



# Static & Dynamic Analysis for Web Applications

**Tony UcedaVelez**  
**OWASP Atlanta, Chapter Lead**  
**& Guest Panel:**

**Jeremiah Grossman (WhiteHat Security)**  
**Chris Eng (Veracode)**  
**Russell Spitler(Fortify)**  
**Matt Wood (HP)**

**OWASP**

Atlanta Chapter  
March 2010 Meeting

Copyright 2009 © The OWASP Foundation  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the OWASP License.

**The OWASP Foundation**  
<http://www.owasp.org>

# Meeting Agenda

- Membership Plea
- Chapter/ Org News
- Topic/ Speaker Introduction
- Panel Intro Slides
- Panel Demo(s)?
- Panel Q&A
- Post Chapter After Hours

**Ri Ra Atlanta**

1080 Peachtree St NE,

**Located at the Corner of Crescent and 12th,**

Atlanta, GA 30309

Phone: 404.477.1700

# Membership Plea

- \$50/ Individual Membership
  - supports Local Chapter
  - \$ used for provisions
  - gift bag/ shirt
  - Move a Speaker program money
  - Tools, labs, and more
- Tax Deductible
- Don't be free loaders!!!
- <http://www.owasp.org/index.php/Membership>



Please  
support your  
Atlanta  
Chapter....

**Sally Struthers – OWASP Member 2010**

# Chapter/ ORG News

## ❑ [Call for papers: OWASP AppSec USA 2010 CA Sept 7-10](#)

OWASP is currently soliciting papers and training proposals for the OWASP AppSec USA, California 2010

- Submission deadline is June 6th at 12PM PST (GMT -8)
- Submit Proposals to:

<http://www.easychair.org/conferences/?conf=appsec2010>

## ■ Opportunity to help the OWASP Development Guide:

- See next slide

## ■ Interested in Static Analysis w/ OWASP solution?

- [Owasp Orizon](#) is source code static analysis tool

- Started in 2006; Standalone tool written in Java

- review over your code making sure it fits recommendations contained into the Owasp Build Guide and the Owasp Code review Guide.



Assignment	Assignee(s)	Status
Role: Template/Worksheet Cop	<a href="#">John</a>	
Section: Front matter	<a href="#">Ken</a> (lead)	Initial draft
Section: Back matter	TBD	Work not started
Section: <a href="#">OWASP-0100 Security Architecture</a>	<a href="#">Mylene</a> (lead), <a href="#">Mike L.</a> , <a href="#">Smriti</a> , <a href="#">Kevin</a> , <a href="#">Charlie</a> , <a href="#">Phil</a>	Initial development
Section: <a href="#">OWASP-0200 Authentication</a>	<a href="#">J.R.</a> (lead), <a href="#">Junilu</a>	Initial development
Section: <a href="#">OWASP-0300 Session Management</a>	<a href="#">Anurag</a> (lead), <a href="#">Mehmet</a>	Initial development
Section: <a href="#">OWASP-0400 Access Control</a>	<a href="#">Vishal</a> (lead), <a href="#">Sandeep</a>	Initial development
Section: <a href="#">OWASP-0500 Input Validation</a>	<a href="#">Tom</a> (lead), <a href="#">Koen</a>	Initial development
Section: <a href="#">OWASP-0600 Output Encoding/Escaping</a>	<a href="#">Theo</a> (lead), Abe	Initial development
Section: <a href="#">OWASP-0700 Cryptography</a>	<a href="#">Anurag</a> (lead), <a href="#">Eric</a> , <a href="#">Kevin</a>	Initial development
Section: <a href="#">OWASP-0800 Error Handling and Logging</a>	<a href="#">Paco</a> (lead)	Initial development
Section: <a href="#">OWASP-0900 Data Protection</a>	<a href="#">Arthur</a> (lead)	Initial development
Section: <a href="#">OWASP-1000 Communication Security</a>	<a href="#">Ketan</a> (lead)	Initial development
Section: <a href="#">OWASP-1100 HTTP Security</a>	<a href="#">Jack</a> (lead), <a href="#">Akash</a>	Initial development
Section: <a href="#">OWASP-1200 Security Configuration</a>	<a href="#">Syed</a> (lead), <a href="#">Ritesh</a>	Initial development
Section: <a href="#">OWASP-1300 Malicious Code Search</a>	<a href="#">Pierre</a> (lead), <a href="#">Akash</a>	



# Feature: Static & Dynamic Analysis for Web Application Security

- What is Static Analysis?

- Review of source or object code without code execution

- Analysis to see how the code is designed to behave (methods, classes, functions, structure, etc)

- Analysis to see what properties are associated with software objects

- Benefit is to catch things prior to building

- What is Dynamic Analysis?

- Review of application behavior via code execution

- Benefits include being able to see what the code actually does

# Our Panel

## • Russ Spitler

Recently Russell has been acting as the Product Manager of the Fortify 360 Suite. During his tenure he has acted as advisor to more than 500 successful deployments of the software and is often a key reference in the design of software security initiatives. In his free time he enjoys skiing, riding motorcycles and drinking whiskey.

## •Jeremiah Grossman

Founder and chief technology officer of WhiteHat Security, is a world-renowned expert in web application security and a founding member of the Web Application Security Consortium (WASC). He is a frequent speaker at industry events including the BlackHat Briefings, ISACA's Networks Security Conference, NASA, ISSA and Defcon

## •Chris Eng

Senior Director of Research at Veracode, is responsible for integrating security expertise into Veracode's technology and helping to define and prioritize the security feature set of Veracode's service offerings. Chris has presented at security conferences such as the Black Hat Briefings and OWASP AppSec.

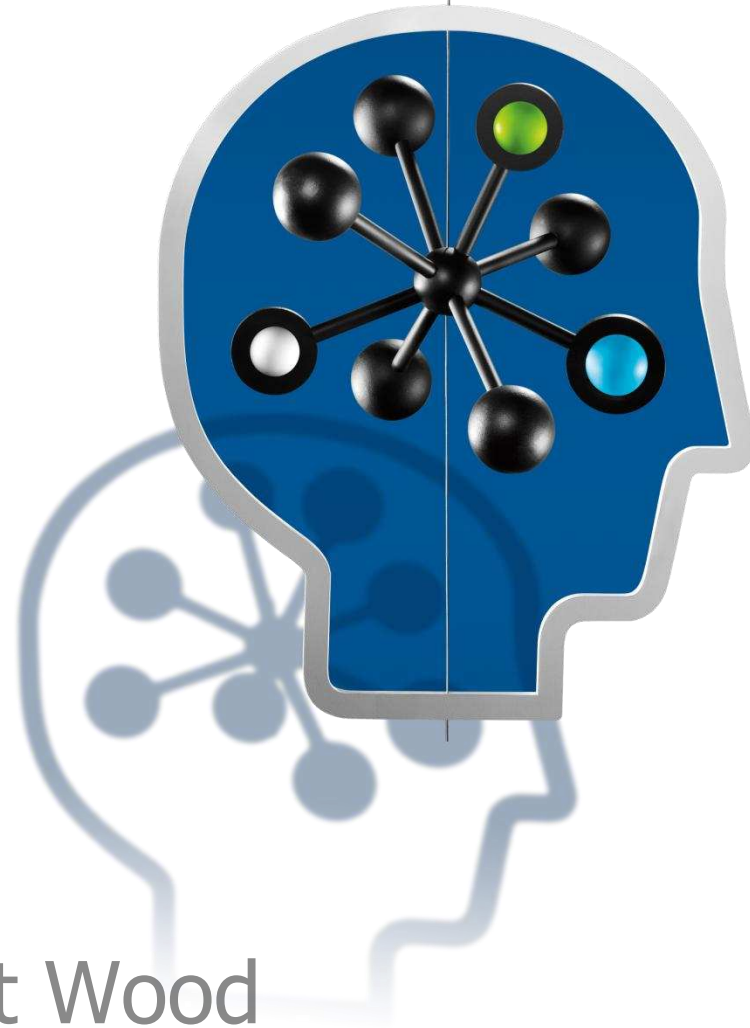
## •Matt Wood

Lead security researcher in HP's Web Security Research Group. Matt has led the development of both HP ScrawlR and HP SWFScan, which are free security tools designed to help organizations find SQL injection and Adobe Flash security vulnerabilities, respectively.





# OWASP Panel



Matt Wood

Lead Researcher – Web Security Research Group

HP Application Security Center



# •Web Security: The Real Challenges

- Web landscape has changed, tools must change
- Attack Surface is King
  - And how to exercise it correctly!
- Reproduce-ability and Root Cause
- Scalability
- Repeatability of Pen-Tests/Audits
- Get Rid of the Lip Service paid to Security

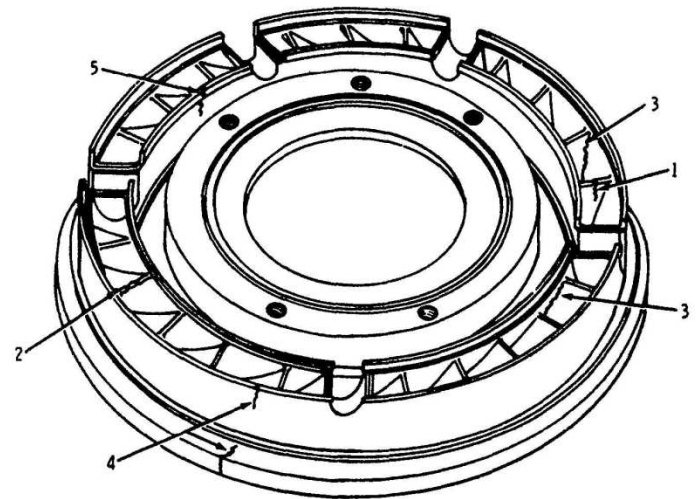
# • **Dynamic + Static = Sexy Hotness**

- Pen-testers/organizations are already doing both
- Root Cause / Runtime Analysis
  - See the line of code & http request to reach it
- Measureable Coverage
- Scan Steering
- Increase Value per Vuln

**VERACODE**

# Static Analysis

- Analysis of software performed without actually executing the program
- Full coverage of the entire source or binary
- In theory, having full application knowledge can reveal a wider range of bugs and vulnerabilities than the “trial and error” of dynamic analysis
- Impossible to identify vulnerabilities based on system configuration that exist only in the deployment environment



# Dynamic Analysis

- Analysis of software performed against a running instance of the program
- Most accurately mimics how a malicious user would attack the application
- Due to the lack of internal application knowledge, discovering vulnerabilities can take longer and coverage may be limited
- Cannot generate and test all possible inputs in reasonable time
- Exposes vulnerabilities in the deployment environment



# •Static vs. Dynamic vs. Manual Tradeoffs

Criteria	Static	Dynamic	Manual
Code Coverage	✓✓		
Ease of Testing	✓		
Noise		✓	✓✓
Ratio of Actionable Vulnerabilities		✓	✓✓
Cost	✓	✓	
Platform/Language Support		✓	✓
Ability to Test Non-Web Applications	✓		✓
Can be Performed in the Cloud	✓	✓	✓

# •Static/Dynamic: Why Correlate?

## Lets get the conversation started

- Goal
  - To reduce false positives
  - To improve test coverage
  - To verify exploitability of flaws
- Challenges
  - Mapping code to URL's
  - Garbage in, Garbage out
    - If your static analysis tool has a lot of false positives, correlation might be a really bad idea
- Questions
  - Is this really the best way to reduce false positives?
  - Do we need to verify exploitability?

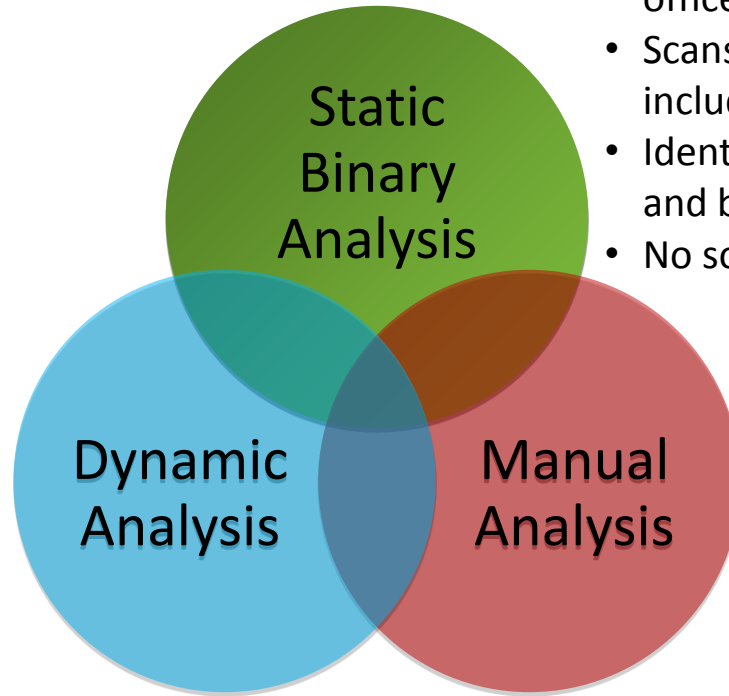


# The Veracode Solution

Good security requires more than one method

## Black Box Testing (Dynamic) “Outside In”

- Tests runtime behavior of application
- Helps secure business critical web applications
- Identifies exploitable security vulnerabilities
- No source code required



## White Box Testing (Static Binary) “Inside Out”

- Helps secure both web facing and back-office applications
- Scans 100% of the application, including 3<sup>rd</sup> party code
- Identifies security flaws, malicious code and backdoors
- No source code required

## Manual Testing

- Identifies design and business logic security risks
- Provides additional layer of security testing for mission-critical applications
- Leverages automation as a baseline to begin assessment
- Source code optional



## •Correlating Static and Dynamic Analysis Results

Jeremiah Grossman  
Founder and CTO  
WhiteHat Security

Russell Spitler  
Product Manager – Fortify 360  
Fortify Software

# •Overview

- Introduction
- Overview of WhiteHat dynamic analysis
- Overview of Fortify static analysis
- Benefits of a combined approach
- Case Study: Fortify *on Demand*
- Questions

# •Motivation

- Between 2005 – 2009 there were:
  - 2,064 reported data security breaches<sup>1</sup>
  - 470 million reported records compromised<sup>1</sup>
  - No industries immune: Finance, retail, government, military, technology, healthcare, telecom, energy, manufacturing, education
- Today, we rely increasingly on software:
  - 114 million active Web sites in the world<sup>2</sup>
  - 17 million software developers in the world <sup>3</sup>

1) <http://www.privacyrights.org/ar/ChronDataBreaches.htm>

2) <http://www.domaintools.com/internet-statistics/>

3) [http://www.fortbes.com/2008/04/03/usa-mobile-developer-tech-wire-cx\\_ew\\_0403ctia.html](http://www.fortbes.com/2008/04/03/usa-mobile-developer-tech-wire-cx_ew_0403ctia.html)

Trillions of lines of code

# •Security Encompasses Many Things



- Dynamic Analysis
- Static Analysis

# •Software Security

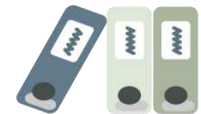
- Developed in-house
- Outsourced to third-parties
- Purchased from ISV (COTS)
- Licensed from open source community



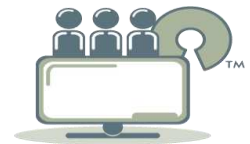
in-house



outsourced



commercial



open source

# •Primary Analysis Techniques

## Dynamic Analysis

- Also known as:
  - Web app scanning
  - Penetration testing
  - Black box testing
- Benefits
  - Quick and easy to get started
  - Simulates a hacker's point of view
- Drawbacks
  - Difficult to exercise the entire application
  - Lacks code-level details

## Static Analysis

- Also known as:
  - Source code analysis
  - Binary or byte-code analysis
- Benefits
  - 100 percent code coverage
  - Early in SDLC
- Drawbacks
  - Results require review

# •Deployment Options

## Software

- Benefits
  - Integrates into SDLC
  - Trains developers to write secure code
- Drawbacks
  - Time, expertise and resources

## Software-as-a-Service (SaaS)

- Benefits
  - Quick and easy to get started
  - Less expertise required
  - Fewer resources used
- Drawbacks
  - Not integrated into SDLC
  - Fails to reinforce security best practices in development



# Dynamic Analysis

# •Know Your Enemy

- Fully Targeted
  - Customize their own tools
  - Focused on business logic
  - Clever and profit driven (\$\$\$)
- Directed Opportunistic
  - Commercial / Open Source Tools
  - Authentication scans
  - Multi-step processes (forms)
- Random Opportunistic
  - Fully automated scripts
  - Unauthenticated scans
  - Targets chosen indiscriminately



# •WhiteHat Security Statistics Report

- 1,364 total websites
- 22,776 verified custom web application vulnerabilities
- Data collected from January 1, 2006 to October 1, 2009
- Vast majority of websites assessed for vulnerabilities weekly

## All Websites

- 83% of websites have had a HIGH, CRITICAL, or URGENT issue
- 64% of websites currently have a HIGH, CRITICAL, or URGENT issue
- 61% vulnerability resolution rate with 8,902 unresolved issues remaining
- Average # of HIGH, CRITICAL, or URGENT severity vulnerabilities per website during the vulnerability assessment lifetime: 16.7
- Average number of serious unresolved vulnerabilities per website: 6.5

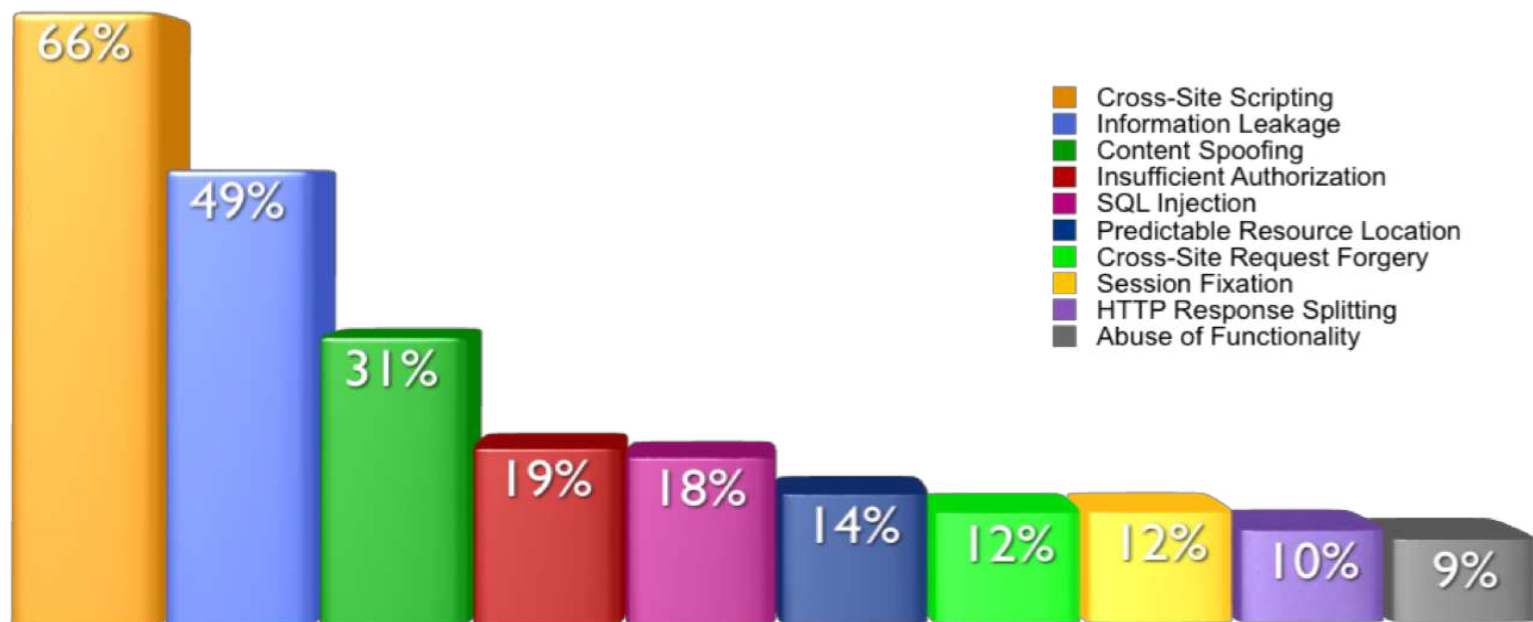
- \* Vulnerability severity naming convention aligns with PCI-DSS
- \* Vulnerabilities classified according to WASC Threat Classification

Percentage likelihood of a website having a vulnerability by severity

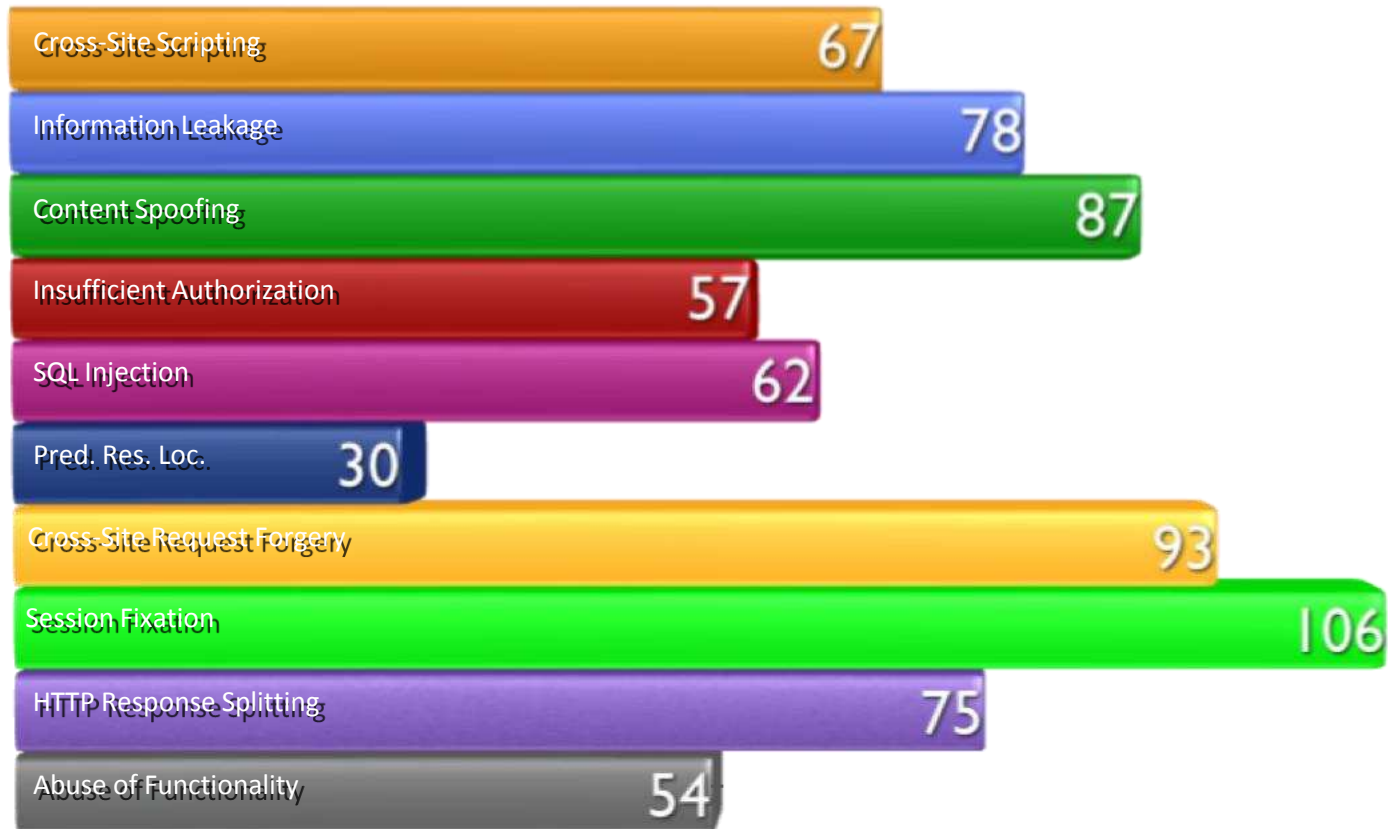


# •WhiteHat Security Top Ten

Percentage likelihood of a website having a vulnerability by class



# •Time-to-Fix



\* Up/down arrows indicate the increase or decrease since the last report.

Best-case scenario: Not all vulnerabilities have been fixed...

# •Resolution Rates

Class of Attack	% resolved	$\Delta$	severity
Cross Site Scripting	12%	8 ↓	urgent
Insufficient Authorization	18%	1 ↓	urgent
SQL Injection	40%	10 ↑	urgent
HTTP Response Splitting	12%	15 ↓	urgent
Directory Traversal	65%	12 ↑	urgent
Insufficient Authentication	37%	1 ↓	critical
Cross-Site Scripting	44%	5 ↑	critical
Abuse of Functionality	14%	14 ↓	critical
Cross-Site Request Forgery	39%	6 ↓	critical
Session Fixation	31%	10 ↑	critical
Brute Force	31%	20 ↑	high
Content Spoofing	46%	21 ↑	high
HTTP Response Splitting	32%	2 ↑	high
Information Leakage	30%	21 ↑	high
Predictable Resource Location	34%	8 ↑	high

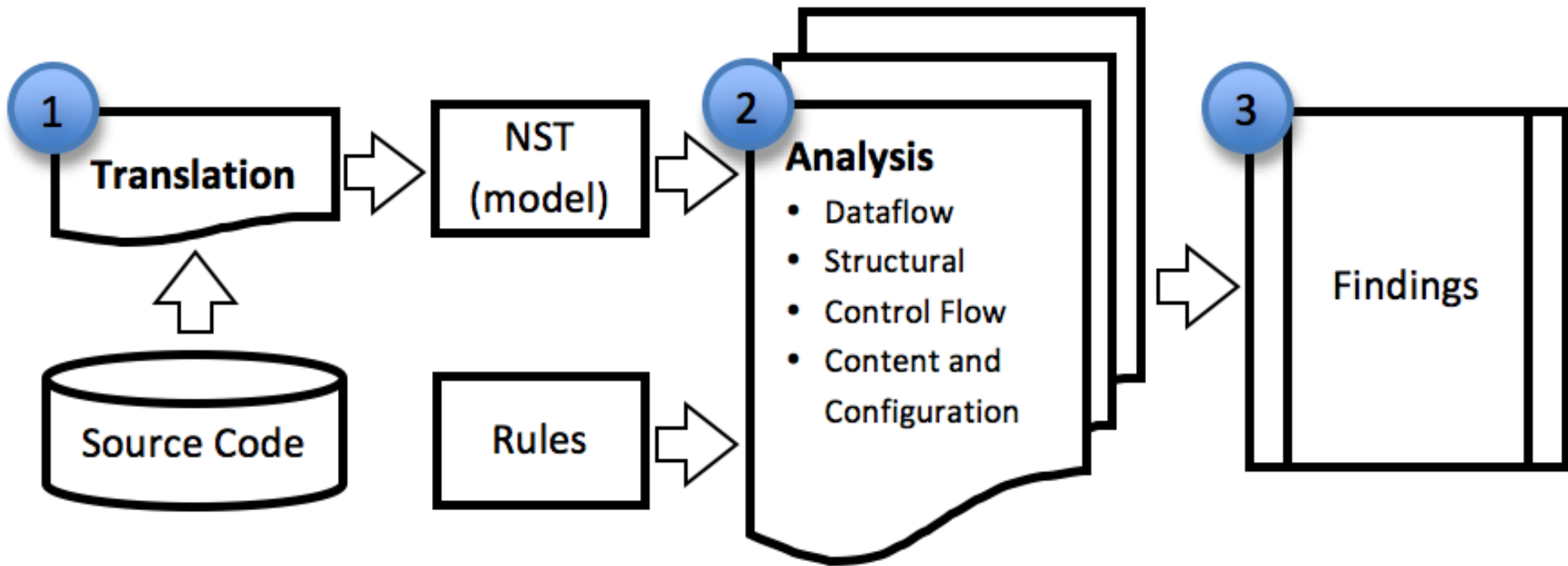
# •Dynamic Analysis Challenges

- Coverage
  - URLs
  - Parameters
- Remediation details
  - Code-level vulnerability details
  - Remediation guidance

# Static Analysis



# •Inside a Static Analysis Engine



1. Translate source code into intermediate model
2. Perform multiple types of analysis
3. Render results for human to review

# •Critical Attributes

- Language support
  - Understands the relevant languages/dialects
- Capacity
  - Ability to gulp down millions of lines of code
- Rule set and analysis algorithms
  - Right rules and techniques to find and prioritize issues
- Results management
  - Allow human to review results
  - Prioritization of issues

# •Why Static Analysis is Good for Security

- Fast compared to manual code review
- Analyze code without executing it
  - Able to contemplate many possibilities
  - Fast compared to testing
  - Complete, consistent coverage
- Integrates into development lifecycle
- Brings security knowledge with it
  - Makes review process easier for non-experts

# •Two Ways to Use the Tools

- #1 Analyze completed programs
  - Large number of results
  - Most people have to start here
  - Good motivator
- #2 Analyze as you write code
  - Run as part of build
  - Nightly/weekly/milestone
  - Fix as you go



# •Static Analysis Challenges

- Completed programs
  - Are not written with security in mind
  - Contain multiple paradigms and technologies
  - Exemplify varying developer skill and techniques
- Which causes static analysis to produce
  - Large numbers of issues
  - Widely varying issues
  - Issues that are difficult to triage
- Until Stage #2, prioritization is hugely important

# Prioritization

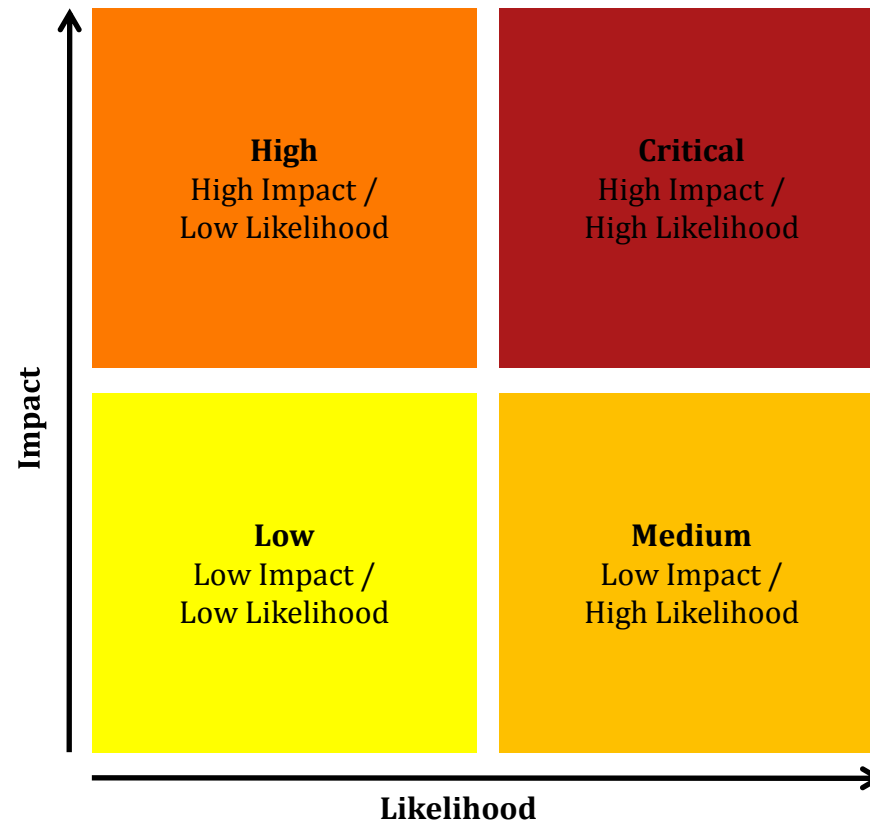
## •Prioritizing Analysis Results

$$\textit{risk} = \textit{impact} \cdot \textit{likelihood}$$

*Impact*: negative outcome resulting from a vulnerability

*Likelihood*: probability that the impact will come to pass

# •Axes Represent Risk



(Whitepaper *Prioritizing Static Analysis Results* at [www.fortify.com](http://www.fortify.com))



# •Fortify Priority Order

- **Critical** – Critical issues have high impact and high likelihood. Critical issues are easy to discover and exploit and result in large asset damage.
- **High** – High-priority issues have high impact and low likelihood. High-priority issues are often difficult to discover and exploit, but can result in large asset damage.
- **Medium** – Medium-priority issues have low impact and high likelihood. Medium-priority issues are easy to discover or exploit, but often result in small asset damage.
- **Low** – Low-priority issues have low impact and low likelihood. Low-priority issues can be difficult to discover and exploit and typically result in small asset damage.

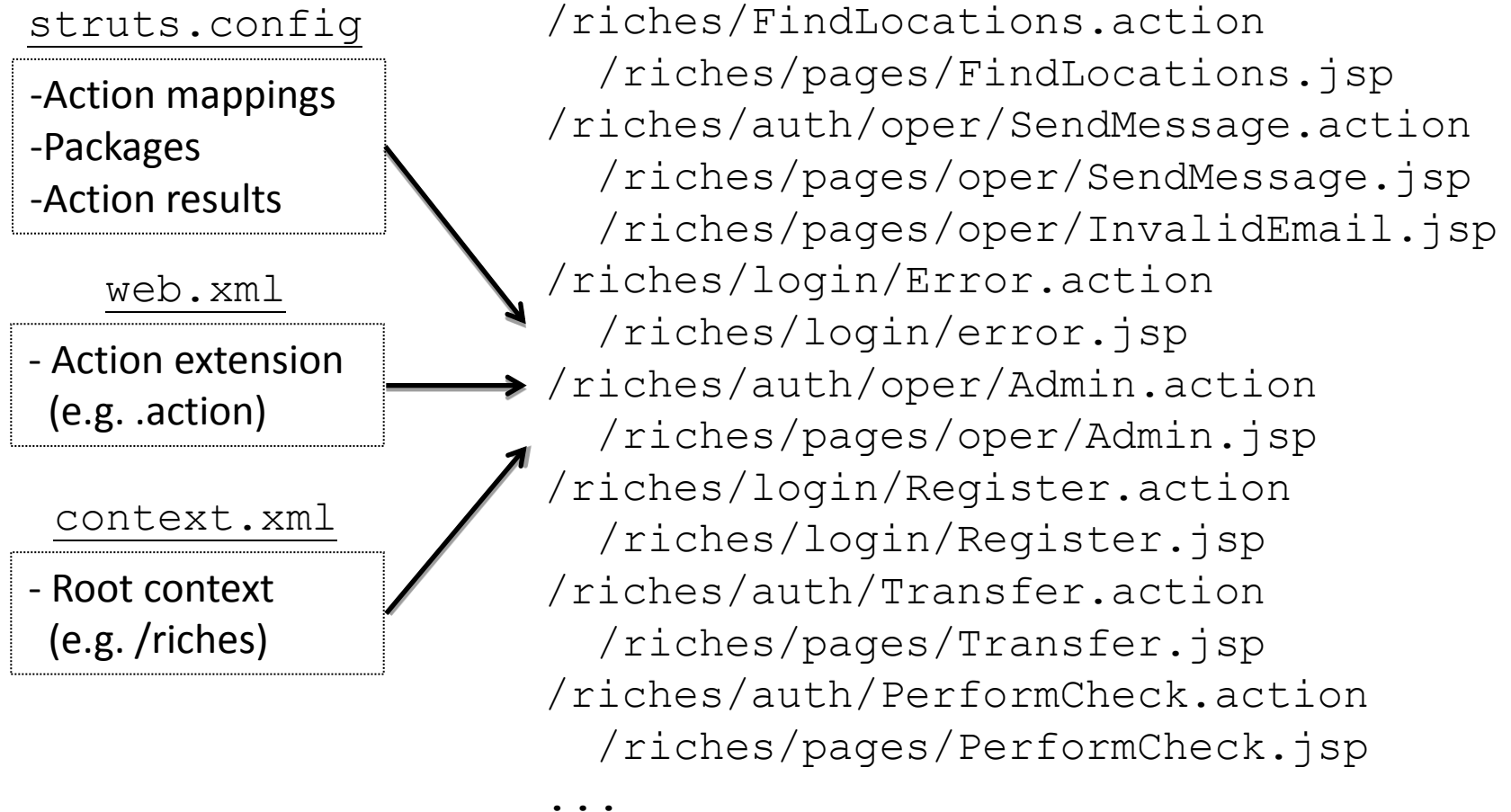
# Correlation

# •Goals

- Expanded dynamic coverage
  - Identify valid URLs
  - List parameters accessed under each URL
- Correlating static and dynamic results
  - Remediation details for dynamic issues
  - Prioritization of static issues
    - Equality
    - Existence
    - Proximity

# •Expanded Dynamic Coverage

## • List valid URLs



# •Expanded Dynamic Coverage

- List parameters for each URL

`/riches/FindLocations.action`

`/riches/auth/oper/SendMessage.action`  
`severity, subject, body, to`

`/riches/login/Error.action`

`/riches/auth/oper/Admin.action`  
`addresses, auth`

`/riches/login/Register.action`

`/riches/auth/Transfer.action`  
`accounts`

`/riches/auth/PerformCheck.action`

`addr, acct, account, memo, name, amount`

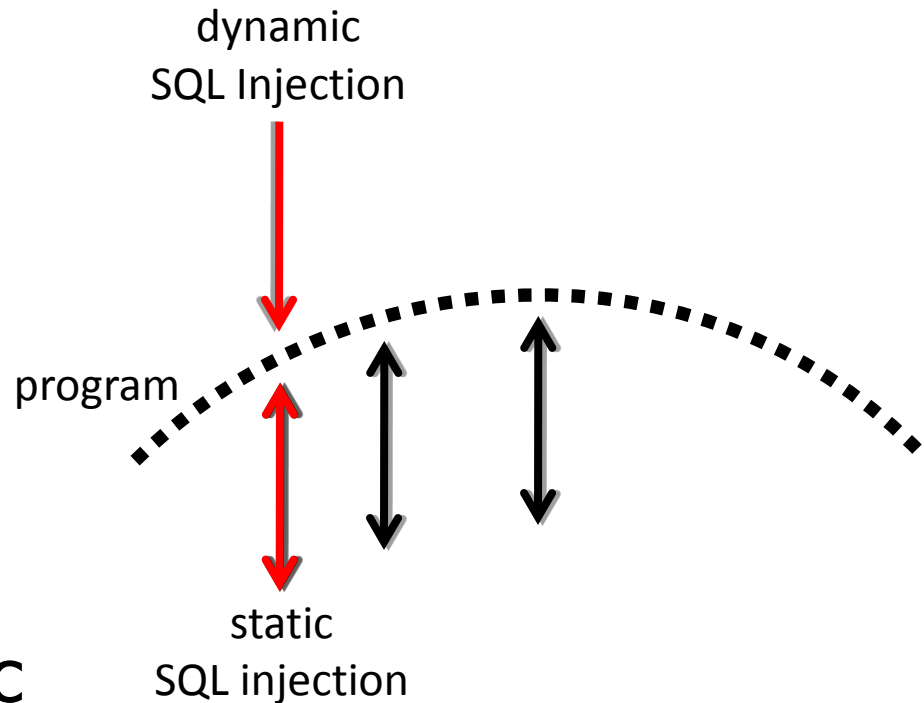
`/riches/ShowLocations.action`

`zip, state, address, type, locations, city`

`/riches/login/Login.action`

# •Correlation: Equality


- Find static and dynamic issues at same URL
- Remediation details for dynamic issues
- Improved prioritization for static issues



# •Remediation Details

SQL Injection			Remediation Effort: Minor	
URL: /riches/auth/AccountDetails.action				
Location	Method	Enc. Method/Params	Priority	Analyzer
AccountDetails.action	GET	acctno= [whs 'check]	Critical	Dynamic

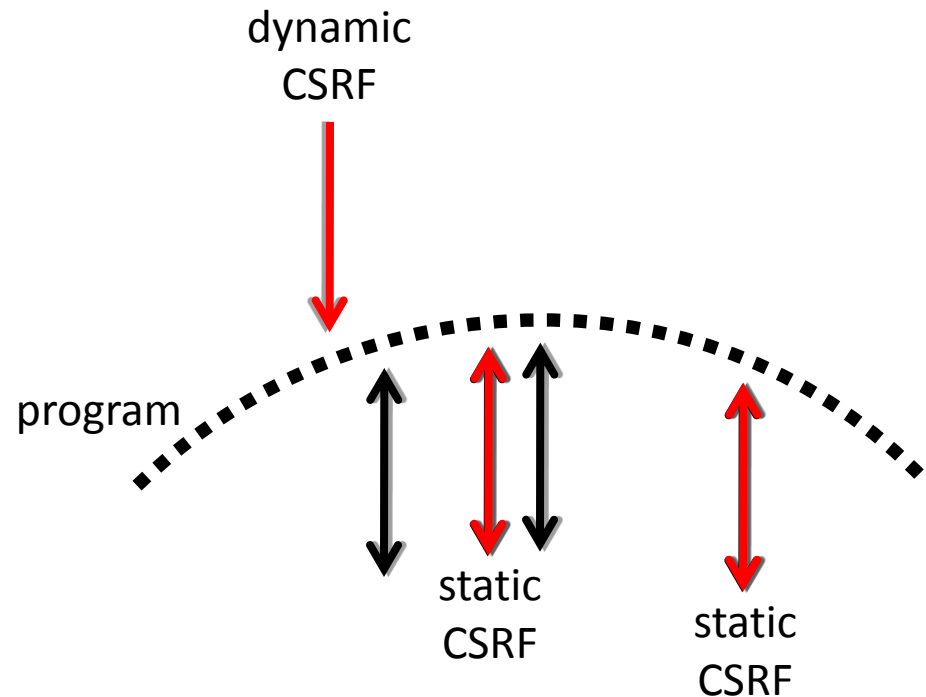
Analysis Evidence



```
<> rsa:0 - Action URL - /riches/auth/AccountDetails.action
-----
➡() AccountDetails.java:56 - setAcctno(0)
Ⓜ AccountDetails.java:57 - Assignment to this.acctno
⚡ AccountDetails.java:40 - Read this.acctno
➡() AccountDetails.java:40 - getTransactions(0)
:= TransactionService.java:17 - Assignment to queryStr
➡() TransactionService.java:19 - createQuery(0)
```

# •Correlation: Existence

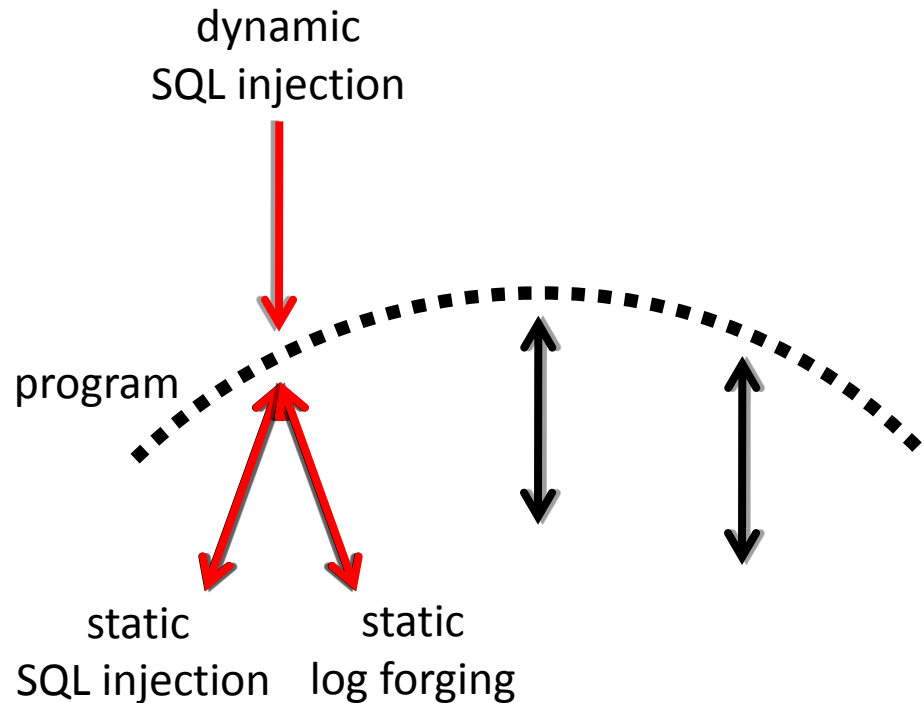
- Find dynamic Session Fixation, CSRF, ... issues
- Prioritize static issues in same category





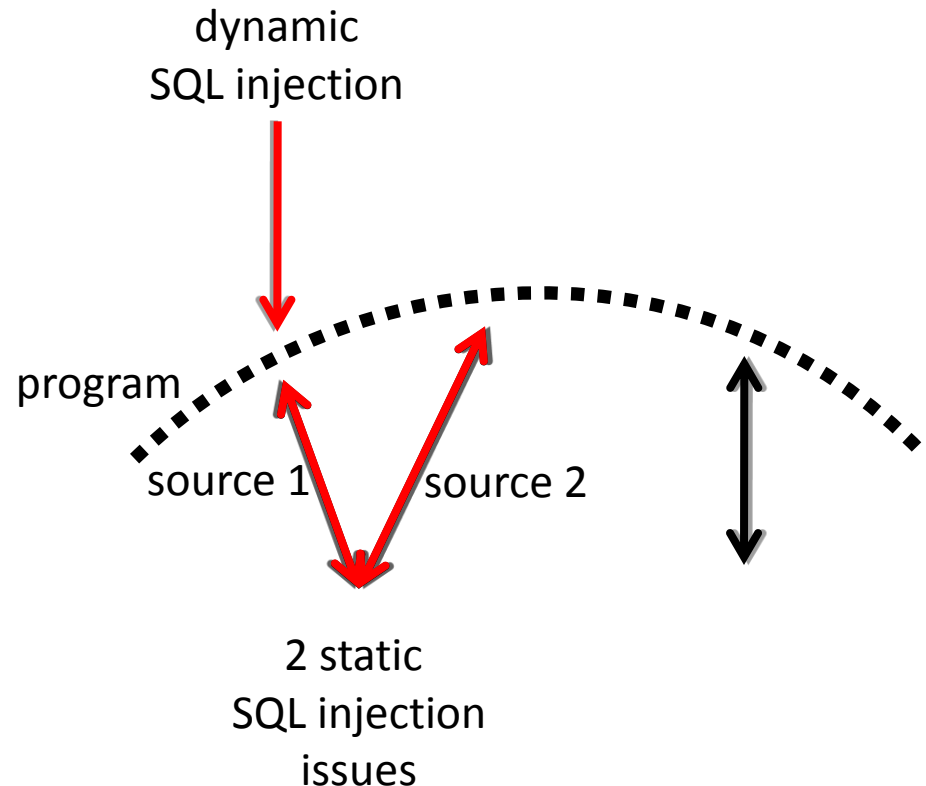
# •Correlation: Proximity (source)

- Find dynamic SQL Injection
- Prioritize static issues with same source



# •Correlation: Proximity (sink)

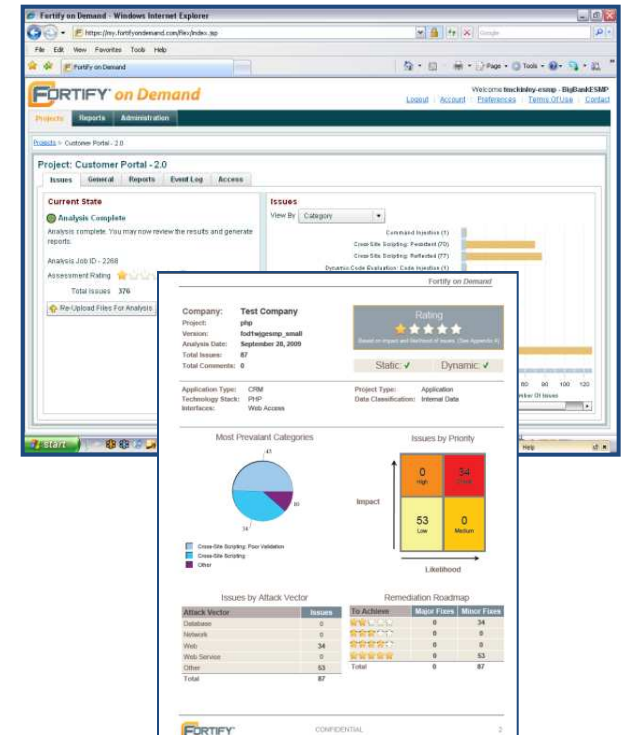
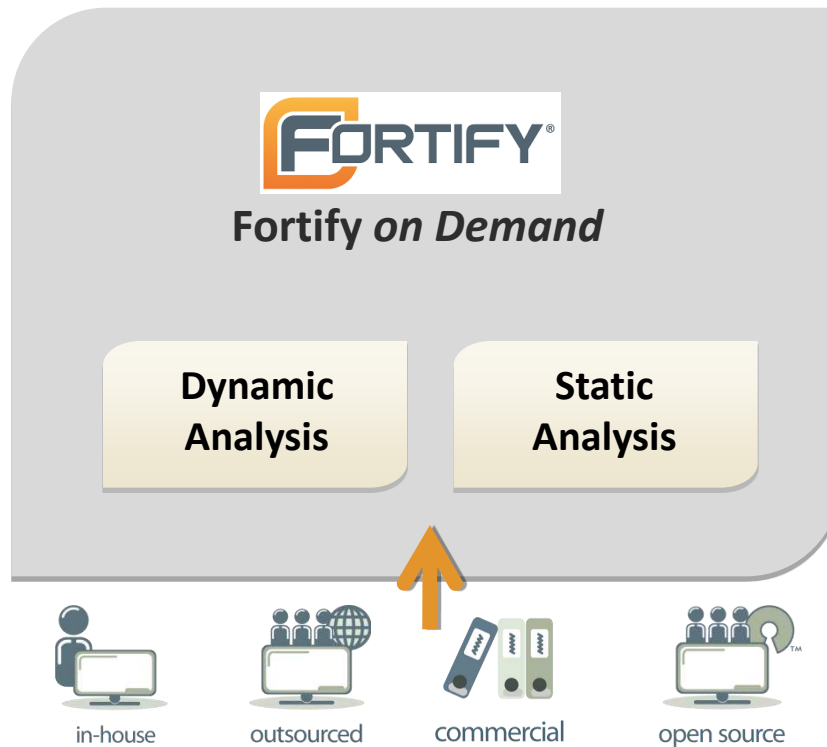
- Find dynamic SQL Injection, XSS, ... issues at URL
- Prioritize static issues in same *category* and file



# Case Study: *Fortify on Demand*

# •Fortify on Demand

## SaaS-based Software Security Testing



# •Riches Wealth Online (RWO)

[Find Locations](#) | [Contact Us](#) | [Site Map](#)



*Your Cornerstone of Stability and Growth*

[Personal](#) | [Small Business](#) | [Commercial](#) | [About RWi](#)

### RWi Online Banking

Username:

Password:

Need to set up online access?  
[Sign Up Now](#) or [Take a Tour](#)

### Account Services

Visit our Tax Center  
Get Mobile Banking  
Try Online Statements  
[>> More](#)

### Find ATMs/Locations

Enter Zip code or City & State

### Fraud Prevention & Online Security

Report Suspicious Email  
Fraud & Identity Theft  
RWi Security Plus™  
Our Online Security Guarantee



## House Hunting?

Find out how much you may  
you may be able to borrow  
with our  
**free** prequalification!



[Learn More...](#)

Banking	Loans	Investing
Checking Savings & CDs Credit Cards Online Banking Bill Pay <a href="#">&gt;&gt; More</a>	Home Mortgage Home Equity Loans Personal Loans Auto Loans Student Loans <a href="#">&gt;&gt; More</a>	Mutual Funds Brokerage Retirement Planning Insurance Private Banking <a href="#">&gt;&gt; More</a>
Open an Account	Check on Today's Rates	Other Services
Checking Account Savings Account Credit Cards CD's Money Market Account <a href="#">&gt;&gt; More</a>	Mortgage Home Equity Credit Cards Personal Loans Auto Loans <a href="#">&gt;&gt; More</a>	Retirement Center Buying a Home College Planning Consolidating Debt Investment Tools <a href="#">&gt;&gt; More</a>

### The RWi Gift Card

A gift every one will enjoy!




### Start Saving for Your Retirement Now!



[>> Learn More](#)

### Thinking about a college plan?



[Learn More](#)

[About RWi](#) | [Careers](#) | [Privacy, Security & Legal](#) | [Report Email Fraud](#) | [Diversity & Accessibility](#)  
[Important Notice on Trading in Fast Markets](#) | [Online Access Agreement \(3/06/2008\)](#) | [Sitemap](#)



# •Static Analysis of RWO

- RWO produces 64 high-impact static issues
  - 26 critical-priority issues (high likelihood)
  - 38 high-priority issues (low likelihood)
- Mapped 21 static issues to URLs
  - 33% of high impact issues
  - 73% of high impact issues that involve web input
- Remaining 43 aren't surprising
  - 14 resource leaks in model code
  - 6 unsafe configuration values
  - 23 "other issues", including database and file system inputs

# •Correlation on RWO

Category	Static	Dynamic	Correlated	Prioritized
SQL Injection	7	2	<b>5</b>	0
Cross-Site Scripting: Reflected	4	3	<b>1</b>	0
Cross-Site Request Forgery	11	3	<b>3</b>	<b>11</b>
Log Forging	2	0	0	<b>1</b>
Session Fixation	1	1	0	<b>1</b>
Total	25	9	<b>9</b>	<b>13</b>

# Conclusion



## •Apply

- Use static analysis to assess and improve completeness of dynamic tests
- Use dynamic analysis to narrow down static analysis results to those that are exploitable
- Don't stop there – use the combined view of the program under test to better inform auditing and remediation activities (existence and proximity)

# **•Q & A OPEN FORUM**

# BEER SOCIAL