# Secure Development Lifecycle

Eoin Keary & Jim Manico

# Jim Manico @manicode

- **OWASP Volunteer**
  - Global OWASP Board Member
  - OWASP Cheat-Sheet Series Manager
- **VP of Security Architecture, WhiteHat Security**
  - 16 years of web-based, database-driven software development and analysis experience
  - Secure coding educator/author
- **Kama'aina Resident of Kauai, Hawaii**
  - Aloha!

# Security in the SCLC

Essential that security is embedded in all stages of the SDLC

- Requirements definition
- Design
- Development
- Testing
- Implementation

**BE FLEXIBLE!**

*"The cost of removing an application security vulnerability during the design phase ranges from 30-60 times less than if removed during production."*

- NIST, IBM, and Gartner Group

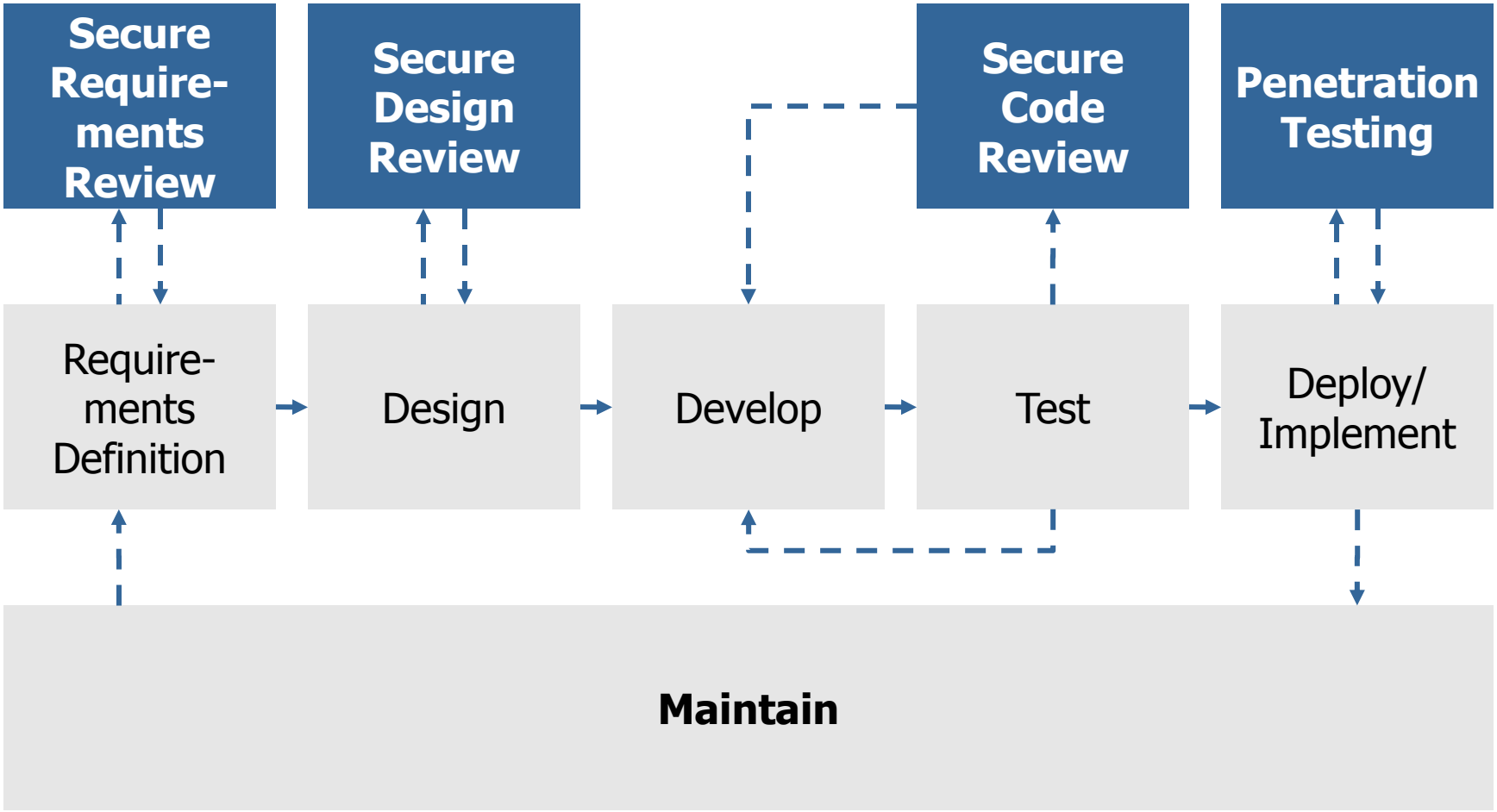
**If you do not have a published SDLC for your organization then you will NOT be successful.**

## SDLC building blocks

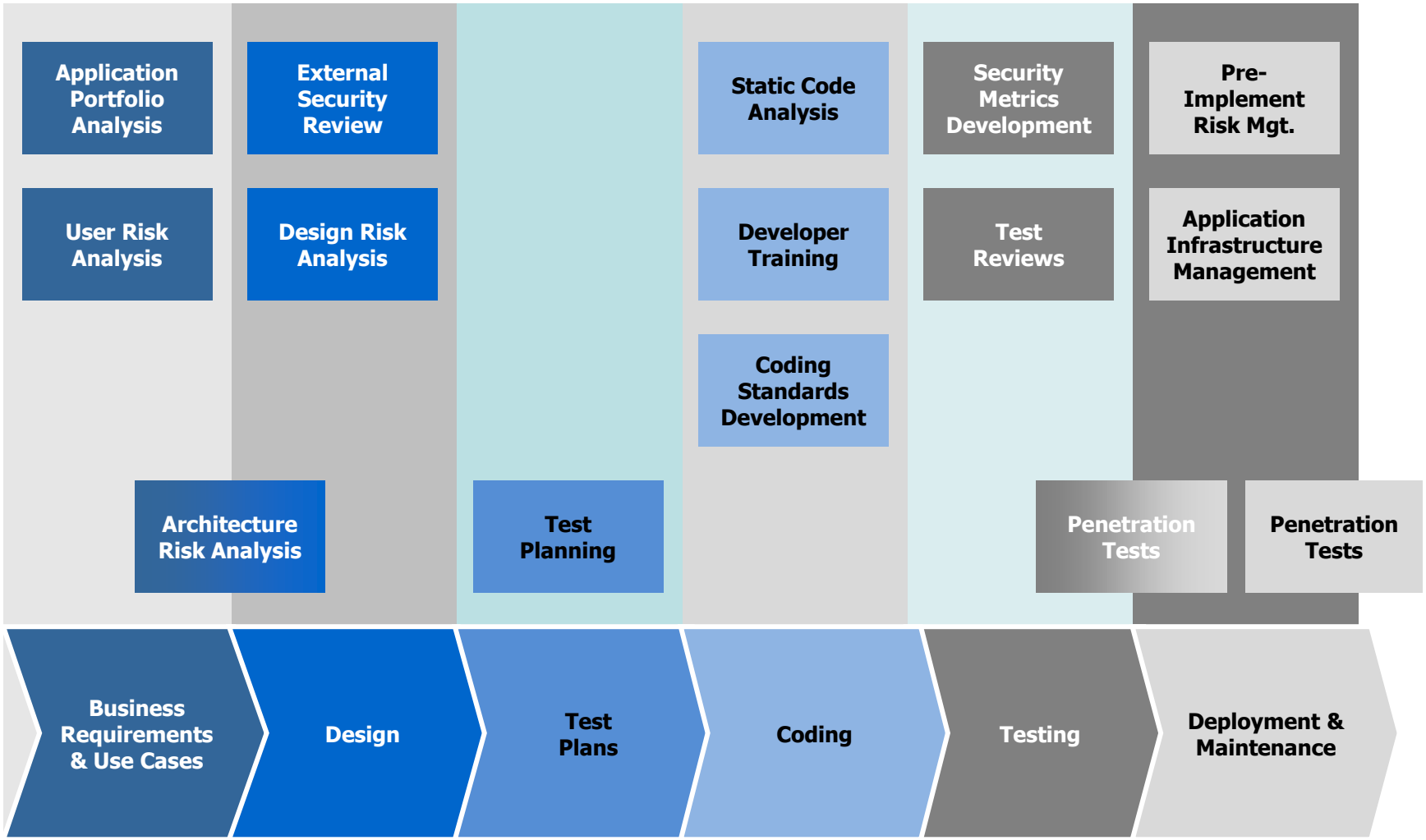
Supporting quotes and research (+)
Secure Coding Guidelines (-)
Secure Coding checklist (+)
Non Functional Requirements (++)
Static Code Analysis (+)
Dynamic Code Analysis (+)
Security Awareness Training (++)
Threat Modeling (+/-)
Application Security Risk Matrix (++)
Published SDLC (++)

Recommended:
Center of Excellence (++)

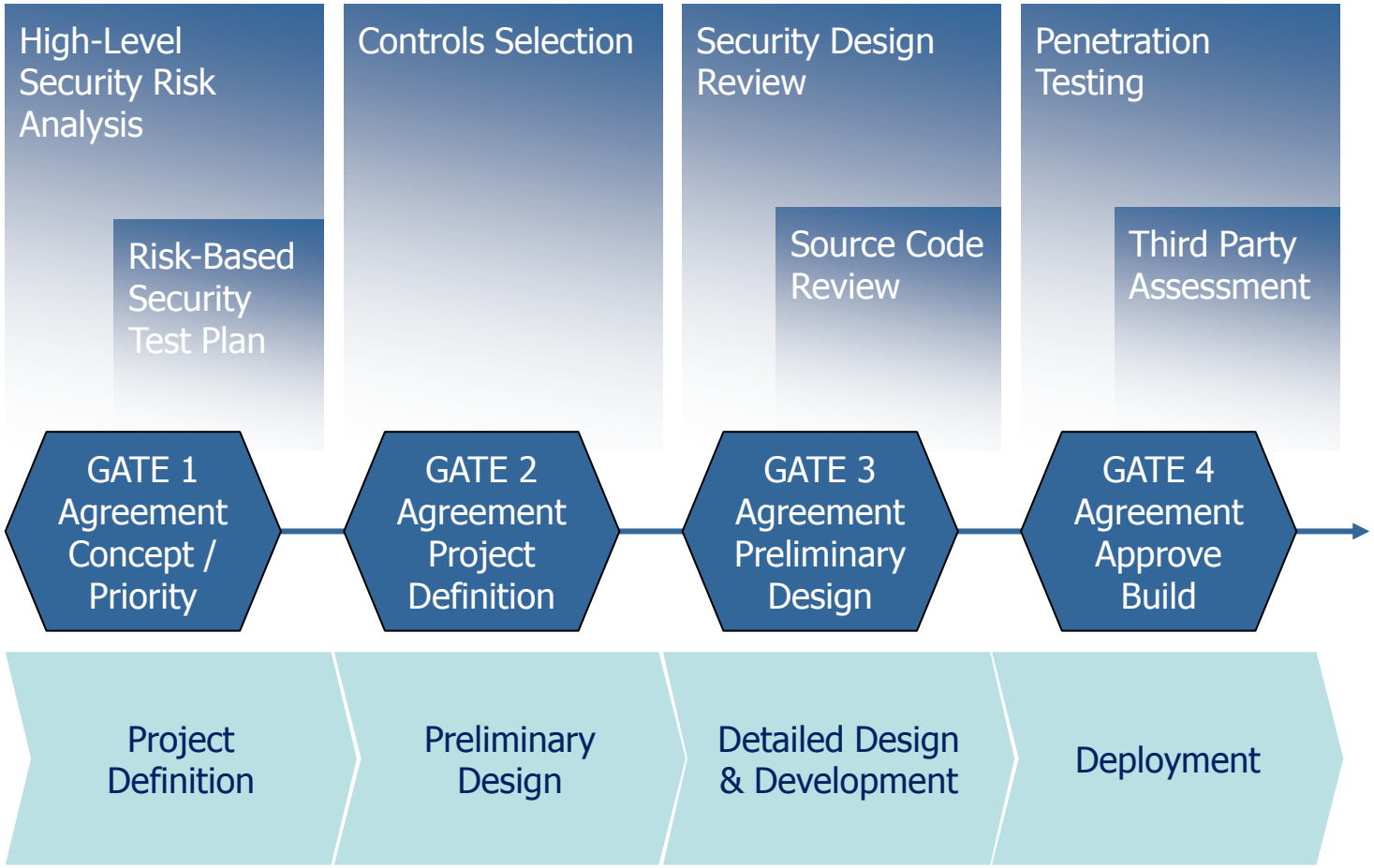
# Security in the SCLC



# Security in the SDLC



# Security quality gates




**COMPANY Software Development Lifecycle (SDLC)**

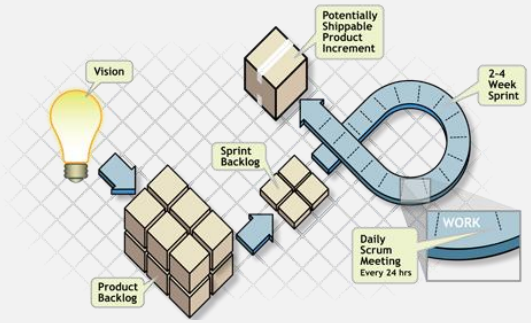
# Agile Security

## Security Sprint Approach

**Security Sprint Approach:**

- Dedicated sprint focusing on application security.
- Stories implemented are security related.
- Code is reviewed.
- Stories may include:
  - Input validation story,
  - Logging story,
  - Authentication story,
  - Authorisation

and some technical risks such as XSS, SQLI etc.




## Every Sprint Approach

**Every Sprint Approach:**

- Similar to Microsoft Security Development Lifecycle (SDL).
- Consists of the requirements and stories essential to security.
- No software should ever be released without requirements being met.
- Sprint is two weeks or two months long.
- Every security requirement in the every-Sprint category must be completed in each and every Sprint.
  - Or the Sprint is deemed incomplete, and the software cannot be released.

# Requirements

## Security Sprint SDL Requirements

**Why?**

- Repetition not necessary
- Must occur at the beginning of the project
- Not possible at the beginning of the project

**Examples:**

- Configure bug tracking system (3 months)
- Identify security/privacy experts (1 month)
- Baseline threat model (3 months)
- Establish a security response plan (6 months)

## Every-Sprint SDL Requirements

**Examples:**

- Update the threat model
- Communicate privacy-impacting design changes to the team's privacy advisor
- Fix all issues identified by code analysis tools for unmanaged code
- Follow input validation and output encoding guidelines to defend against cross-site scripting attacks

## Non-Functional Requirements (++)

Most effective of all building blocks

‘Container’ for other SDLC building blocks.

Can include application security guidelines, secure coding checklist, security policies, etc.

Effective NFRs will document the requirement *and* explain why the requirement is necessary.

# Security requirements

Establishing the security requirements for the application

What are the key security risks within the application?

- Type of information application is processing
- Functionality
- Use case modelling

Involve group risk and/or internal audit to avoid later conflict

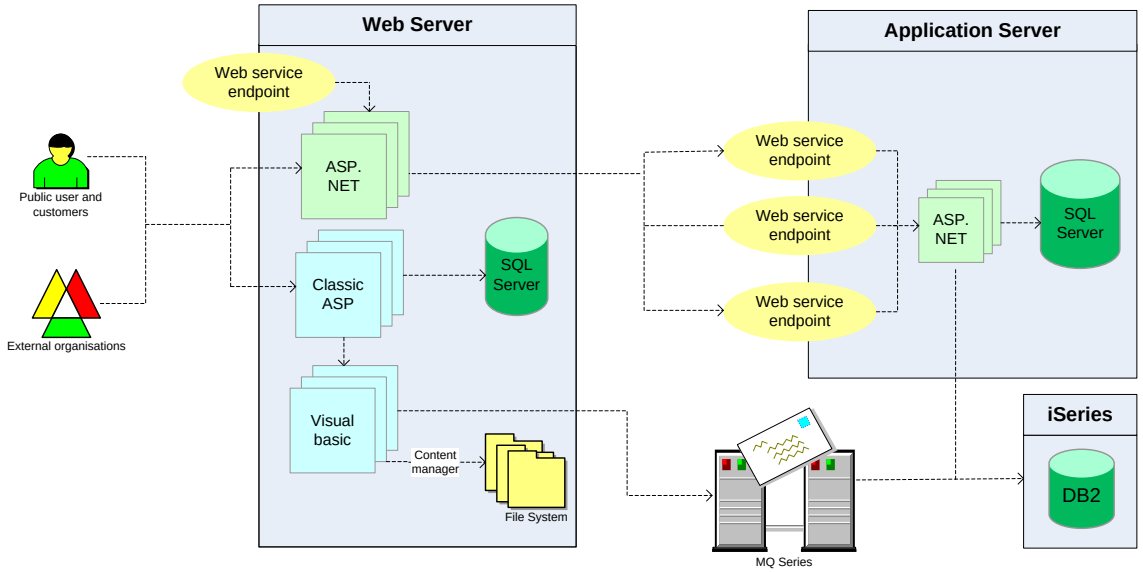
What are the Group standards (e.g. password lengths, security schemes), legal and regulatory security requirements?

Is the project acceptable from an information security perspective and what are the key security requirements which should be deployed?

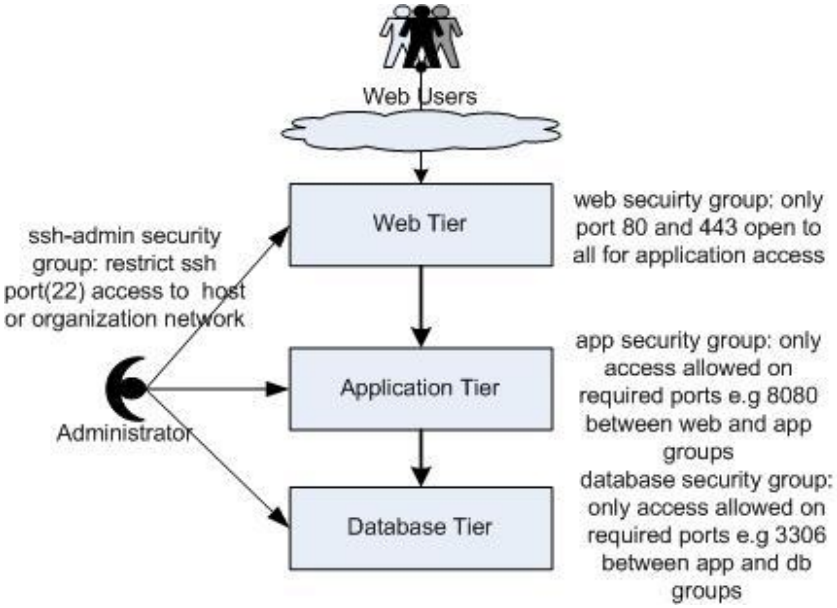
# Security architecture

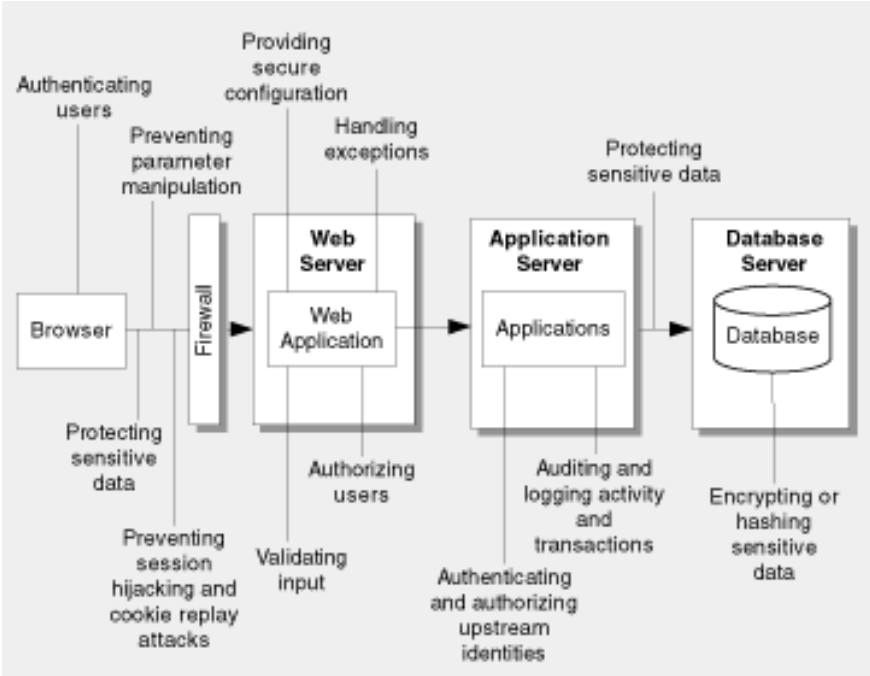
How do the application components fit together

- Web server
- Database
- Underlying operating systems
- Middleware
- Interfaces with backend systems



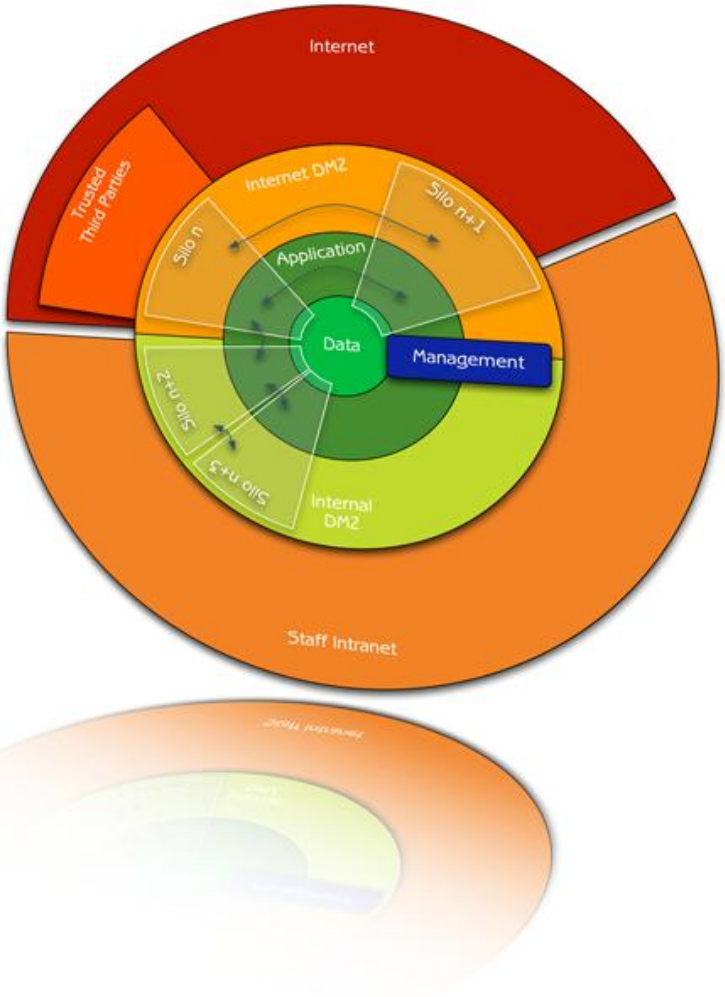
# Deployment


Web Users
Web Tier
Application Tier
Database Tier
Administrator
ssh-admin security group: restrict ssh port(22) access to host or organization network
web secuirty group: only port 80 and 443 open to all for application access
app security group: only access allowed on required ports e.g 8080 between web and app groups
database security group: only access allowed on required ports e.g 3306 between app and db groups


Authenticating users
Providing secure configuration
Preventing parameter manipulation
Handling exceptions
Protecting sensitive data
Browser
Firewall
Web Server
Web Application
Application Server
Applications
Database Server
Database
Protecting sensitive data
Preventing session hijacking and cookie replay attacks
Validating input
Authorizing users
Auditing and logging activity and transactions
Authenticating and authorizing upstream identities
Encrypting or hashing sensitive data

# Logical Zones

Compartmentalise
Minimise attack surface
Levels of trust
Defence in depth



# Security design

Building security into the design of the application

Threat modeling has four major steps:

- Decomposing the application
- Categorizing threats
- Ranking threats
- Mitigation

Designing the countermeasures to mitigate threats identified and address the security requirements

Planning the security testing phase
(i.e. how to test the countermeasures designed)

Output is the security technical specifications and security test plans

## Threat modeling (+/-)

Hit or miss at most locations

Can be informal process

Combines nicely with NFRs

Discussing NFR often leads to threat modeling discussion

# Application Security Risk Matrix (++)

| | | |
|---|---|---|
| External facing | Data:<br>Non sensitive | Data:<br>sensitive |
| Internal facing | Data:<br>Non sensitive | Data:<br>sensitive |

# Development

Ensuring that code is developed securely and implementing the security controls identified during the design phase

Developer security awareness programs

Unit testing of security features of the application

Security audit and code reviews

- Secure coding standards
- Automated code review tools
- Independent code review by third party or IT security

**Security awareness training (++)**

Instructor-led training

Course curriculum for each job responsibility

Very useful for educating on attack techniques and unexpected behavior

Rewards for training

**Your goal should be to provide anyone that can influence application security, e.g. project managers, development managers, application developers, server configuration, release management, QA, etc. with the training, awareness and resources they need to be successful.**

## Secure Coding Guidelines (-)

Overlooked by developers

“Static and not helpful”

100+ pages that can be language specific

Most surprising discovery over the last 5 years

*Can* be successful if collaborative/wiki format and regularly updated

## Secure Coding Checklist (+)

Simple 1-2 page document

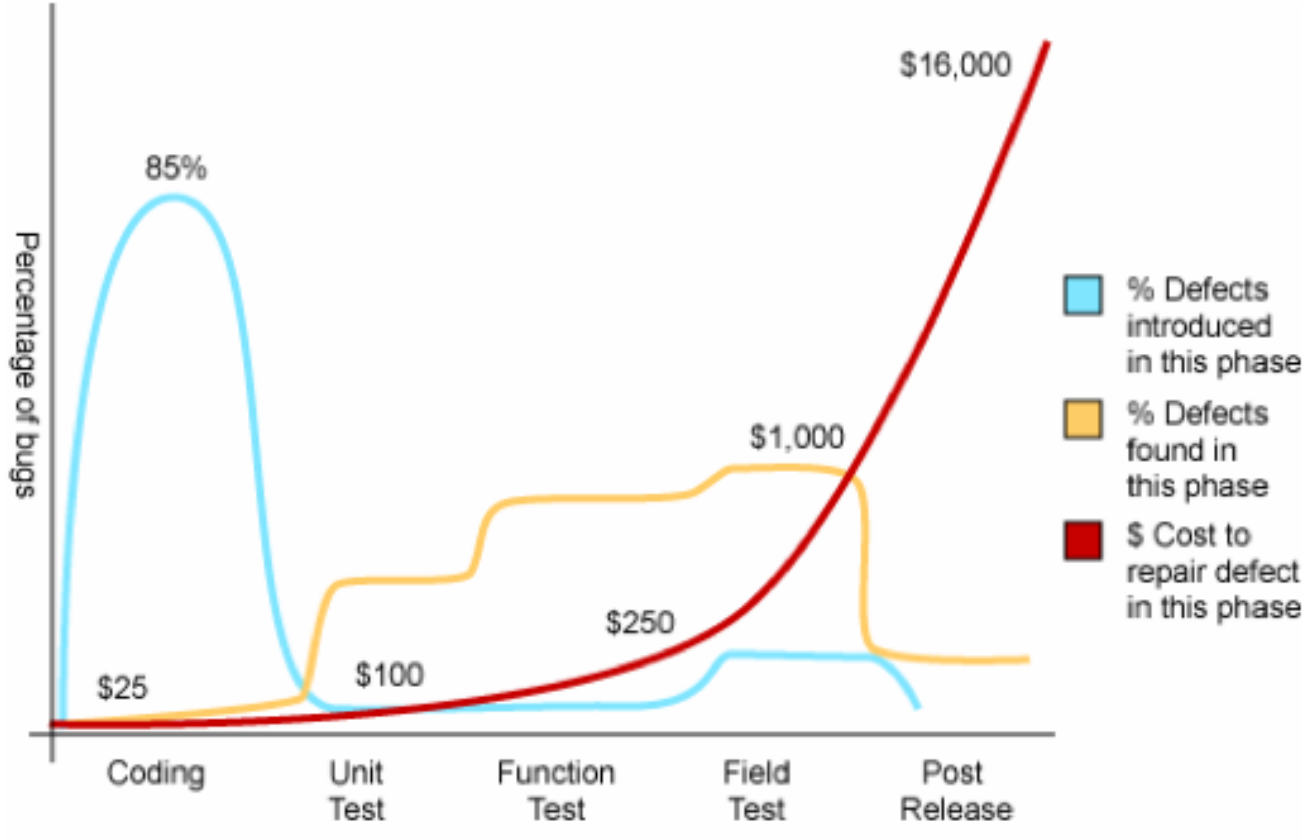
Useful if combined with a peer code review prior to code check-in.

# Testing

- Ensure the application meets the security standards and security testing is performed
- Has the security design been implemented correctly in the application components?
- Execution of test plans created during the design phase
- Independent penetration testing, including infrastructure assessment
- Security release and sign off before deployment to the production environment

# Why Code review

## The Cost of Software Bugs




**"We cant hack our-selves secure, and if we could it would cost too much"**

Source: Applied Software Measurement, Capers Jones, 1996

## Static Code Analysis (SCA) (+)

SDLC requires SCA

Must be baked into acceptance criteria for code to leave the SDLC.

Assurance to QA that code is ready for testing

SCA can be integrated into the build process (each automated build spawns Static Code Analysis)

## Dynamic Code Analysis (+)

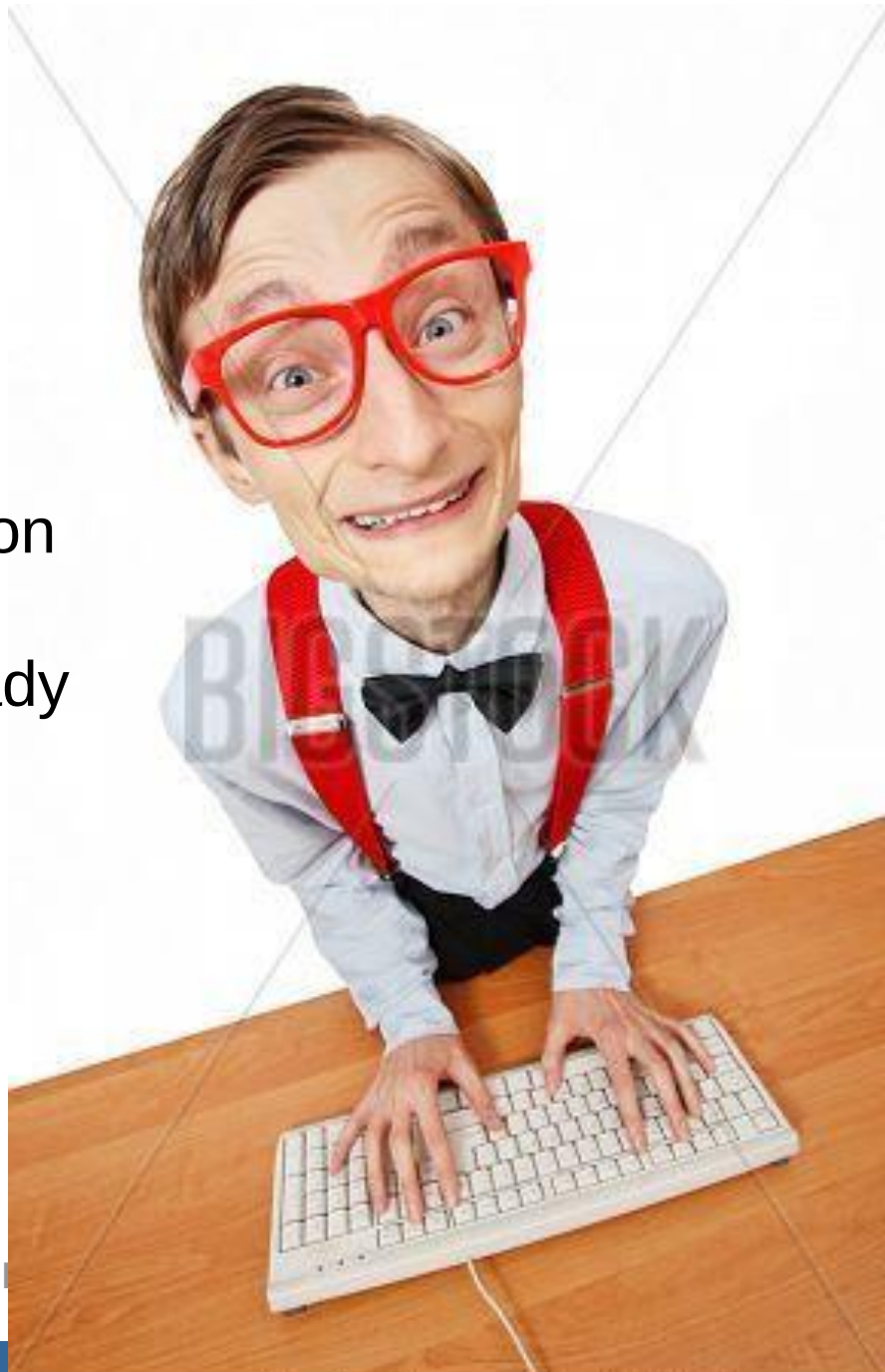
Looks for unexpected application behavior within the interface

Dynamic analysis can happen multiple times during each iteration

Assurance to QA that code is ready for testing

Dynamic analysis can offer 24/7 monitoring

Be tied to incident management process



## Center of Excellence (++)

COE Steering Committee

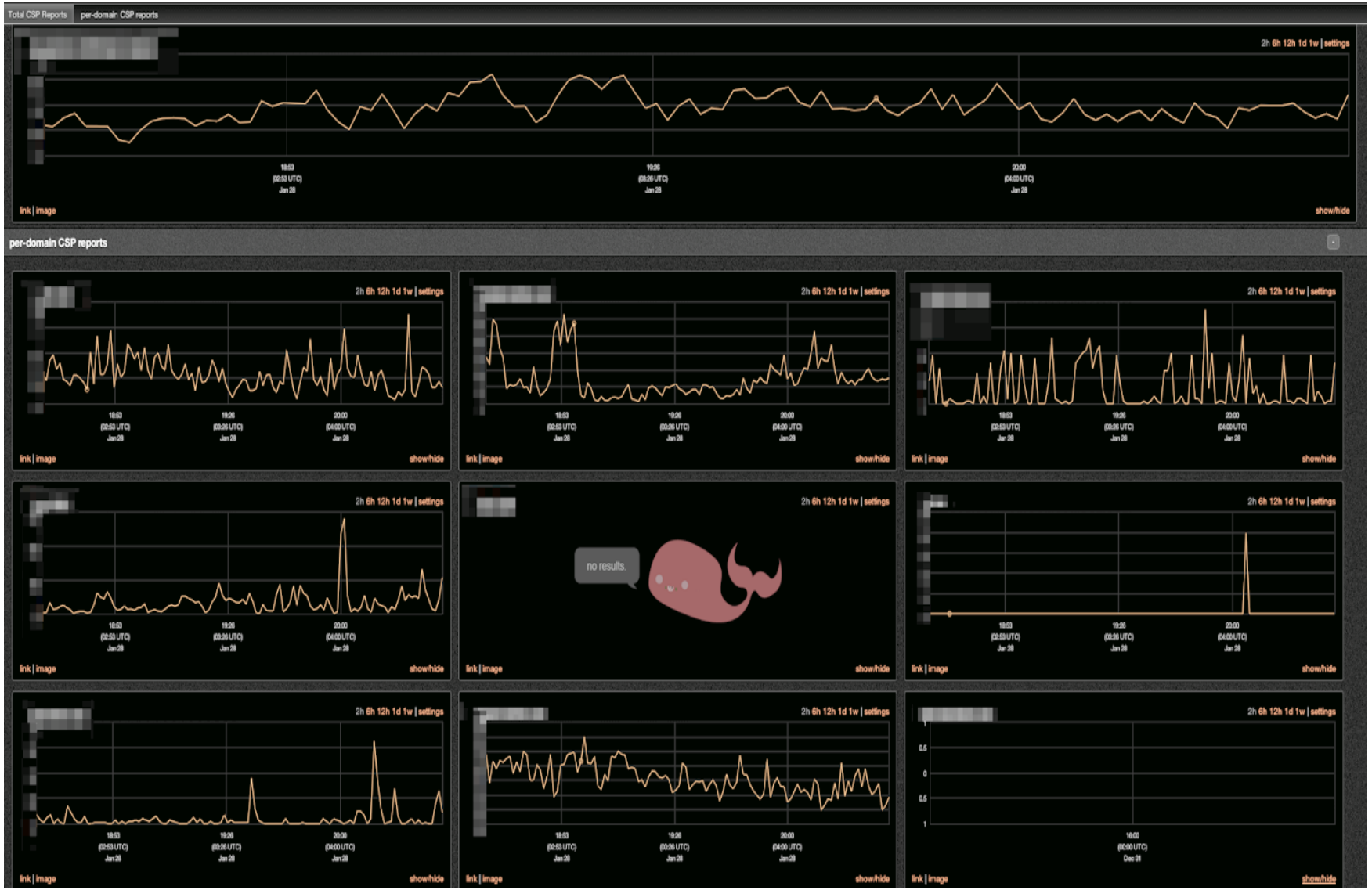
COE Drivers

COE Members

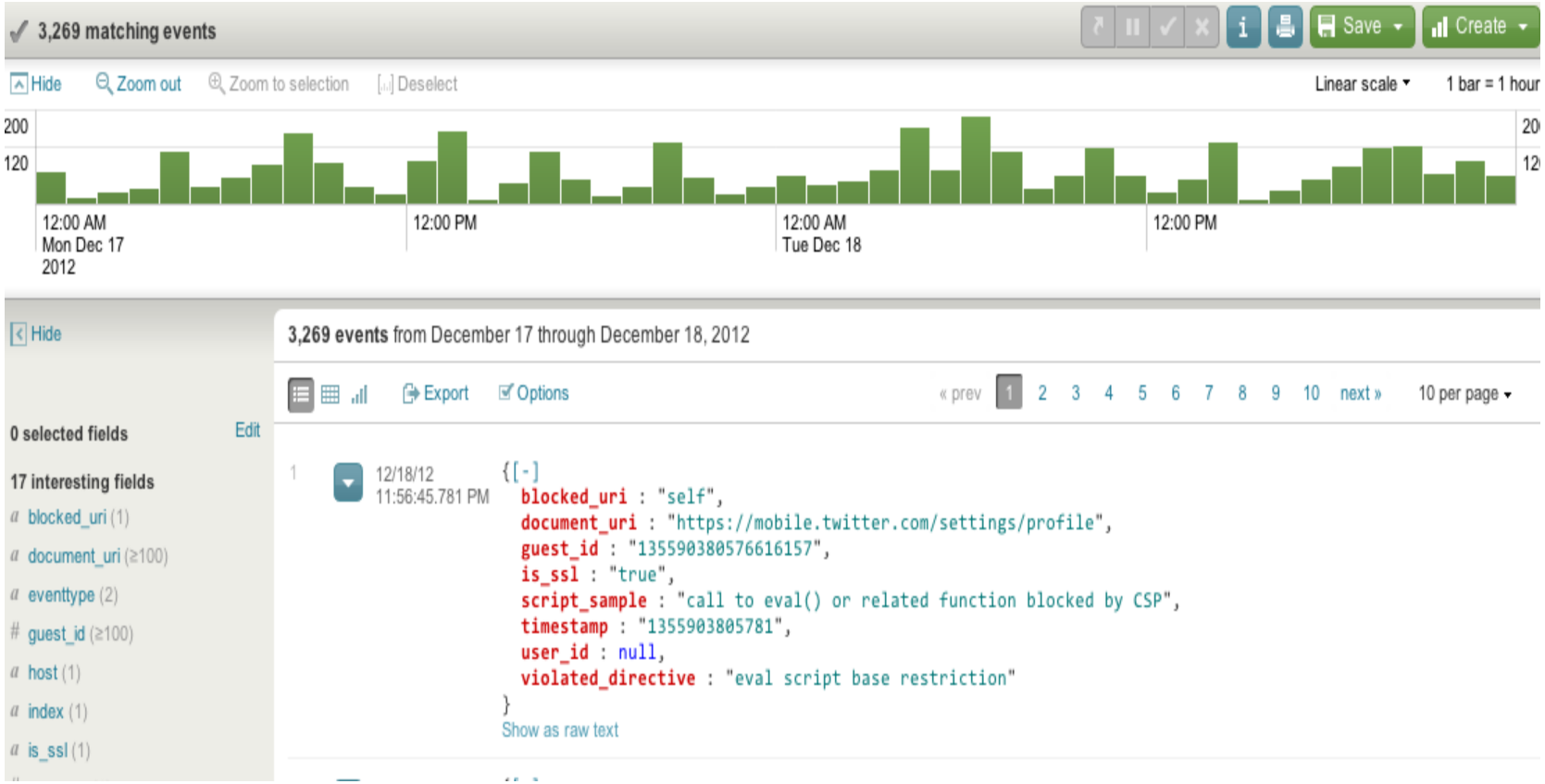
Remove barriers between departments

Positively impact change

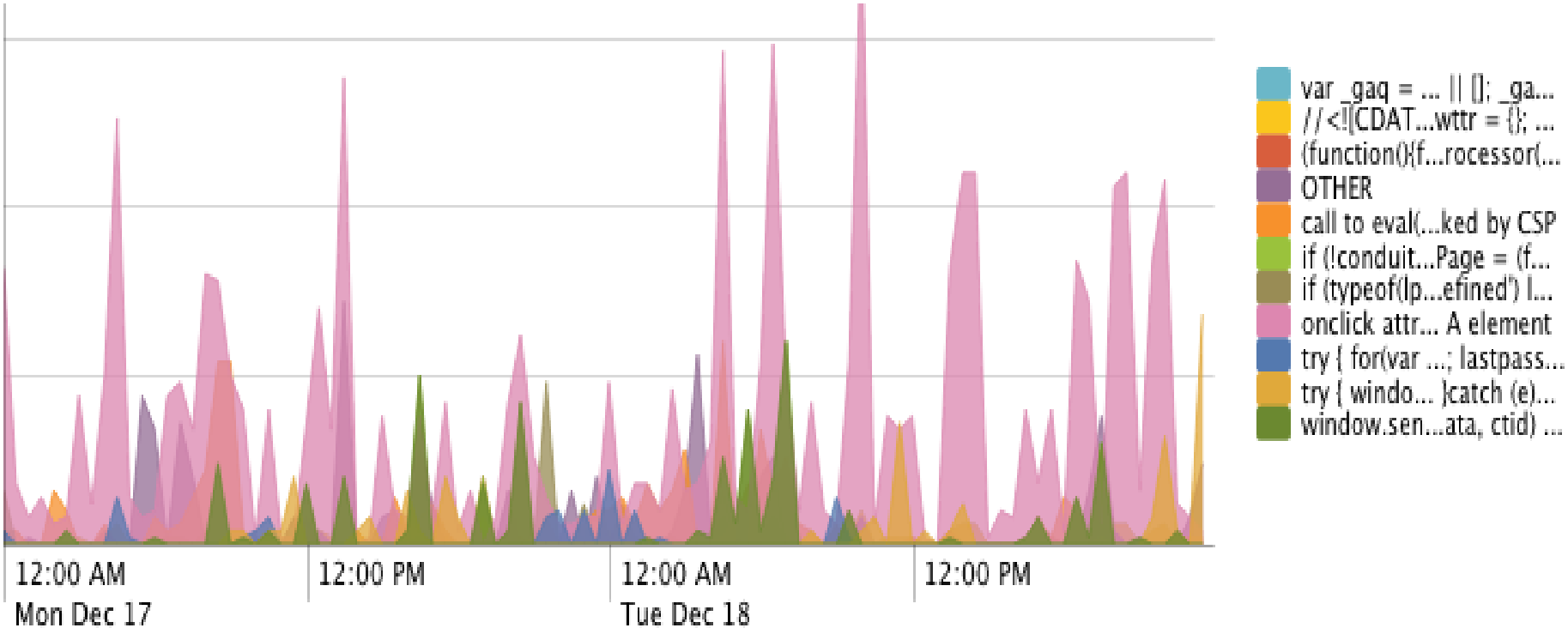
# Monitor and Tune ALL the things



# Splunk


3,269 matching events
Save
Create
Hide
Zoom out
Zoom to selection
Deselect
Linear scale
1 bar = 1 hour
200
120
12:00 AM
Mon Dec 17
2012
12:00 PM
12:00 AM
Tue Dec 18
12:00 PM
Hide
3,269 events from December 17 through December 18, 2012
Export
Options
« prev
1
2
3
4
5
6
7
8
9
10
next »
10 per page
0 selected fields
Edit
17 interesting fields
a blocked_uri (1)
a document_uri (≥100)
a eventtype (2)
# guest_id (≥100)
a host (1)
a index (1)
a is_ssl (1)
1
12/18/12
11:56:45.781 PM
{[-]
blocked_uri : "self",
document_uri : "https://mobile.twitter.com/settings/profile",
guest_id : "135590380576616157",
is_ssl : "true",
script_sample : "call to eval() or related function blocked by CSP",
timestamp : "1355903805781",
user_id : null,
violated_directive : "eval script base restriction"
}
Show as raw text

# Trending and anomalies



# Continuing Education

**Websites**

- The Open Web Application Security Project (http://www.owasp.org)
- OWASP SAMM (http://www.opensamm.org)

**Online Documents & Books**

- OWASP Code Review Guide & OWASP Testing Guide

# Continuing Education

**Tools**

- Burp suite (http://www.portswigger.net)
- HTTrack (http://www.httrack.com/)
- Webscarab(http://www.owasp.org/)

Conferences

- OWASP (http://www.owasp.org/conferences.html)

PodCasts

- OWASP (http://www.owasp.org/index.php/OWASP_Podcast)
- PaulDotCom (http://pauldotcom.com/podcast/)