DENIM GROUP

the leading secure software development firm

Managing Your Application Security Program with the ThreadFix Ecosystem

Dan Cornell
@danielcornell

This presentation contains information about DHS-funded research:
Topic Number: H-SB013.1-002 - Hybrid Analysis Mapping (HAM)
Proposal Number: HSHQDC-13-R-00009-H-SB013.1-002-0003-I

My Background

- Dan Cornell, founder and CTO of Denim Group
- Software developer by background (Java, .NET, etc)
- OWASP San Antonio

Denim Group Background

- Secure software services and products company
  - *Builds secure software*
  - *Helps organizations assess and mitigate risk of in-house developed and third party software*
  - *Provides classroom training and e-Learning so clients can build software securely*
- Software-centric view of application security
  - *Application security experts are practicing developers*
  - *Development pedigree translates to rapport with development managers*
  - ***Business impact: shorter time-to-fix application vulnerabilities***
- Culture of application security innovation and contribution
  - *Develops open source tools to help clients mature their software security programs*
    - *Remediation Resource Center, ThreadFix*
  - *OWASP national leaders & regular speakers at RSA, SANS, OWASP, ISSA, CSI*
  - *World class alliance partners accelerate innovation to solve client problems*

Application Vulnerability Management

- Application security teams uses automated static and dynamic test results as well as manual testing results to assess the security of an application
- Each test delivers results in different formats
- Different test platforms describe same flaws differently, creating duplicates
- Security teams end up using spreadsheets to keep track manually
- It is extremely difficult to prioritize the severity of flaws as a result
- Software development teams receive unmanageable reports and only a small portion of the flaws get fixed

The Result

- Application vulnerabilities persist in applications:

  ***Average serious vulnerabilities found per website per year is 79*

  ***Average days website exposed to one serious vulnerability is 231 days*

  ***Overall percentage of serious vulnerabilities that are fixed annually is only 63%*

- Part of that problem is there is no easy way for the security team and application development teams to work together on these issues

- Remediation quickly becomes an overwhelming project

- Trending reports that track the number of reduced vulnerabilities are impossible to create

****WhiteHat Statistics Report (Summer 2012):**
https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf

Vulnerability Fun Facts:

Industry	Annual Avg. Vulnerabilities	Avg. Time-to-Fix (Days)	Average Remediation	Window of Exposure (Days)
ALL	79	38	63%	231
Banking	17	45	74%	185
Education	53	30	46%	261
Financial Services	67	80	63%	227
Healthcare	48	35	63%	239
Insurance	92	40	58%	211
IT	85	35	57%	208
Manufacturing	30	17	50%	252
Retail	121	27	66%	238
Social Networking	31	41	62%	264
Telecom	52	50	69%	271
Non-Profit	37	94	56%	320
Energy	31	4	40%	250

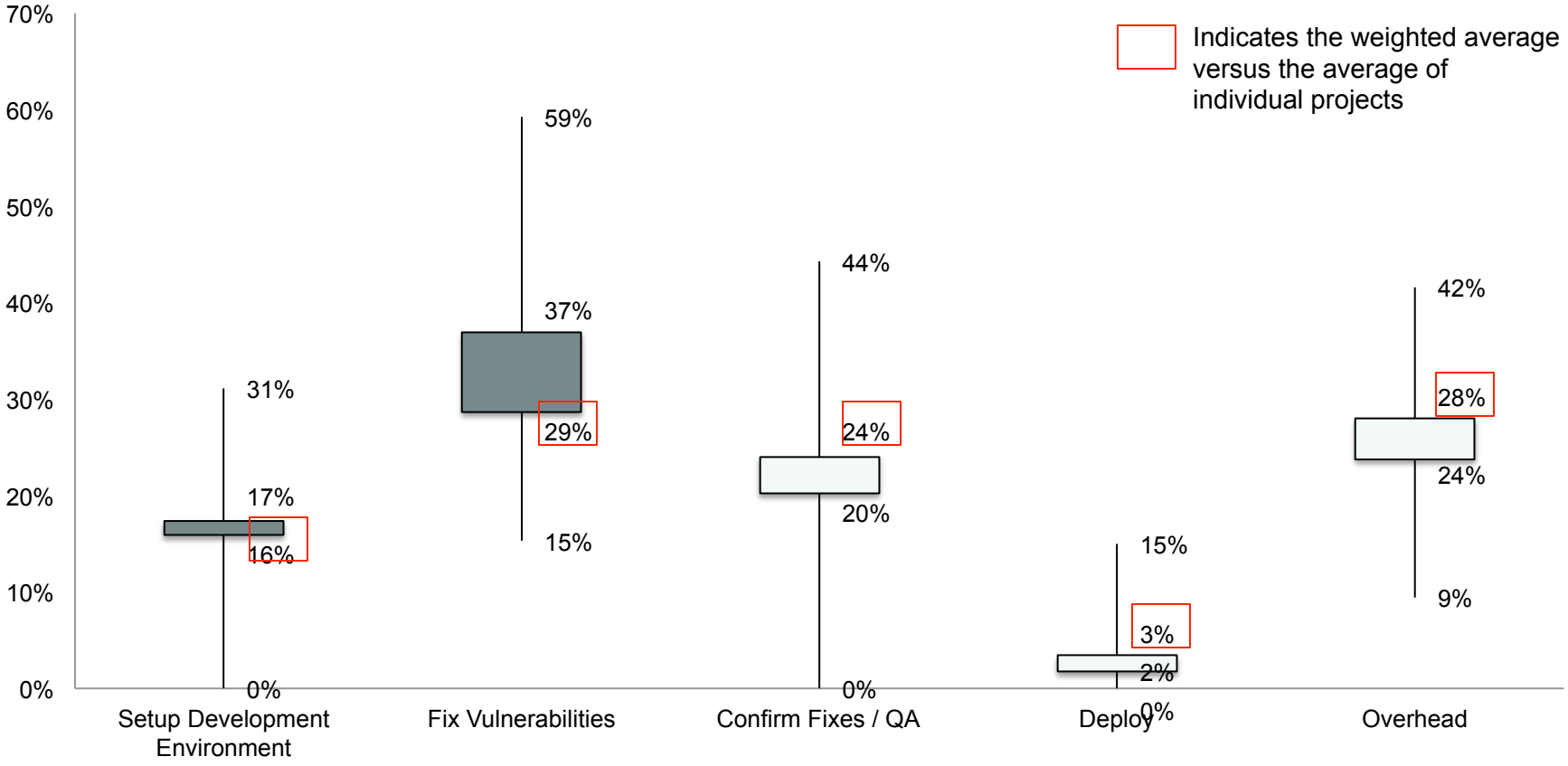
- Average number of serious vulnerabilities found per website per year is 79 **
- Serious Vulnerabilities were fixed in ~38 days **
- Percentage of serious vulnerabilities fixed annually is only 63% **
- Average number of days a website is exposed, at least one serious vulnerability ~231 days

WhiteHat Statistics Report (Summer 2012):
https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf

Vulnerability Remediation Data

Vulnerability Type	Sample Count	Average Fix (minutes)
Dead Code (unused methods)	465	2.6
Poor logging: system output stream	83	2.9
Poor Error Handling: Empty catch block	180	6.8
Lack of Authorization check	61	6.9
Unsafe threading	301	8.5
ASP.NET non-serializable object in session	42	9.3
XSS (stored)	1023	9.6
Null Dereference	157	10.2
Missing Null Check	46	15.7
XSS (reflected)	25	16.2
Redundant null check	21	17.1
SQL injection	30	97.5

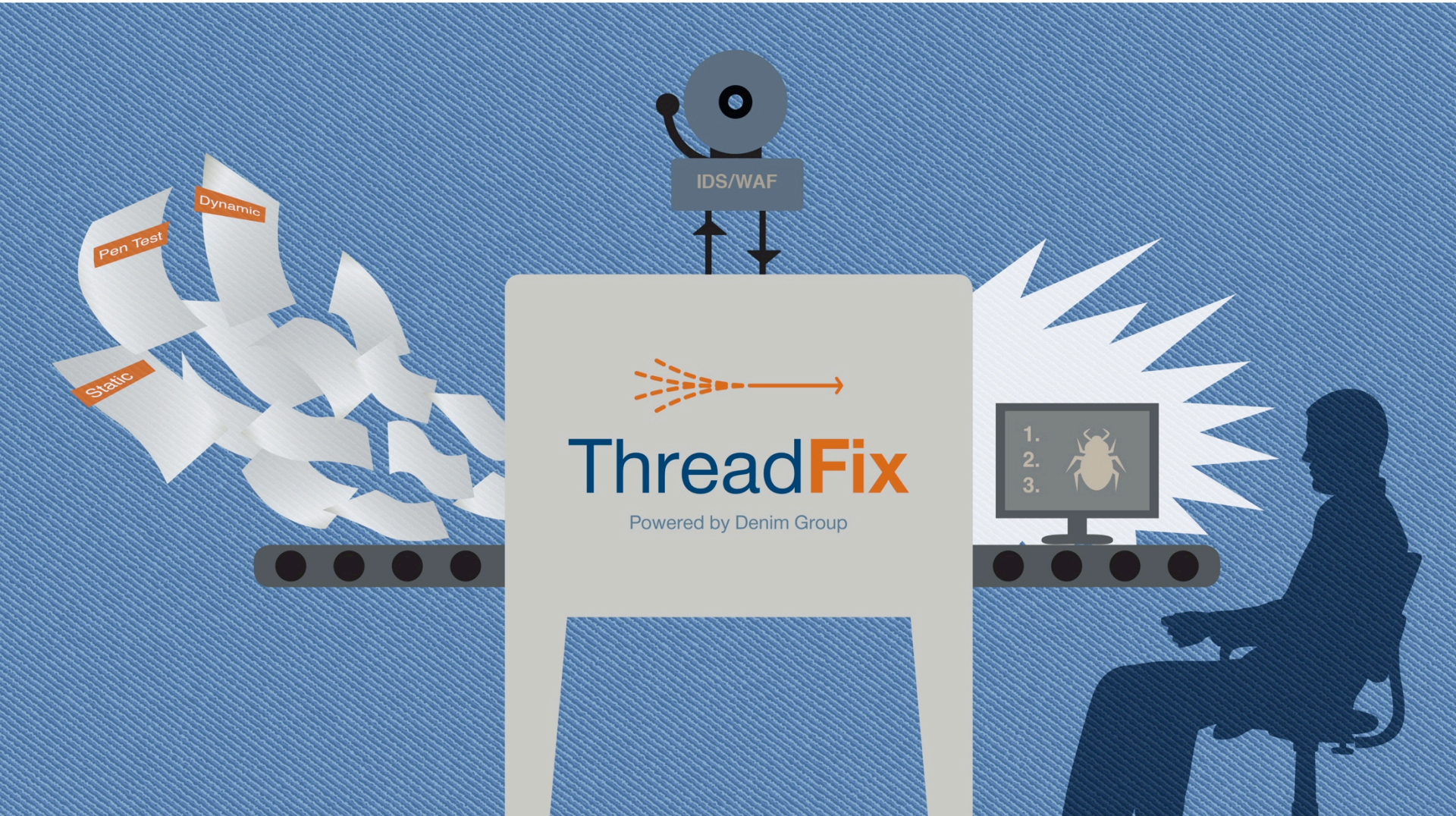
Where Is Time Being Spent?



ThreadFix

Accelerate Software Remediation

ThreadFix is a software vulnerability aggregation and management system that helps organizations aggregate vulnerability data, generate virtual patches, and interact with software defect tracking systems.



- **Open source vulnerability management and aggregation platform:**
  - *Allows software security teams to reduce the time to remediate software vulnerabilities*
  - *Enables managers to speak intelligently about the status / trends of software security within their organization.*
- **Features/Benefits:**
  - *Imports dynamic, static and manual testing results into a centralized platform*
  - *Removes duplicate findings across testing platforms to provide a prioritized list of security faults*
  - *Eases communication across development, security and QA teams*
  - *Exports prioritized list into defect tracker of choice to streamline software remediation efforts*
  - *Auto generates web application firewall rules to protect data during vulnerability remediation*
  - *Empowers managers with vulnerability trending reports to pinpoint issues and illustrate application security progress*
  - *Benchmark security practice improvement against industry standards*
- Freely available under the Mozilla Public License (MPL) 2.0
- Download available at: www.denimgroup.com/threadfix

List of Supported Tools / Technologies:

Dynamic Scanners
Acunetix
Arachni
Burp Suite
HP WebInspect
IBM Security AppScan Standard
IBM Security AppScan Enterprise
Mavituna Security Netsparker
NTO Spider
OWASP Zed Attack Proxy
Tenable Nessus
Skipfish
w3aF

Static Scanners
FindBugs
IBM Security AppScan Source
HP Fortify SCA
Microsoft CAT.NET
Brakeman

SaaS Testing Platforms
WhiteHat
Veracode
QualysGuard WAS

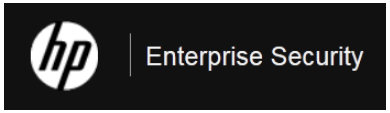
IDS/IPS and WAF
DenyAll
F5
Imperva
Mod_Security
Snort

ThreadFix
Powered by Denim Group

Defect Trackers
Atlassian JIRA
Microsoft Team Foundation Server
Mozilla Bugzilla

Known Vulnerable Component Scanner
Dependency Check

Large Range of Tool Compatibility



What is a Unique Vulnerability?

- (CWE, Relative URL)
  - *Predictable resource location*
  - *Directory listing misconfiguration*
- (CWE, Relative URL, Injection Point)
  - *SQL injection*
  - *Cross-site Scripting (XSS)*
- Injection points
  - *Parameters – GET/POST*
  - *Cookies*
  - *Other headers*

Why Common Weakness Enumeration (CWE)?

- Every tool has their own “spin” on naming vulnerabilities
- OWASP Top 10 / WASC 24 are helpful but not comprehensive

- CWE is exhaustive (though a bit sprawling at times)
- Reasonably well-adopted standard
- Many tools have mappings to CWE for their results

- Main site: http://cwe.mitre.org/

What Can We Do With ThreadFix?

- Create a consolidated view of your applications and vulnerabilities
- Prioritize application risk decisions based on data
- Translate vulnerabilities to developers in the tools they are already using

Create a consolidated view of your applications and vulnerabilities

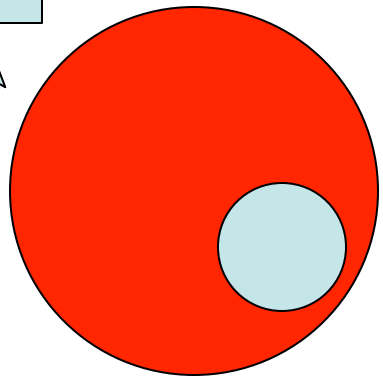
What Is Your Software Attack Surface?

What?
- Critical legacy systems
- Notable web applications

Why?
- Lots of value flows through it
- Auditors hassle you about it
- Formal SLAs with customers mention it
- Bad guys found it and caused an incident (oops)

What Is Your Software Attack Surface?

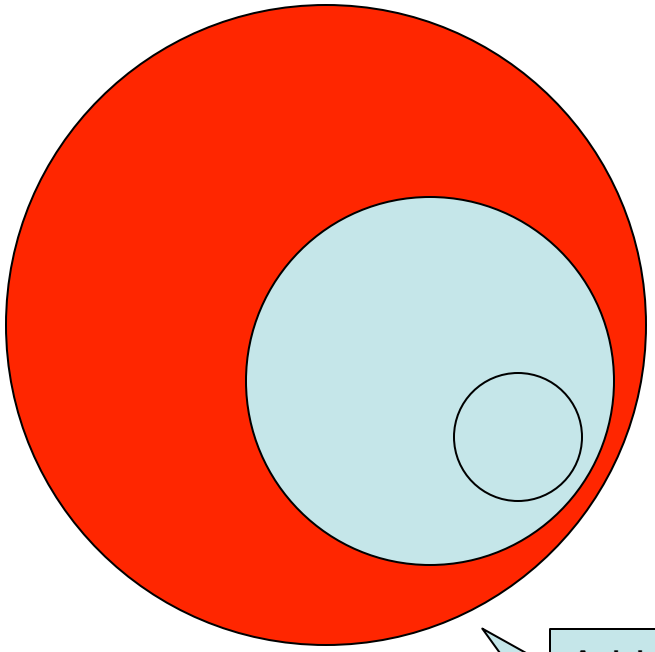
Add In the Rest of the Web Applications You Actually Develop and Maintain



What?
- Line of business applications
- Event-specific applications

Why Did You Miss Them?
- Forgot it was there
- Line of business procured through non-standard channels
- Picked it up through a merger / acquisition

What Is Your Software Attack Surface?

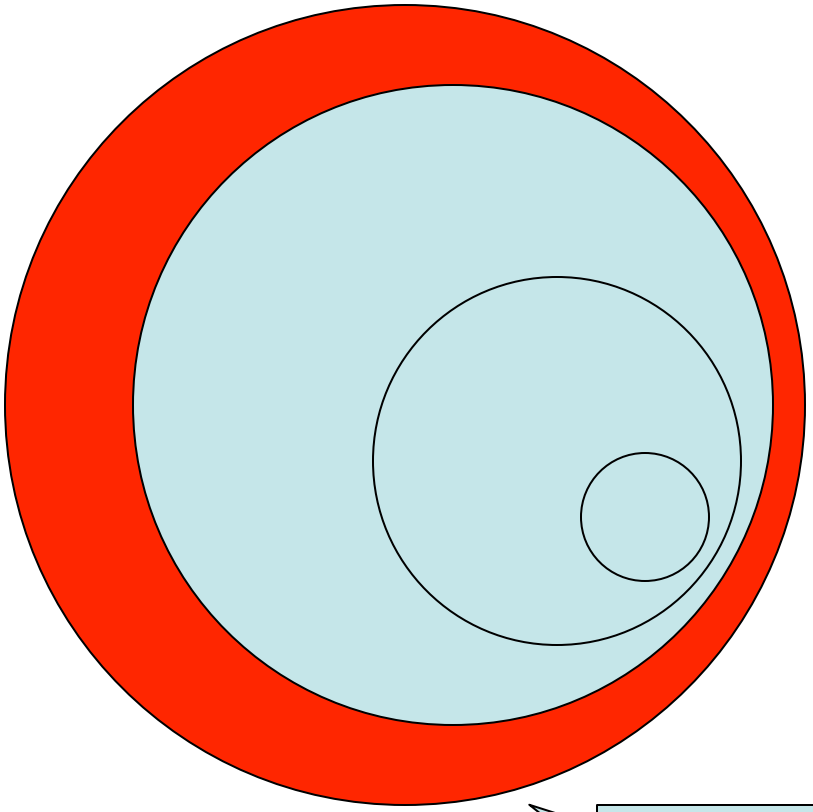


Add In the Software You Bought from Somewhere

What?
- More line of business applications
- Support applications
- Infrastructure applications

Why Did You Miss Them?
- Most scanner only really work on web applications so no vendors pester you about your non-web applications
- Assume the application vendor is handling security

What Is Your Software Attack Surface?

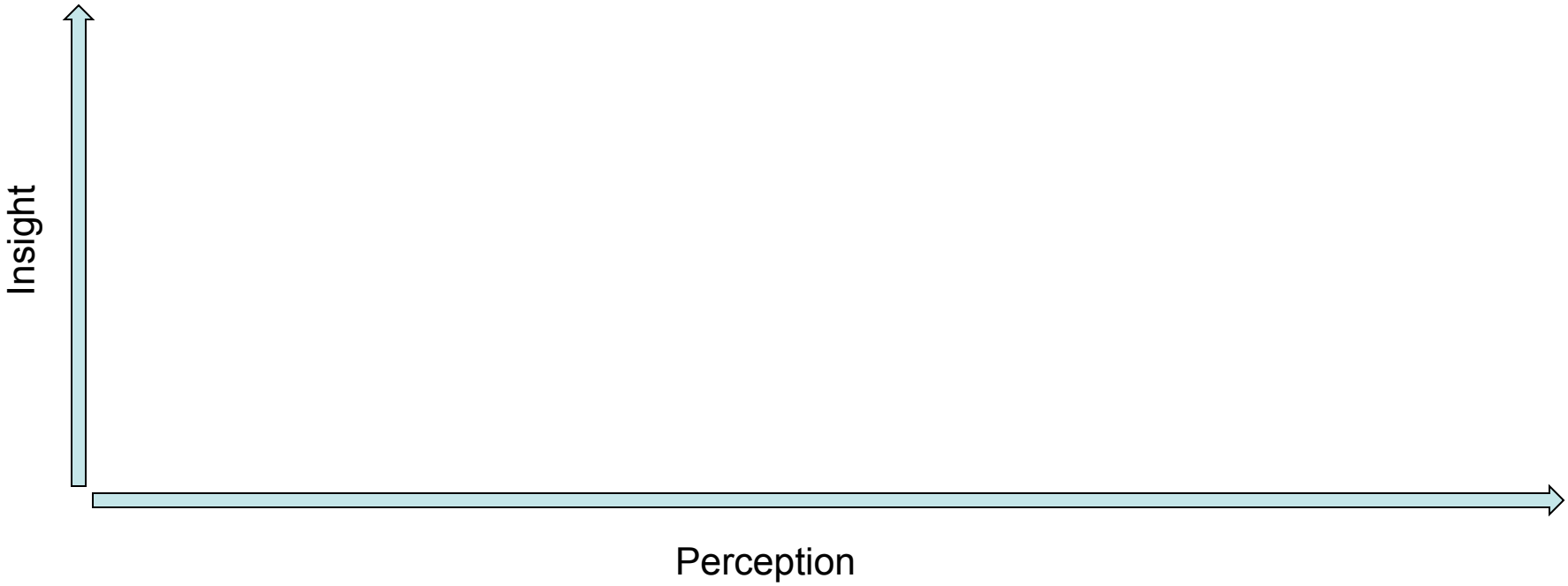


What?
- Support for line of business functions
- Marketing and promotion

Why Did You Miss Them?
- Any jerk with a credit card and the ability to submit an expense report is now runs their own private procurement office

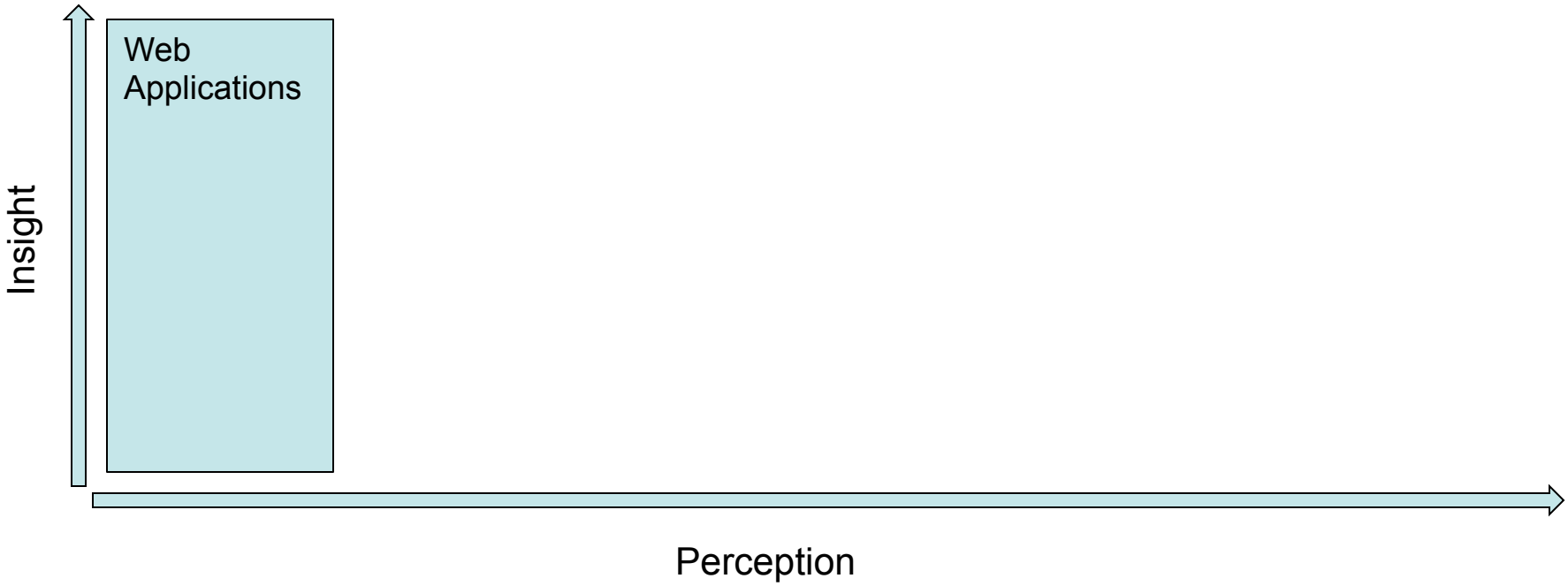
Attack Surface: The Security Officer's Journey

- Two Dimensions:
  - *Perception of Software Attack Surface*
  - *Insight into Exposed Assets*



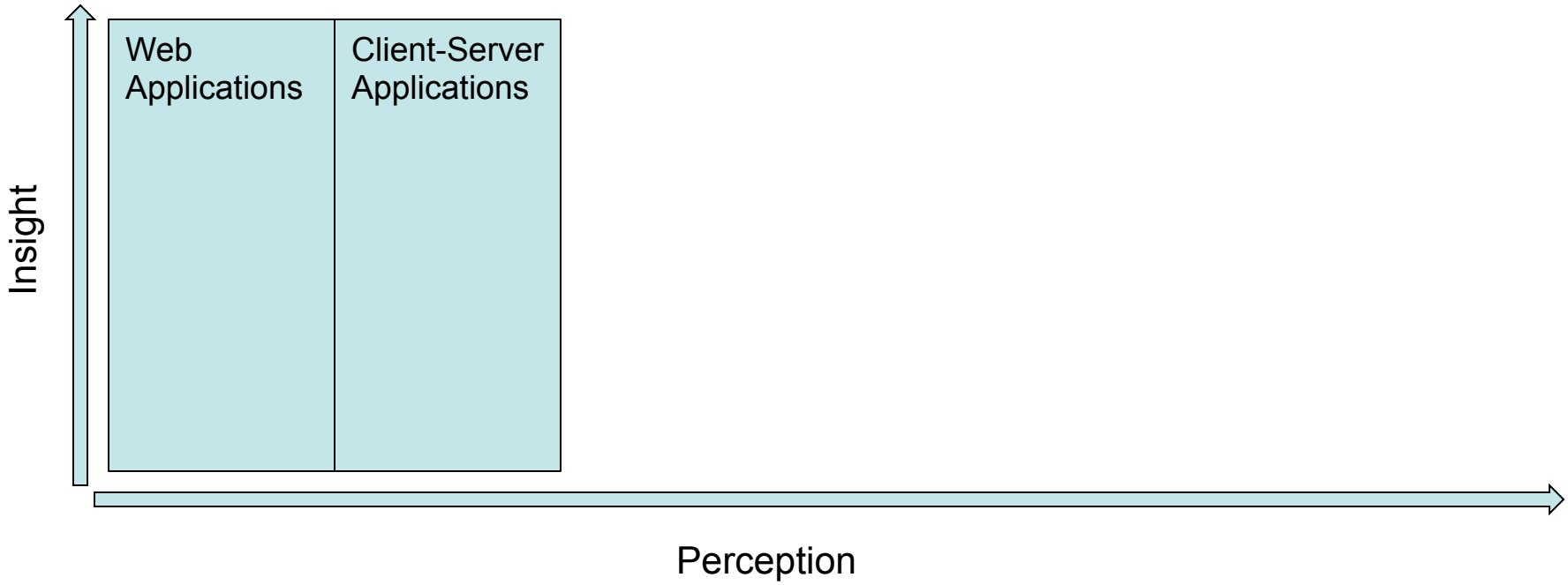
Attack Surface: The Security Officer's Journey

- As perception of the problem of attack surface widens the scope of the problem increases



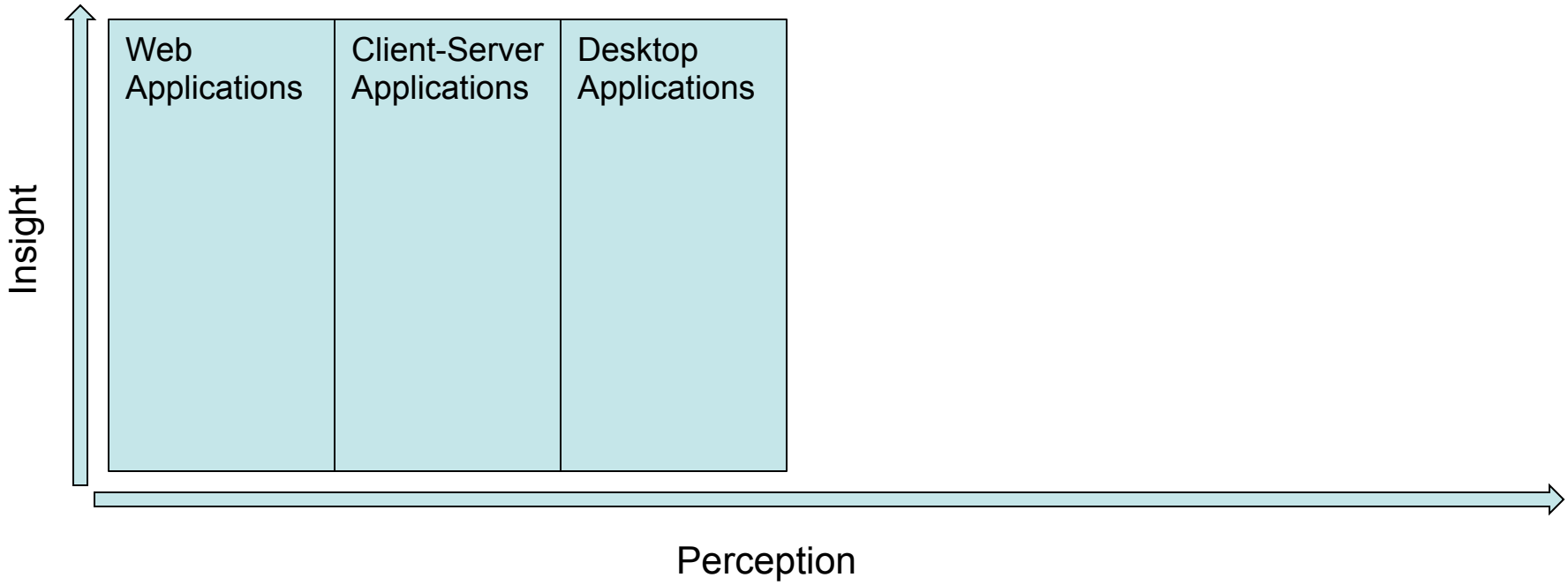
Attack Surface: The Security Officer's Journey

- As perception of the problem of attack surface widens the scope of the problem increases



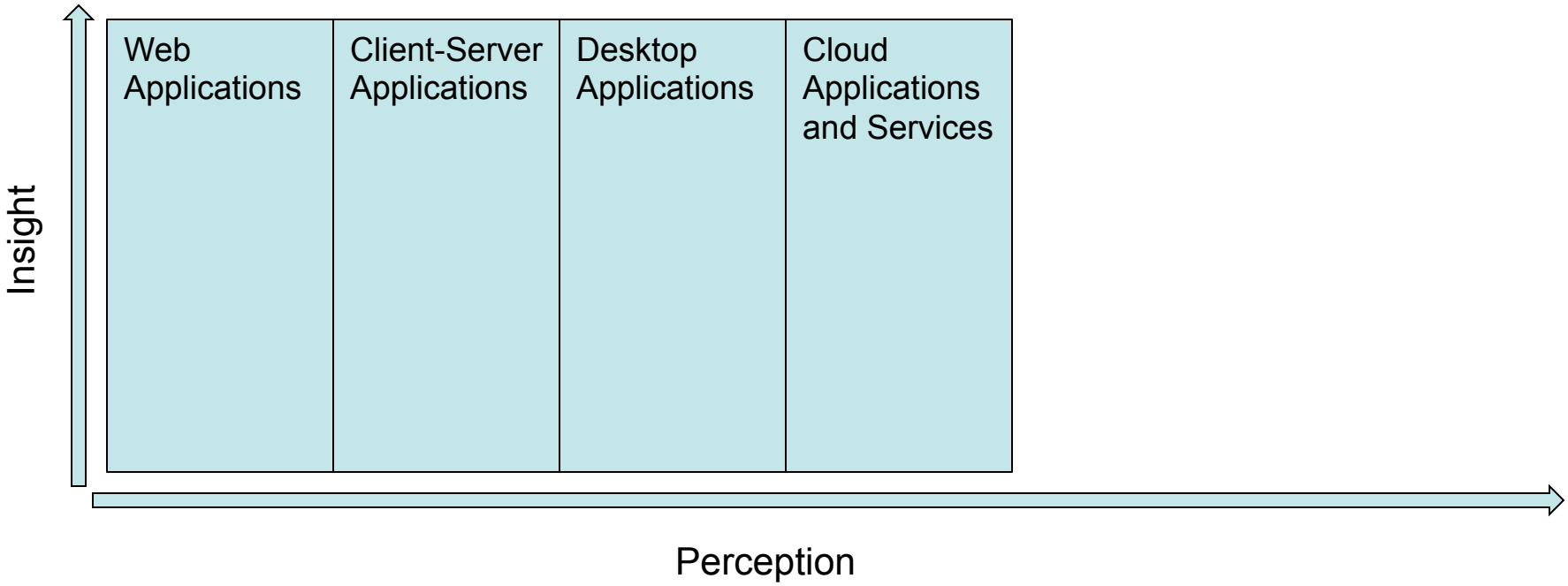
Attack Surface: The Security Officer's Journey

- As perception of the problem of attack surface widens the scope of the problem increases



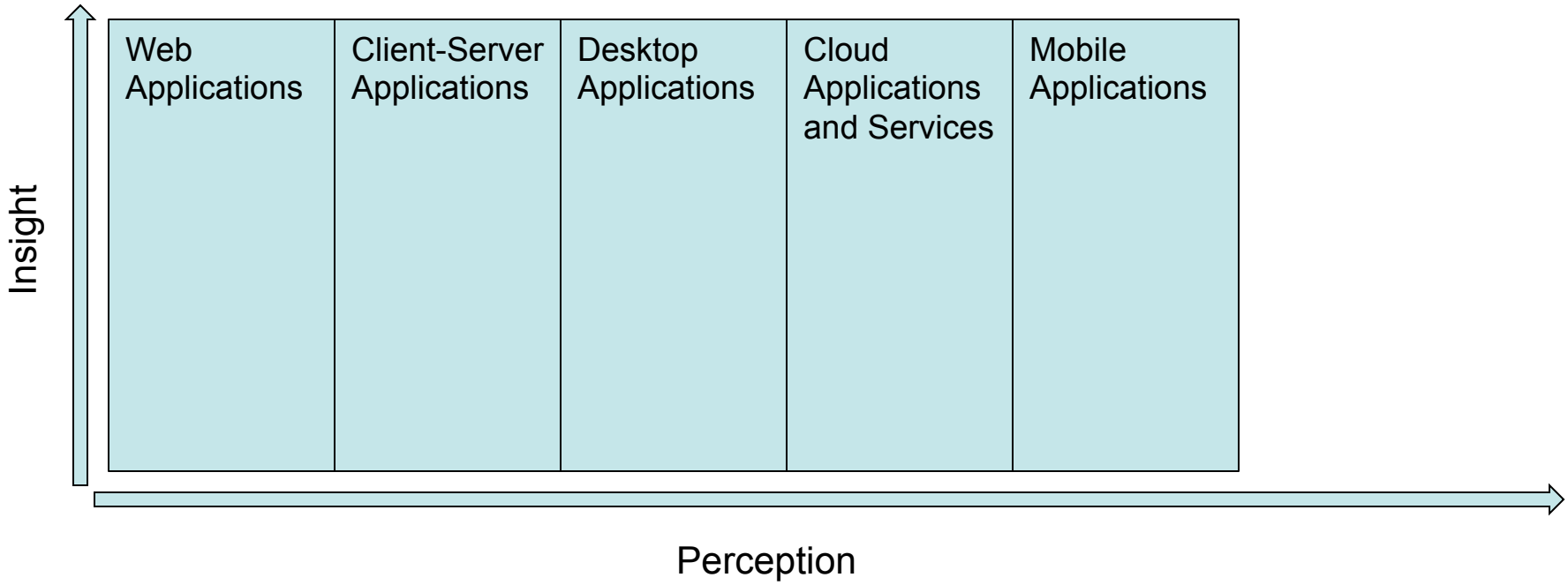
Attack Surface: The Security Officer's Journey

- As perception of the problem of attack surface widens the scope of the problem increases



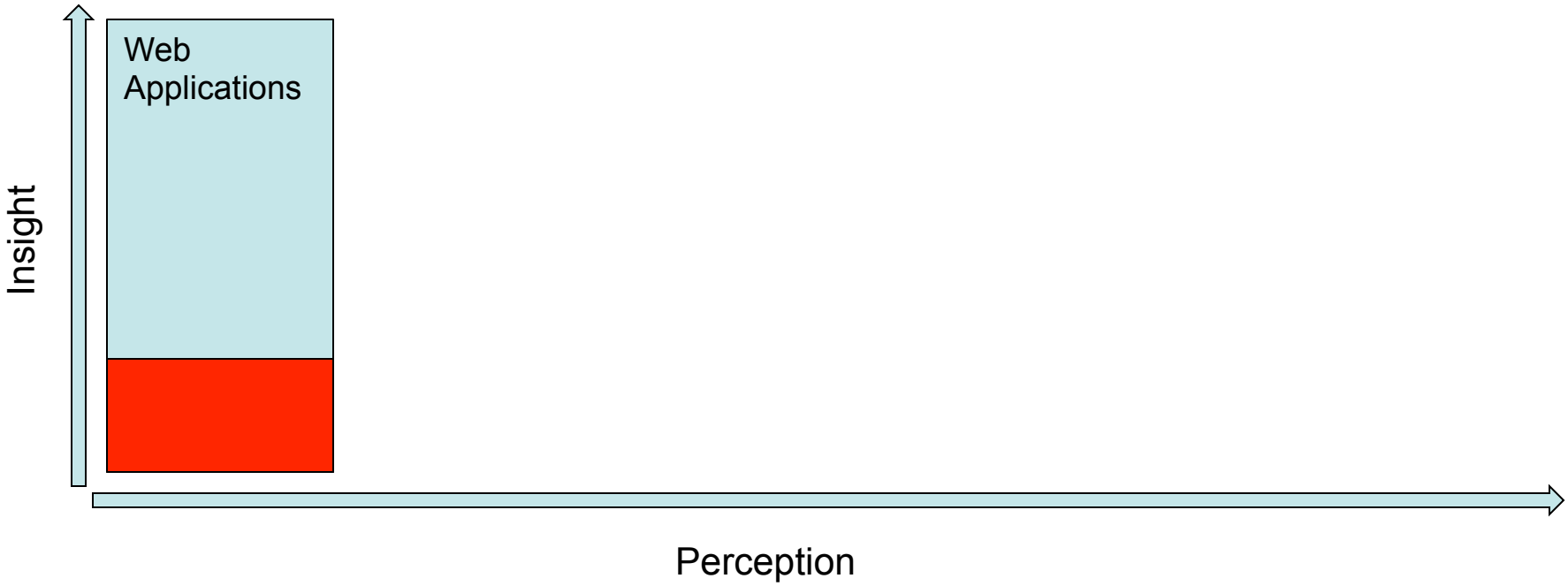
Attack Surface: The Security Officer's Journey

- As perception of the problem of attack surface widens the scope of the problem increases



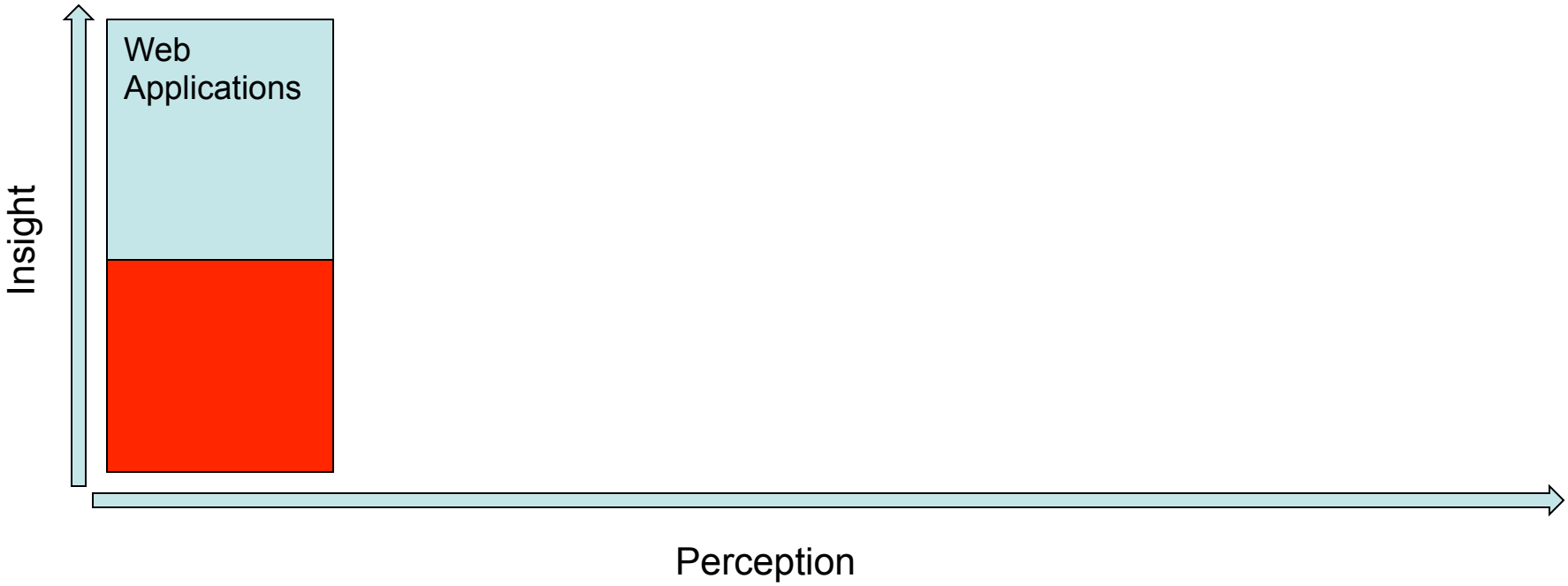
Attack Surface: The Security Officer's Journey

- Discovery activities increase insight



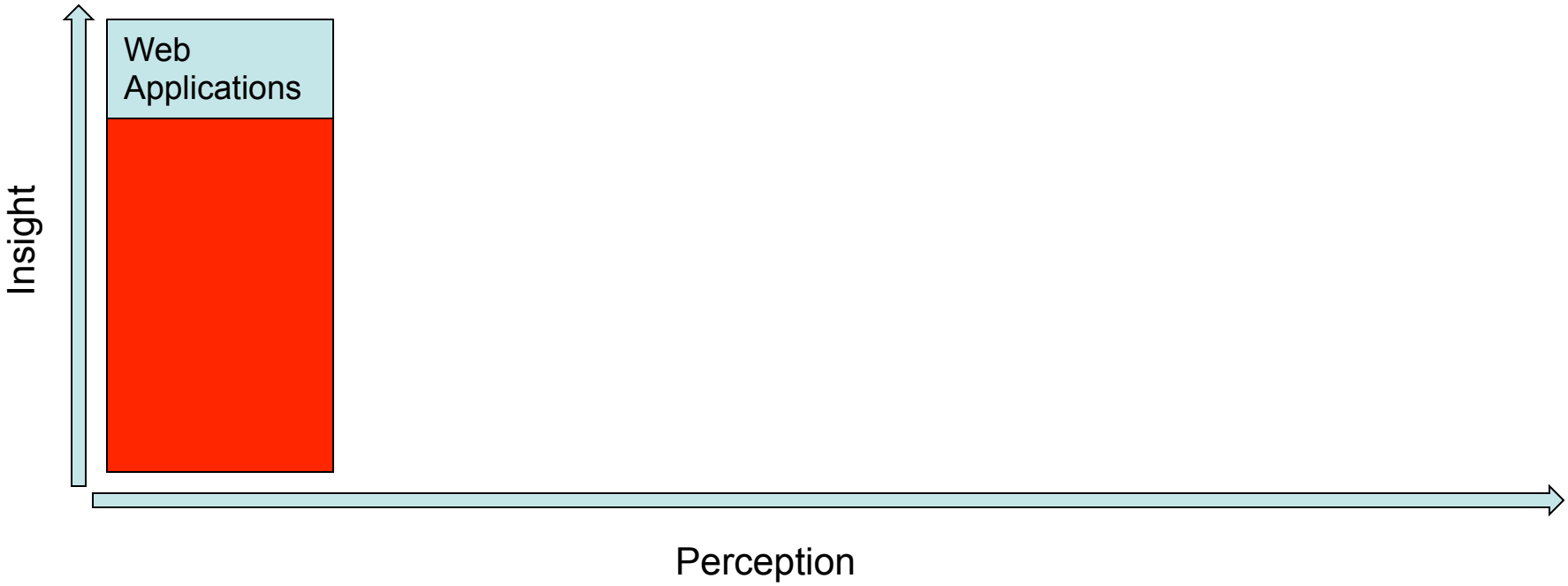
Attack Surface: The Security Officer's Journey

- Discovery activities increase insight



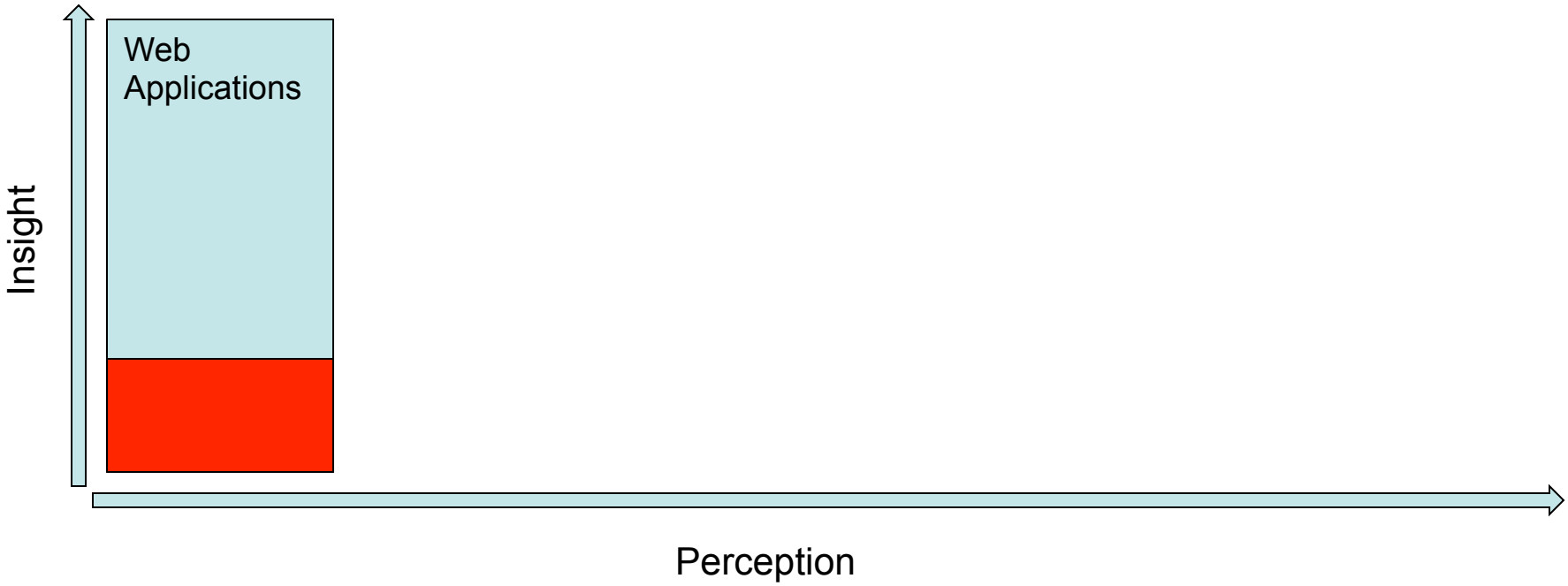
Attack Surface: The Security Officer's Journey

- Discovery activities increase insight



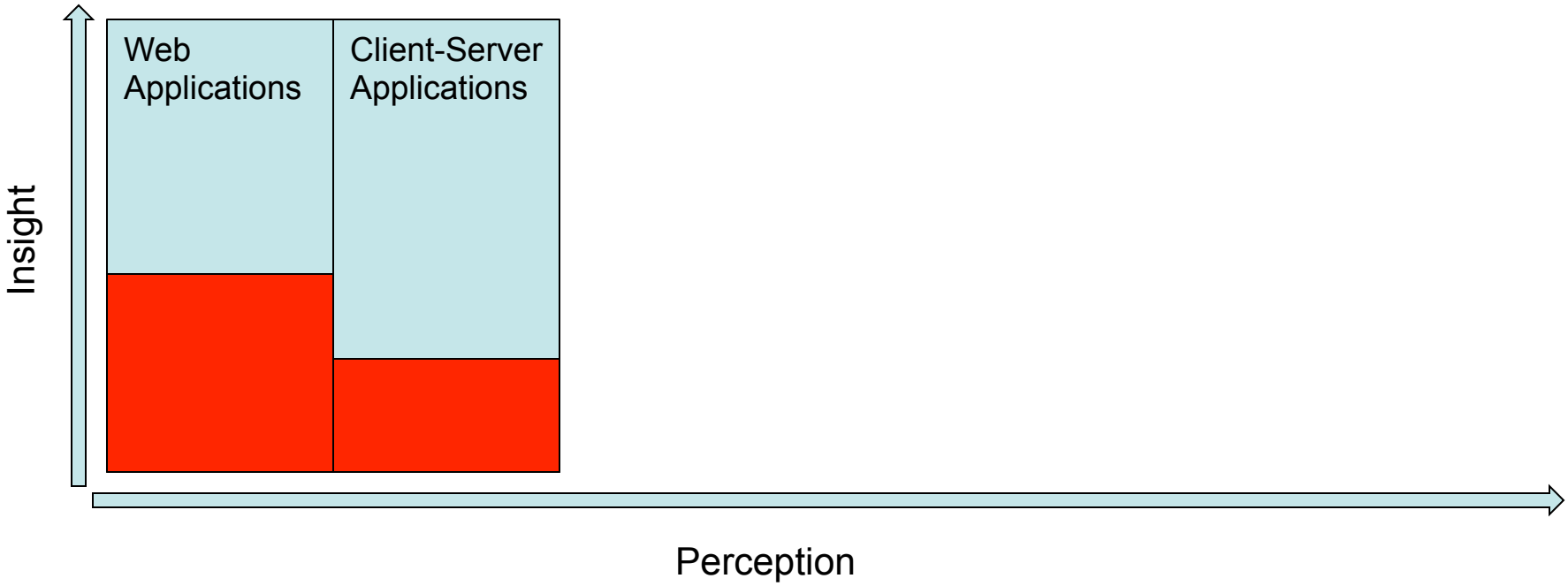
Attack Surface: The Security Officer's Journey

- Over time you end up with a progression



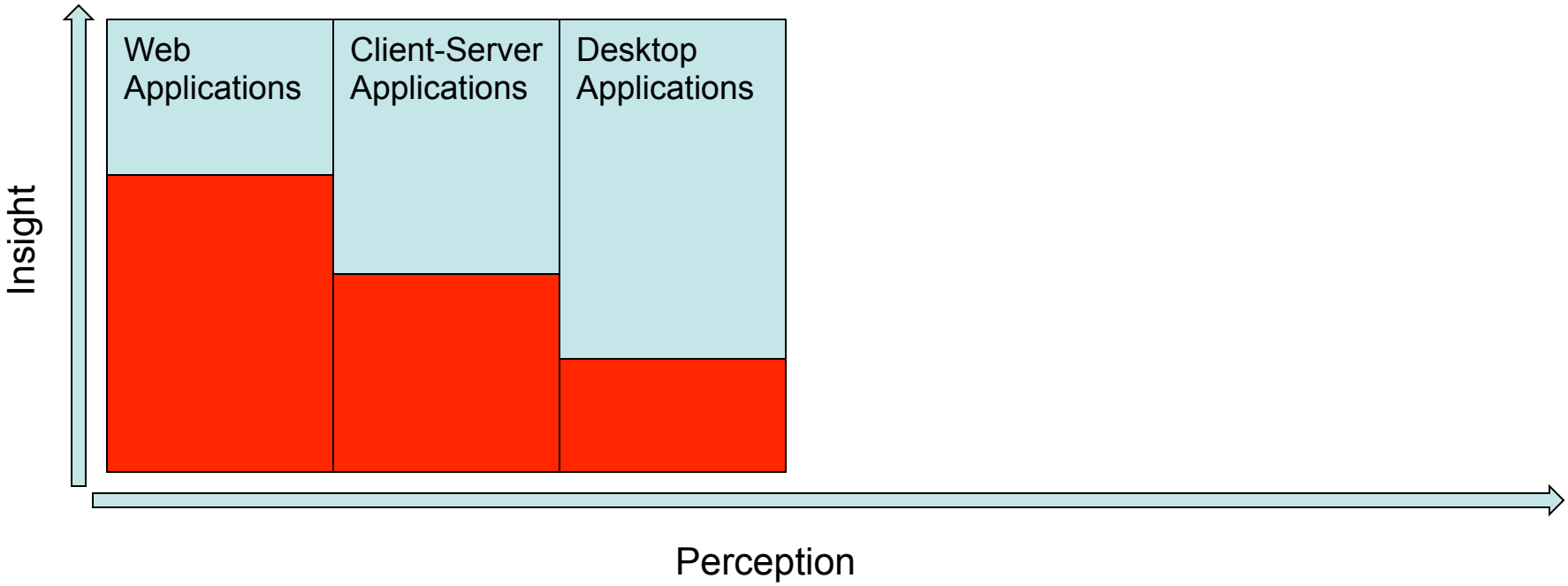
Attack Surface: The Security Officer's Journey

- Over time you end up with a progression



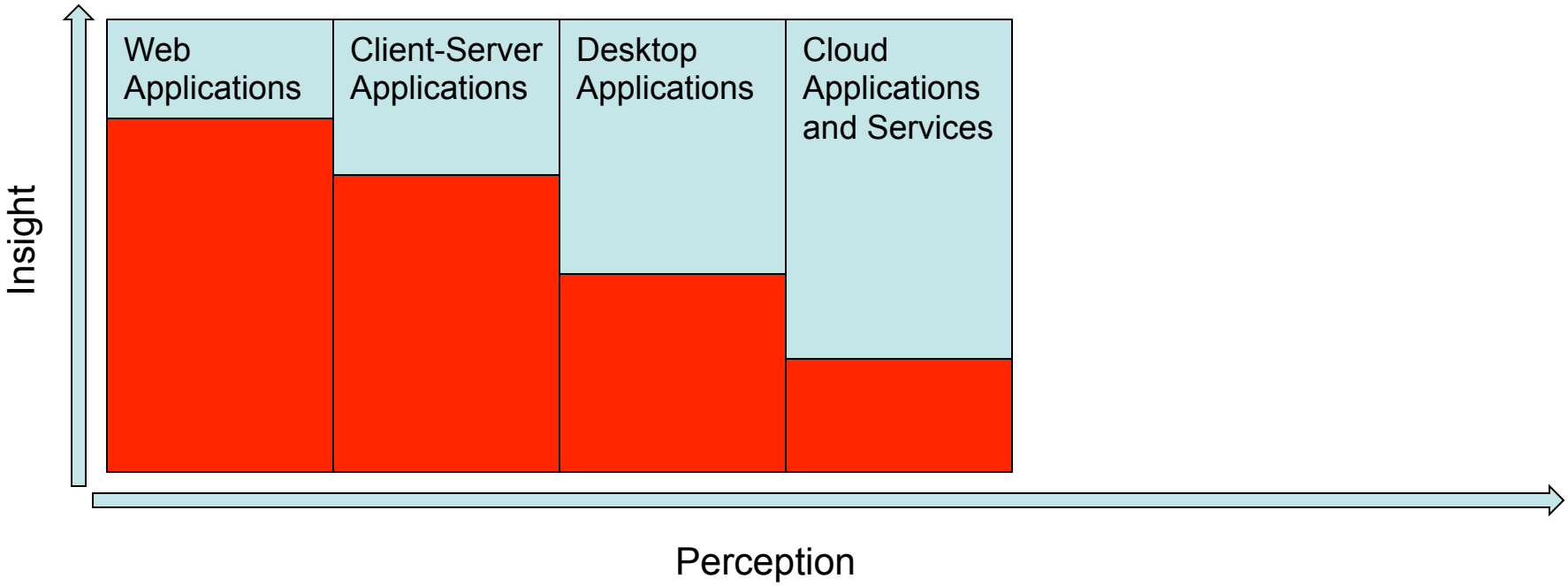
Attack Surface: The Security Officer's Journey

- Over time you end up with a progression



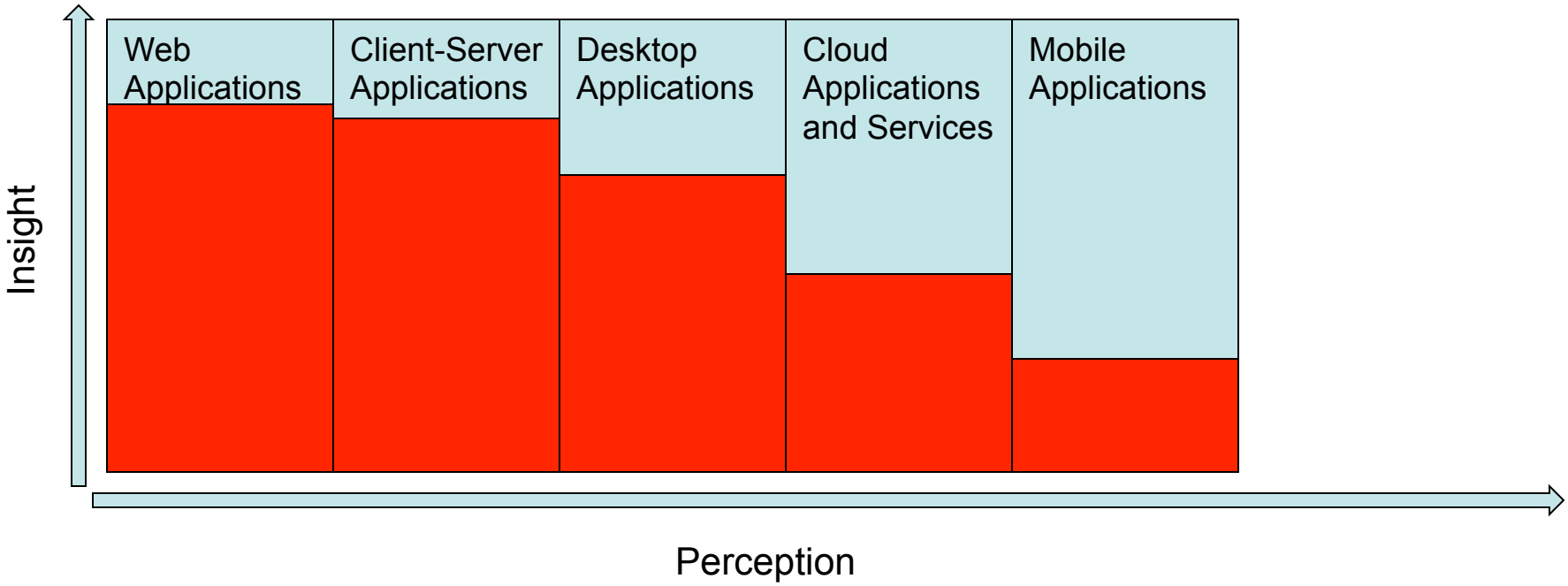
Attack Surface: The Security Officer's Journey

- Over time you end up with a progression



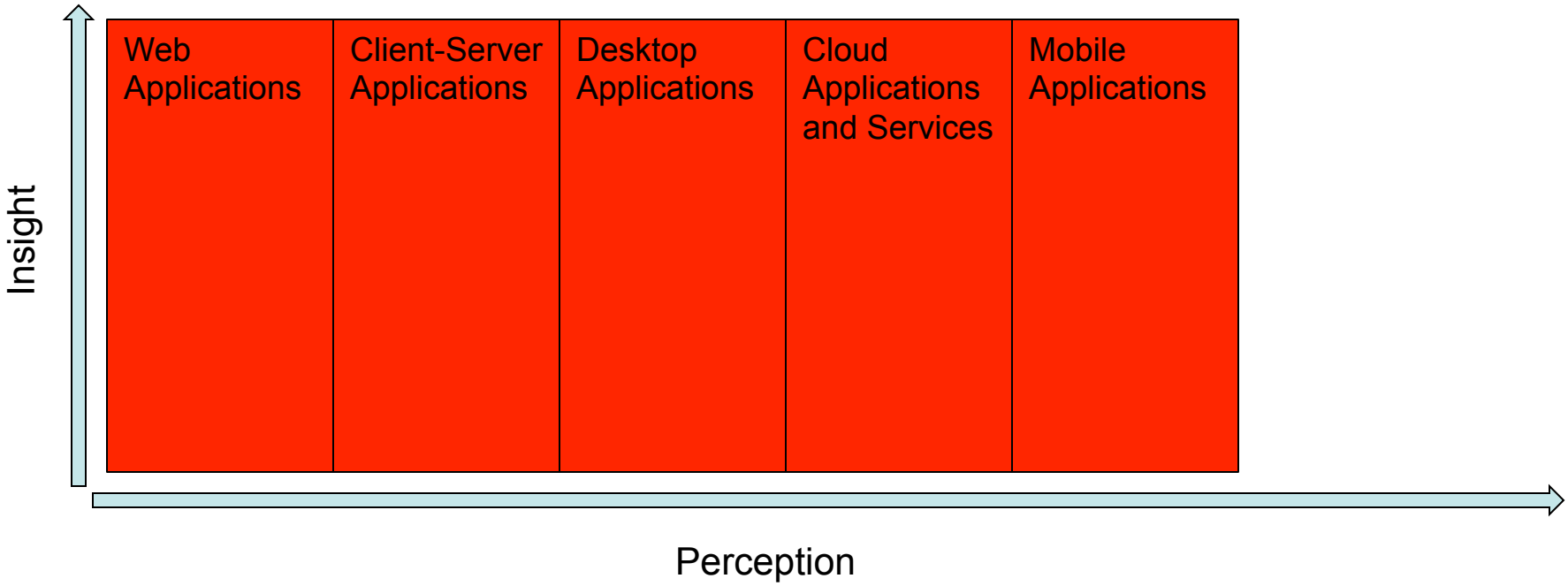
Attack Surface: The Security Officer's Journey

- Over time you end up with a progression



Attack Surface: The Security Officer's Journey

- When you reach this point it is called "enlightenment"
- You won't reach this point



Value and Risk Are Not Equally Distributed

- Some Applications Matter More Than Others
  - *Value and character of data being managed*
  - *Value of the transactions being processed*
  - *Cost of downtime and breaches*

- Therefore All Applications Should Not Be Treated the Same
  - *Allocate different levels of resources to assurance*
  - *Select different assurance activities*
  - *Also must often address compliance and regulatory requirements*

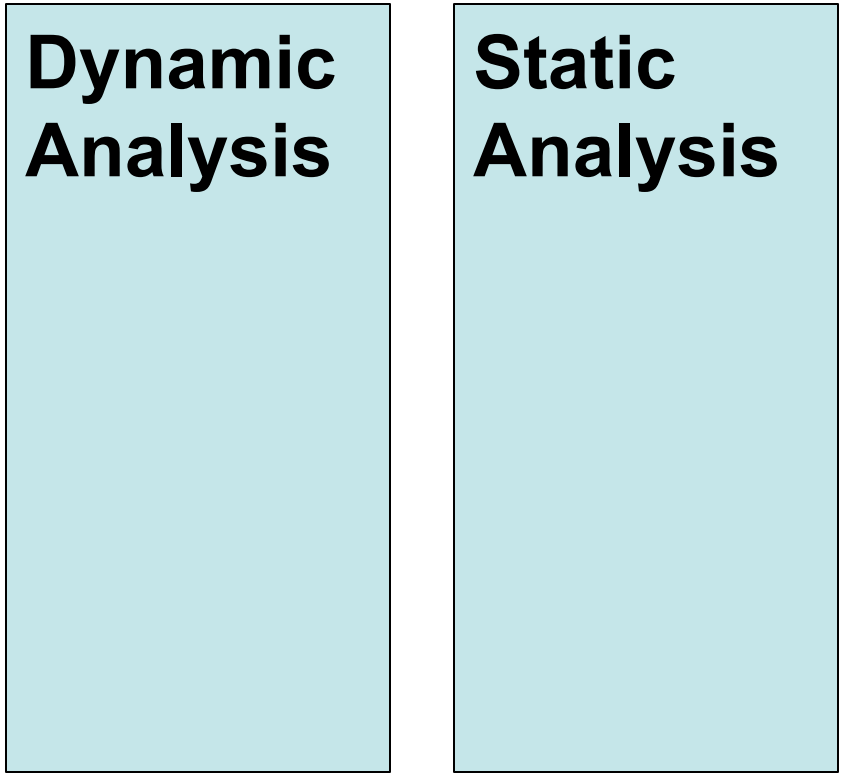
Do Not Treat All Applications the Same

- Allocate Different Levels of Resources to Assurance
- Select Different Assurance Activities

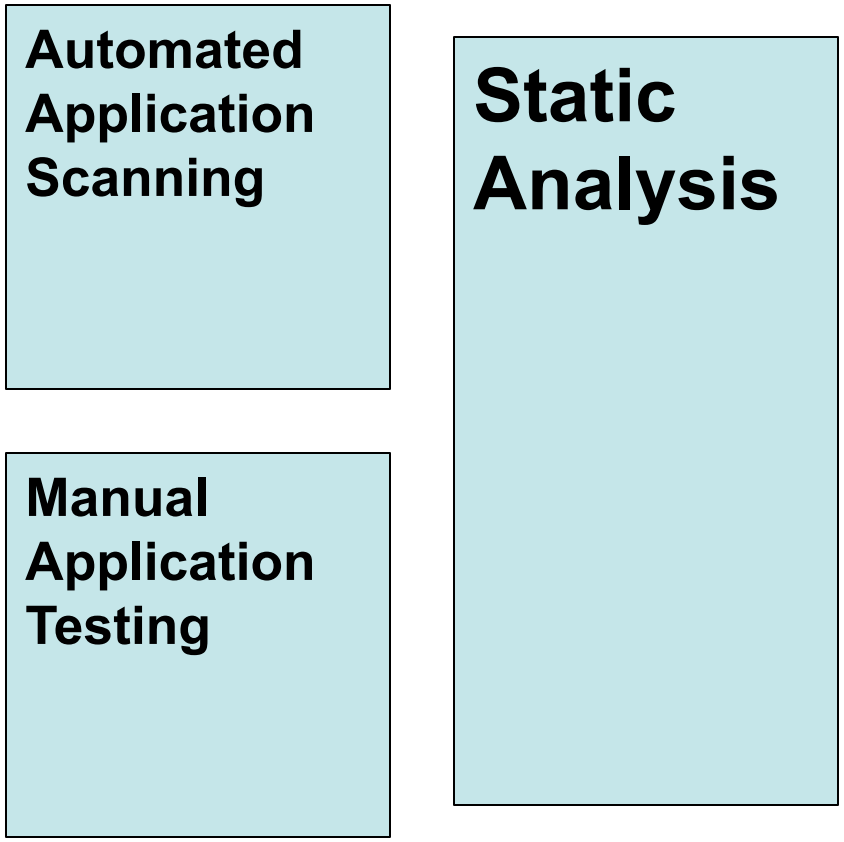
- Also Must Often Address Compliance and Regulatory Requirements

What Goes Into An Application Test?

What Goes Into An Application Test?



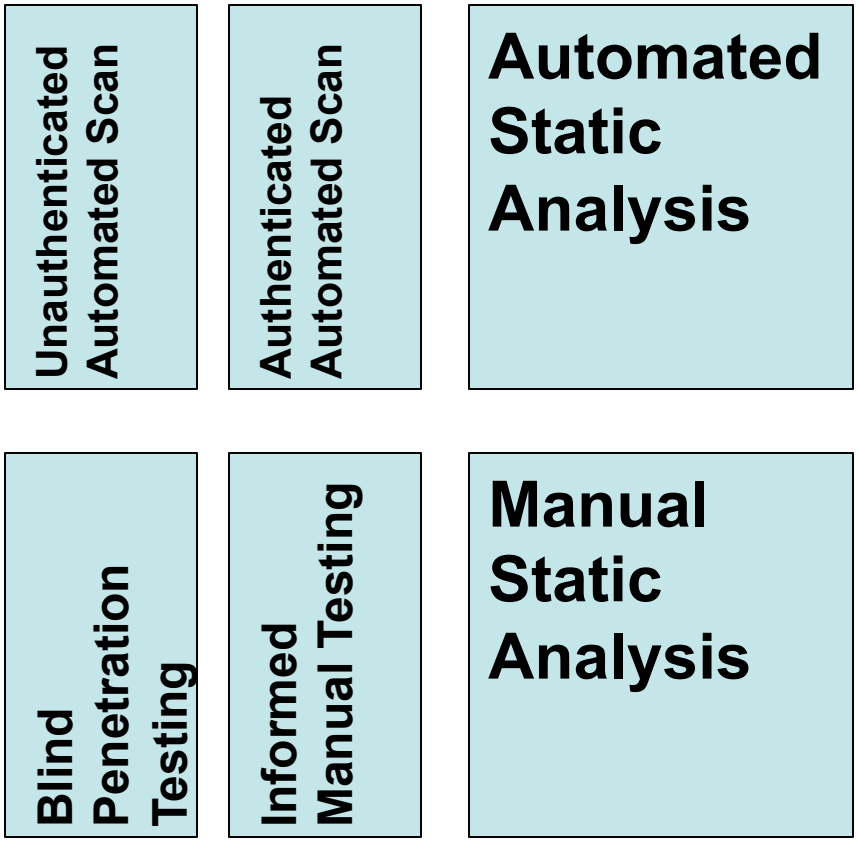
What Goes Into An Application Test?



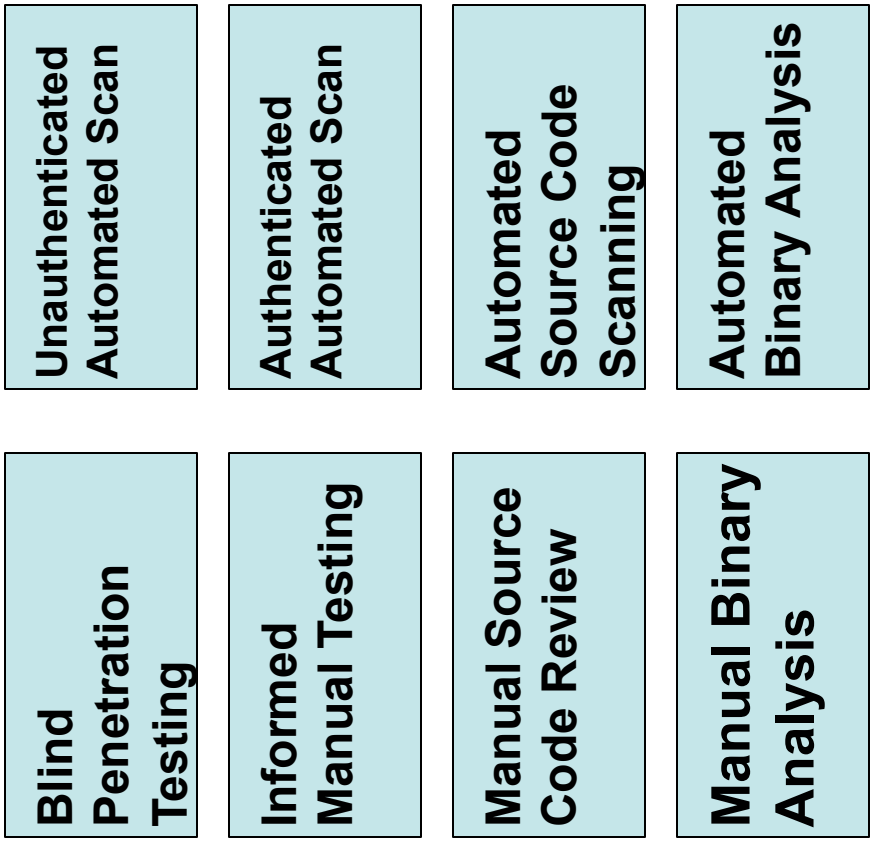
What Goes Into An Application Test?

Automated Application Scanning	Automated Static Analysis
Manual Application Testing	Manual Static Analysis

What Goes Into An Application Test?



What Goes Into An Application Test?



How To Allocate Scarce Resources?

- What Do You HAVE To Do?
  - *What discretion do you have within these constraints?*
- What Is Left Over?
- Strategies
  - *Breadth-first*
  - *Depth-first*
  - *Hybrid*

Breadth-First

- Do Base-level Security Testing of Everything
  - *Well, everything you can find*
  - *And everything you test with automation*
- Automation is key
- Understand the limitations
  - *Some applications cannot be effectively scanned*
  - *Often scans are unauthenticated*
  - *Whole classes of vulnerabilities are out of testing scope*

Depth-First

- Do Deeper Testing of Critical Applications
- Typically Combination of Automation and Manual Testing
- Understand the Limitations
  - *Some applications remain unexamined*
  - *And breaches to those applications put shared resources and infrastructure at risk*

Hybrid

- Combination of Automation and Manual Testing Across Portfolio
- This is where most organizations end up
  - *Often because regulatory and compliance mandates*
- Know Your Gaps

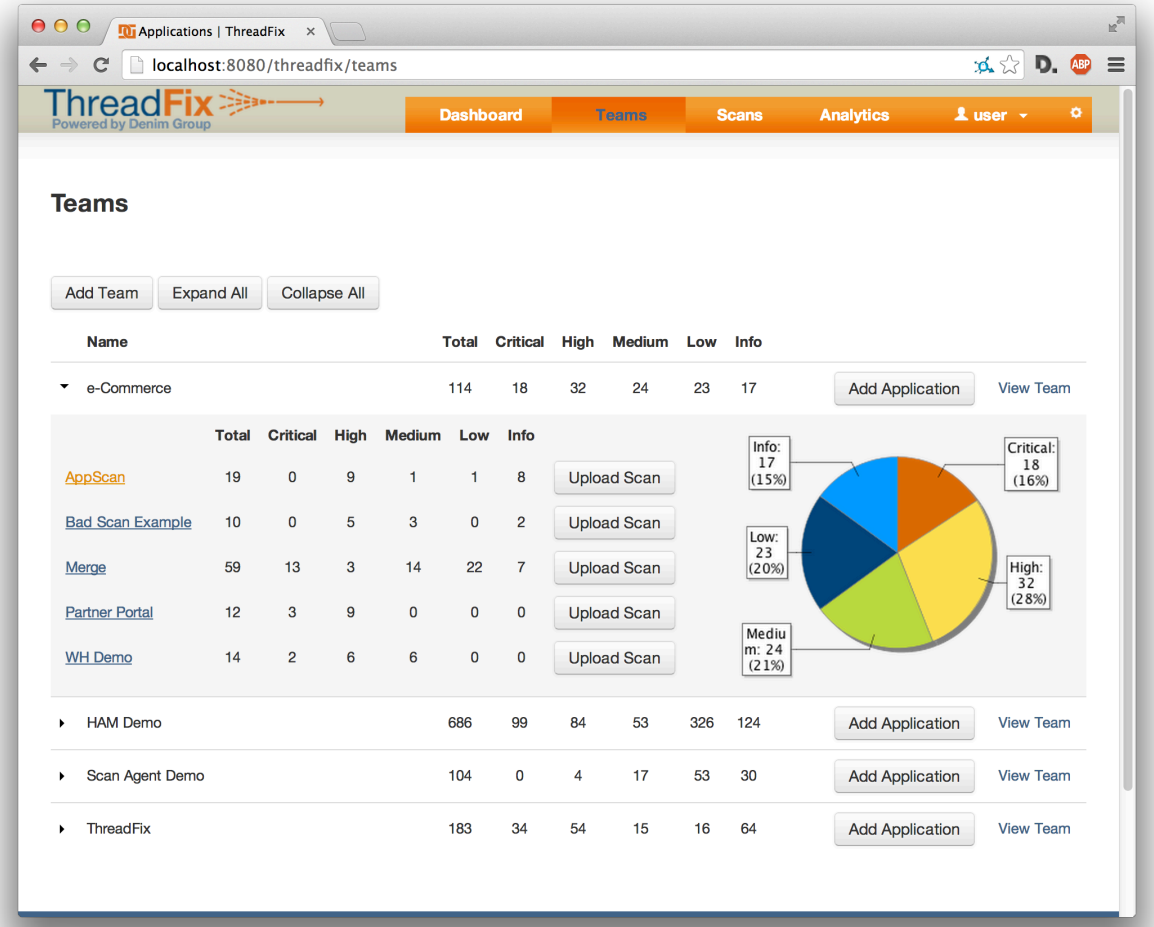
Application Portfolio Tracking

- Track multiple “Teams”
  - *Arbitrary distinction – geography, line of business, common tools and practices*
- Track multiple “Applications” per “Team”
  - *Unit of scanning or testing*

- Track Application metadata
  - *Criticality, hosted URL, source code location*

- Reporting can be done at the organization, Team or Application level

Demo: Application Portfolio Tracking



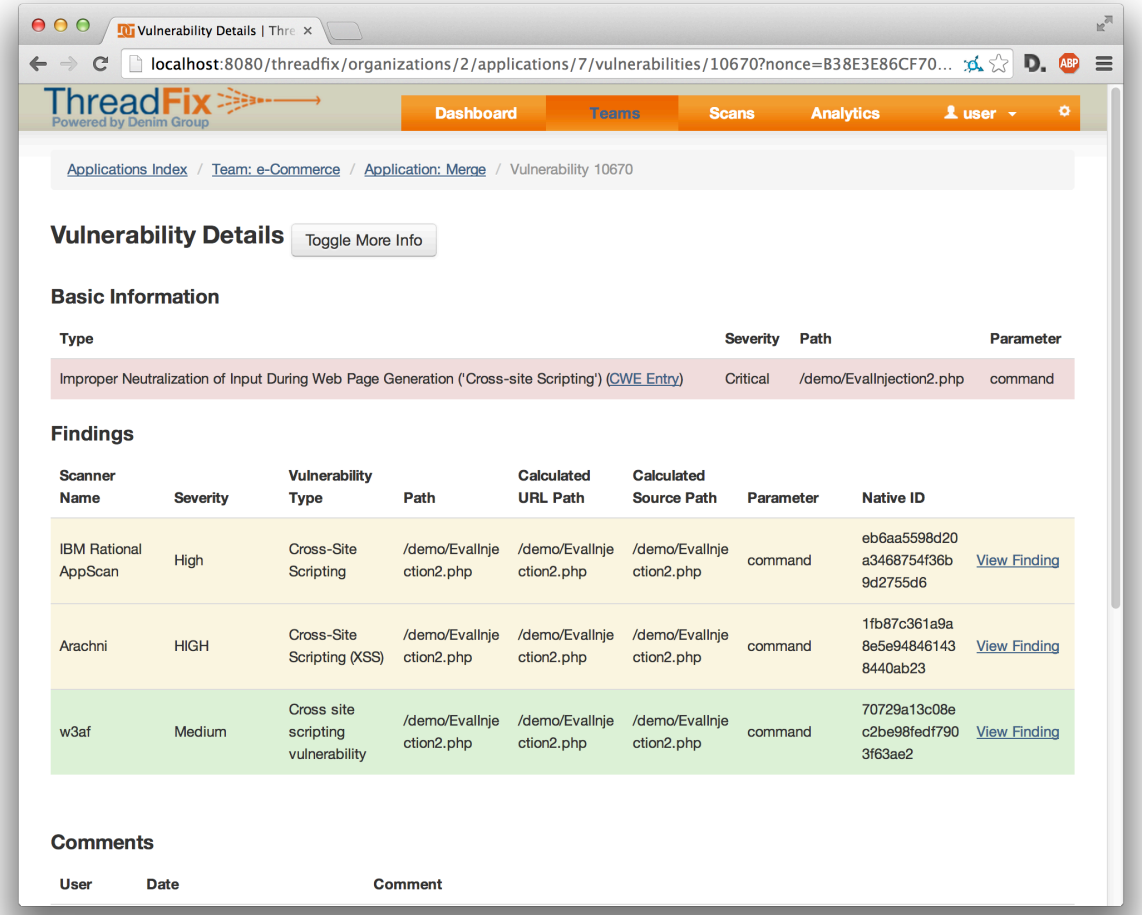
Fill ThreadFix Up With Vulnerability Data

- Manual file upload
- REST API
  - https://github.com/denimgroup/threadfix/wiki/Threadfix-REST-Interface
- Command Line Interface (CLI)
  - https://github.com/denimgroup/threadfix/wiki/Command-Line-Interface
  - *JAR can also be used as a Java REST client library*
- Jenkins plugin
  - *Contributed from the ThreadFix community (yeah!)*
  - https://github.com/automationdomination/threadfix-plugin

What Does ThreadFix Do With Scan Results

- Diff against previous scans with same technology
  - *What vulnerabilities are new?*
  - *What vulnerabilities went away?*
  - *What vulnerabilities resurfaced?*
- Findings marked as false positive are remembered across scans
  - *Hopefully saving analyst time*
- Normalize and merge with other scanners' findings
  - *SAST to SAST*
  - *DAST to DAST*
  - *SAST to DAST via Hybrid Analysis Mapping (HAM)*

Demo: Vulnerability Merge



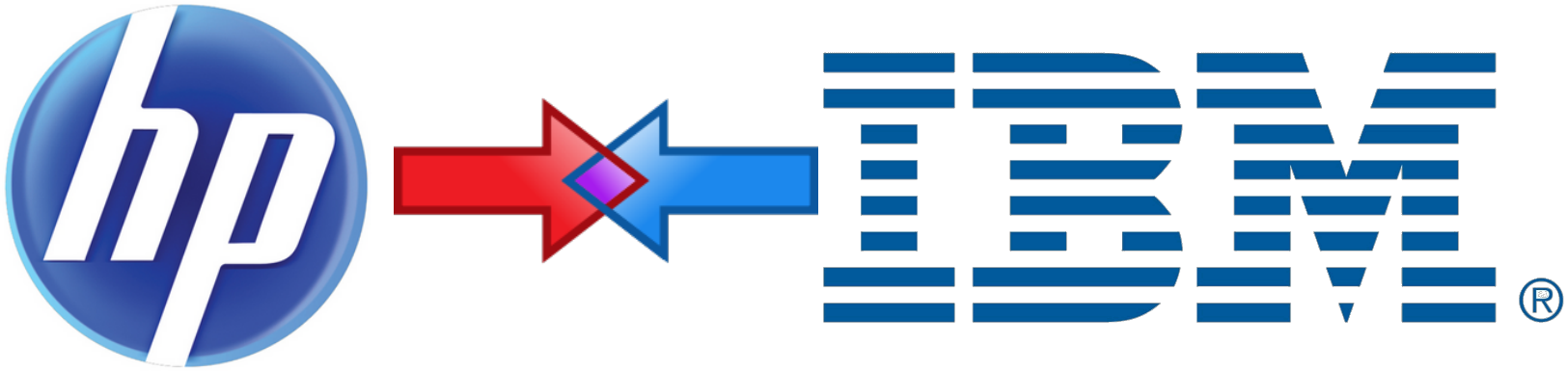
Hybrid Analysis Mapping (HAM)

- Initial research funded by the US Department of Homeland Security (DHS) Science and Technology (S&T) Directorate via a Phase 1 and (now) Phase 2 Small Business Innovation Research (SBIR) contract
  - *Acronyms!*

- Initial goal: SAST to DAST merging
- Results: That, plus other stuff

Hybrid Analysis Mapping – Phase 1 Goal

- **Determine the feasibility of developing a system that can reliably and efficiently correlate and merge the results of automated static and dynamic security scans of web applications.**



HP Fortify SCA

IBM AppScan Standard

Dynamic Application Security Testing

- Spider to enumerate attack surface
- Fuzz to identify vulnerabilities based on analysis of request/response patterns

Static Application Security Testing

- Use source or binary to create a model of the application
  - *Kind of like a compiler or VM*
- Perform analysis to identify vulnerabilities and weaknesses
  - *Data flow, control flow, semantic, etc*

```
String username = request.getParameter("username");
String sql = "SELECT * FROM User WHERE username = '" + username + "'";

Statement stmt;
stmt = con.createStatement();
stmt.execute(sql);
```

Hybrid Analysis Mapping – Phase 1 Sub-Goals

- Standardize vulnerability types
- Match dynamic and static locations
- Improve static parameter parsing

Hybrid Analysis Mapping
Phase 1 - Technical Objectives

- **Technical Objective 1: Create common data structure standards for both automated static and dynamic security scanning results.**
  - *Task 1: Create a Data Structure for Automated Dynamic Security Scanning Results*
  - *Task 2: Create a Data Structure for Automated Static Security Scanning Results*

- **Technical Objective 2: Research and prototype methods of mapping the results of automated static and dynamic security scanning.**
  - *Task 1: Create a Structured Model for Hybrid Analysis Mapping*
  - *Task 2: Investigate Approaches for Vulnerability Type Mapping*
  - *Task 3: Investigate Approaches for Mapping Source Code Files to URLs*
  - *Task 4: Investigate Approaches for Determining Injection Points*

Information Used

- Source Code (Git URL)
- Framework Type (JSP, Spring)
- Extra information from Fortify (if available)

Vulnerability Types

- Successful CWE standardization
- Investigation into trees and Software Fault Patterns
  - *Meant to correct for human errors*
  - *Hard to do in an automated fashion*

Unified Endpoint Database (Static and Dynamic)

- EndpointQuery
  - *dynamicPath*
  - *staticPath*
  - *Parameter*
  - *httpMethod*
  - *codePoints [List<CodePoint>]*
  - *informationSourceType*

- EndpointDatabase
  - *findBestMatch(EndpointQuery query): Endpoint*
  - *findAllMatches(EndpointQuery query): Set<Endpoint>*
  - *getFrameworkType(): FrameworkType*

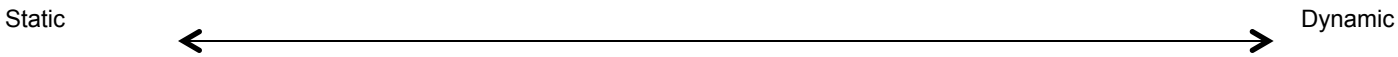
Parsing Attack Surface Locations

- JSP: Start with root JSP folder
- Spring: Parse @Controller classes

Parsing Parameters

- JSP: Look for request.getParameter() calls
  - *Coupled with lightweight dataflow analysis*
- Spring: Parse @RequestParam, @PathVariable, @Entity annotations

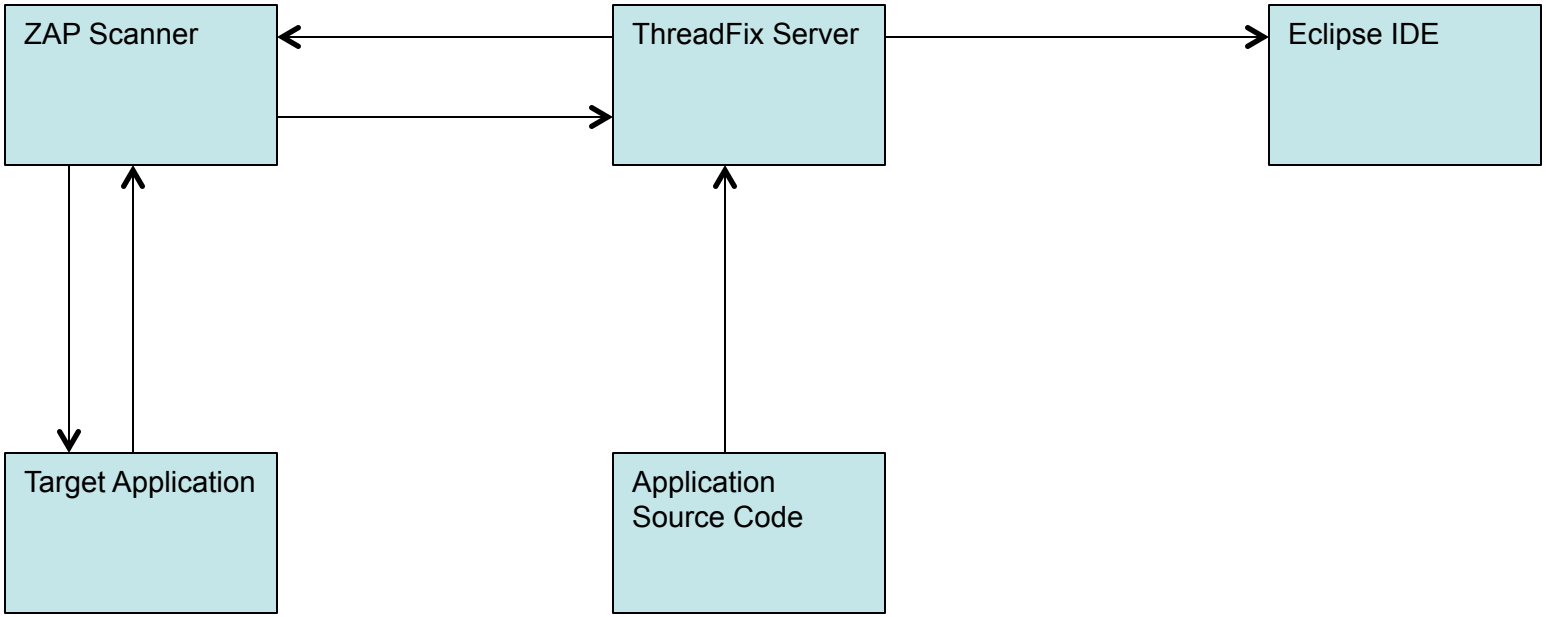
HAM Bridge



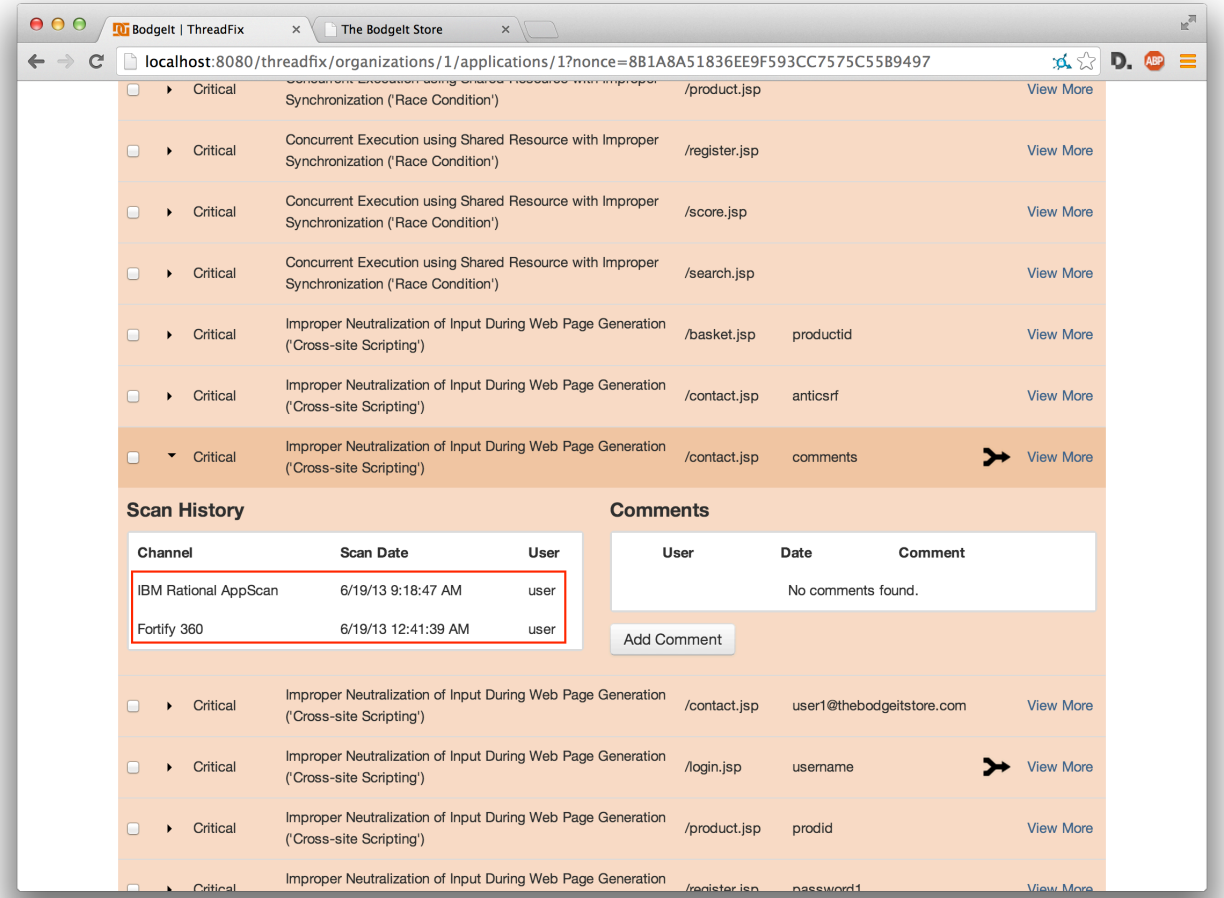
- EndpointDatabase enables more than merging

- Scanner integration allows smarter scanning
- IDE plugin shows all vulnerabilities inline

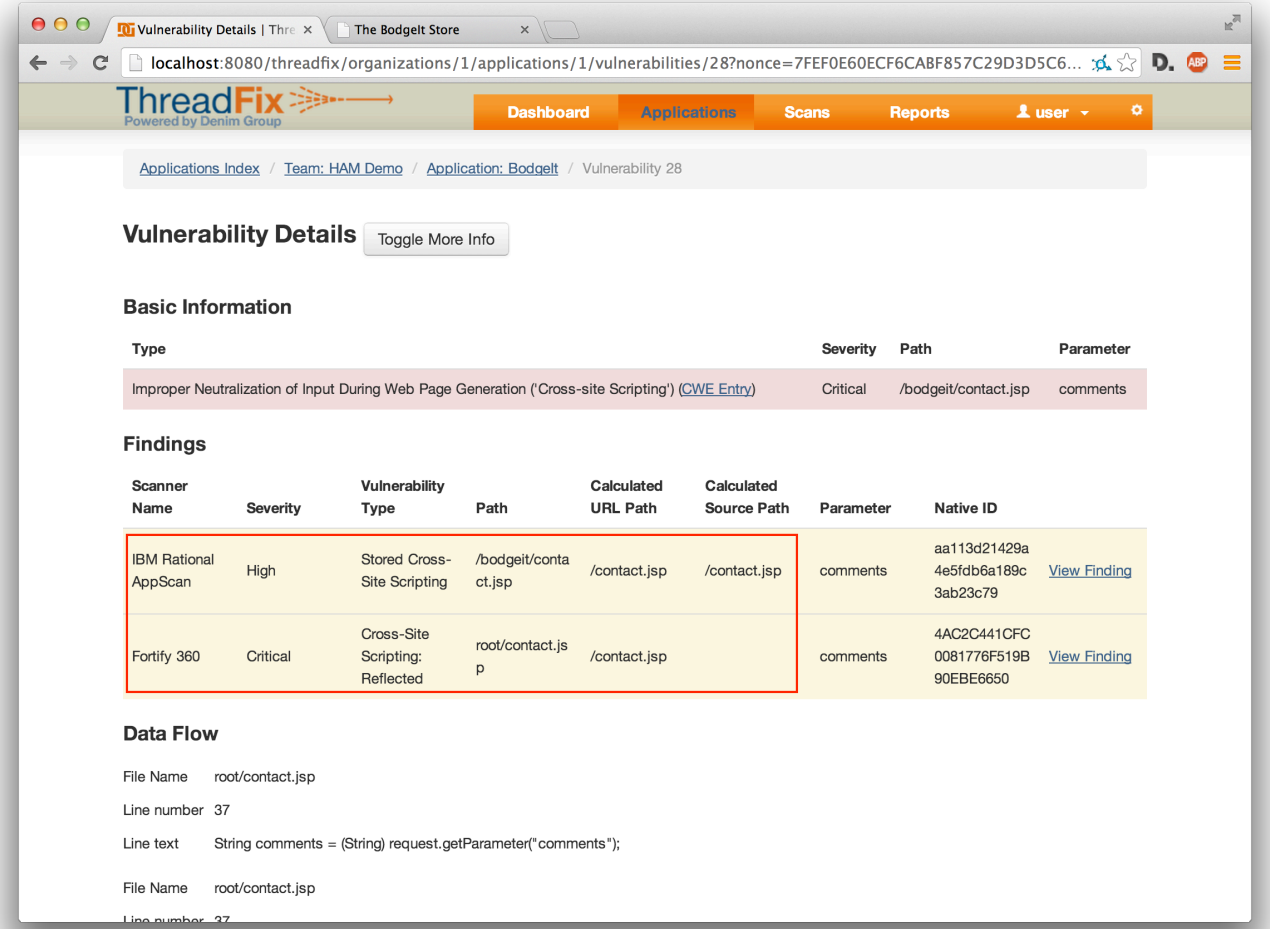
System Structure



Demo: Merging Static and Dynamic Scanner Results



Demo: Merging Static and Dynamic Scanner Results



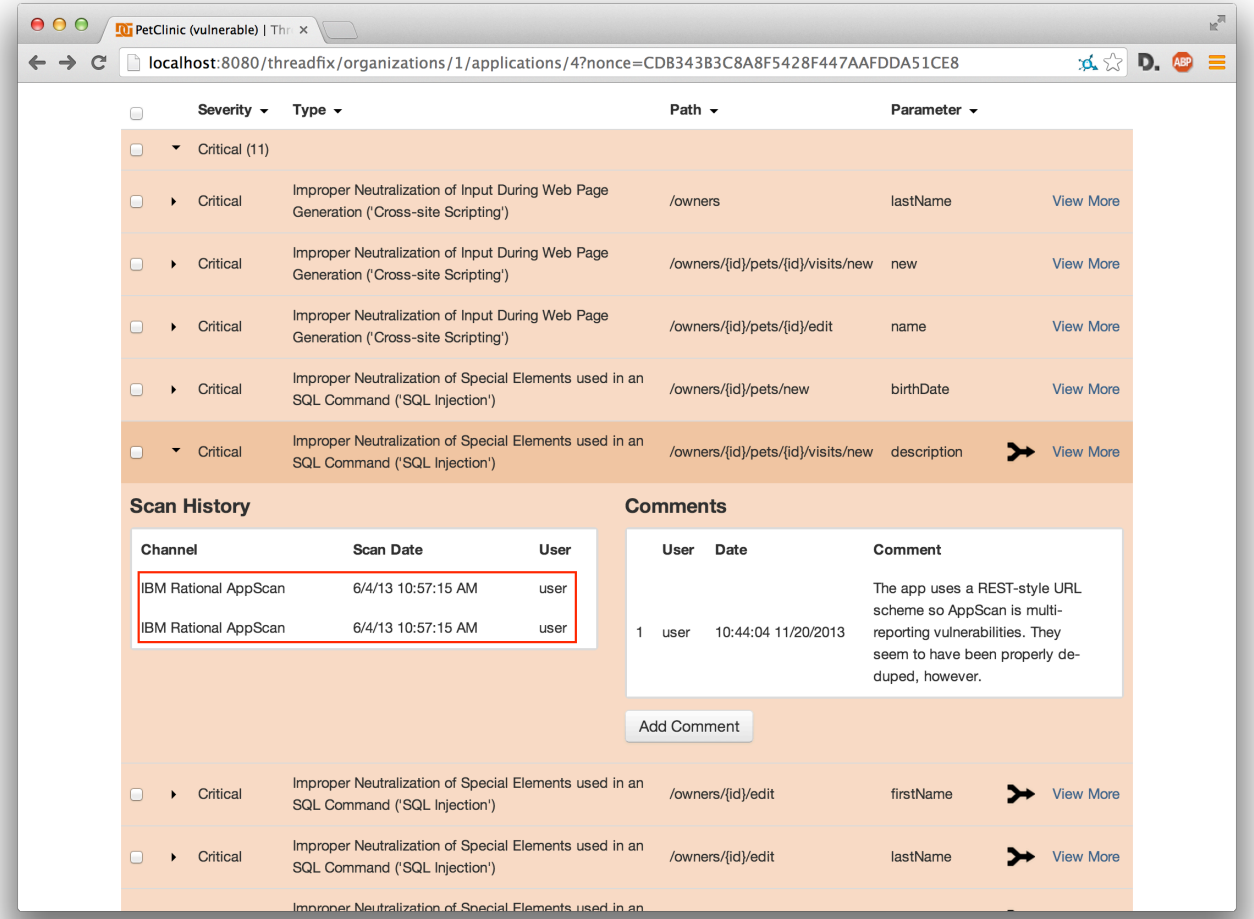
Merging Static and Dynamic Results Is Cool…

…But I want more

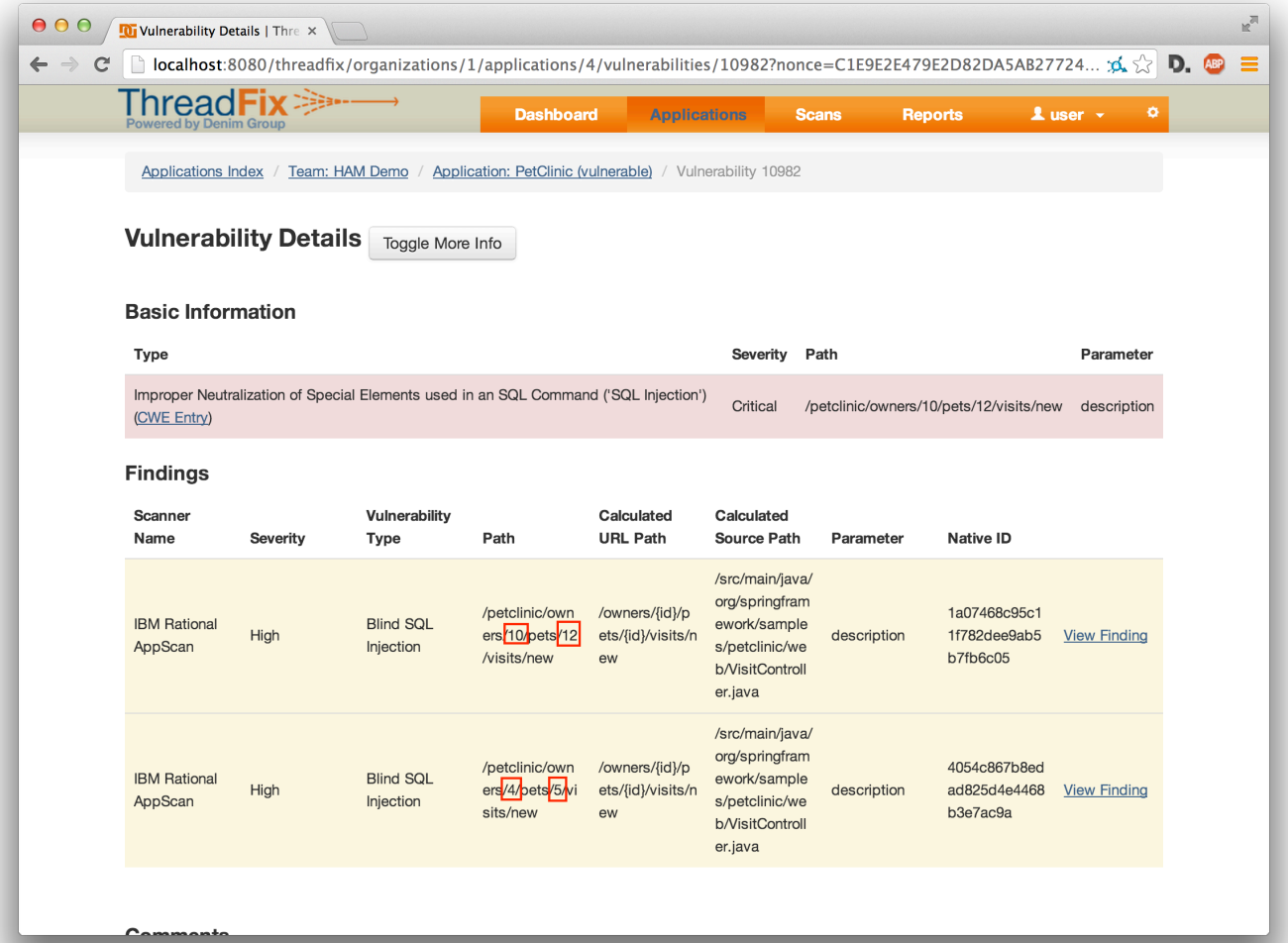
- Problem: Many DAST scanners handle applications with RESTful URLs poorly
- Problem: Many applications have “hidden” landing pages and parameters that will not be found by standard crawling
- Problem: DAST scanner results can be hard for developers to act on

- What else can we do with this attack surface model / database?
  - *Clean up scanner results*
  - *Enumerate application attack surface*
  - *Map dynamic results to specific lines of code*

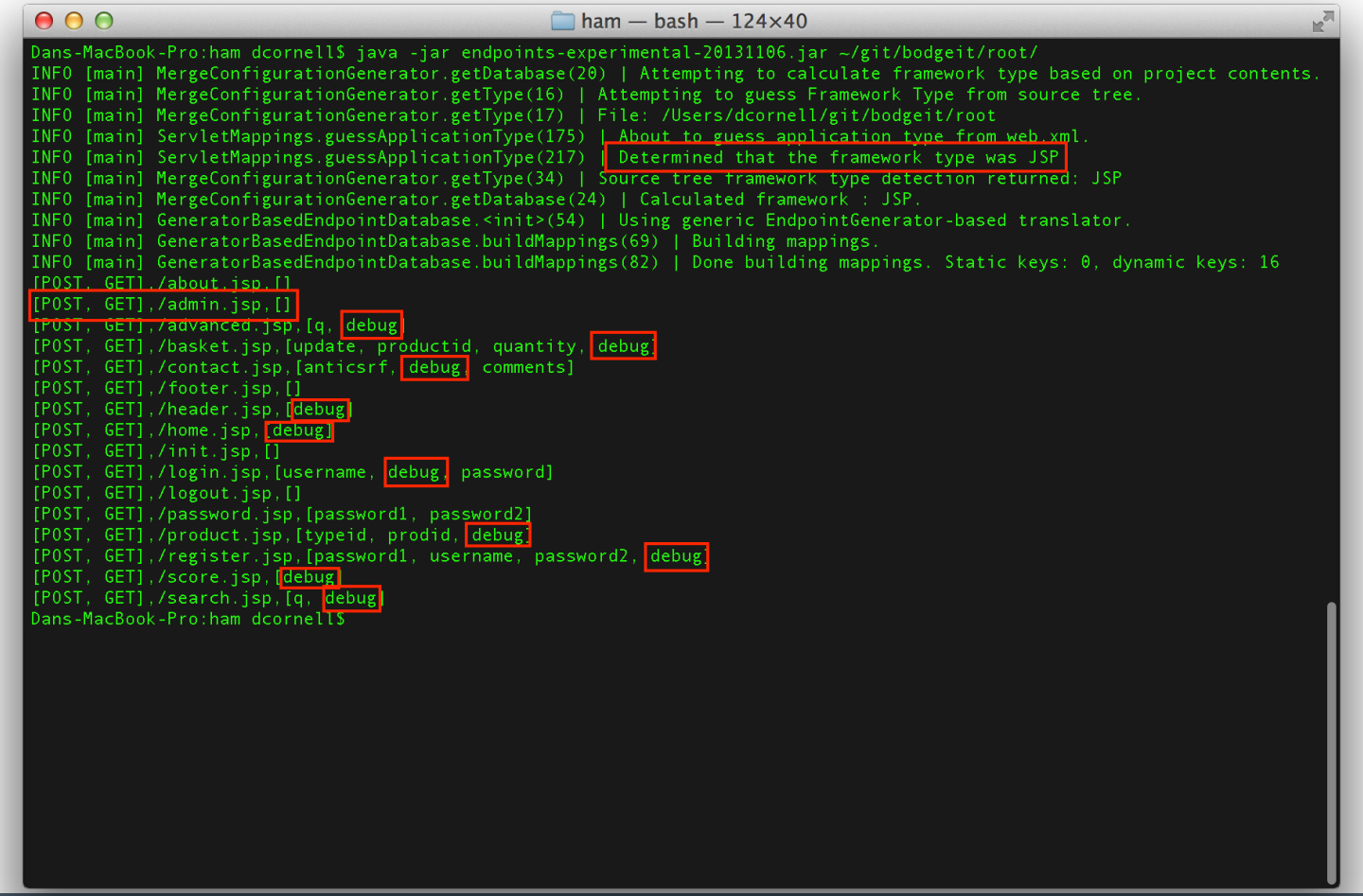
Demo: De-Duplicate Dynamic RESTful Scanner Results



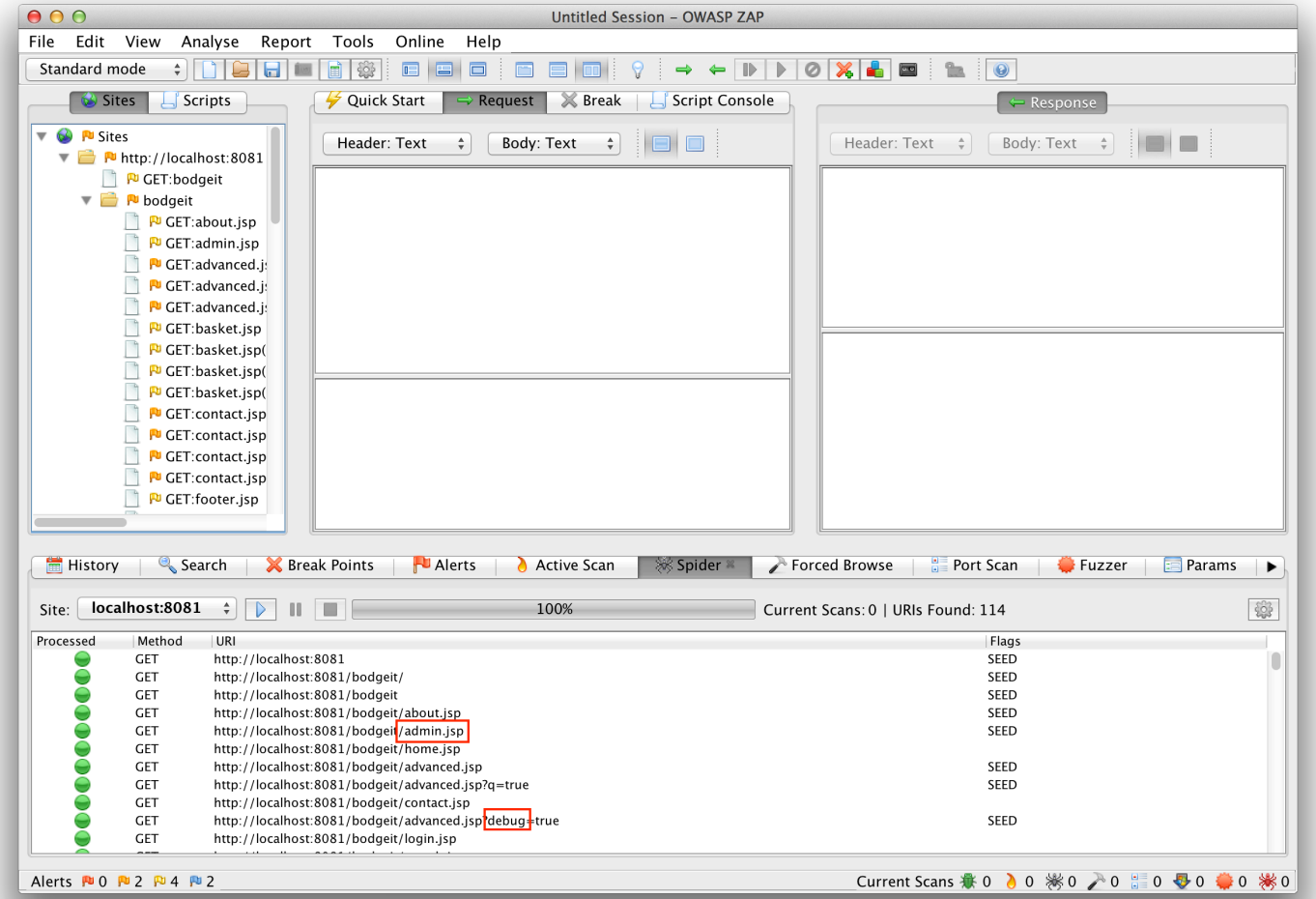
Demo: De-Duplicate Dynamic RESTful Scanner Results



Demo: Application Attack Surface (CLI)



Demo: Seed Scanner with Attack Surface



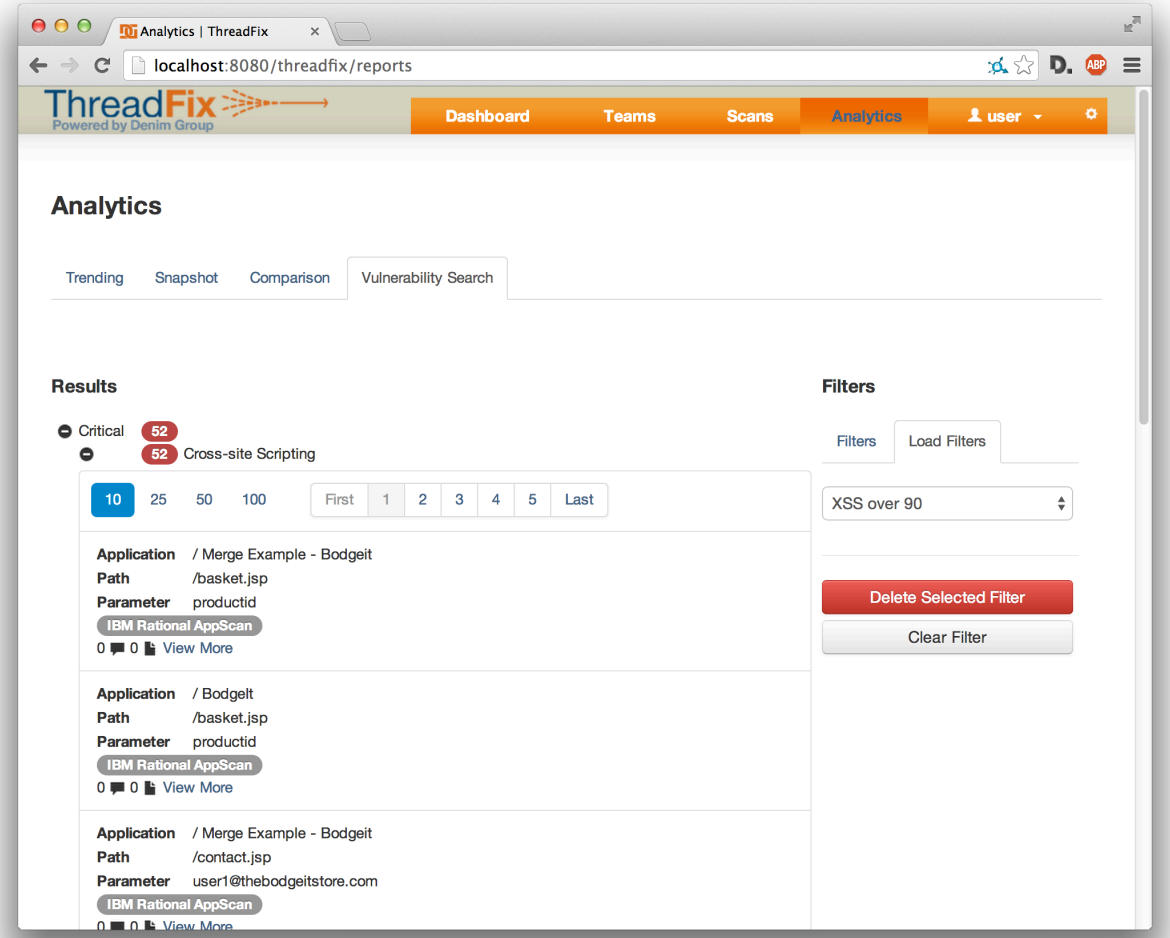
Prioritize application risk decisions based on data

Vulnerability Filtering

- Filter vulnerability data
  - *Scanner, scanner count*
  - *Vulnerability type*
  - *Path, parameter*
  - *Severity*
  - *Status*
  - *Aging*

- Save filters for future use

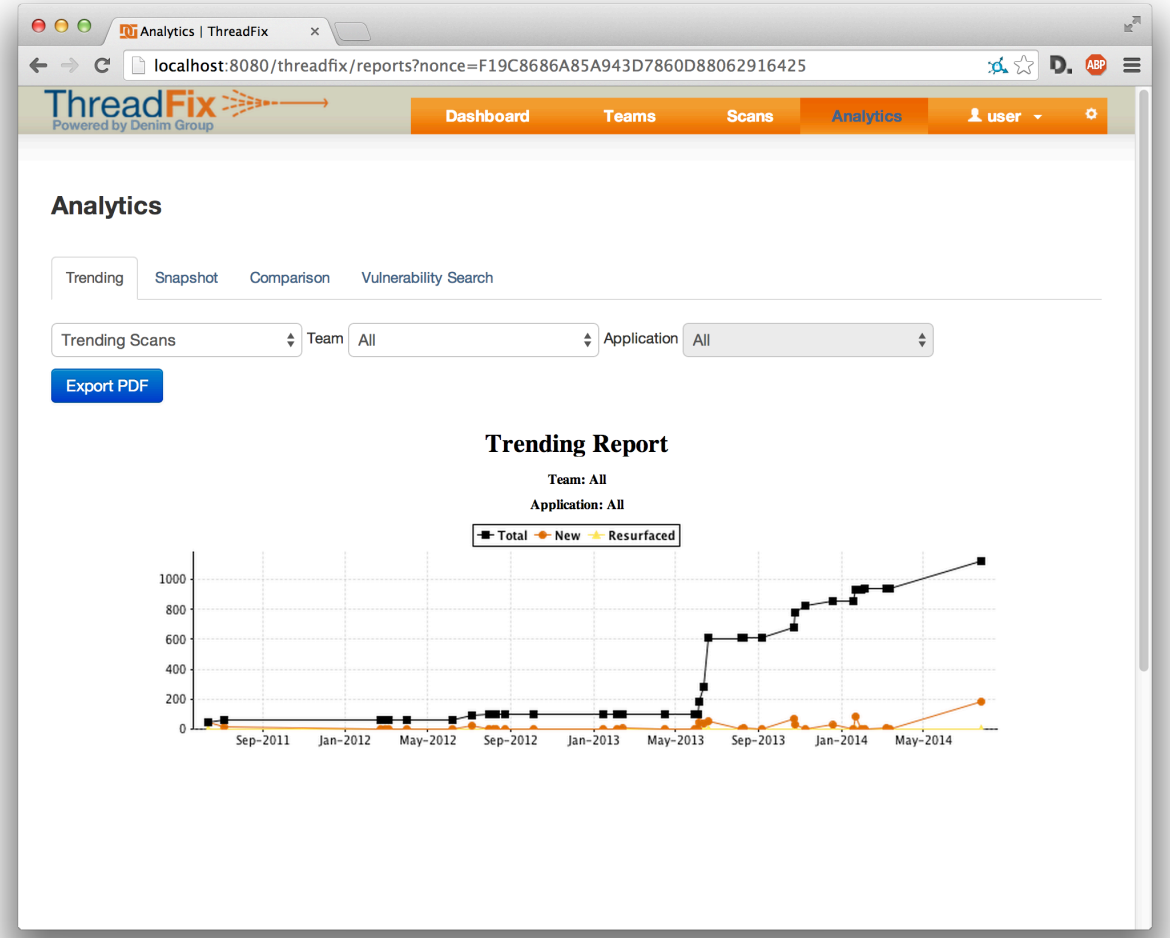
Demo: Vulnerability Filtering



Reporting

- Trending
- Progress by Vulnerability
  - *For program benchmarking*
- Portfolio Report
  - *For resource prioritization*
- Comparison
  - *For scanner/technology benchmarking*

Demo: Reporting



Translate vulnerabilities to developers in the tools they are already using

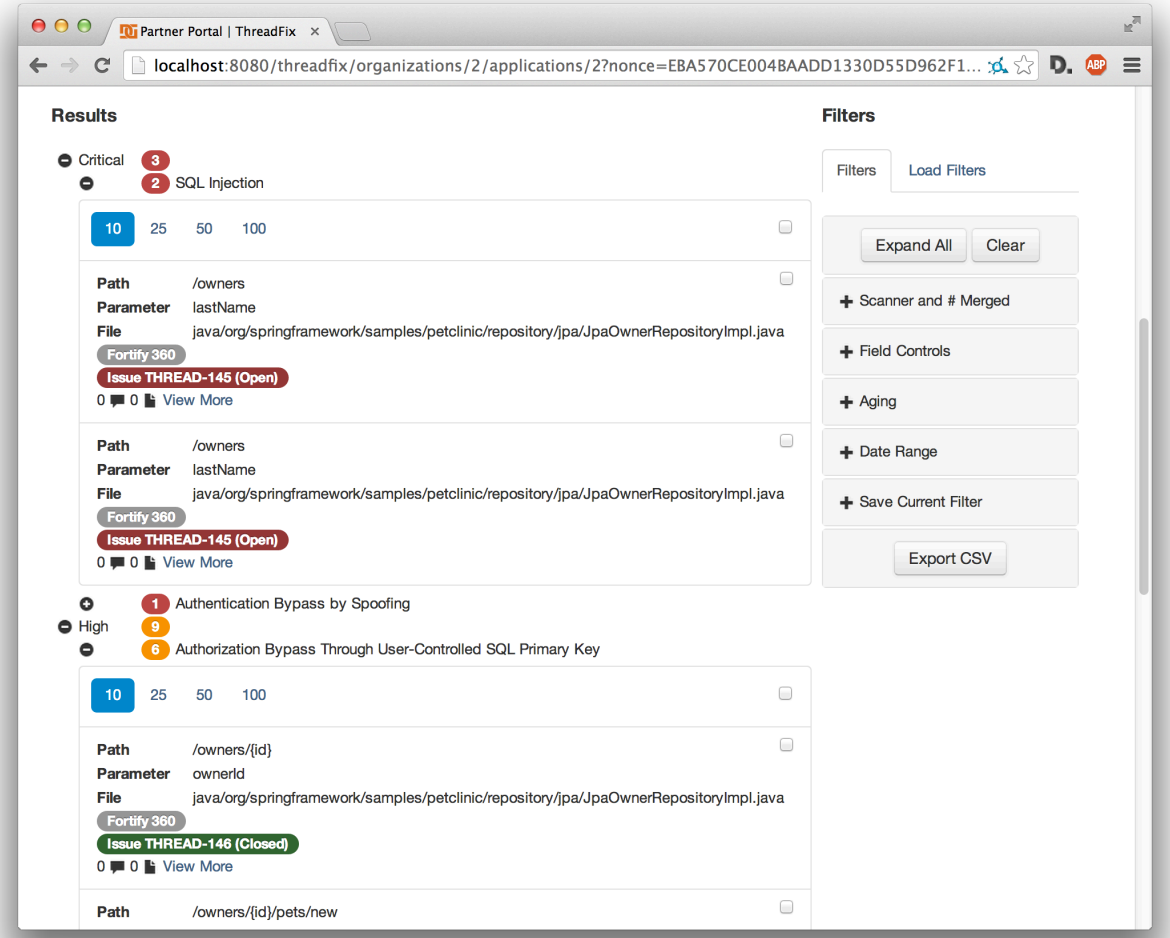
Mapping Vulnerabilities to Defects

- 1:1 mapping is (usually) a horrible idea
  - *500 XSS turned into 500 defects?*
  - *If it takes longer to administer the bug than it does to fix the code…*
- Cluster like vulnerabilities
  - *Using the same libraries / functions*
  - *Cut-and-paste remediation code*
  - *Be careful about context-specific encoding*
- Combine by severity
  - *Especially if they are cause for an out-of-cycle release*
- Which developer “owns” the code?

Defect Tracker Integration

- Bundle multiple vulnerabilities into a defect
  - *Using standard filtering criteria*
- ThreadFix periodically updates defect status from the tracker

Demo: Defect Tracker Integration

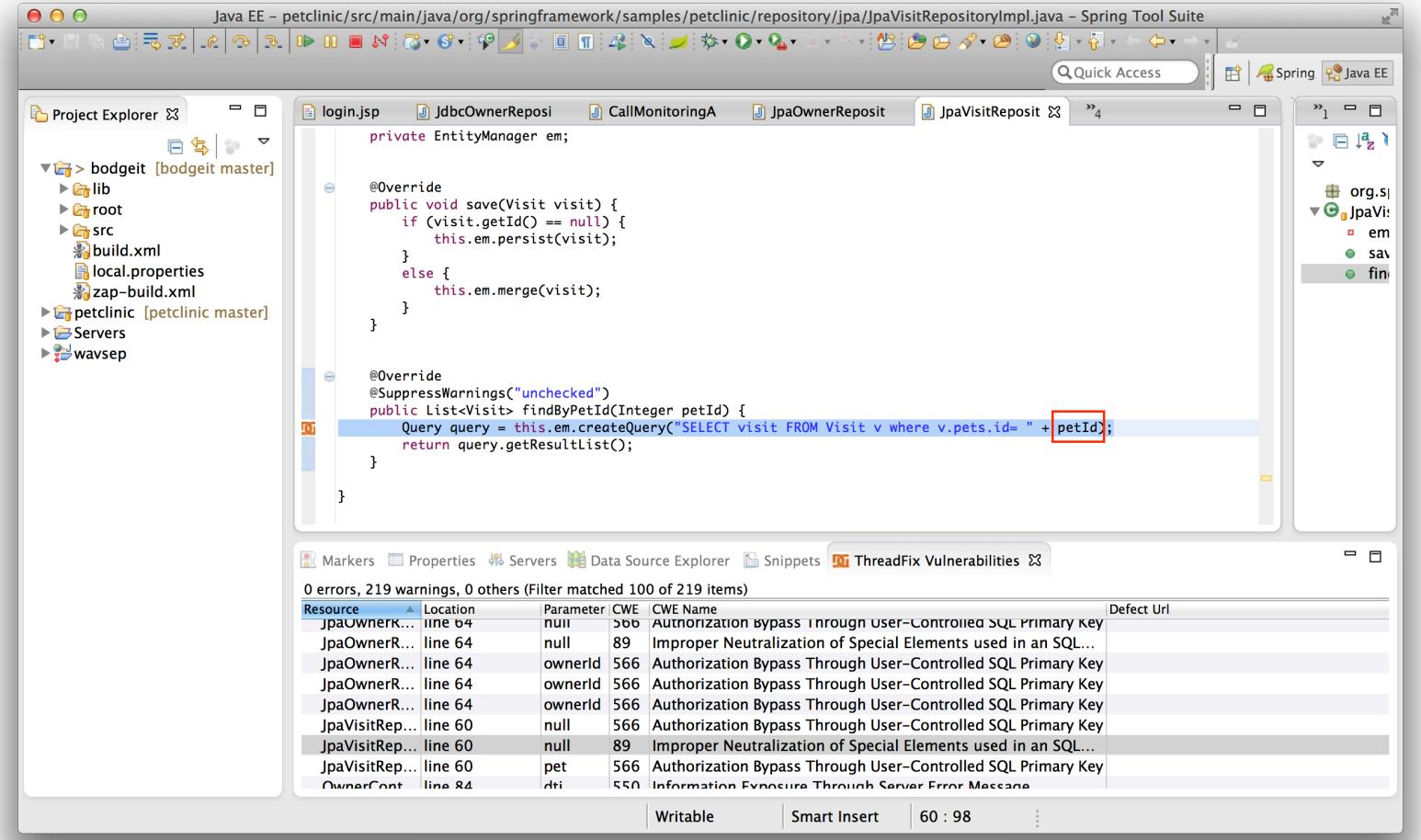


IDE Plug Ins

- Import vulnerability data to integrated development environments (IDEs)

- Static (SAST) scanners
  - *Easy*
- Dynamic (DAST) scanners
  - *Possible using Hybrid Analysis Mapping (HAM)*

Map Dynamic Scan Results to LoC in IDE



Important Links

- Main ThreadFix website: www.threadfix.org
  - *General information, downloads*
- ThreadFix GitHub site: www.github.com/denimgroup/threadfix
  - *Code, issue tracking*
- ThreadFix GitHub wiki: https://github.com/denimgroup/threadfix/wiki
  - *Project documentation*
- ThreadFix Google Group: https://groups.google.com/forum/?fromgroups#!forum/threadfix
  - *Community support, general discussion*

Questions / Contact Information

Dan Cornell
Principal and CTO
dan@denimgroup.com
Twitter @danielcornell
(210) 572-4400

www.denimgroup.com
www.threadfix.org