI is findings early in the dev lifecycle

#1 - Workflow

Local optimizations with a global view

Your fix cannot be my new problem

Ensure no single-step optimizations degrade overall performance
Spending time optimizing anything other than the critical resource is an illusion.
Find the bottle neck in your workflow and start there
 - Upstream changes will just back things up
 - Downstream changes won't manifest since input is limited
Each new optimization creates a new bottleneck – iterate on this

Increase the flow of work

Now go faster

Make sure you have a well-defined, repeatable process first

Look for manual steps that can be automated

Look for duplicate work that can be removed/eliminated

Measuring/tracking time taken at each step is crucial

Where does the flow ebb?

The 3 Ways of DevOps

1. Workflow
2. Improve Feedback
3.

#2 – Improve Feedback

Open yourself to upstream and downstream information

Feedback loops occur when information is gathered from

- upstream (business / development)
- downstream (customer / operations)

Make visible problems, concerns, potential improvements

– share this publicly within your company

Learn as you move left to right so improvements aren't lost

Requests are opportunities to better fulfill the needs of the business

There is rarely enough feedback, capture and look for more

Feedback collected can be used to optimally improve the system

#2 – Improve Feedback

Understand and respond to your customers

Customers are also inside your business

Customer is more then the 'consumer' at the end of the process

- Each step is the customer of the previous step
- Understand what the next steps need from you to succeed

Remember, feedback isn't guaranteed - encourage it by responding

Make feedback & responding quick, easy and readily available

Embed knowledge when needed

Go all in

Keep specialized knowledge out of people's heads and into the system

- Check it into source control – automatically versioned.

Moving left to right, keep needed info in the stage that requires it

The 3 Ways of DevOps

1 Workflow

2 Improve Feedback

3 Continual Experimentation and Learning

#3 – Continual Experimentation & Learning

Create a culture of innovation and experimentation

The fundamentals are now solid, what can your new knowledge buy you?

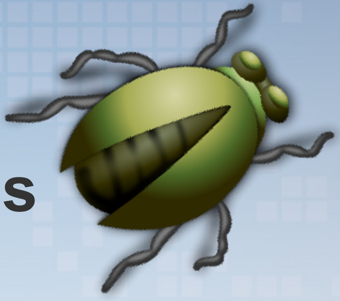
The business culture must allow for and embrace innovation & experimentation

Two essential things must be understood by the business and all involved

- We can learn from the failed experiments and risks we take
- Mastery comes with repetition and practice

and you won't be a master the first N times you practice

Findings directly to bug trackers



- PDFs are great, bugs are better
  - Work with developer teams to submit bugs
  - Security category needs to exist
  - Bonus points if the bug tracker has an API
- Security issues are now part of the normal work flow
  - Beware of death by backlog - do security sprints
  - Learn how the team treats issues
- ThreadFix is nice for metrics and pumping issues into issue trackers - http://code.google.com/p/threadfix/

For the reticent: nag, nag, nag

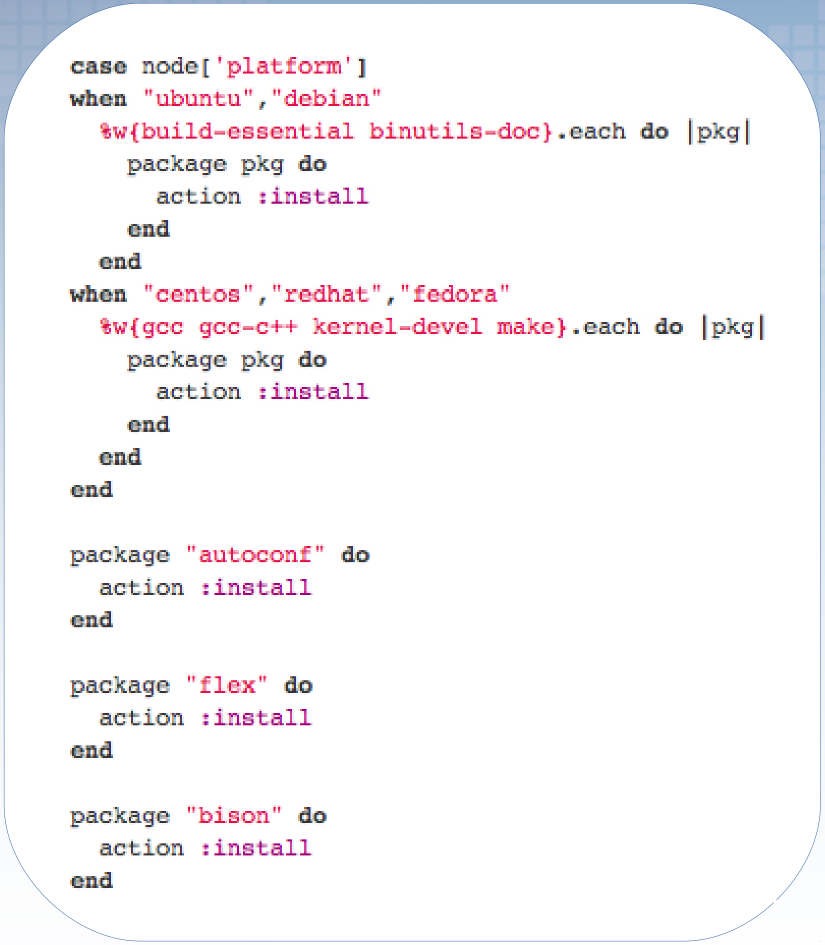
- Attach a SLA to each severity level for findings
  - Remediation plan vs Fixed
  - "Age" all findings against these SLAs
  - Politely warn when SLA dates are close

- Walk up the Org chart as things get older
- Bonus points for dashboards and bug tracker APIs
- Get management sold first

Automating Infrastructure

- Declarative configuration language
- Plain-text configuration in source control
- Fully programmatic, no manual interactions

Cookbooks, Stacks, Playbooks, ...

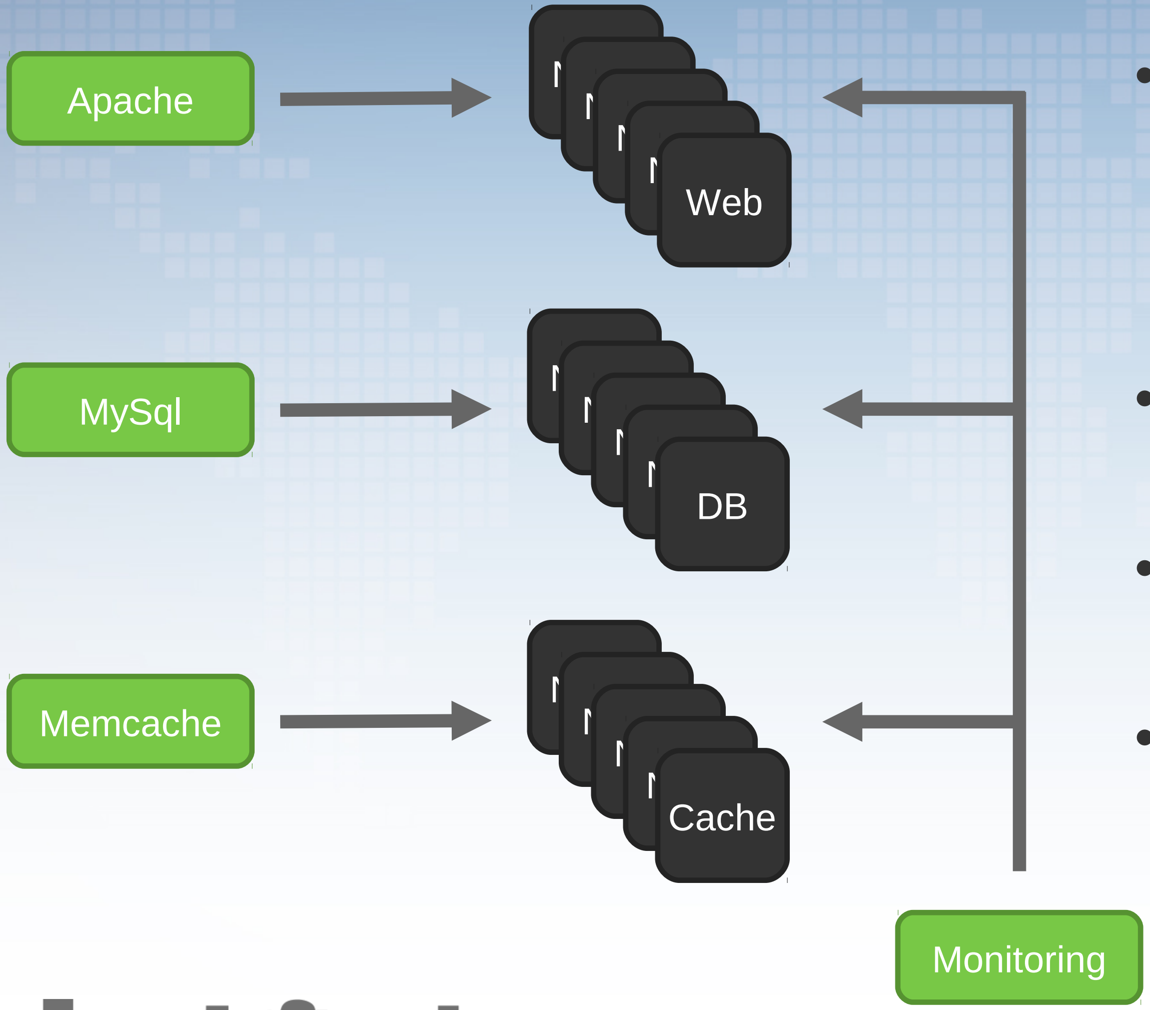


```
case node['platform']
when "ubuntu","debian"
  %w{build-essential binutils-doc}.each do |pkg|
    package pkg do
      action :install
    end
  end
when "centos","redhat","fedora"
  %w{gcc gcc-c++ kernel-devel make}.each do |pkg|
    package pkg do
      action :install
    end
  end
end

package "autoconf" do
  action :install
end

package "flex" do
  action :install
end

package "bison" do
  action :install
end
```

- Most have methods to bundle / share automation routines
- You will have to write your own / customize
- Good place to spend security cycles
  - Merge patches upstream for extra good karma points.

OWASP AppSecEU 15
Amsterdam, The Netherlands

Grouping & Tagging



- Tagging your servers applies the required set of automation
- A base set of for all servers
- Each server can have multiple tags
- Map tags to security requirements

Works for Clouds Too!

Inspector – you need one

- For each group and/or tag
    - Review the recipe, do a PR aka Pull Request
    - Hook provisioning for post deploy review
- Focus on checking for code compliance
    - Not perfection, bare minimums
- Can include multiple facets
    - Security, Scalability, Compliance
- Vuln scanners – manual or auto
- Jenkins Job + Lynis (open source)

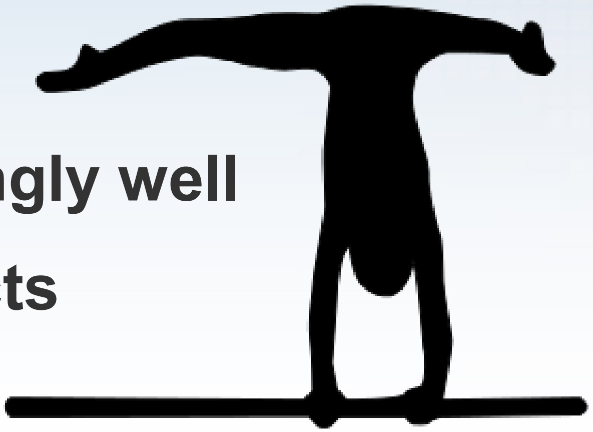
Agent – one mole to rule them all

- Add an agent to the standard deploy
    - Read-only helps sell to SysAdmin
    - Looks at the state of the system
    - Reports the state to the “mothership”
- Add a dashboard to visualize state of infrastructure
    - Change policy, servers go red
    - Watch the board go green as patches roll-out
- Roll your own or find a vendor

Mozilla MIG CloudPassage

Turn Vuln scanning on its head

- Add value for your ops teams
  - Subscribe and parse vuln emails for key software
  - Get this info during threat models or config mgmt
  - Provide an early warning and remove panic from software updates
- Roll your own or find a vendor
  - Gmail + filters can work surprisingly well
  - Secunia VIM covers 40K+ products
- Reverse the scan then report standard



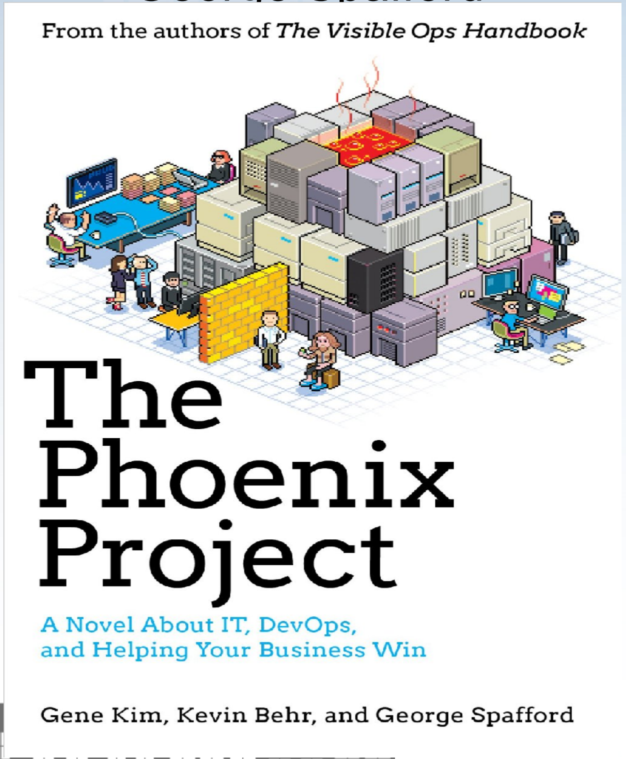
Key Take Aways

- Automate, automate, automate
  - Look for “paper cuts” and fix those first
- Finding workflow – your AppSec Pipeline
  - Figure this out and standardize / optimize
- Create systems which can *grow organically*
  - App is never done, its just created to easily be added to over time
  - Finding blocks become templates for next time
- Learn to talk “dev”

Books to read

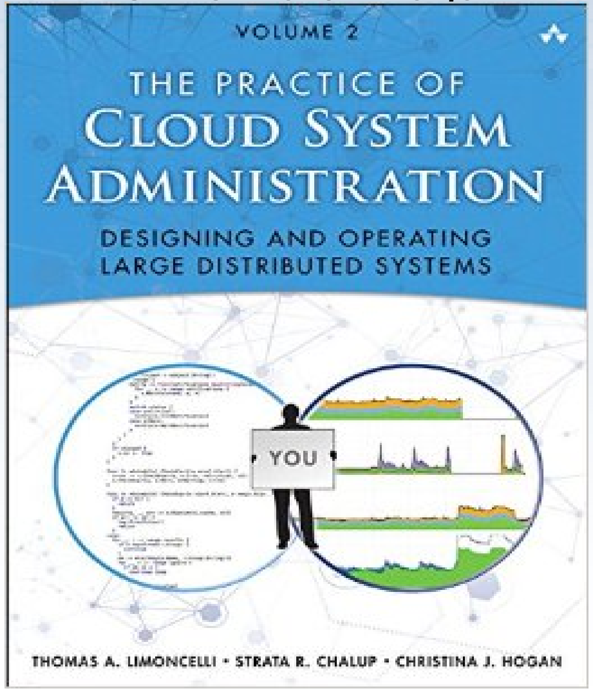
The Phoenix Project

Gene Kim, Kevin Behr and George Spafford


From the authors of The Visible Ops Handbook
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford

The Practice of Cloud System Administration

Thomas A. Limoncelli, Strata R. Chalup, Christina J. Hogan


VOLUME 2
THE PRACTICE OF CLOUD SYSTEM ADMINISTRATION
DESIGNING AND OPERATING LARGE DISTRIBUTED SYSTEMS
YOU
THOMAS A. LIMONCELLI • STRATA R. CHALUP • CHRISTINA J. HOGAN

OWASP AppSecEU 15
Amsterdam, The Netherlands

Questions?

Thank You