

Architecture Analysis

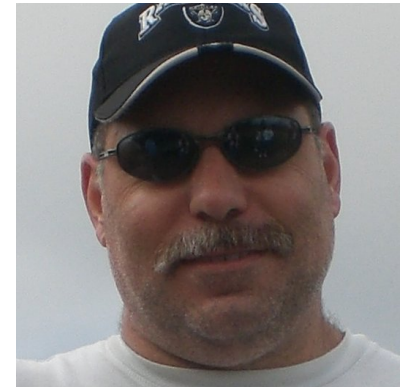
JIM DELGROSSO

@JIMDELGROSSO
DELNET2013(AT)CIGITAL.COM

Introduction

Jim DelGrosso

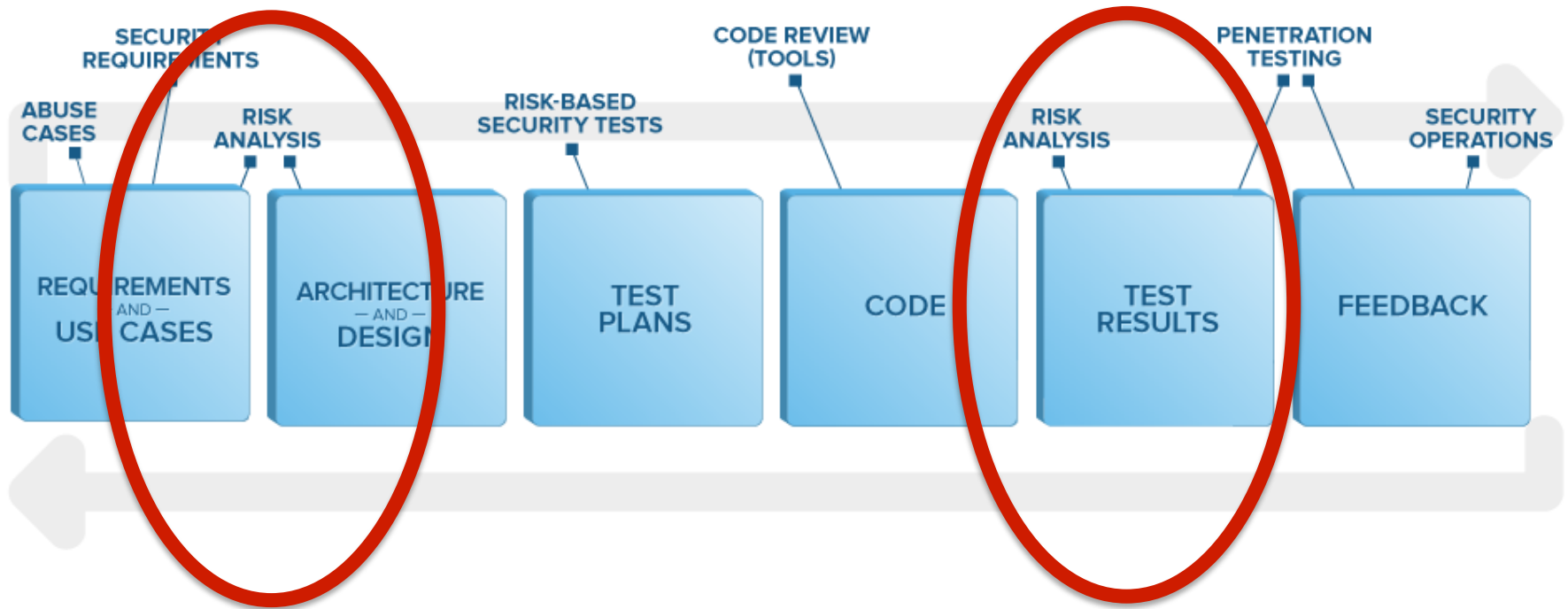
- Spend a great deal of time working with companies to find security design flaws
- Run Cigital's Architecture Analysis practice
- 20+ years in software development in many different domains
- ~15 years focusing on software security
- Executive Director of IEEE CS CSD initiative



@jimdelgrosso



Software Security In The SDLC



The Software Defect Universe



Cross Site Scripting
Buffer Overflow

(Implementation) BUGS

Code Review

Penetration Testing

Architecture Analysis







Weak/Missing/Wrong
Security Control



(Design) FLAWS

Bugs vs. Flaw Comparison





Cryptography Defects

| Description | Bug | Flaw |
|---|---|---|
| Use a weak IV or key with a crypto primitive |  | |
| Use a confidentiality control where an integrity control is necessary | |  |
| Hardcoded key in source code |  |  |

Authentication Defects

| Description | Bug | Flaw |
|--|---|--|
| LDAP Injection |  | |
| Two-step authentication process with hidden user account, performed on client side | |  |

Logging Defects

| Description | Bug | Flaw |
|--|--|---|
| Allow logs to be altered without detection | |  |
| Writing sensitive data to logs |  | |
| Log Injection |  | |
| Not tokenizing sensitive data for easy log aggregation | |  |

How To Find Flaws?

- Code review?
 - Unlikely with tool; maybe by manual review
- Pen-testing?
 - Unlikely without deep knowledge of system; and possibly a lot of test time; and possibly access to back-end systems
- Need something else...
 - Analysis that is not code-based
 - Analysis focusing on how system is designed

How To Find Flaws?

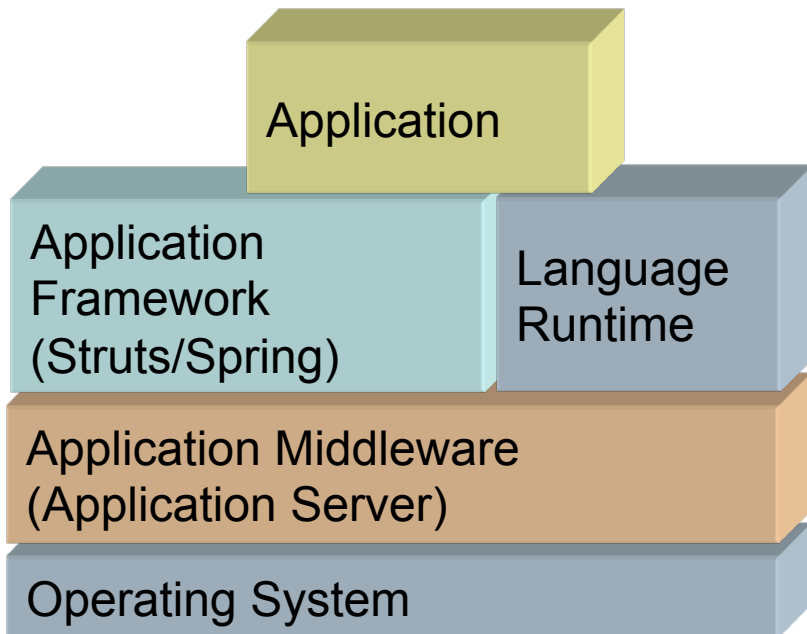
- **Dependency Analysis**
- **Known Attack Analysis**
- **System Specific Analysis**

Finding Flaws

DEPENDENCY ANALYSIS

Dependency Analysis

Software is built upon layers of other software



What kind of flaws are found?

- Known vulnerabilities in open-source or product versions
- Weak security controls provided with the framework
- Framework features that must be disabled or configured to their secure form

Dependency Analysis

Sponsored by DHS/NCCIC/US-CERT

NIST National Institute of Standards and Technology

National Vulnerability Database

automating vulnerability management, security measurement, and compliance checking

| | | | | | | | |
|-----------------|------------|------------------|--------------------|----------------|------------|-----------------|------|
| Vulnerabilities | Checklists | Products/800-53A | Product Dictionary | Impact Metrics | Data Feeds | Statistics | FAQs |
| Home | SCAP | Related Tools | SCAP Events | About | Contact | Vendor Comments | |

Mission and Overview

NVD is the U.S. government repository of standards based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

Search Results (Refine Search)

There are **42** matching records.
Displaying matches **1** through **20**.

Search Parameters:

- **Keyword (text search):** ruby rails
- **Search Type:** Search Last 3 Years
- **Contains Software Flaws (CVE)**

1 2 3 > >>

Resource Status

NVD contains:

- 68647 [CVE Vulnerabilities](#)
- 278 [Checklists](#)
- 248 [US-CERT Alerts](#)
- 4326 [US-CERT Vuln Notes](#)
- 10286 [OVAL Queries](#)
- 100871 [CPE Names](#)

Last updated: 2/9/2015 12:03:15 PM
CVE Publication rate: 24.13

Email List

NVD provides four mailing lists to the public. For information and subscription instructions please visit [NVD Mailing Lists](#)

CVE-2014-7829

Summary: Directory traversal vulnerability in actionpack/lib/action_dispatch/middleware/static.rb in Action Pack in Ruby on Rails 3.x before 3.2.21, 4.0.x before 4.0.12, 4.1.x before 4.1.8, and 4.2.x before 4.2.0.beta4, when serve_static_assets is enabled, allows remote attackers to determine the existence of files outside the application root via vectors involving a \ (backslash) character, a similar issue to CVE-2014-7818.
Published: 11/18/2014 6:59:03 PM

CVSS Severity: 5.0 MEDIUM

CVE-2014-7819

Summary: Multiple directory traversal vulnerabilities in server.rb in Sprockets before 2.0.5, 2.1.x before 2.1.4, 2.2.x before 2.2.3, 2.3.x before 2.3.3, 2.4.x before 2.4.0, 2.5.x before 2.5.1, 2.6.x and 2.7.x before 2.7.1, 2.8.x before 2.8.3, 2.9.x before 2.9.4, 2.10.x before 2.10.2, 2.11.x before 2.11.3, 2.12.x before 2.12.3, and 3.x before 3.0.0.beta.3, as distributed with Ruby on Rails 3.x and 4.x, allow remote attackers to determine the existence of files outside the application root via a ../ (dot dot slash) sequence with (1) double slashes or (2) URL encoding.
Published: 11/8/2014 6:55:03 AM

CVSS Severity: 5.0 MEDIUM

CVE-2014-7818

Summary: Directory traversal vulnerability in actionpack/lib/action_dispatch/middleware/static.rb in Action Pack in Ruby on Rails 3.x before 3.2.20, 4.0.x before 4.0.11, 4.1.x before 4.1.7, and 4.2.x before 4.2.0.beta3, when serve_static_assets is enabled, allows remote attackers to determine the existence of files outside the application root via a /..%2F sequence.
Published: 11/8/2014 6:55:02 AM

Finding Flaws

KNOWN ATTACK ANALYSIS

Known Attack Analysis

Understanding known attacks provide insight

- Designers – what controls are needed to prevent them
- Attackers – what to try again



Known Attack Analysis

What defects show up “often”?

- Client-side trust
- Missing or weak control
 - XSS
 - CSRF
 - Logging and auditing
 - Click-jacking
- Session management

Known Attack Analysis

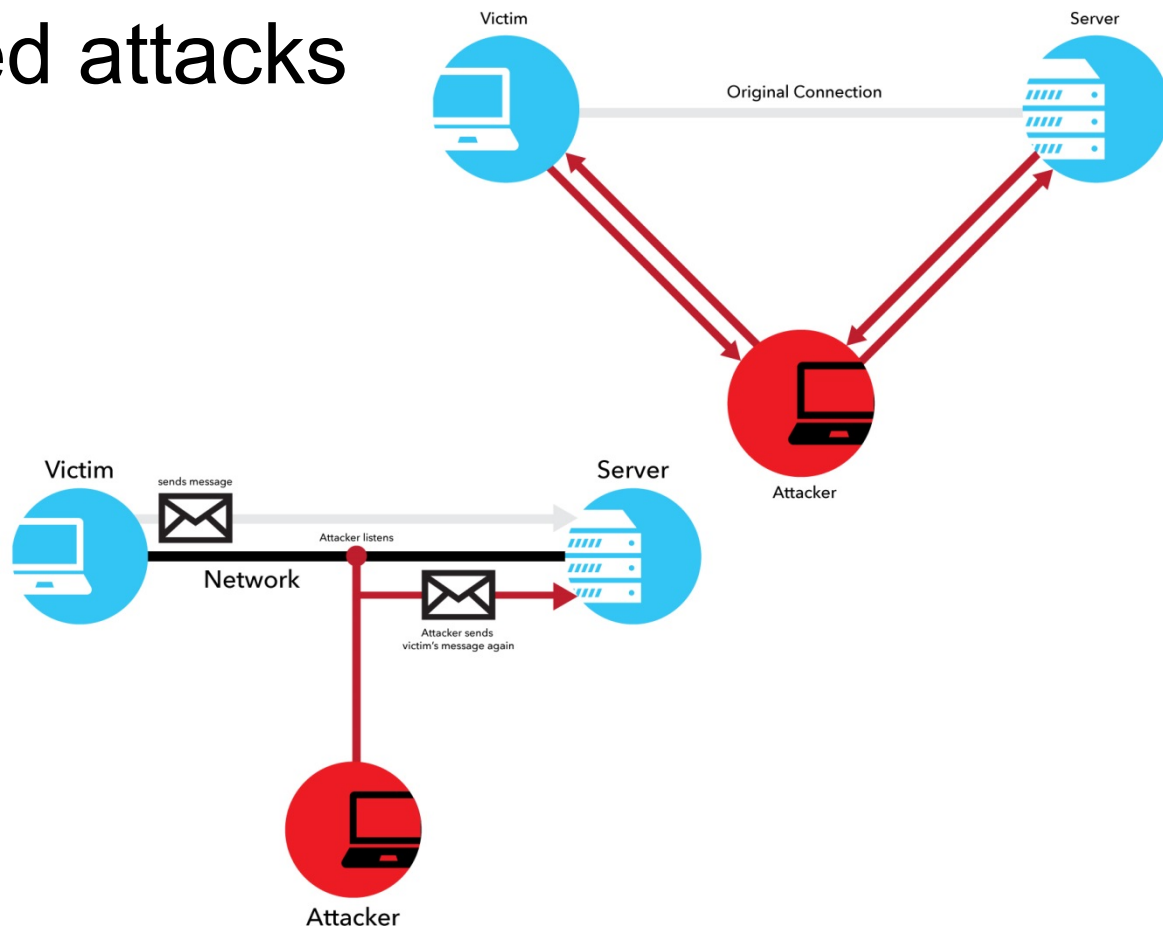
Identify design elements historically vulnerable to attack

- Distributed architecture
- Dynamic code generation and interpretation
- APIs across stateless protocols
- Client code – RIA, Mobile, ...
- Service-Oriented Architecture

Distributed Architecture

- Distributed systems are susceptible to network-based attacks

- Eavesdrop
- Tamper
- Spoof
- Hijack
- Observe
- Replay



Dynamic Code Generation and Interpretation

- Languages and programming environments are moving more decisions from design-time to run-time
- Many attacks involve misinterpretation of data as code in these environments
- When and how will user input be used by runtime language interpreters?

APIs Across Stateless Protocols

- Identifiers representing state can be abused
 - Prediction
 - Capture
 - Fixation
- State sent to the client between requests is altered or replayed

Client Code – RIA, Mobile, ...

- Processing moved to the client
 - RIA
 - Mobile
 - HTML5
- It is still a client
- It is still an untrusted platform
- An exposed server endpoint is exposed to everyone – not just for your purposes

Service-Oriented Architecture (SOA)

- Security needed for SOA components
 - Web-services: SOAP/WSDL/UDDI
 - Message-oriented middleware
 - Enterprise Service Bus
- Common Problems
 - Exposing backend code to dynamic attacks
 - Channel versus message security

Finding Flaws

SYSTEM SPECIFIC ANALYSIS

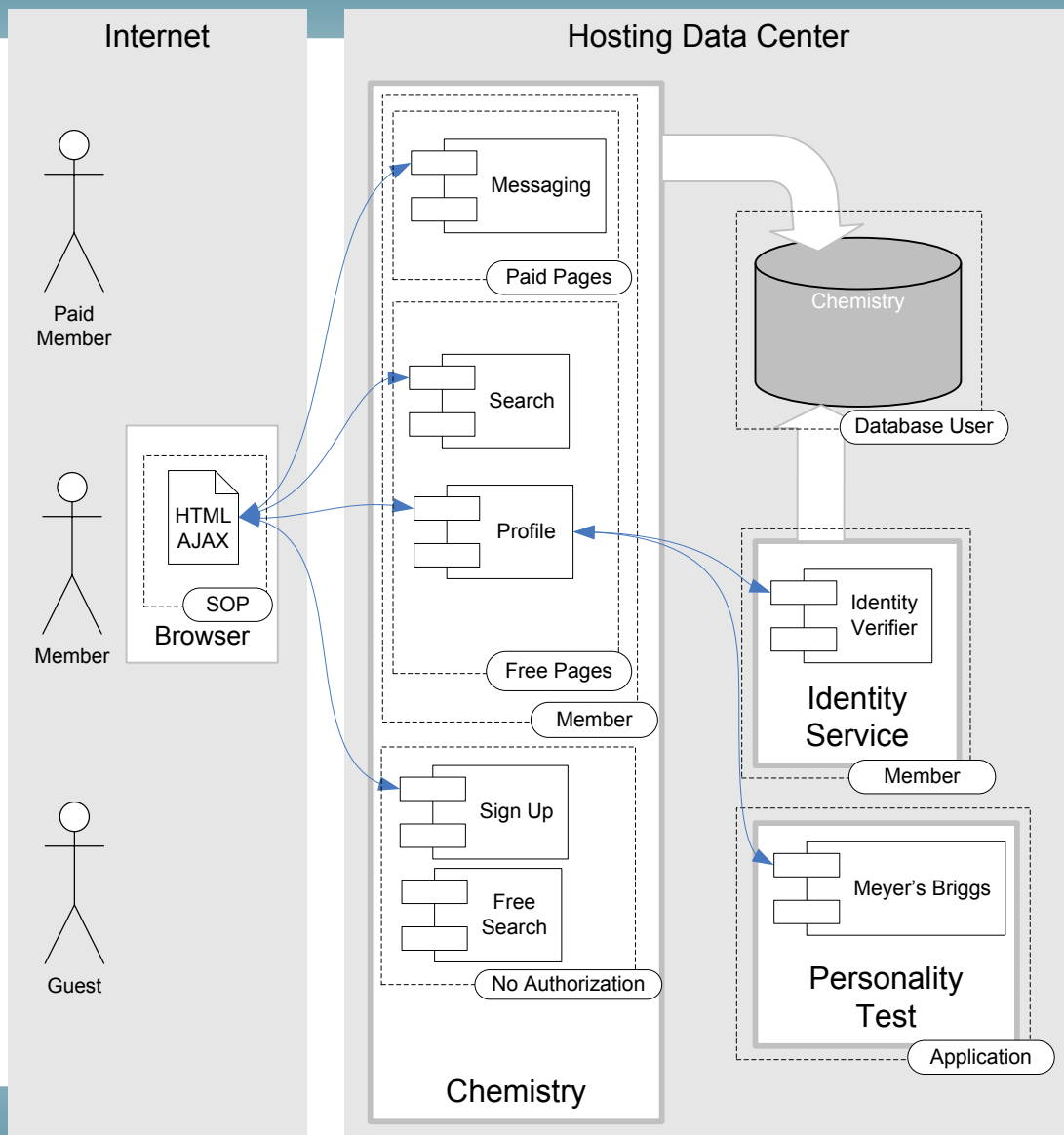
System Specific Analysis Flaws

- Weakness in a custom protocol
- Reusing authentication credentials
- Not following good software security design principles

Model the software by understanding

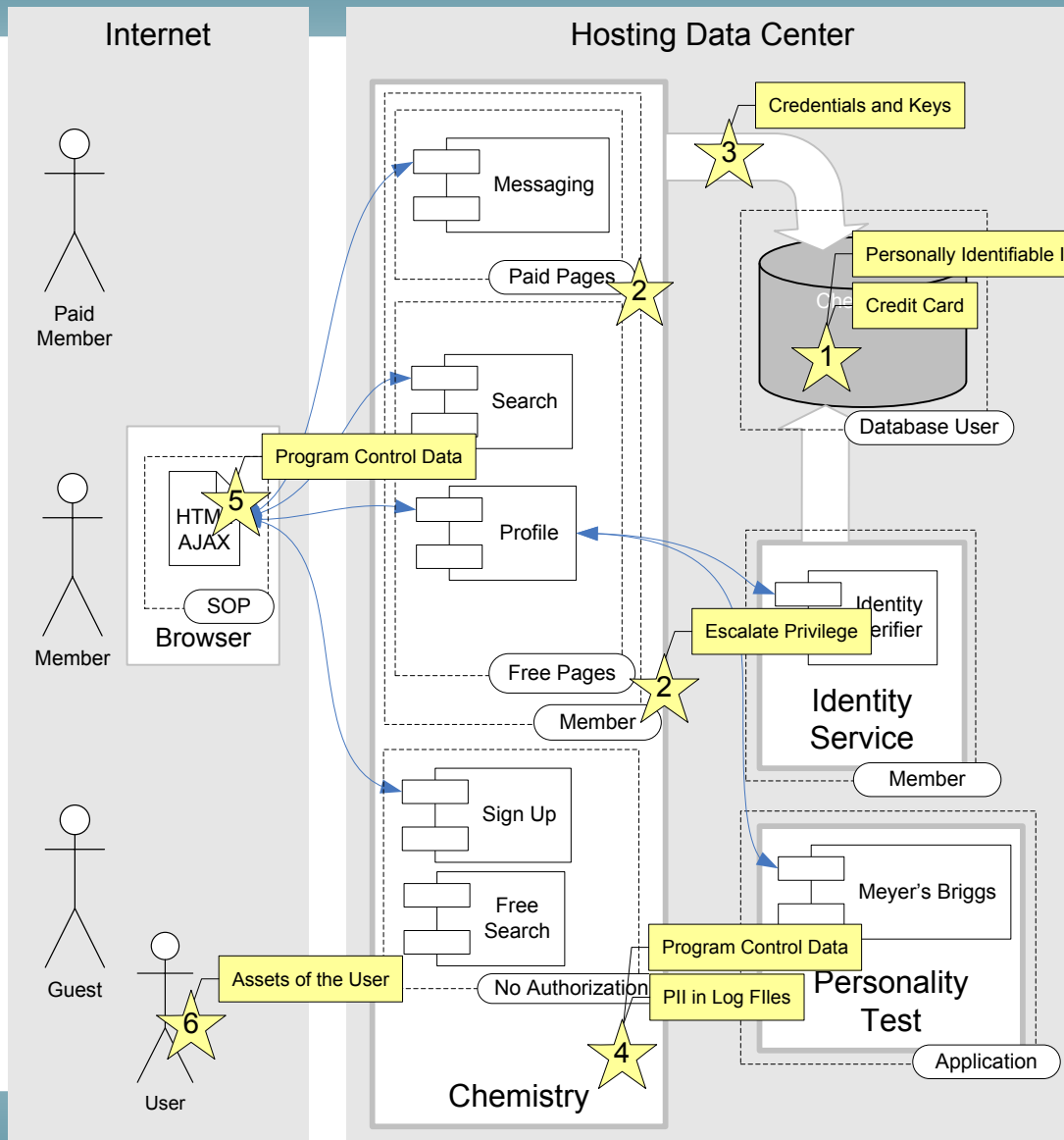
- Threat agent
- Asset
- Attack
- Attack surface
- Attack goal
- Security control

Who Is Attacking You?



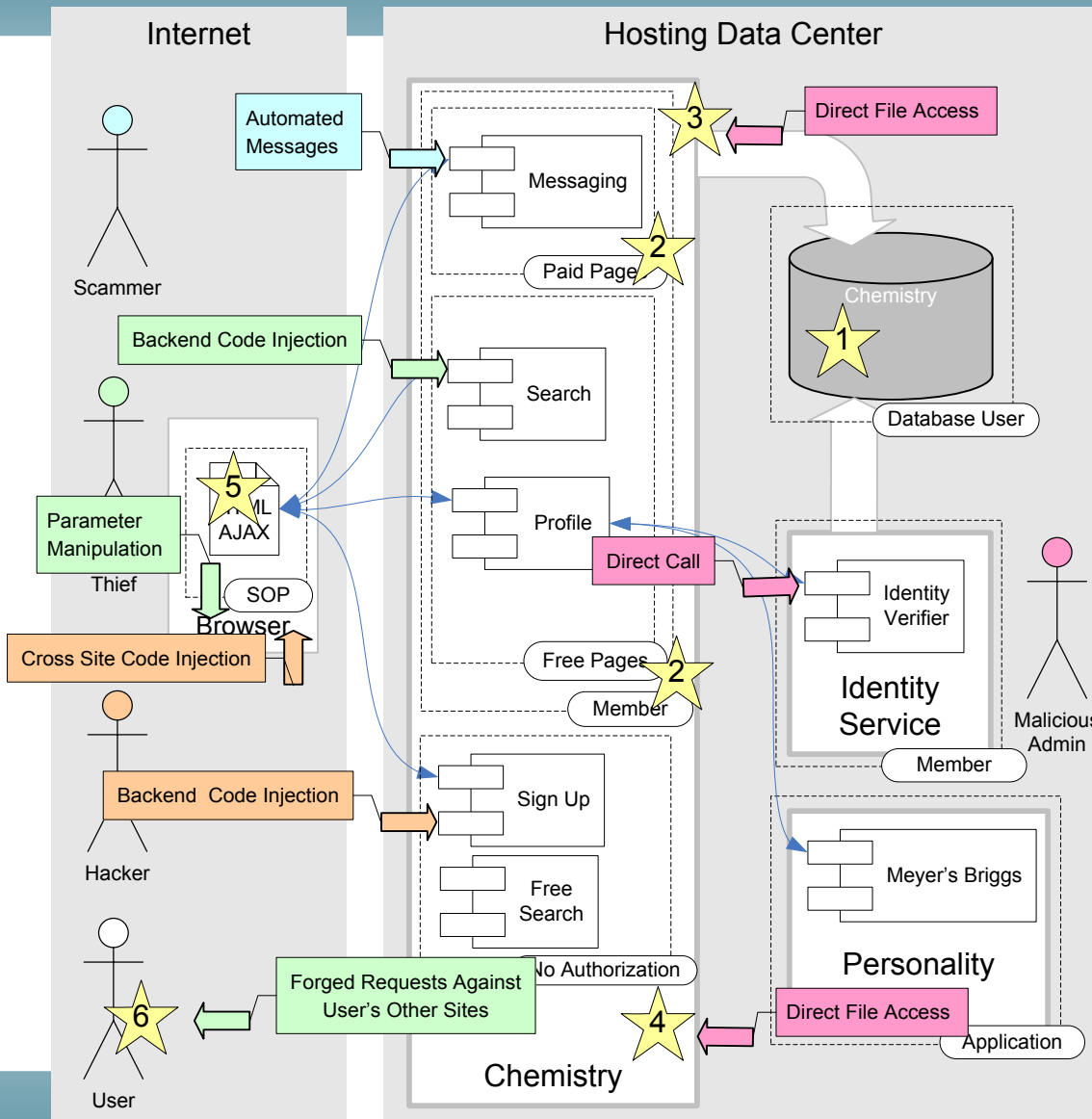
- Threat Agents are users that have malicious intent
- Like users they have capabilities within the system
- Threat Agents have a goal that usually involves subverting a security control

What Are You Trying To Protect?



- Assets are the application's functions
- Assets are the application's sensitive data
- Assets are the application's users, and assets of other systems the user can access

How Will You Be Attacked?



- Examine how a Threat Agent will try to reach an Asset
- Threat Agents will attack nearest, easiest targets first
- Designers: look to place controls around Assets
- Threat Agents: start with direct attacks and graduate to multi-step

Why Architecture Analysis Is Necessary

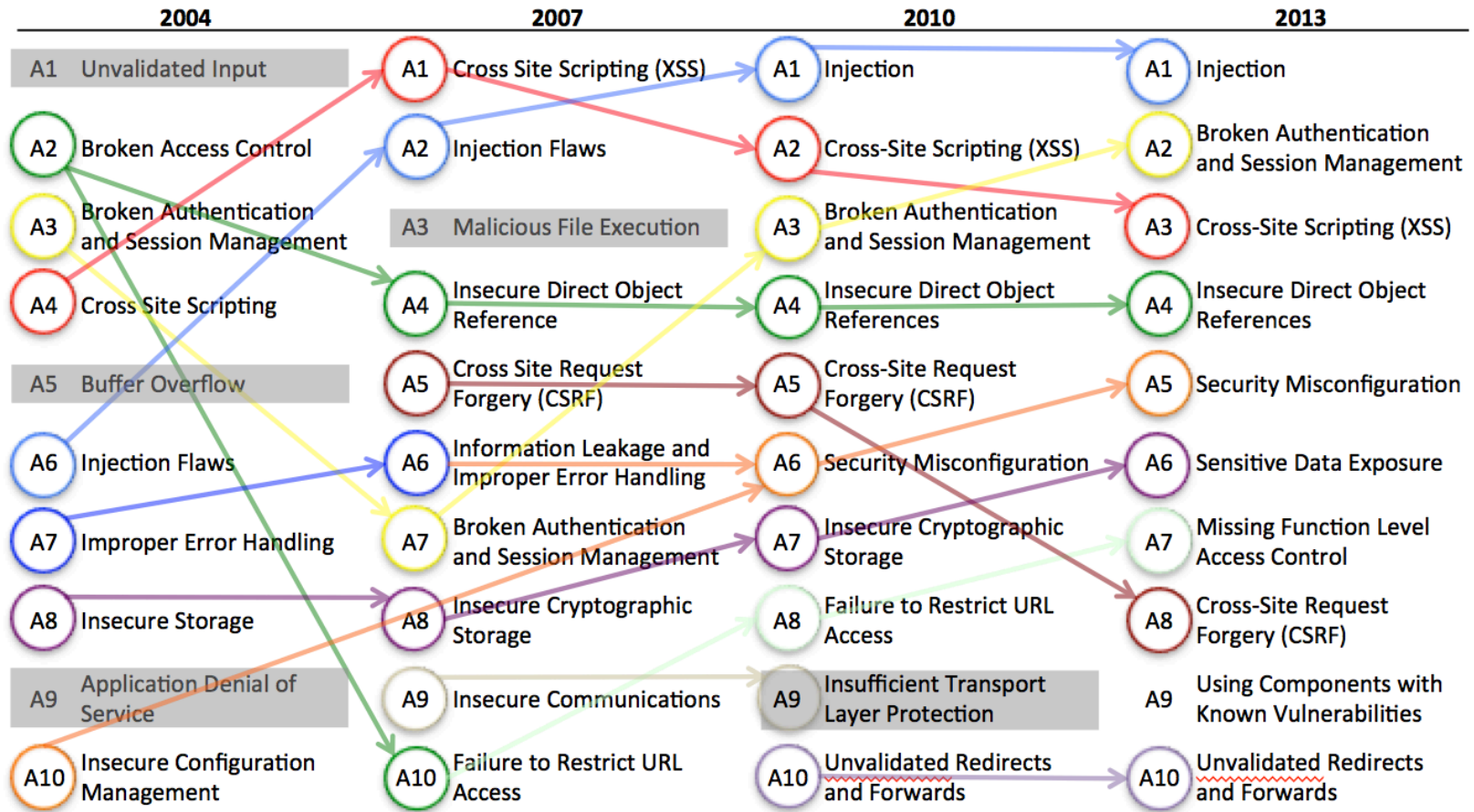
Architecture Analysis Finds Flaws

Poor key management example

- Hard-coded crypto keys
- PT would likely miss this
- SCR would probably flag it as a key management issue
- AA would fix the design



History Repeating Itself



Knowing != Avoiding

Some Flaws You Might Be Missing – IEEE CSD

- Earn or give, but never assume, trust
- Use an authentication mechanism that cannot be bypassed or tampered with
- Authorize after you authenticate
- Strictly separate data and control instructions, and never process control instructions received from untrusted sources
- Define an approach that ensures all data are explicitly validated
- Use cryptography correctly
- Identify sensitive data and how they should be handled
- Always consider the users
- Understand how integrating external components changes your attack surface
- Be flexible when considering future changes to objects and actors

Challenges

Challenge – Assumptions Are Evil

Assuming systems are hardened

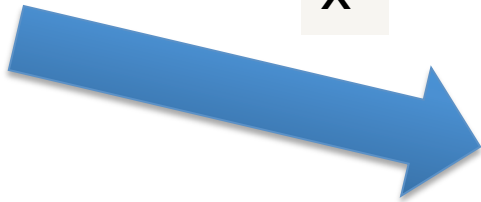
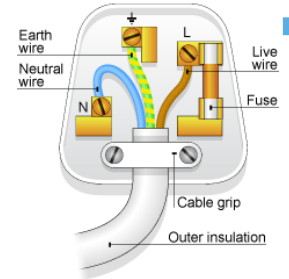
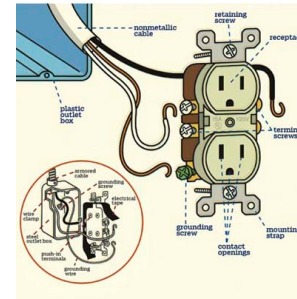
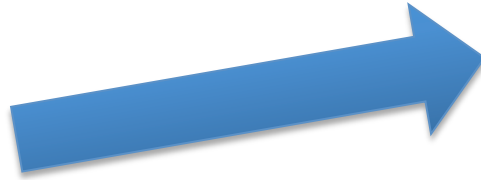
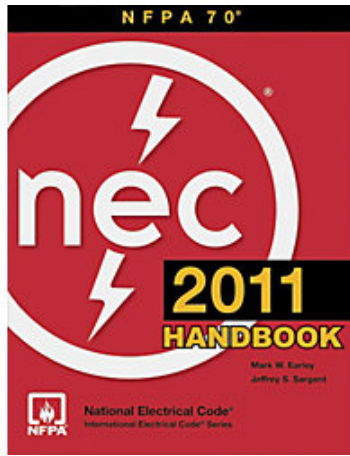
Assuming nothing sensitive is sent to the client

Assuming the fundamentals are done well

Assuming the overall design has been looked at after the initial design – maybe many years ago

Assuming that because your organization has a secure process defined, that process is followed

Challenge – Some of This Is Hard Stuff



Not just “book” training
Some of this requires apprenticeship

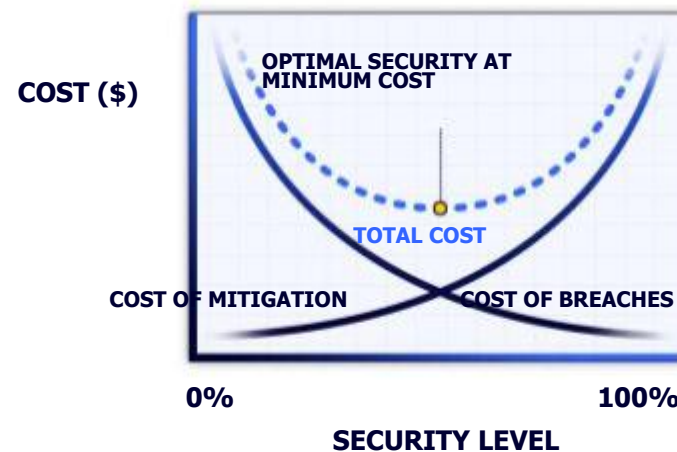
Challenge – Too Much Too Soon



Wrap-Up

Modern Security Is About *Managing* Risks

- There is no such thing as 100% secure
 - Must make tradeoffs
 - Should be business decisions
- Proactive security is about building things right
 - Software security
 - Security in the SDLC
- Security is not a *function*
- Most security problems are caused by software bugs and flaws
- We must build secure software



Architecture Analysis Wrap-Up

- Helps you find flaws
- Does **NOT** replace other techniques
- Human-driven analysis (minimal tool support)
- Some fixes require long-term solutions
 - Risk mitigation is key
- Apprenticeship

The text "Thank You" is written in a large, white, sans-serif font. To the left of the text, there are several overlapping blue and white squares of varying sizes, creating a decorative graphic element.

Thank You