RSAConference2018

San Francisco | April 16 – 20 | Moscone Center

NOW MATTERS

SESSION ID: DEV-F03

DEVOPS AND THE FUTURE OF ENTERPRISE SECURITY

Frank Kim

Founder
ThinkSec
@fykim
www.frankkim.net

Security Perceptions

"DevOps is just another excuse for developers to have root access in production."

Walls of Confusion

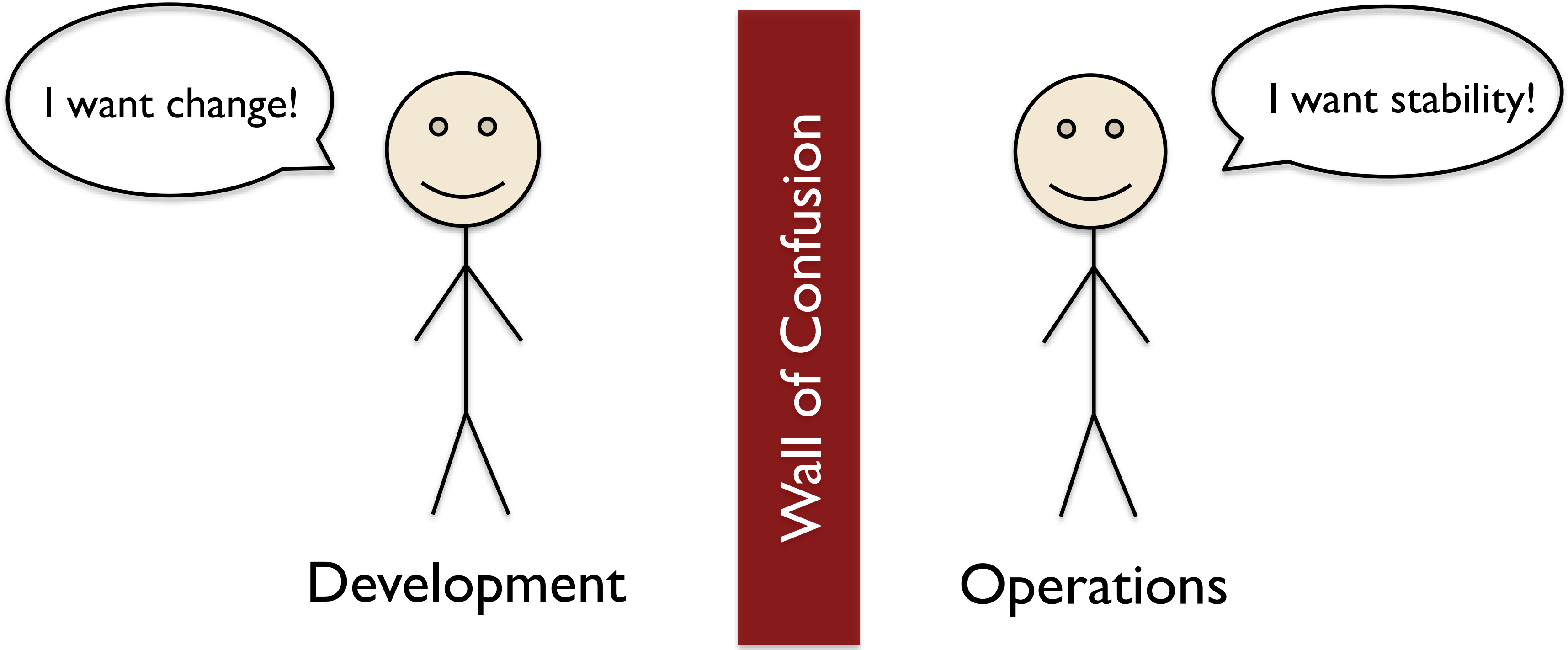


Image concept: http://dev2ops.org/2010/02/what-is-devops

#1 Understand DevOps

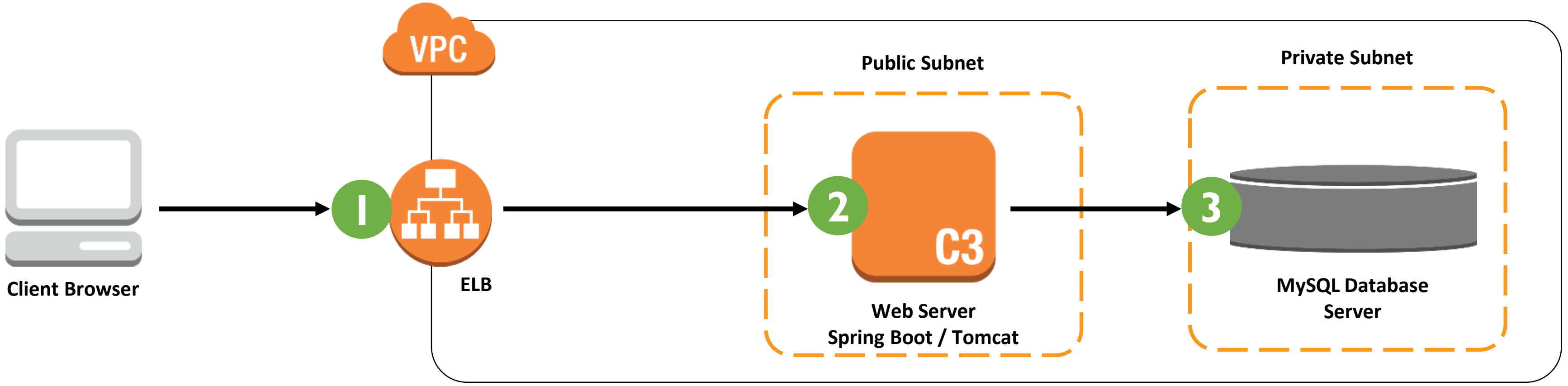
Security Perceptions

"It's not the strongest that survive or
the most intelligent that survive.
It's the ones that are most adaptable to change."

- Charles Darwin

Monolith Architecture Security Controls

- Common security controls are applied to each trust boundary in the monolith architecture:



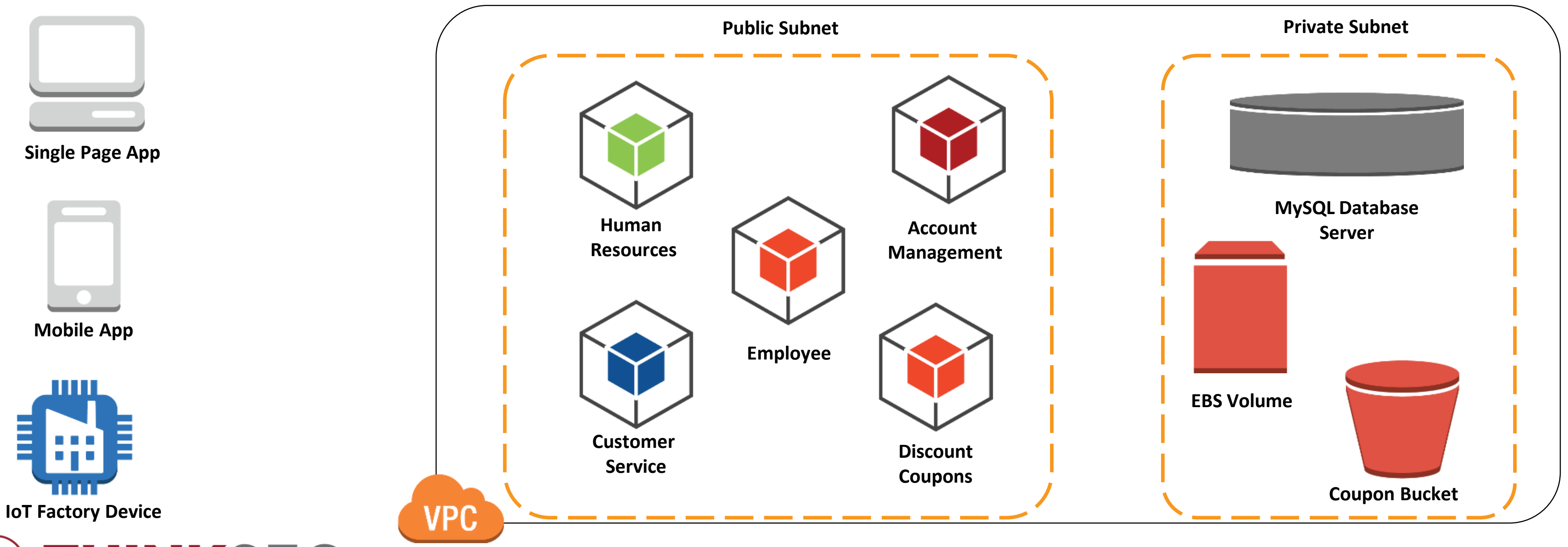
1. Security Controls
Web Application Firewall
HTTPS, Rate Limiting

2. Security Controls
Authentication, Authorization
Access control, Validation

3. Security Controls
System Authentication, TLS
Encryption at rest

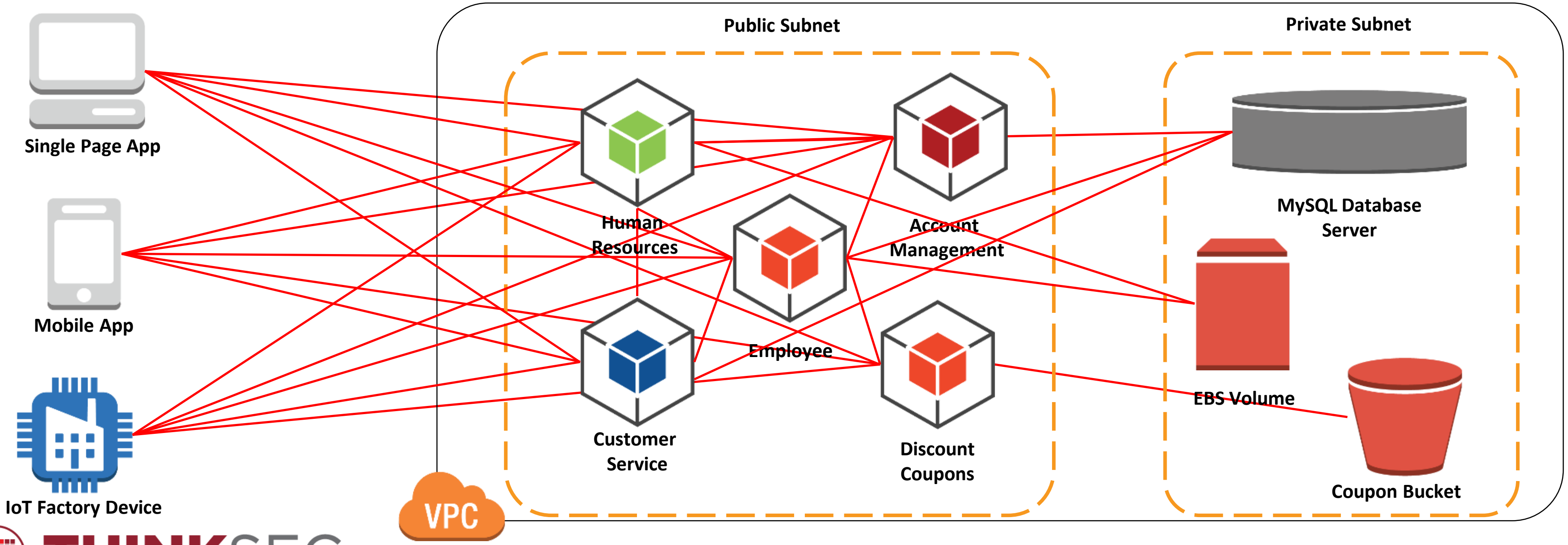
Microservice Architecture

- How does this change in a microservice architecture?



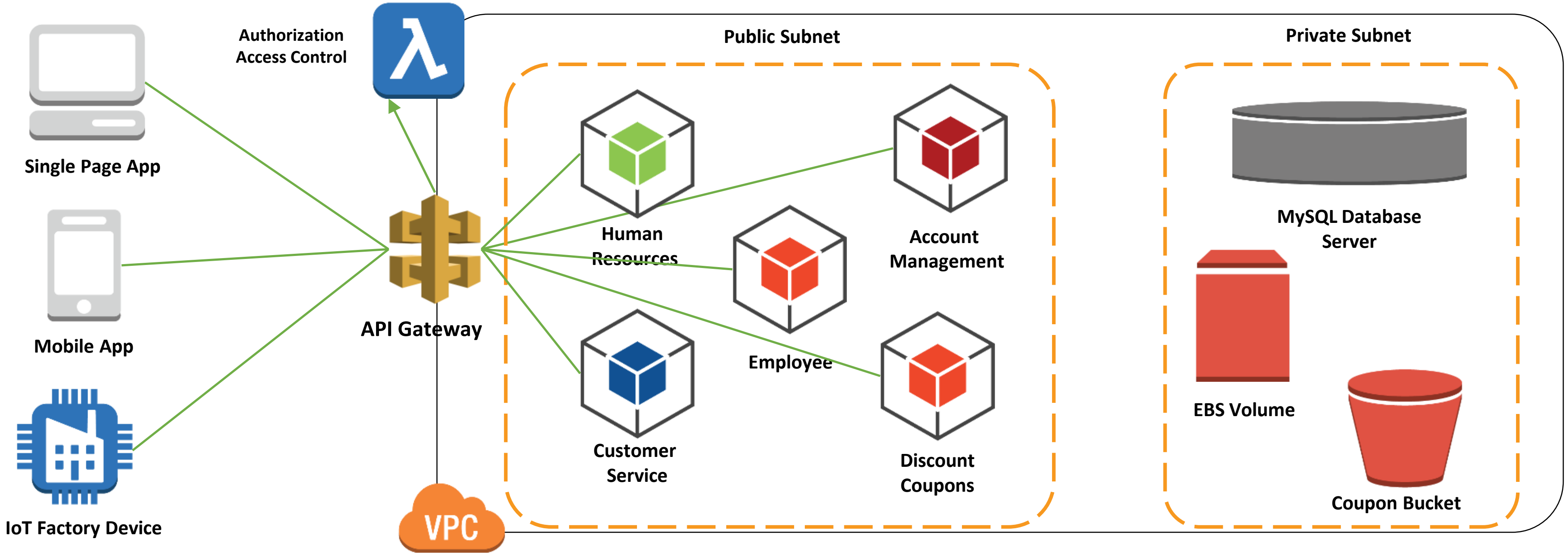
Microservice Architecture Attack Surface

- Consider the attack surface in a modern microservice architecture:



Microservice API Gateway Architecture

- Adding an API Gateway to provide perimeter security controls:



Serverless Computing

- Serverless refers to new, non-traditional architecture
  - Does not use dedicated containers
  - Event-triggered computing requests
  - Ephemeral environment
  - Servers fully managed by 3rd party (e.g. AWS)
  - Referred to as Functions as a service (FaaS)
- Example Technologies
  - AWS Lambda, MS Azure Functions, Google Cloud Functions
  - Amazon API Gateway

Serverless Security Benefits

- How does serverless improve security?
  - Attack surface is smaller
  - No servers to patch
  - No long running servers
    - That can be scanned or attacked
    - That can have malware installed on them
  - Fewer compromised servers
    - If malware is installed the next request brings up new, clean "server"

Serverless Security Concerns

- How does serverless make security harder?
  - Attack surface is bigger (but different)
  - Authentication and access control
  - Compliance

Serverless and Application Security

- Application security is even more important with serverless
  - If attackers have less infrastructure to attack
  - The focus naturally shifts to the application
- Every function crosses a trust boundary
  - Functions are designed to independent
  - Therefore each function must be secured independently
- Apply application security best practices
  - Input validation / sanitization must be performed in each function
  - Perform code review and automated scans
  - Review dependencies and libraries used by functions

AWS WAF Security Automations

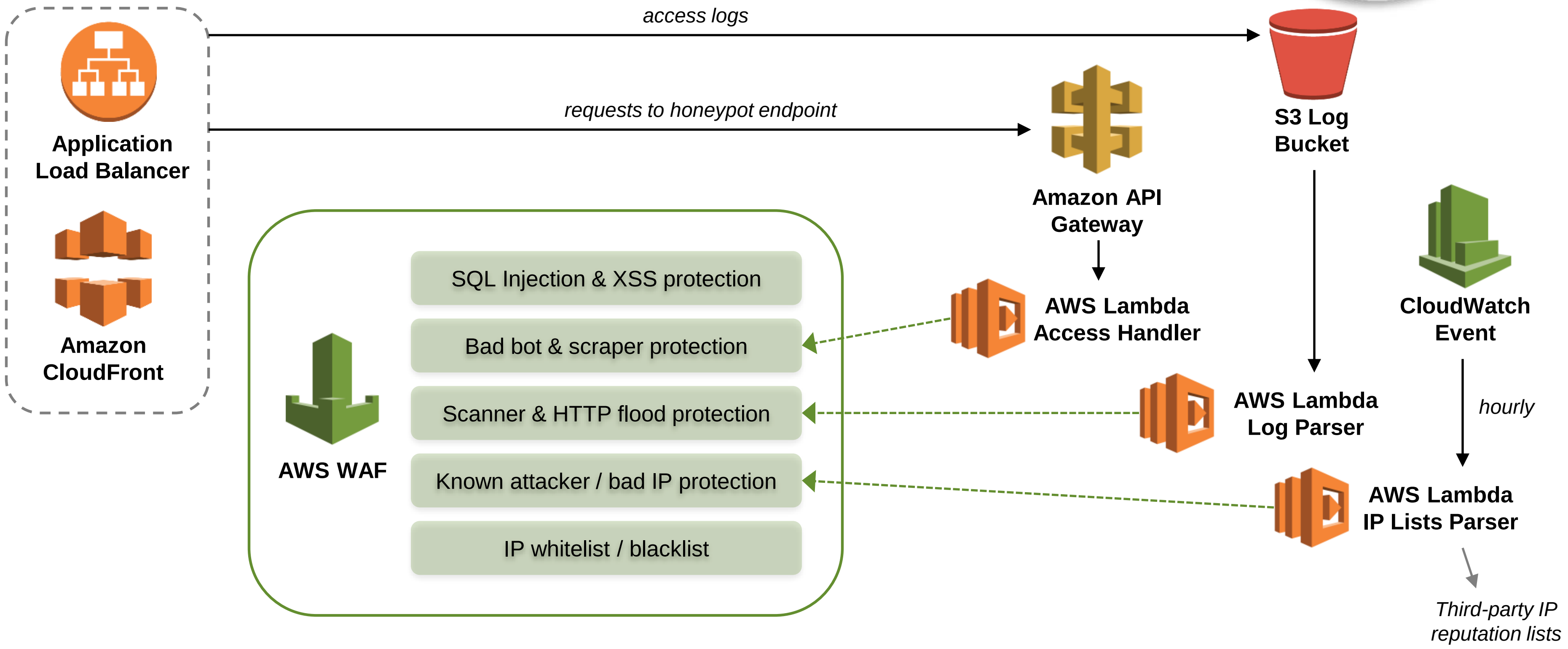

access logs
requests to honeypot endpoint
Application Load Balancer
Amazon CloudFront
Amazon API Gateway
S3 Log Bucket
AWS Lambda Access Handler
AWS Lambda Log Parser
CloudWatch Event
hourly
AWS Lambda IP Lists Parser
Third-party IP reputation lists
AWS WAF
SQL Injection & XSS protection
Bad bot & scraper protection
Scanner & HTTP flood protection
Known attacker / bad IP protection
IP whitelist / blacklist

Image credit: http://docs.aws.amazon.com/solutions/latest/aws-waf-security-automations/architecture.html

#2 Support DevOps

Hunt the Bug

- Can you identify the bug in this code snippet?

```
<%
String theme = request.getParameter("look");
if (theme == null && session != null) {
    theme = (String)session.getAttribute("look");
}

if (session !=null) session.setAttribute("look", theme);
%>

<link rel="stylesheet" type="text/css" media="all"
    href="<%= request.getContextPath() %>
         /ui/theme/<%= theme %>/colors.css" />
```

Hunt the Bug – XSS

- Can you identify the bug in this code snippet?

```
<%
String theme = request.getParameter("look");
if (theme == null && session != null) {
    theme = (String)session.getAttribute("look");
}

if (session !=null) session.setAttribute("look", theme);
%>

<link rel="stylesheet" type="text/css" media="all"
    href="<%= request.getContextPath() %>
         /ui/theme/<%= theme %>/colors.css" />
```

AngularJS Output Encoding

- ngBind for HTML tags

```
<div ng-controller="ExampleController">
  <label>Enter name: <input type="text" ng-model="name"></label><br>
  Hello <span ng-bind="name"></span>!
</div>
```

- Output from AngularJS expressions

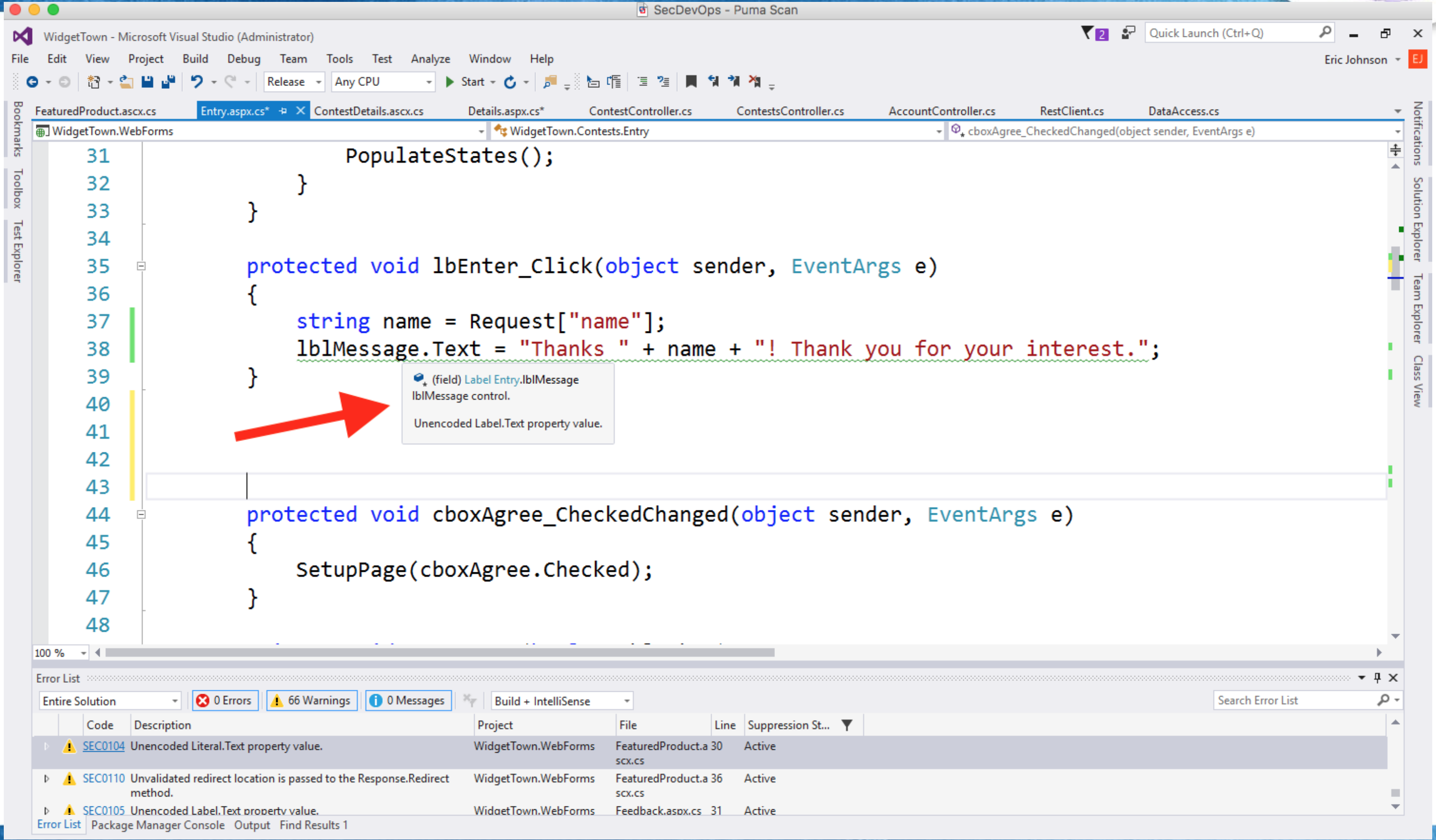
```
<div ng-controller="ExampleController" class="expressions">
  Expression:<input type='text' ng-model="expr" size="80"/>
  <button ng-click="addExp(expr)">Evaluate</button>
  <ul>
   <li ng-repeat="expr in exprs track by $index">
     [ <a href="" ng-click="removeExp($index)">X</a> ]
     <code>{{expr}}</code> => <span ng-bind="$parent.$eval(expr)"></span>
    </li>
  </ul>
</div>
```

Static Analysis Tools

- Free / open source:
  - Find Security Bugs, Phan, Puma Scan, Brakeman, Bandit, Flawfinder, QARK
- Commercial:
  - HP Fortify, Checkmarx, Coverity, IBM AppScan Source, Klocwork, Veracode, Brakeman Pro

Secure Code Spell Checker

# RSAC



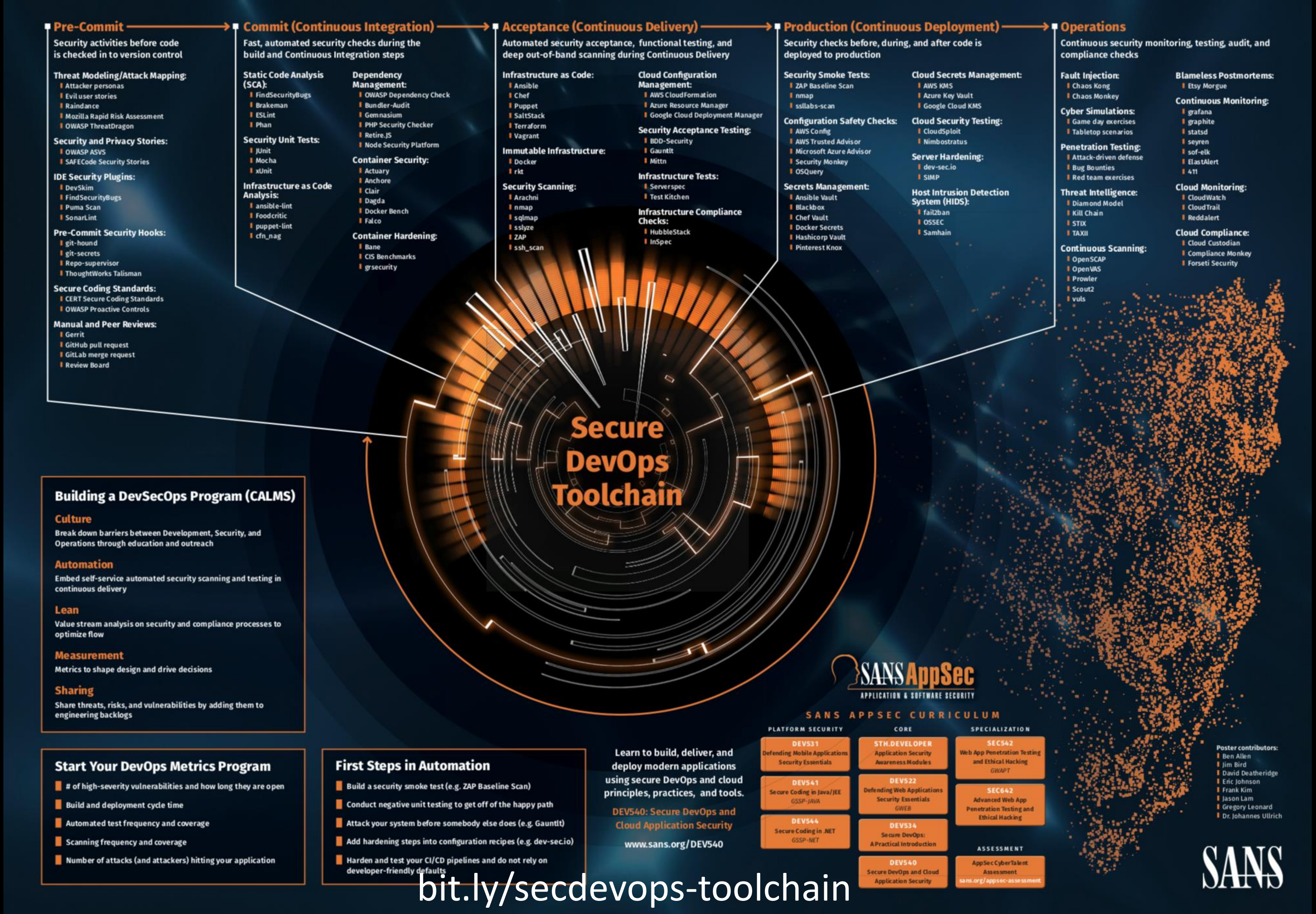
```
            PopulateStates();
        }
    }

    protected void lbEnter_Click(object sender, EventArgs e)
    {
        string name = Request["name"];
        lblMessage.Text = "Thanks " + name + "! Thank you for your interest.";
    }




    protected void cboxAgree_CheckedChanged(object sender, EventArgs e)
    {
        SetupPage(cboxAgree.Checked);
    }

```

(field) Label Entry.lblMessage
lblMessage control.

Unencoded Label.Text property value.

Error List

Code	Description	Project	File	Line	Suppression St...
SEC0104	Unencoded Literal.Text property value.	WidgetTown.WebForms	FeaturedProduct.ascx.cs	30	Active
SEC0110	Unvalidated redirect location is passed to the Response.Redirect method.	WidgetTown.WebForms	FeaturedProduct.ascx.cs	36	Active
SEC0105	Unencoded Label.Text property value.	WidgetTown.WebForms	Feedback.aspx.cs	31	Active



Pre-Commit

Security activities before code is checked in to version control

Threat Modeling/Attack Mapping:
- Attacker personas
- Evil user stories
- Raindance
- Mozilla Rapid Risk Assessment
- OWASP ThreatDragon

Security and Privacy Stories:
- OWASP ASVS
- SAFECode Security Stories

IDE Security Plugins:
- DevSkim
- FindSecurityBugs
- Puma Scan
- SonarLint

Pre-Commit Security Hooks:
- git-hound
- git-secrets
- Repo-supervisor
- ThoughtWorks Talisman

Secure Coding Standards:
- CERT Secure Coding Standards
- OWASP Proactive Controls

Manual and Peer Reviews:
- Gerrit
- GitHub pull request
- GitLab merge request
- Review Board

Commit (Continuous Integration)

Fast, automated security checks during the build and Continuous Integration steps

Static Code Analysis (SCA):
- FindSecurityBugs
- Brakeman
- ESLint
- Phan

Security Unit Tests:
- JUnit
- Mocha
- xUnit

Infrastructure as Code Analysis:
- ansible-lint
- Foodcritic
- puppet-lint
- cfn_nag

Dependency Management:
- OWASP Dependency Check
- Bundler-Audit
- Gemnasium
- PHP Security Checker
- Retire.JS
- Node Security Platform

Container Security:
- Actuary
- Anchore
- Clair
- Dagda
- Docker Bench
- Falco

Container Hardening:
- Bane
- CIS Benchmarks
- grsecurity

Acceptance (Continuous Delivery)

Automated security acceptance, functional testing, and deep out-of-band scanning during Continuous Delivery

Infrastructure as Code:
- Ansible
- Chef
- Puppet
- SaltStack
- Terraform
- Vagrant

Immutable Infrastructure:
- Docker
- rkt

Security Scanning:
- Arachni
- nmap
- sqlmap
- sslyze
- ZAP
- ssh_scan

Cloud Configuration Management:
- AWS CloudFormation
- Azure Resource Manager
- Google Cloud Deployment Manager

Security Acceptance Testing:
- BDD-Security
- Gauntlt
- Mittn

Infrastructure Tests:
- Serverspec
- Test Kitchen

Infrastructure Compliance Checks:
- HubbleStack
- InSpec

Production (Continuous Deployment)

Security checks before, during, and after code is deployed to production

Security Smoke Tests:
- ZAP Baseline Scan
- nmap
- ssllabs-scan

Configuration Safety Checks:
- AWS Config
- AWS Trusted Advisor
- Microsoft Azure Advisor
- Security Monkey
- OSQuery

Secrets Management:
- Ansible Vault
- Blackbox
- Chef Vault
- Docker Secrets
- Hashicorp Vault
- Pinterest Knox

Cloud Secrets Management:
- AWS KMS
- Azure Key Vault
- Google Cloud KMS

Cloud Security Testing:
- CloudSploit
- Nimbostratus

Server Hardening:
- dev-sec.io
- SIMP

Host Intrusion Detection System (HIDS):
- fail2ban
- OSSEC
- Samhain

Operations

Continuous security monitoring, testing, audit, and compliance checks

Fault Injection:
- Chaos Kong
- Chaos Monkey

Cyber Simulations:
- Game day exercises
- Tabletop scenarios

Penetration Testing:
- Attack-driven defense
- Bug Bounties
- Red team exercises

Threat Intelligence:
- Diamond Model
- Kill Chain
- STIX
- TAXII

Continuous Scanning:
- OpenSCAP
- OpenVAS
- Prowler
- Scout2
- vuls

Blameless Postmortems:
- Etsy Morgue

Continuous Monitoring:
- grafana
- graphite
- statsd
- seyren
- sof-elk
- ElastAlert
- 411

Cloud Monitoring:
- CloudWatch
- CloudTrail
- Reddalert

Cloud Compliance:
- Cloud Custodian
- Compliance Monkey
- Forseti Security

Secure DevOps Toolchain

Building a DevSecOps Program (CALMS)

Culture
Break down barriers between Development, Security, and Operations through education and outreach

Automation
Embed self-service automated security scanning and testing in continuous delivery

Lean
Value stream analysis on security and compliance processes to optimize flow

Measurement
Metrics to shape design and drive decisions

Sharing
Share threats, risks, and vulnerabilities by adding them to engineering backlogs

Start Your DevOps Metrics Program
- # of high-severity vulnerabilities and how long they are open
- Build and deployment cycle time
- Automated test frequency and coverage
- Scanning frequency and coverage
- Number of attacks (and attackers) hitting your application

First Steps in Automation
- Build a security smoke test (e.g. ZAP Baseline Scan)
- Conduct negative unit testing to get off of the happy path
- Attack your system before somebody else does (e.g. Gauntlt)
- Add hardening steps into configuration recipes (e.g. dev-sec.io)
- Harden and test your CI/CD pipelines and do not rely on developer-friendly defaults

Learn to build, deliver, and deploy modern applications using secure DevOps and cloud principles, practices, and tools.

DEV540: Secure DevOps and Cloud Application Security

www.sans.org/DEV540

SANS AppSec
APPLICATION & SOFTWARE SECURITY

SANS APPSEC CURRICULUM

PLATFORM SECURITY	CORE	SPECIALIZATION
DEV531 Defending Mobile Applications Security Essentials	STH.DEVELOPER Application Security Awareness Modules	SEC542 Web App Penetration Testing and Ethical Hacking GWAPT
DEV541 Secure Coding in Java/JEE GSSP-JAVA	DEV522 Defending Web Applications Security Essentials GWEB	SEC642 Advanced Web App Penetration Testing and Ethical Hacking
DEV544 Secure Coding in .NET GSSP-NET	DEV534 Secure DevOps: A Practical Introduction	ASSESSMENT
	DEV540 Secure DevOps and Cloud Application Security	AppSec CyberTalent Assessment sans.org/appsec-assessment

Poster contributors:
- Ben Allen
- Jim Bird
- David Deatheridge
- Eric Johnson
- Frank Kim
- Jason Lam
- Gregory Leonard
- Dr. Johannes Ullrich

SANS

bit.ly/secdevops-toolchain

#3

Adopt DevOps

Critical Security Controls (CSC)

CIS Controls

First 5 CIS Controls
Eliminate the vast majority of your organization's vulnerabilities

1: **Inventory of Authorized and Unauthorized Devices**
2: **Inventory of Authorized and Unauthorized Software**
3: **Secure Configurations for Hardware and Software**
4: **Continuous Vulnerability Assessment and Remediation**
5: **Controlled Use of Administrative Privileges**

Infrastructure as Code

- Different approaches to set up and manage systems
  - Traditional: manual checklists and scripts, ad hoc changes/fixes made by system administrators at runtime
  - Modern: treating Infrastructure as Code and configuration management as system engineering
- Configuration management with scripts is not scalable
  - Error prone and leads to configuration drift over time
- Configuration management tools
  - Chef, Puppet, Ansible, Salt/Saltstack, CFEngine

Automate Standard Configurations

- AWS CloudFormation to create EC2 instance

```
InstancePublic:
  Type: AWS::EC2::Instance
  Properties:
  IamInstanceProfile: !Ref InstanceProfilePhotoReadOnly
  ImageId: !FindInMap [Images, !Ref "AWS::Region", ecs]
  InstanceType: "t2.micro"
  KeyName: "secretKey"
  SecurityGroupIds:
    - !Ref SecurityGroupPublic
  SubnetId: !Ref SubnetPublic
  UserData:
    Fn::Base64:
      !Sub |
      #!/bin/bash -xe
      yum update -y
```

Conduct Asset Inventory

- Command line call to retrieve all EC2 instances

```
aws ec2 describe-instances --output json | jq
'.Reservations[].Instances[] | [.LaunchTime, .InstanceType, .InstanceId,
.SecurityGroups[].GroupId, .Tags[].Value]'
```

- Output

```
[ "2017-01-08T18:51:46.000Z", "t2.micro", "i-0500510e3f808d2ee", "sg-7caf4600"
  , "prod-springline-aws-web", "Springboot MVC target application"
  , "SANS\\app.user"
]
[
  "2017-01-08T18:55:02.000Z", "t2.micro", "i-0e74e490c2ebc5d37", "sg-79af4605"
  ,"qa-springline-aws-web", "QA Springboot MVC target application"
  , "SANS\\app.user"
]
```

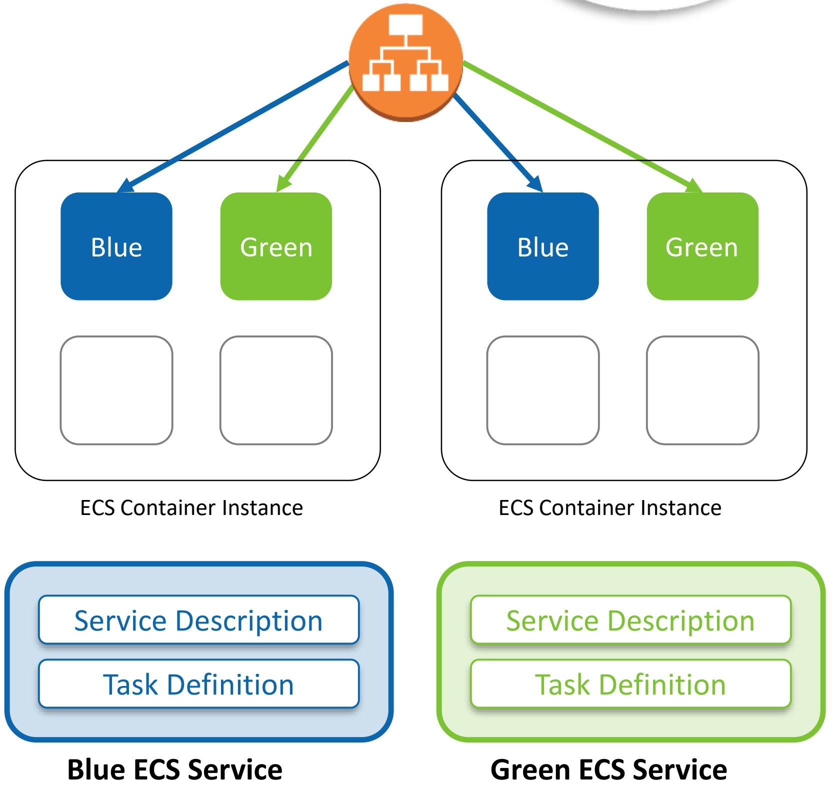
THINKSEC

Continuous Vulnerability Remediation

- Blue/Green Deployment
  - Divert traffic from one environment to another
  - Each running a different version of the application
- Benefits of blue/green deployments
  - Reduced downtime
  - Improved ability to rollback
  - Faster deployment of features and bug fixes
- Use blue/green deploys when you have:
  - Immutable infrastructure
  - Well defined environment boundary
  - Ability to automate changes

AWS Elastic Container Service (ECS)

- Deployment process
  - Use original blue service and task def
  - Create new green service and task def
  - Map new green service to the Application Load Balancer (ALB)
  - Scale up green service by increasing number of tasks
  - Remove blue service, setting tasks to 0



Deploying Application Updates

- Create a new “green” ECS Service

```
aws cloudformation deploy --template-file green-web-ecs-service.yaml --stack-name green-web-ecs-service
```

- Increase the desired count for the “green” service

```
aws ecs update-service --cluster DM-ecs --service $GreenService --desired-count 1
```

- Turn off the “blue” service when ready

```
aws ecs update-service --cluster DM-ecs --service $BlueService --desired-count 0
```

Key Takeaways

- Understand DevOps
  - Next week: Begin to understand the DevOps CI/CD pipeline and modern architectures used in your organization
- Support DevOps
  - In three months: Inject security into the CI/CD pipeline in an easy to use way
- Adopt DevOps
  - In six months: Leverage DevOps principles and practices to improve your security program

RSAConference2018

#RSAC

Frank Kim
@fykim
www.frankkim.net

Material based on SANS DEV540
Secure DevOps and Cloud Application Security