cigital

# Threat Modeling & Architectural Analysis

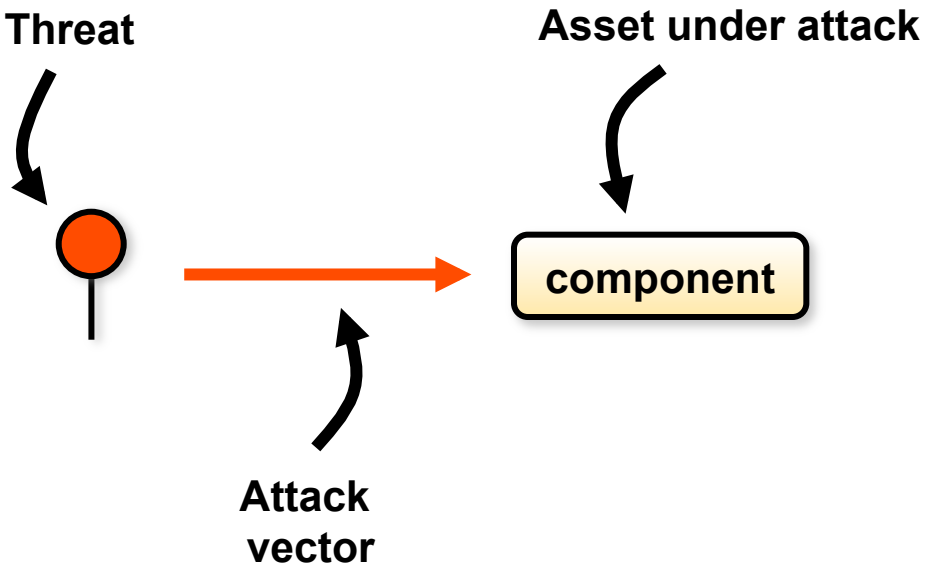
## Understanding Software Architecture to Expose Risk


www.cigital.com
info@cigital.com
+1.703.404.9293

# What is a Threat?

- An agent who attacks you?
- An attack?
- An attack's consequence?
- A risk?

- An agent




cigital

# Confusion Over "Threat"

- Some literatures equates "threat" to mean "a potential event that will have an unwelcome consequence"
  - Attacker views sensitive data
  - Attacker elevates privilege
- Devolves modeling to a checklist of events
- Expands thinking about possible abuse
  - Threats help
    - Encourage thorough thought about how intentions for misuse
    - Determine "out of bounds" scenarios
- **We refer to "threat" as a person or agent**

cigital

cigital

# Introduction

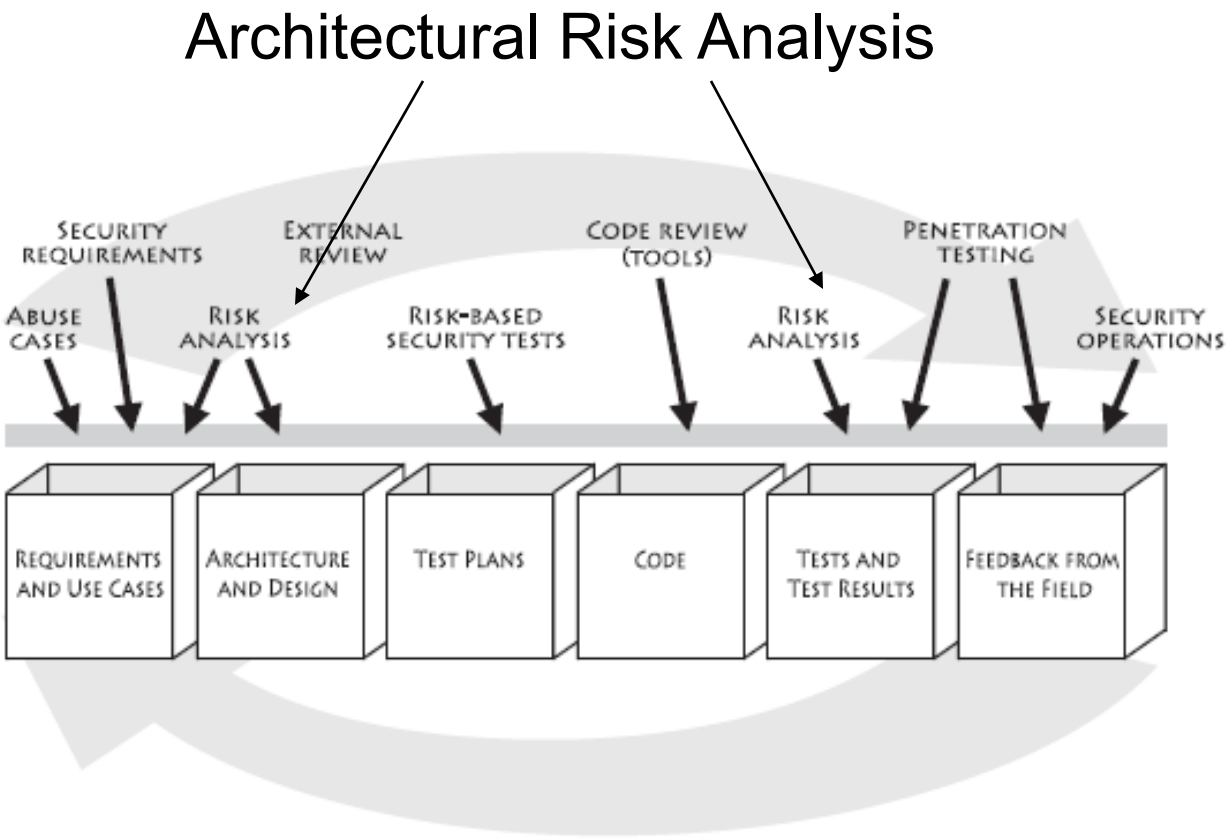
## What Is a Secure Architecture?

# Learning Objectives

- Learn how to evaluate and prioritize threats and attacks
- Learn how to build a Trust and Threat Model for a software system
- Learn how to analyze threats and attacks using a Trust and Threat Model
- Learn techniques for designing security controls

cigital

# You Are Here



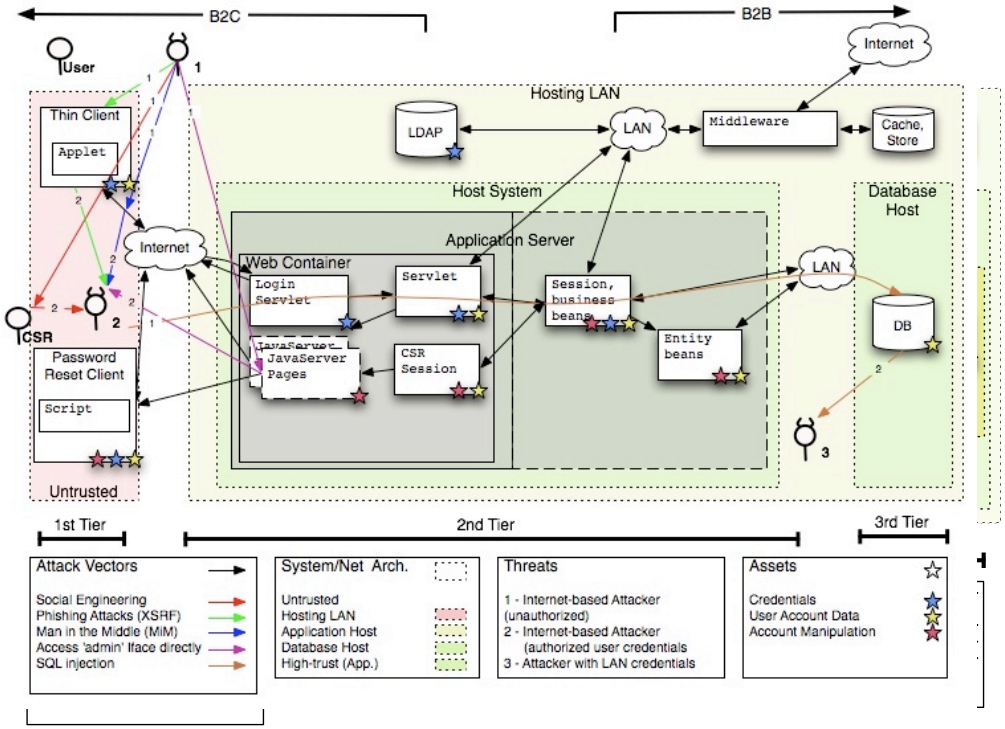
# What is a Threat Model

# What is a Threat Model

- Depiction of:
  - The system's *attack surface*
  - *Threats* who can attack the system
  - *Assets* threats may compromise
- Some leverage risk management practices
  - Estimate *probability* of attack
  - Weight *impact* of successful attack

cigital

# Elements of a Threat Model




- Structural view
- Behavioral Views
- Threat Actors
- Assets
- Attack Vectors
- Privilege / 'trust'

cigital

# The most effective Threat Modeling Tool

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| | | | | |

cigital

# How much is enough?

- Incrementally improve *from wherever you are*
- Think about organization's 'arch-types'
  - B2C, n-tier*
  - Mobile
  - B2B, Legacy
  - ATMs
  - RIA**
- Within each step, resist urge to do other steps
- Start with step for *corresponding SDL activity*
- Threat model what's new and different

cigital

# Alternative Models / Methods

# Security Goals

CIA

- Confidentiality

  limiting access and disclosure to "the right people“; preventing access by or disclosure to "the wrong people”.

- Integrity

  the trustworthiness of information resources

- Availability

  information systems provide access to authorized users

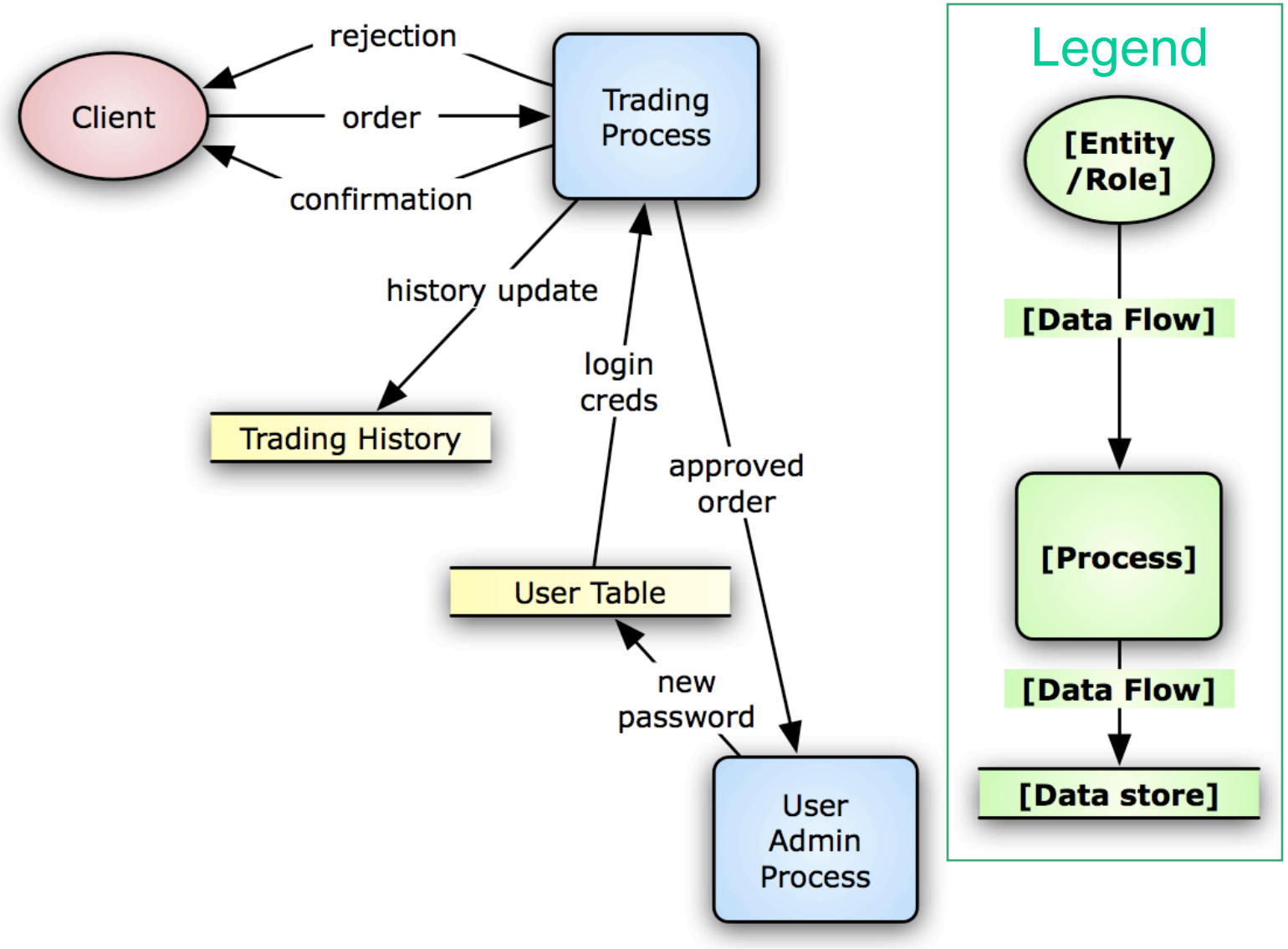
# A Few Words on STRIDE

- A conceptual attack checklist:
  - **S**poofing
  - **T**ampering
  - **R**epudiation
  - **I**nformation Disclosure
  - **D**enial of Service
  - **E**scalation of Privilege

  - Backed by DFDs

cigital

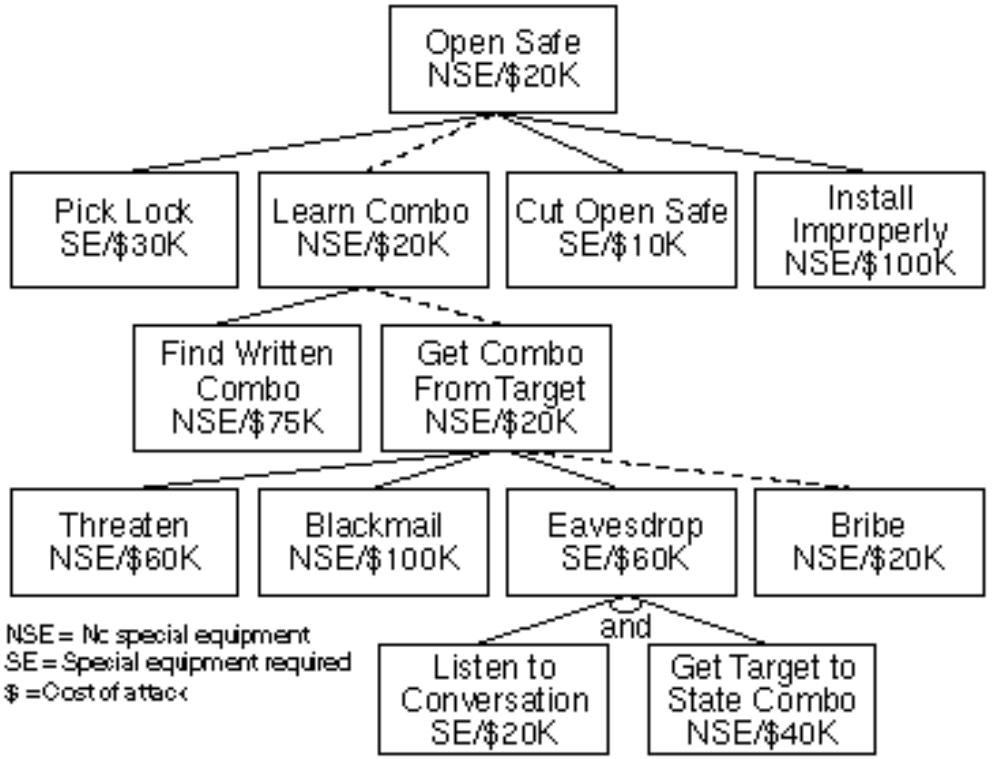
# An Example DFD




cigital

# Attack Trees

- Aggregate attack possibilities
- Use OR,AND
- Allow for decoration
  - Probability
  - Cost
  - Skills required, etc




From Bruce Schneier's Blog

cigital

# SSL Discussion

- SSL is infrastructure control to application
- SSL provides secure channel to do authentication
- SSL provides privacy after initial handshake
- Usually, only server is authenticated
- Client-side certs don't work (doesn't scale)

cigital

# Use Threat Modeling to Identify…

- Where potential threats exist relative to the architecture
- How threats escalate privilege
  - …become more formidable
- Specify vectors of attack
- Identifies components and assets worth protecting

… Ties technical risk & business assets to application design;

…Ties attacks to role, privilege, and capability;

…Drives security analysis, testing.

cigital

# The Early Cigital Model

Viega and McGraw

- Prevention
- Traceability and Auditing
- Monitoring
- Privacy and Confidentiality
- Multilevel Security
- Anonymity
- Authentication
- Integrity

- Evolve towards “Secure Design”
- *Ross Anderson’s* Security Engineering

cigital

# Threat Modeling as a Process

# Threat Modeling – High-level process

1. Diagram structure
2. Identify assets
3. Identify Threats

4. Enumerate doomsday scenarios
5. Document misuse/abuse

6. Architectural Risk Analysis
7. Iterate

cigital

# 1 - Diagram Software Structure

# Application Structure: No 'One Size Fits All'

- Network topology is a failure mode
- UML doesn't provide sufficient context
- "Boxes and spiders" don't provide value

- A 'one pager' is very important…
- Aspects of security will require different views

cigital

# 1.1 - Anchor in Software Architecture

Consider where attacks occur

Top-down

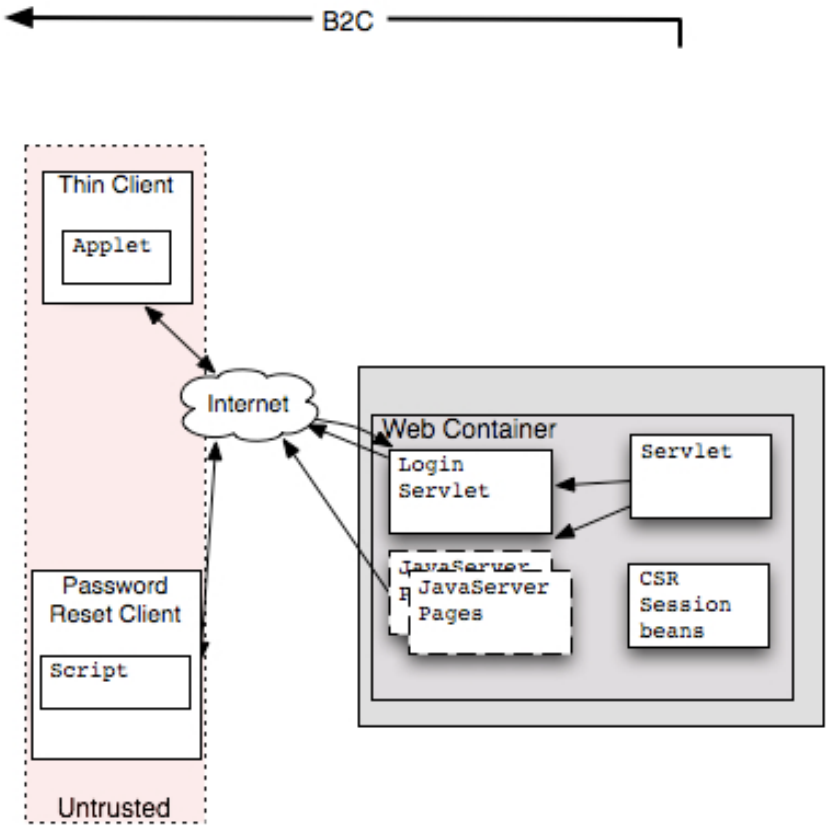
- Enumerate business objects
  - Sensitive data
  - Privileged functionality

Bottom-up

- Enumerate application entities
  - Sensitive data
  - Privileged functionality

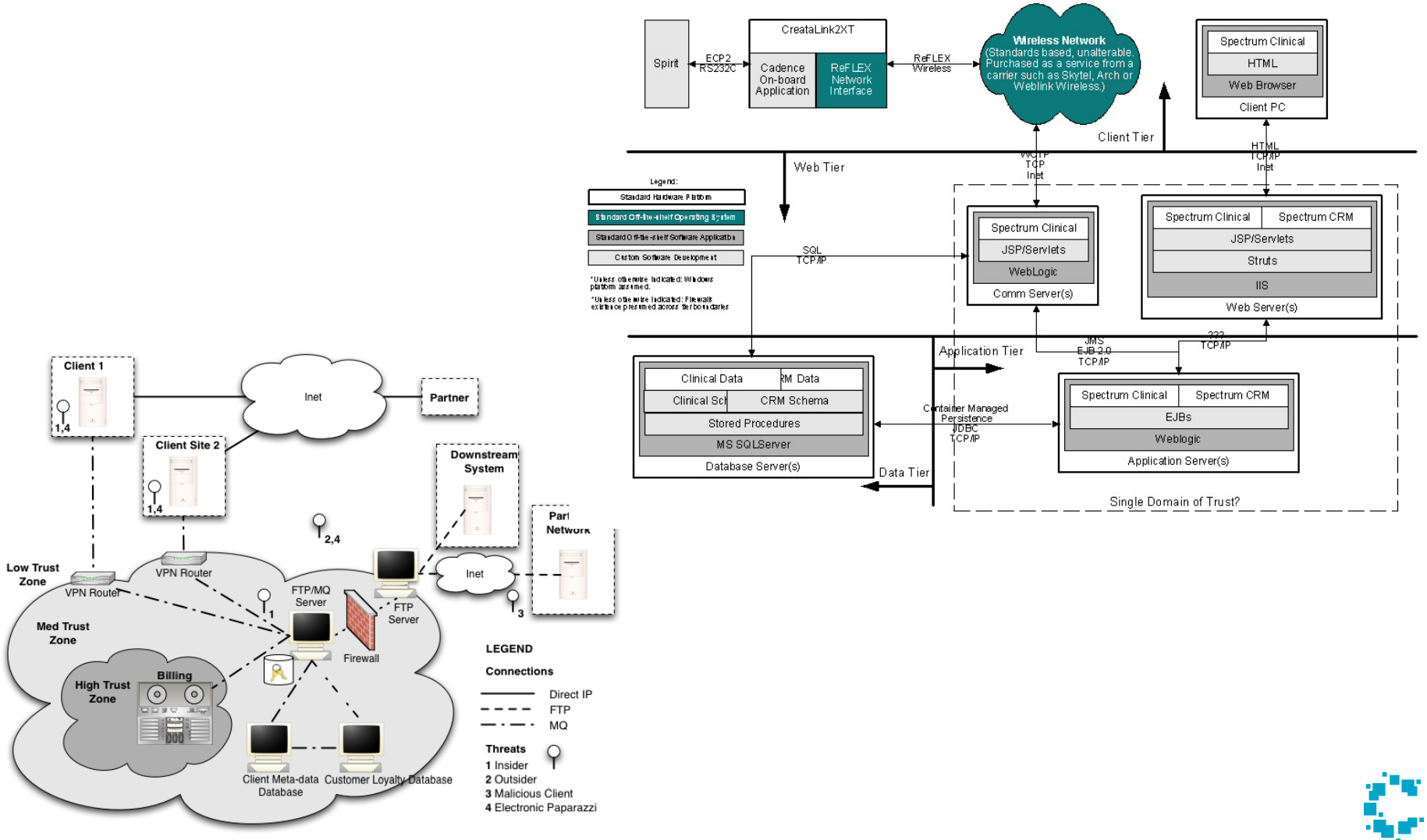
Look for

- Middleware
- Open source
- Frameworks

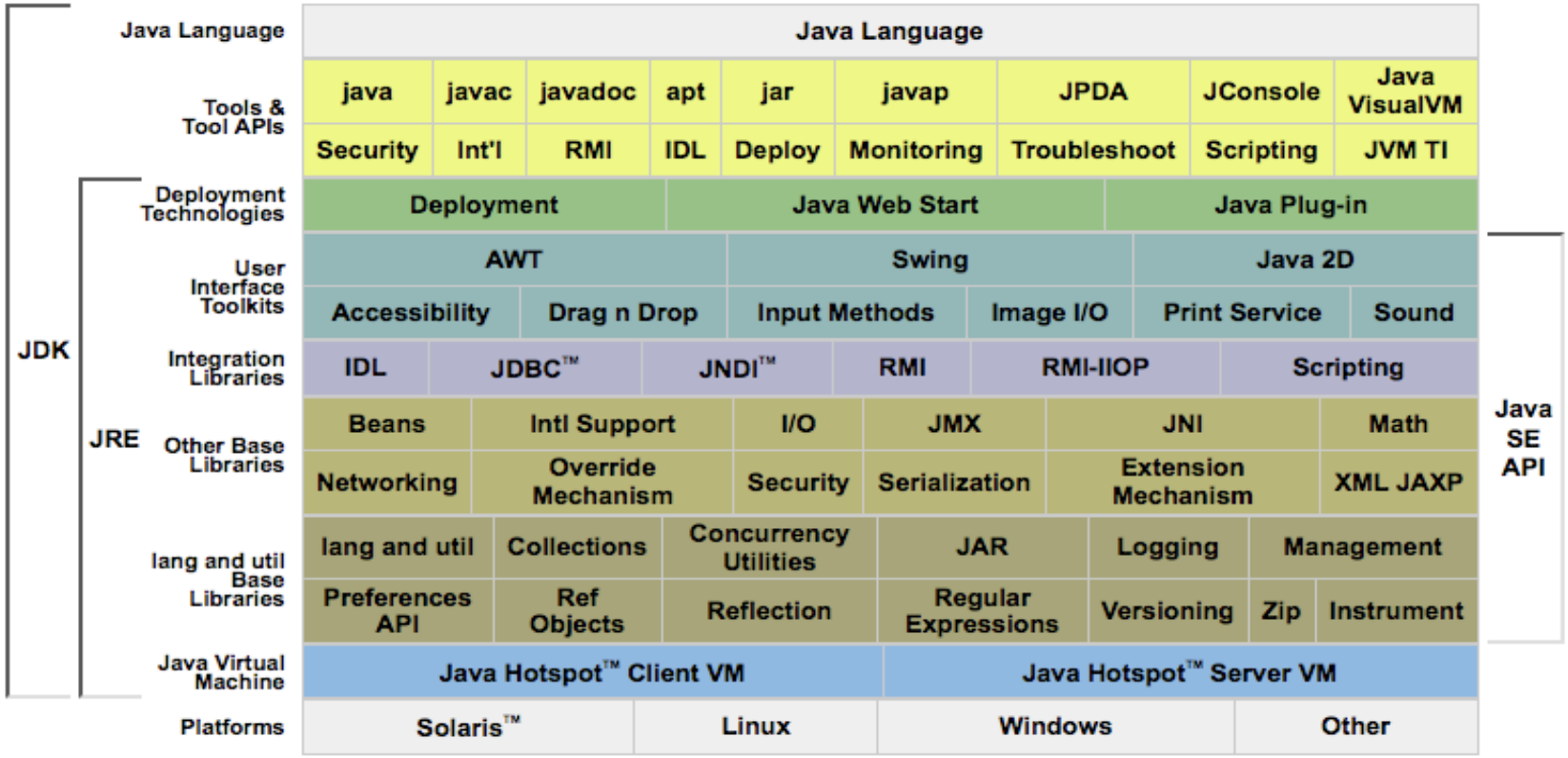



cigital

# Diagram Software at Correct Level



# Avoid ‘the stack’




What does this diagram tell you about component interaction?

cigital

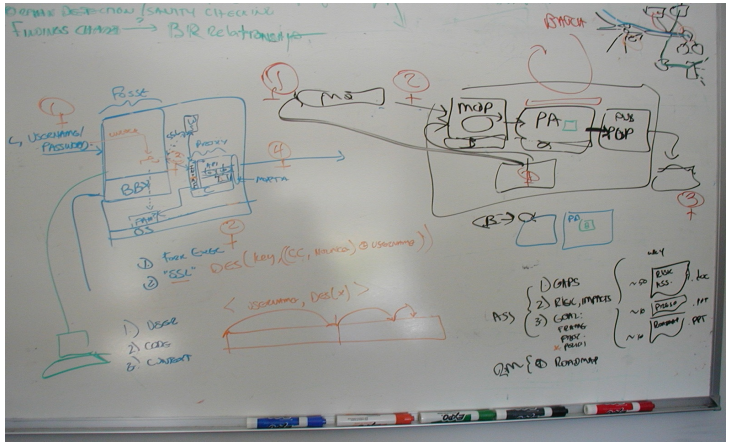
# Application Structure: No 'One Size Fits All'

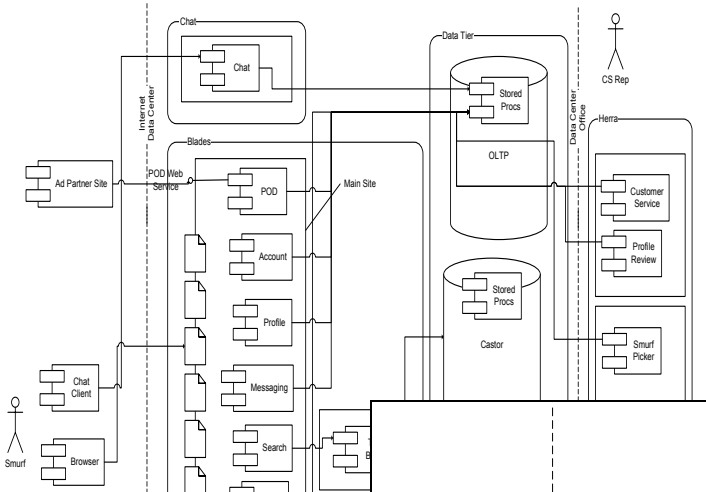
- Network topology is a failure mode
- UML doesn't provide sufficient context
- "Boxes and spiders" don't provide value

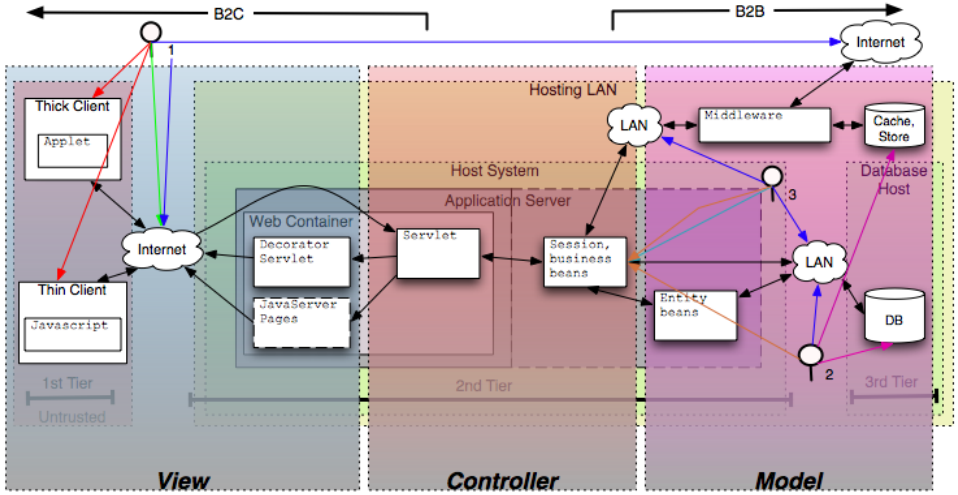
- A 'one pager' is very important…
- Aspects of security will require different views

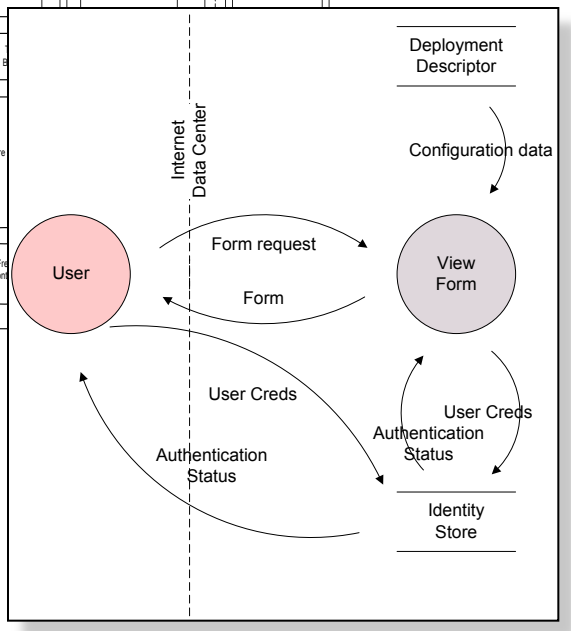
cigital

# Architecture Diagrams

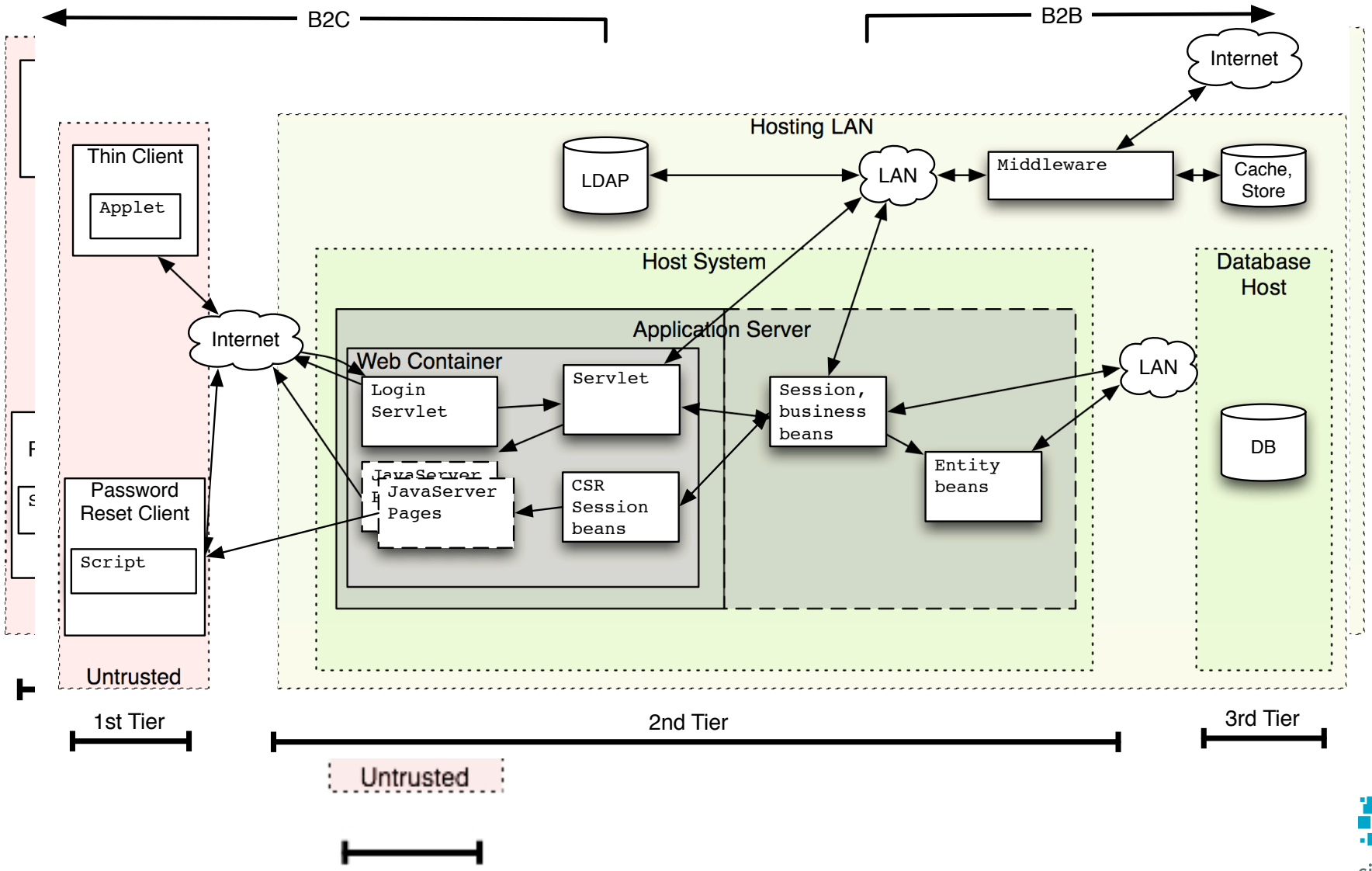




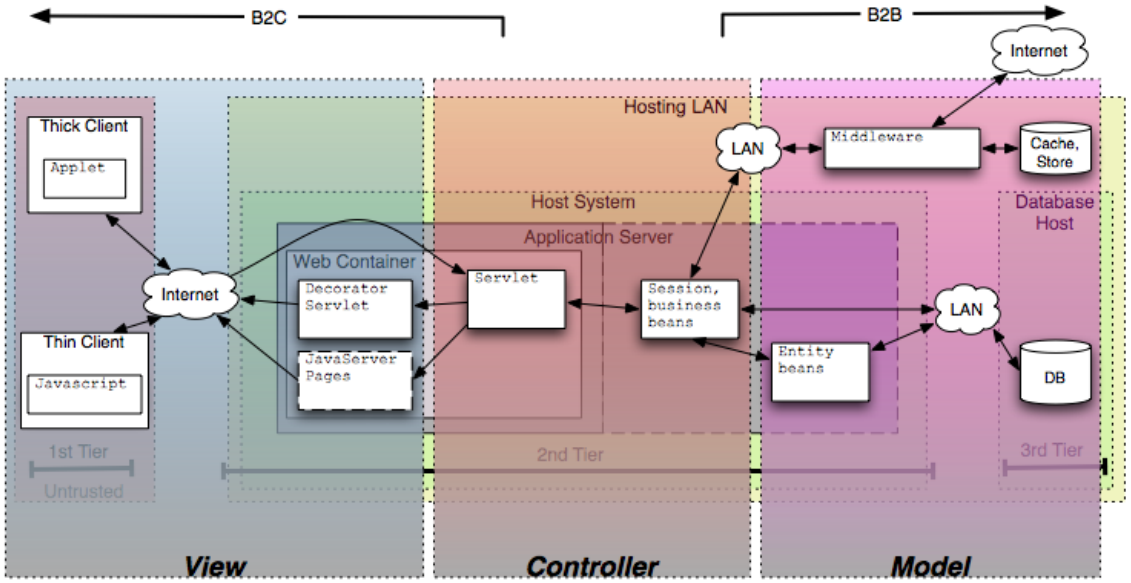



# 1.2 – Identify Application Attack Surface



# Exercise: Attack Surfaces in your Mobile App

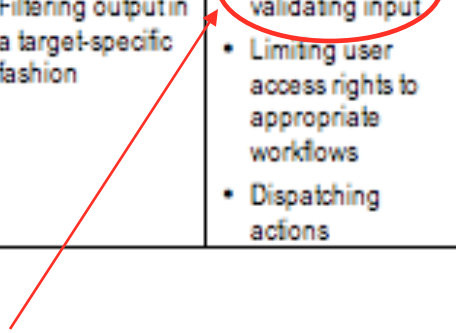
# 1.3 - Annotate with design patterns




cigital

# 1.4 – Consider Patterns' responsibilities

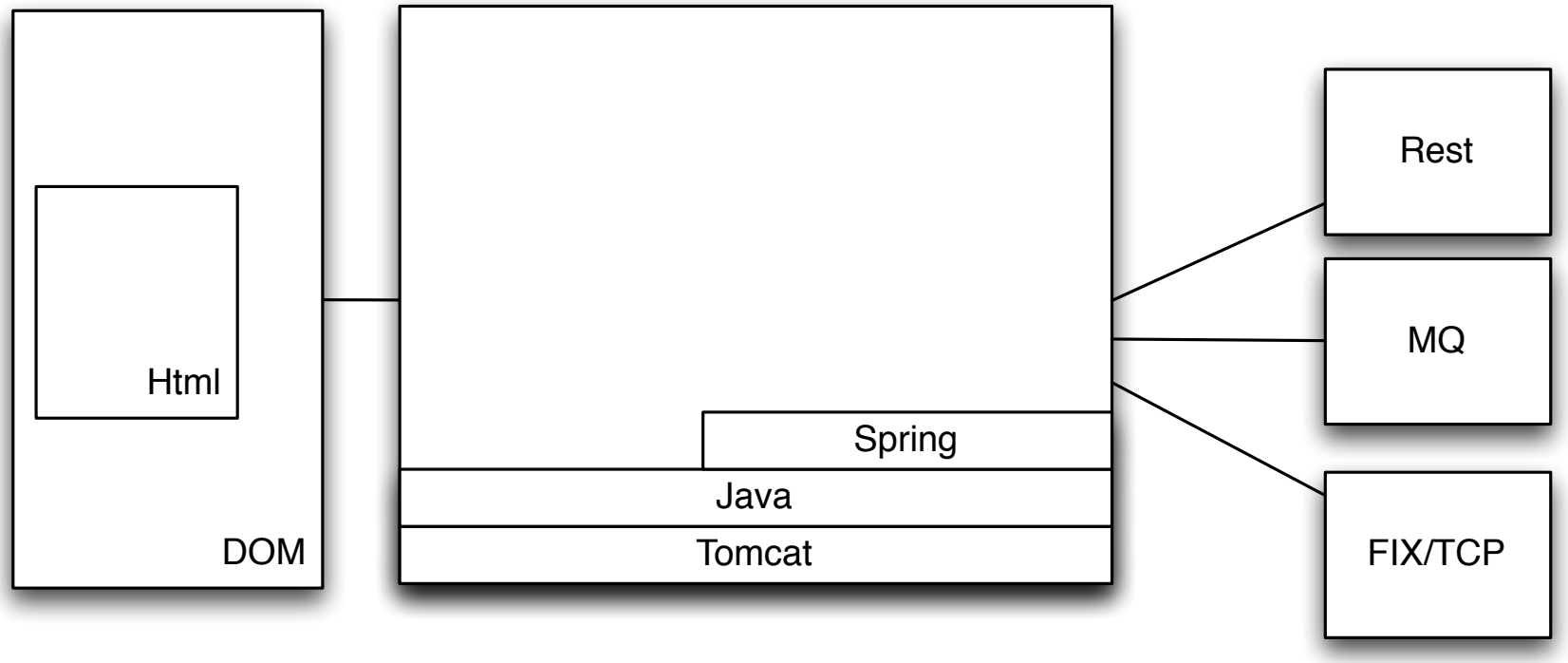
| MVC Element | View | | Controller | | Model |
|---|---|---|---|---|---|
| Component | Client-side Script' | Decorator Servlet | Controller Servlet | Action Servlet | Persistent Store |
| Responsibility | • Aspects of User experience | • Consuming and hiding error conditions<br>• Filtering output in a target-specific fashion | • Authenticating requests<br>• Filtering / validating input<br>• Limiting user access rights to appropriate workflows<br>• Dispatching actions | • Processing requests<br>• Generating content<br>• Redirecting sessions to different views<br>• Coarse-grain transaction boundary | • ACID transaction properties<br>• Hold data |



- Document specific standards for implementing each responsibility

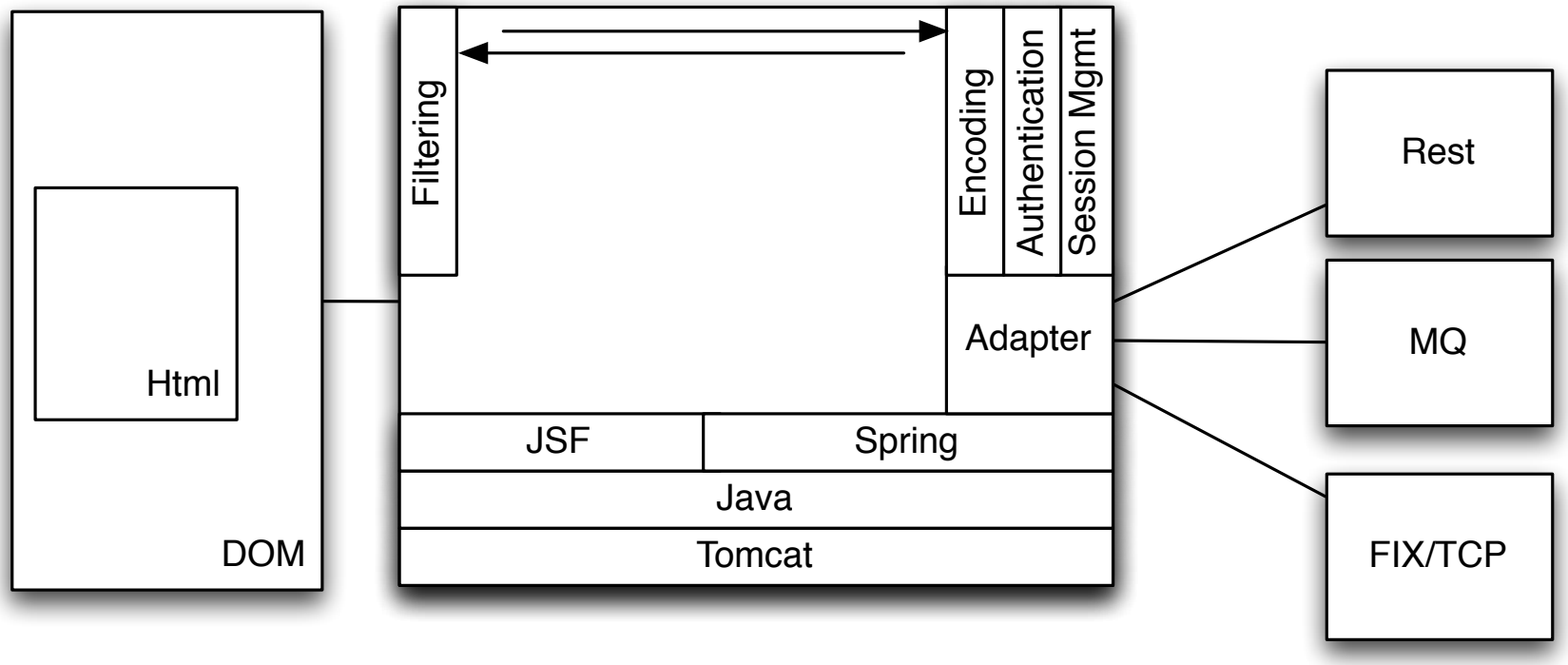
cigital

# Exercise: Find repsonsibilities



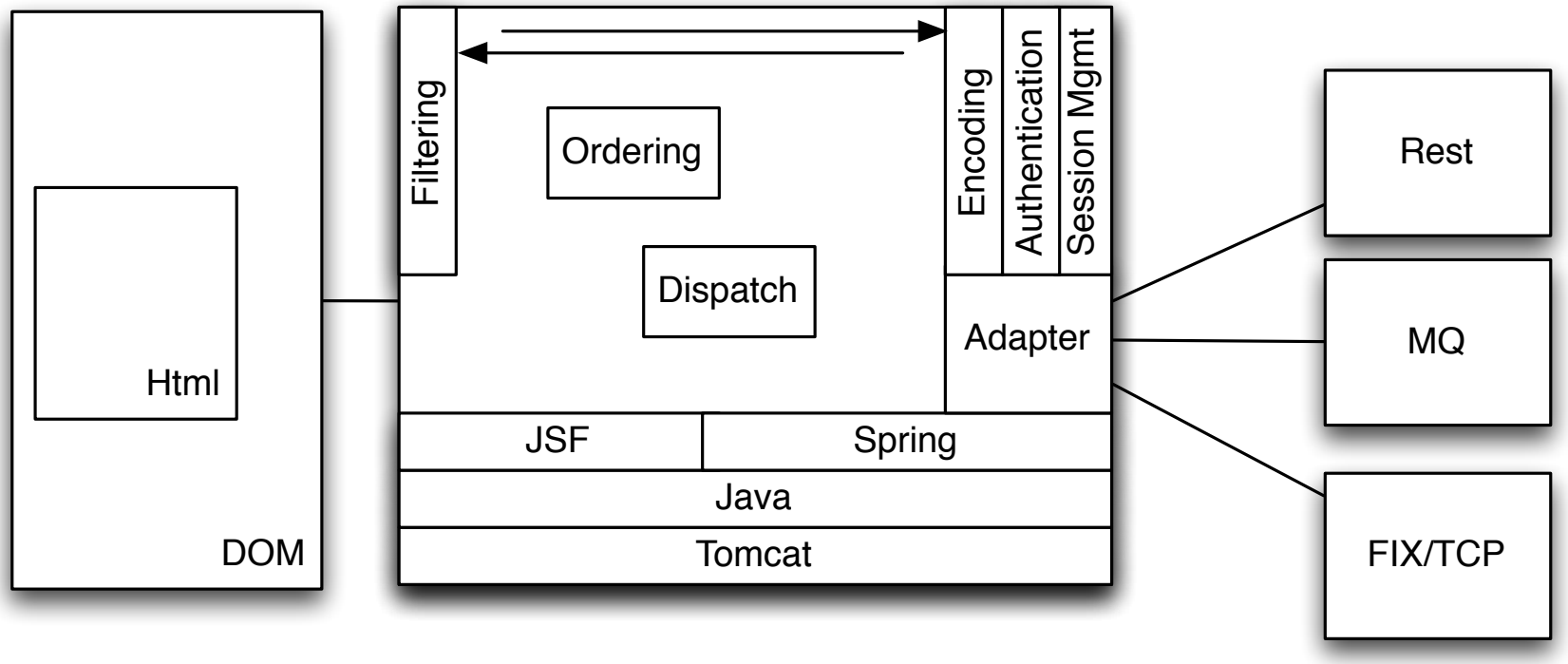

cigital

# Exercise: Find repsonsibilities




cigital

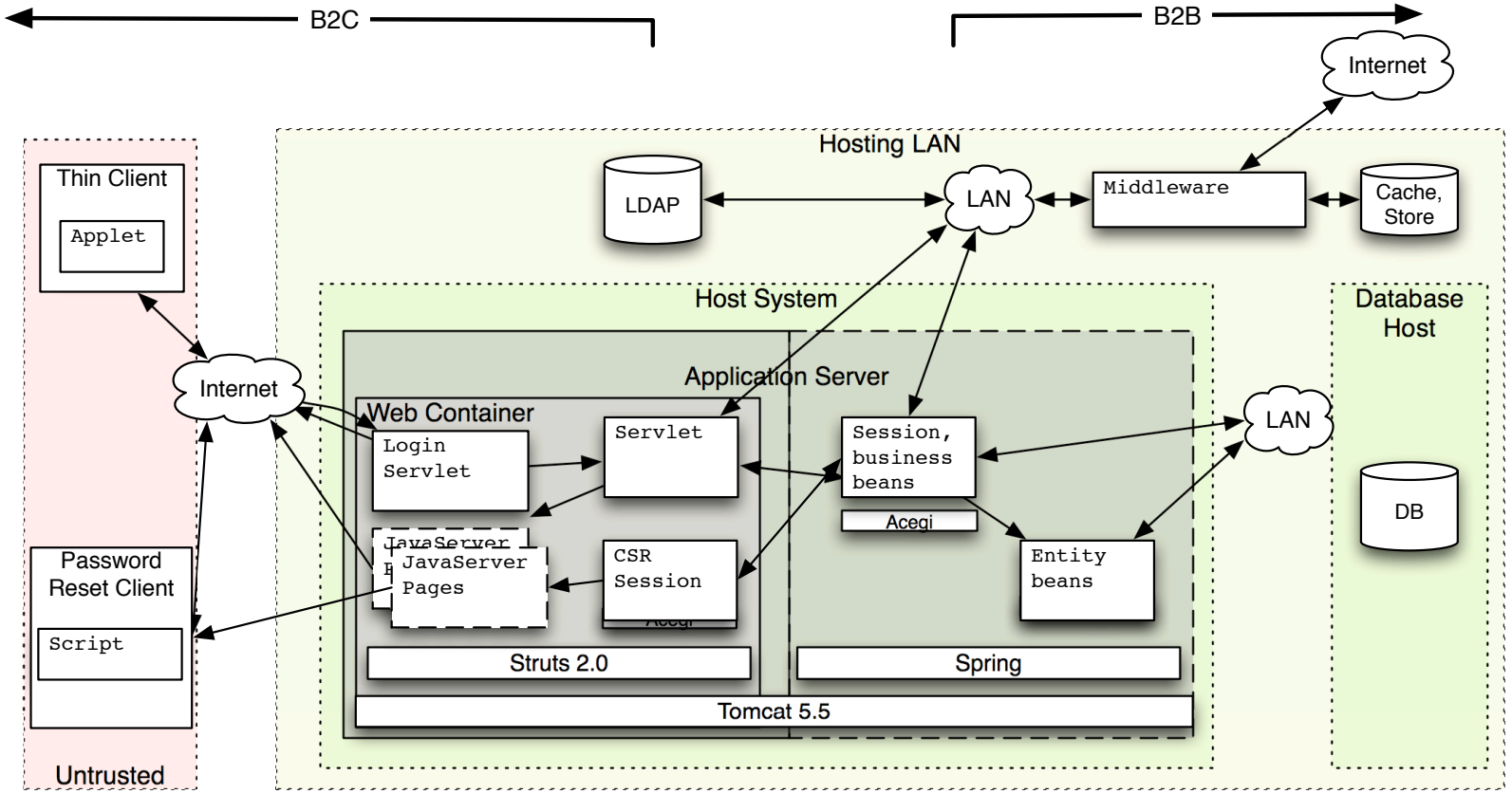
# Exercise: Find repsonsibilities




cigital

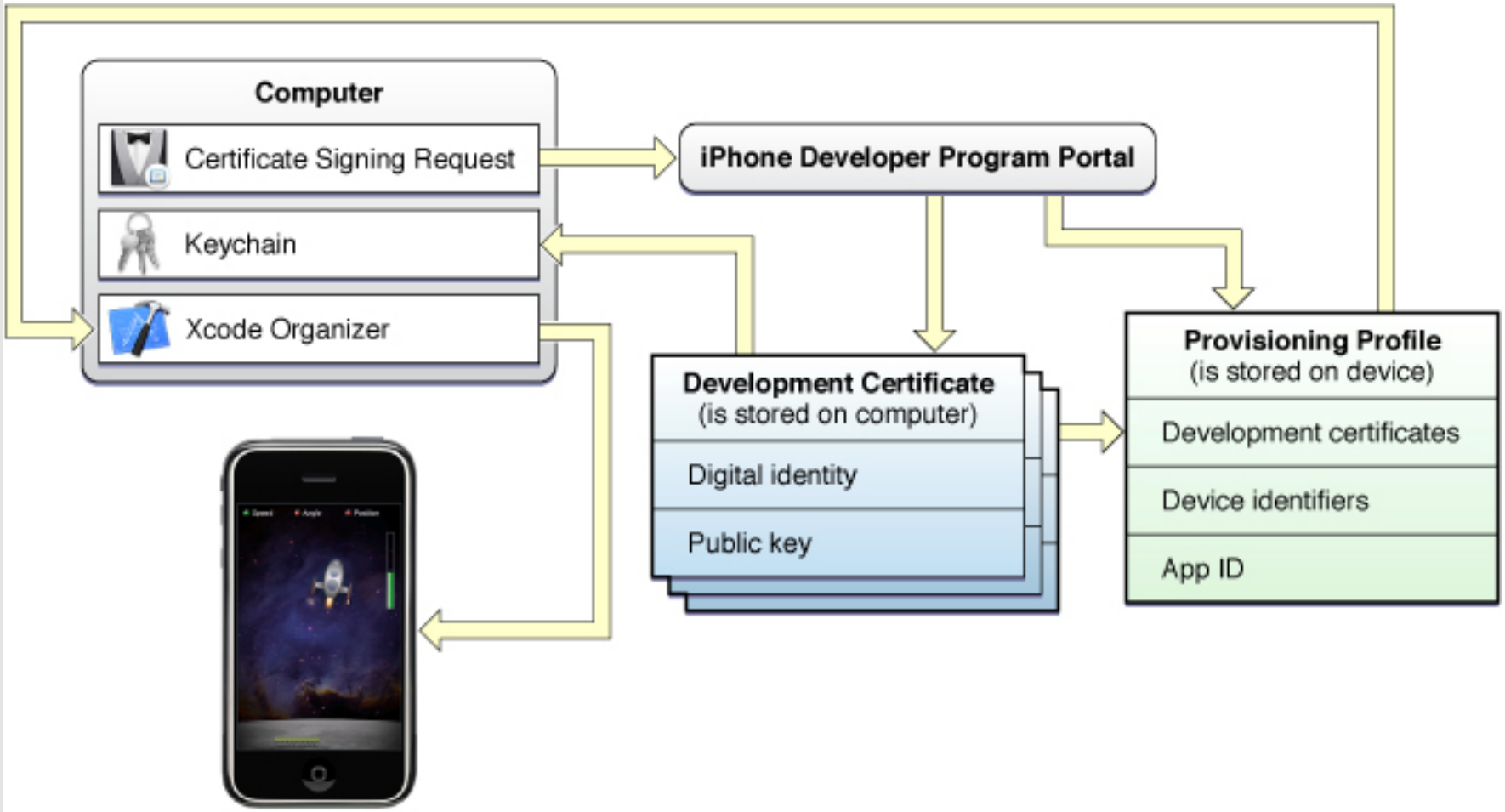
# 1.5 – Identify Frameworks

**Showing frameworks indicates where important service contracts exist 'up' and 'down'**




cigital

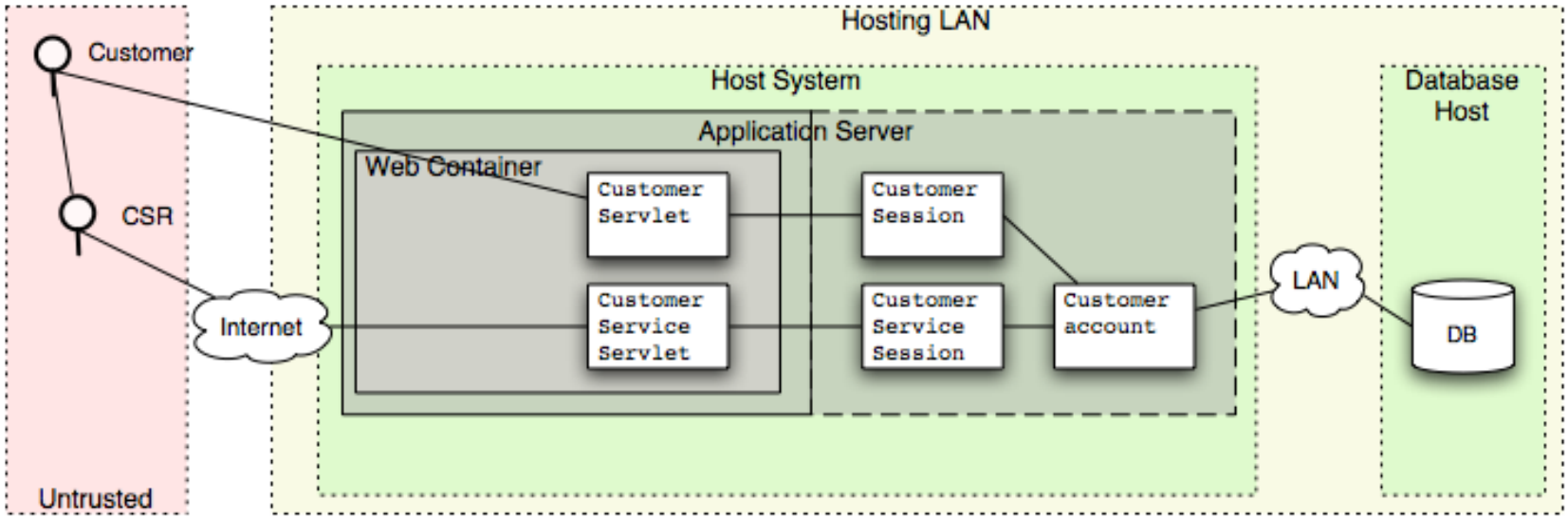
# 1.6 – Identify Controls Explicitly




cigital

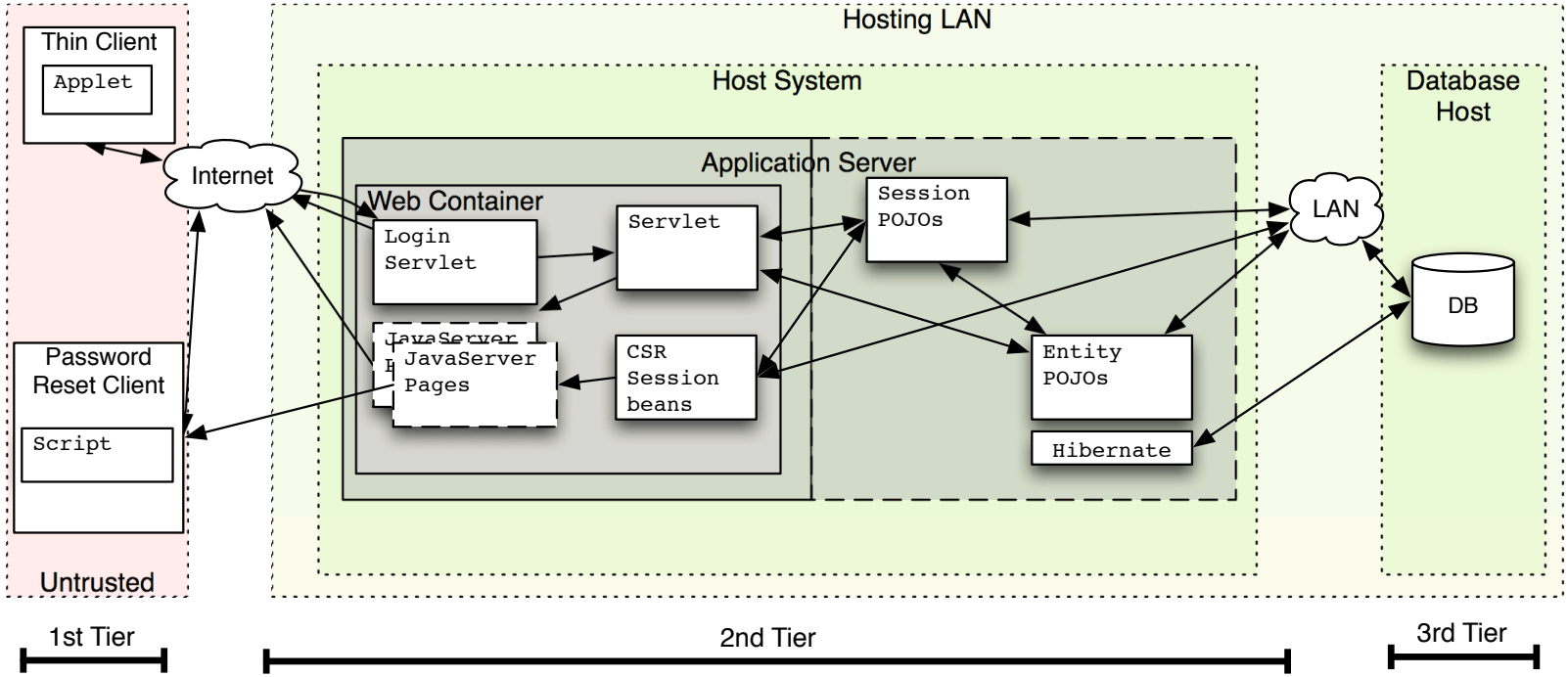
# 2 – Identifying Assets

# 2.1 Identify Critical Data Assets



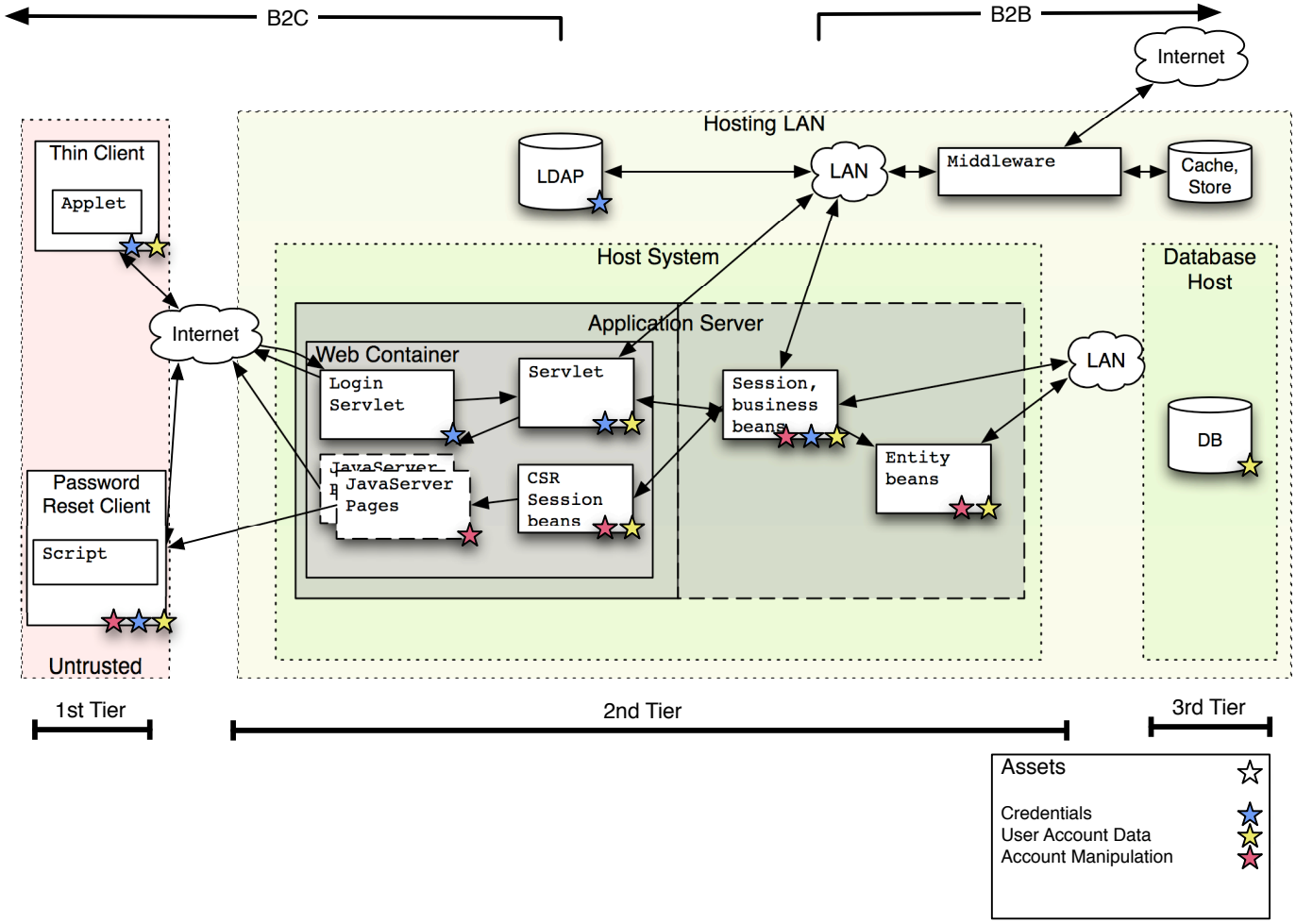

cigital

# 2.2 - Identify Interfaces as Proxies for Data

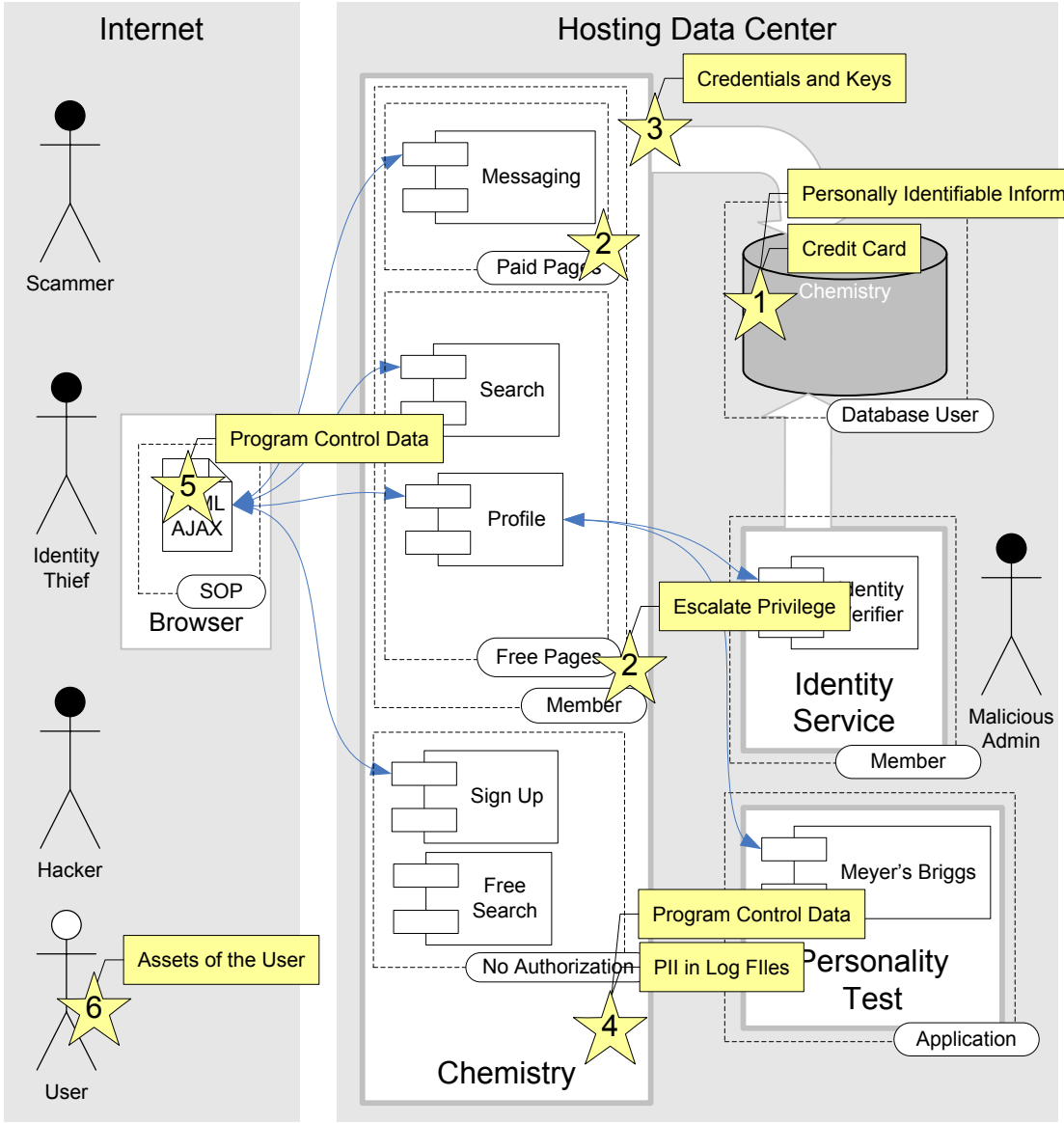


# 2.3 – Identify Assets flow through the system

## Assets exist not only in rest, but also flow through the system



# Exercise: Determine the Application's Assets




- Assets are application's functions
- Assets are the application's sensitive data
- Assets are data controlling the application's state
- Assets are the application users and the assets of the other systems the users access

cigital

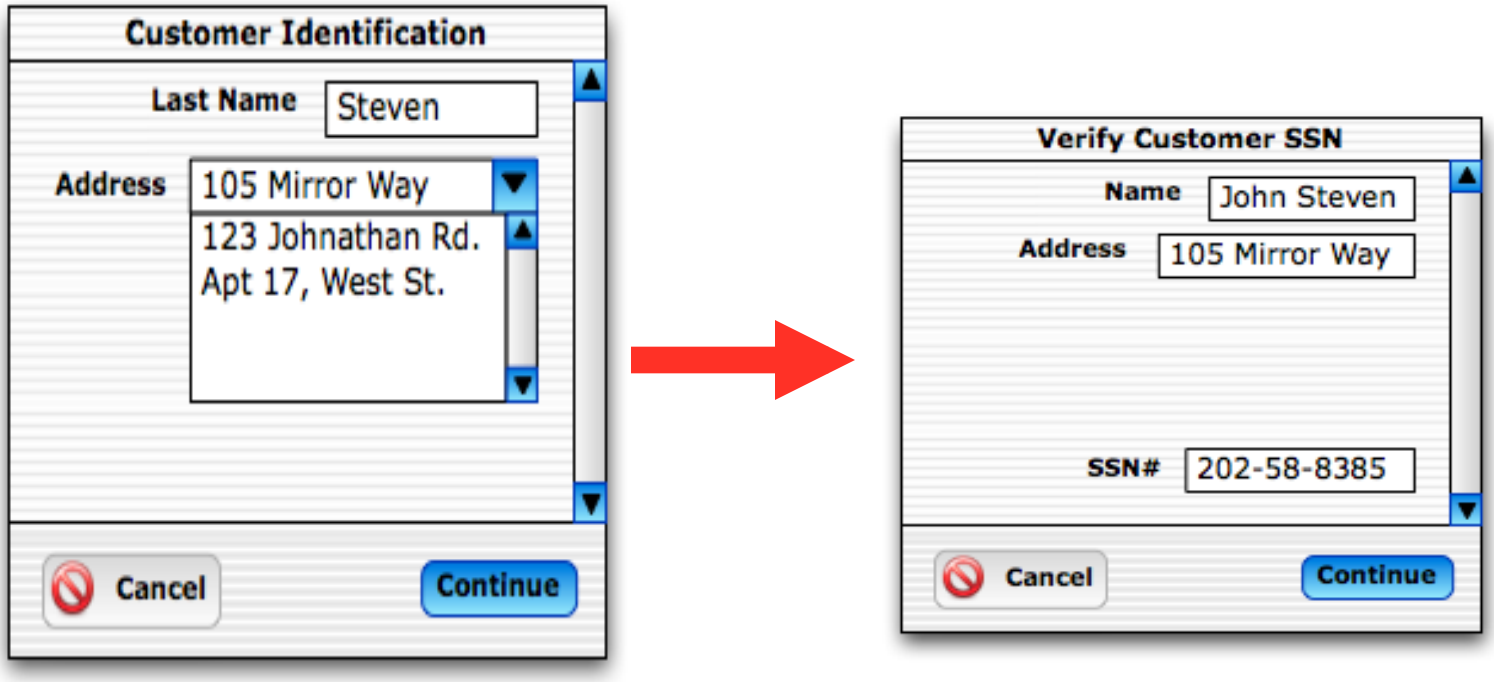
# Example: Design for Sensitive Information

- *Steal credentials or secrets embedded in the client*
  - *Read client-cached values, even from different users*

- Example: Build a web-based customer service application:
  - Supports: account maintenance, password reset, etc.
  - Customer identified by Social Security Number (SSN)
  - Customer has a password for web application

cigital

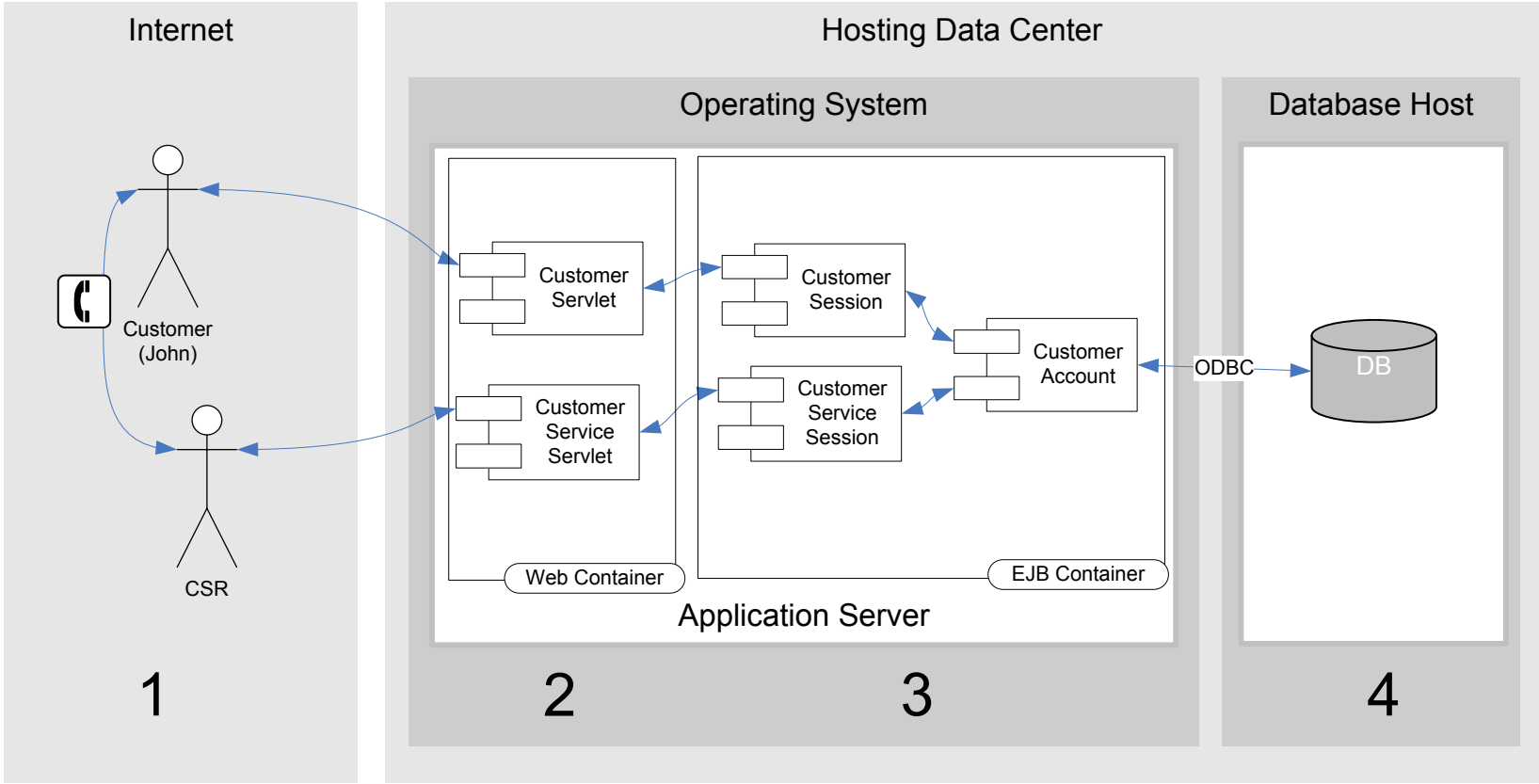
# Example Application Flow Begins




- "What's your last name Sir?"
- "Verify your address."
- "For security purposes, verify the last four digits of your social security number."

cigital

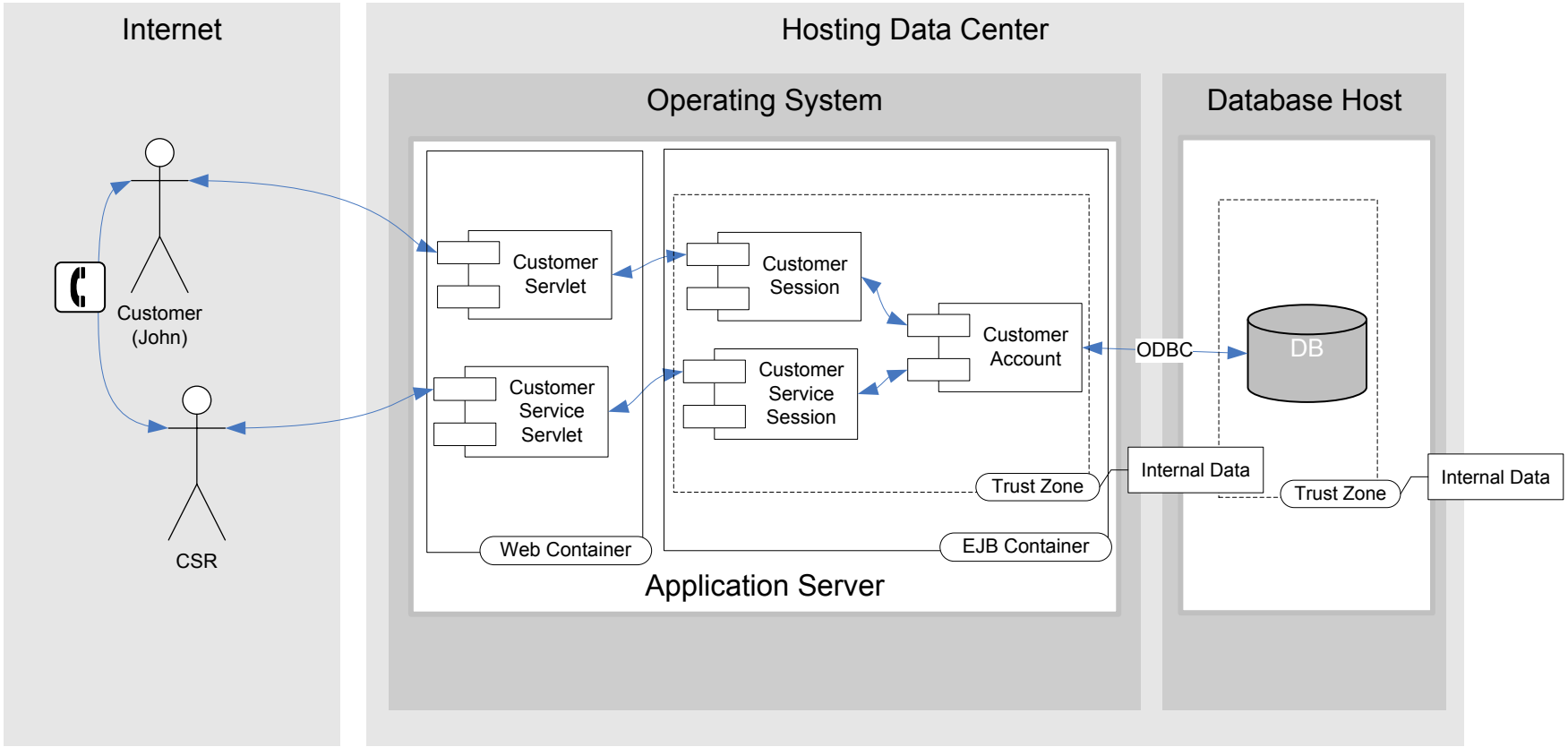
# What That Implies About Data in This System




- In what zones (by number) is what information present?
- What problems have we created for ourselves?
- What are we going to do about it?

cigital

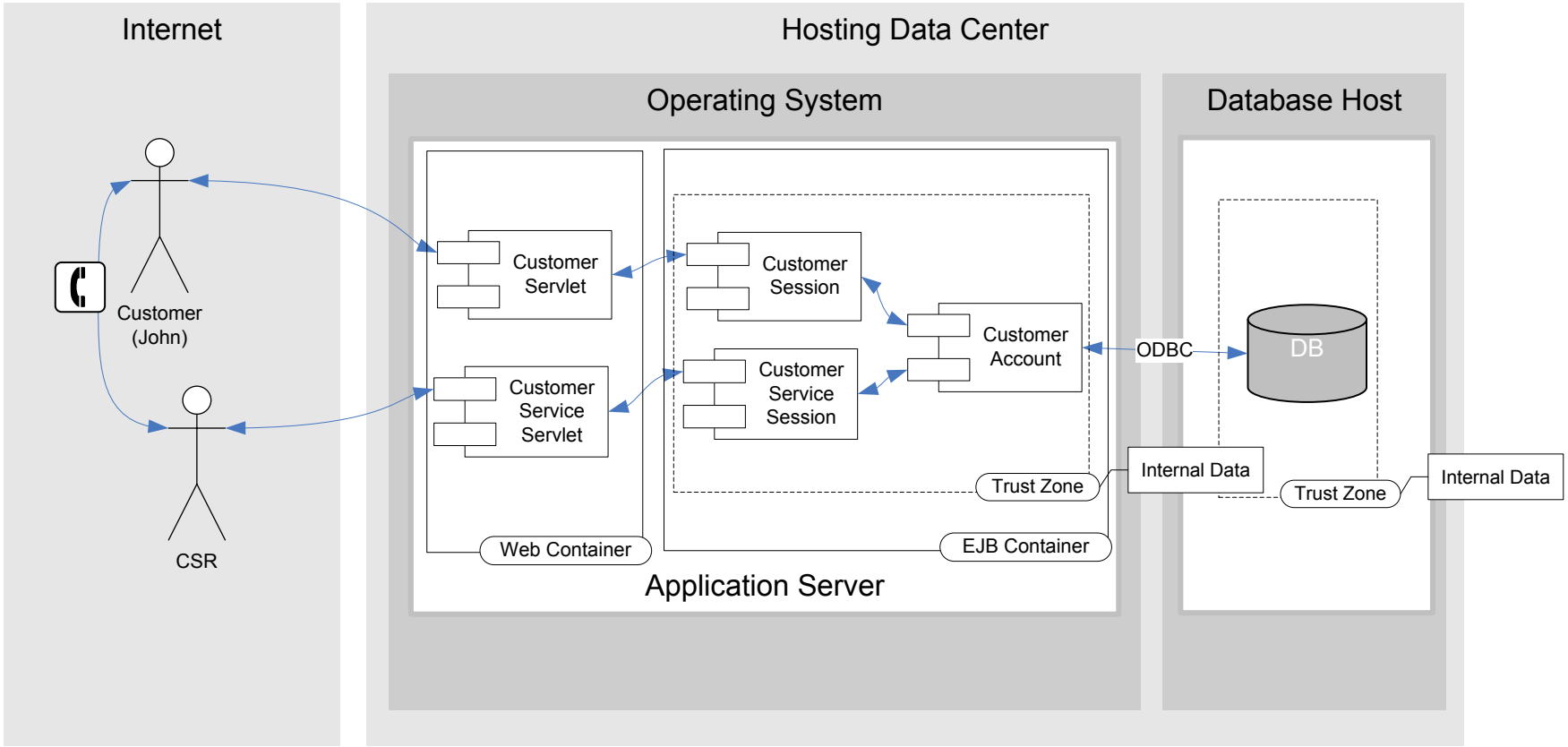
# Build Trust Zones to Protect Sensitive Data



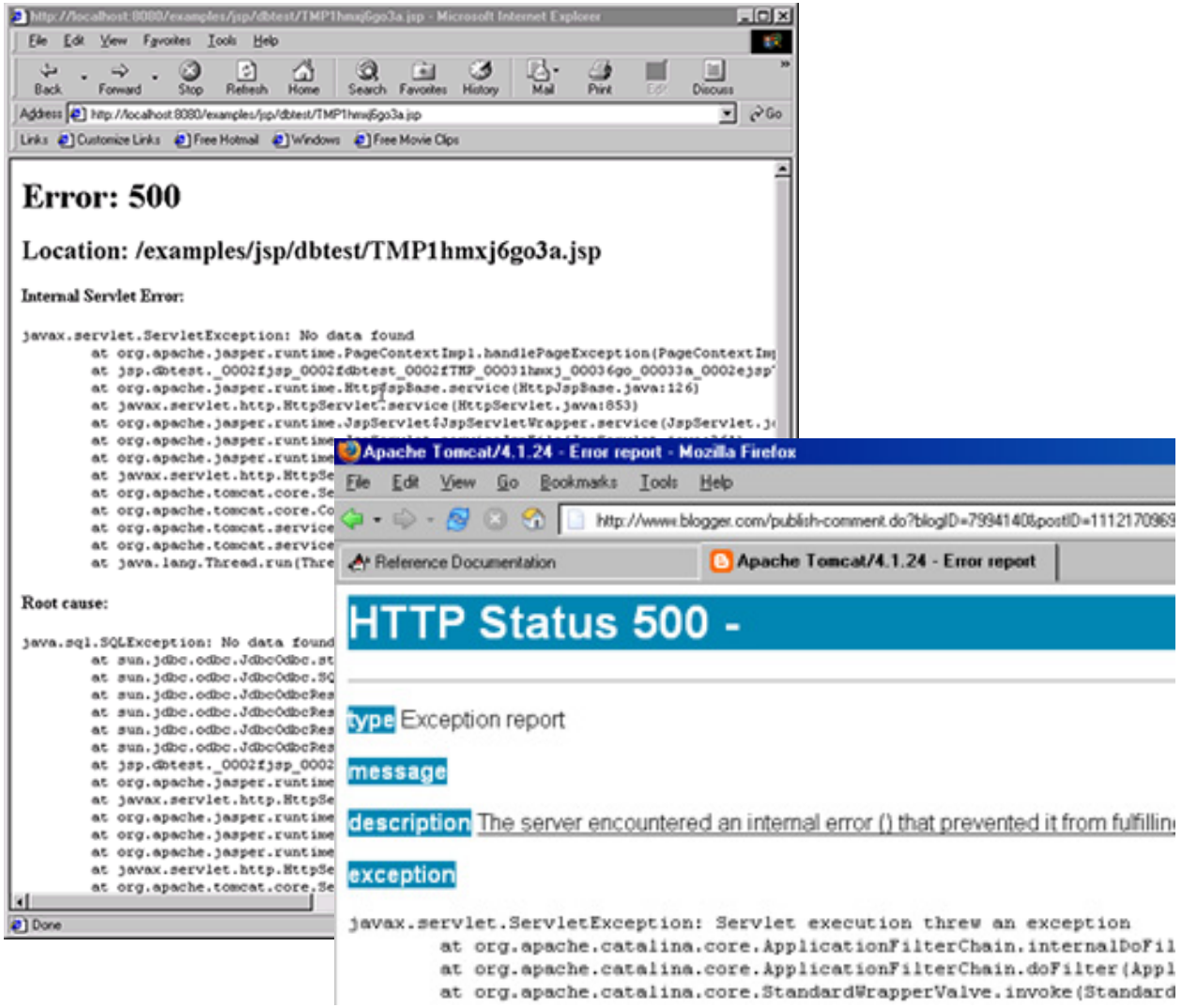

- Why should sensitive information leave the Internal Data zone?
  1. The customer knows their SSN, don't present them with it
  2. The CSR should ask for and enter the SSN, not be presented with it
  3. Use programmaic means to verify SSN in the application server

cigital

# Exercise: What can do we do about it?



# What kinds of users do these messages help?




cigital

# Users often make bad decisions


cigital

# Exercise

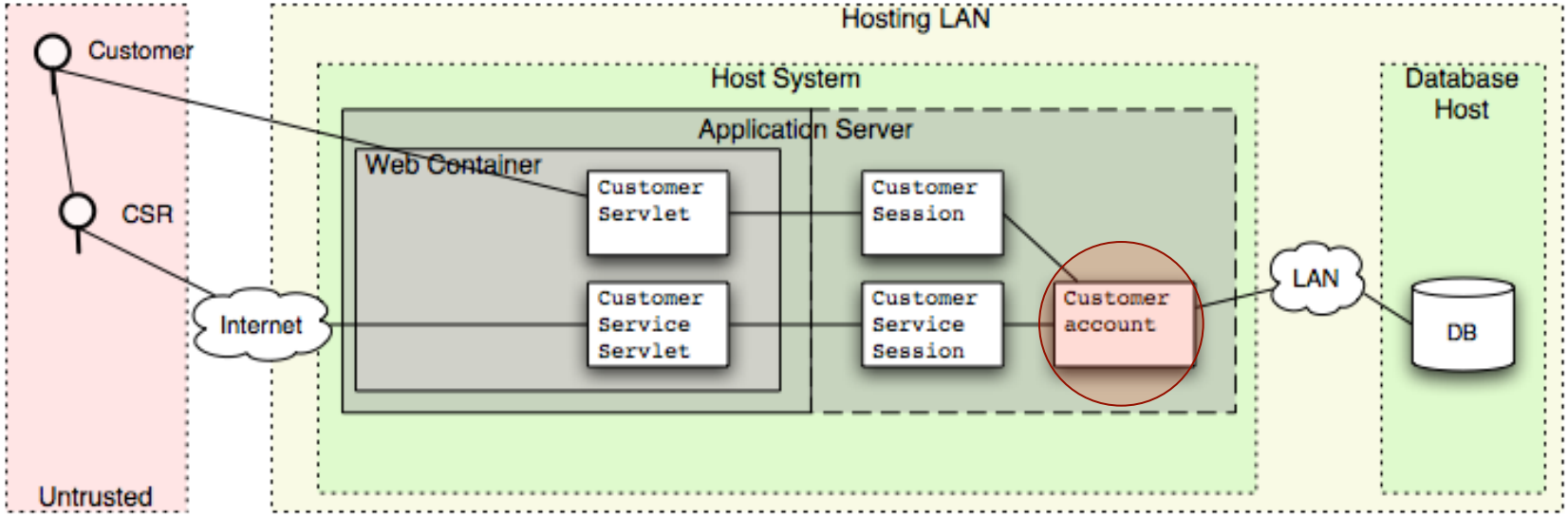
- Couched as a trust boundary problem what do security guys suggest?

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Malicious CSR | Use interface to harvest PII, SSN | Lever[illegible]unctional desig[illegible] case | Manda[illegible]disclosure & cov[illegible] | Have the CSR enter information |

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Malicious CSR | Use i[illegible]ce to harve[illegible], SSN | Levera[illegible]nctional design[illegible]case | Manda[illegible]isclosure & cove[illegible] | Support self-service PW reset |

cigital

# 2.4 - Identify Critical Application Entities



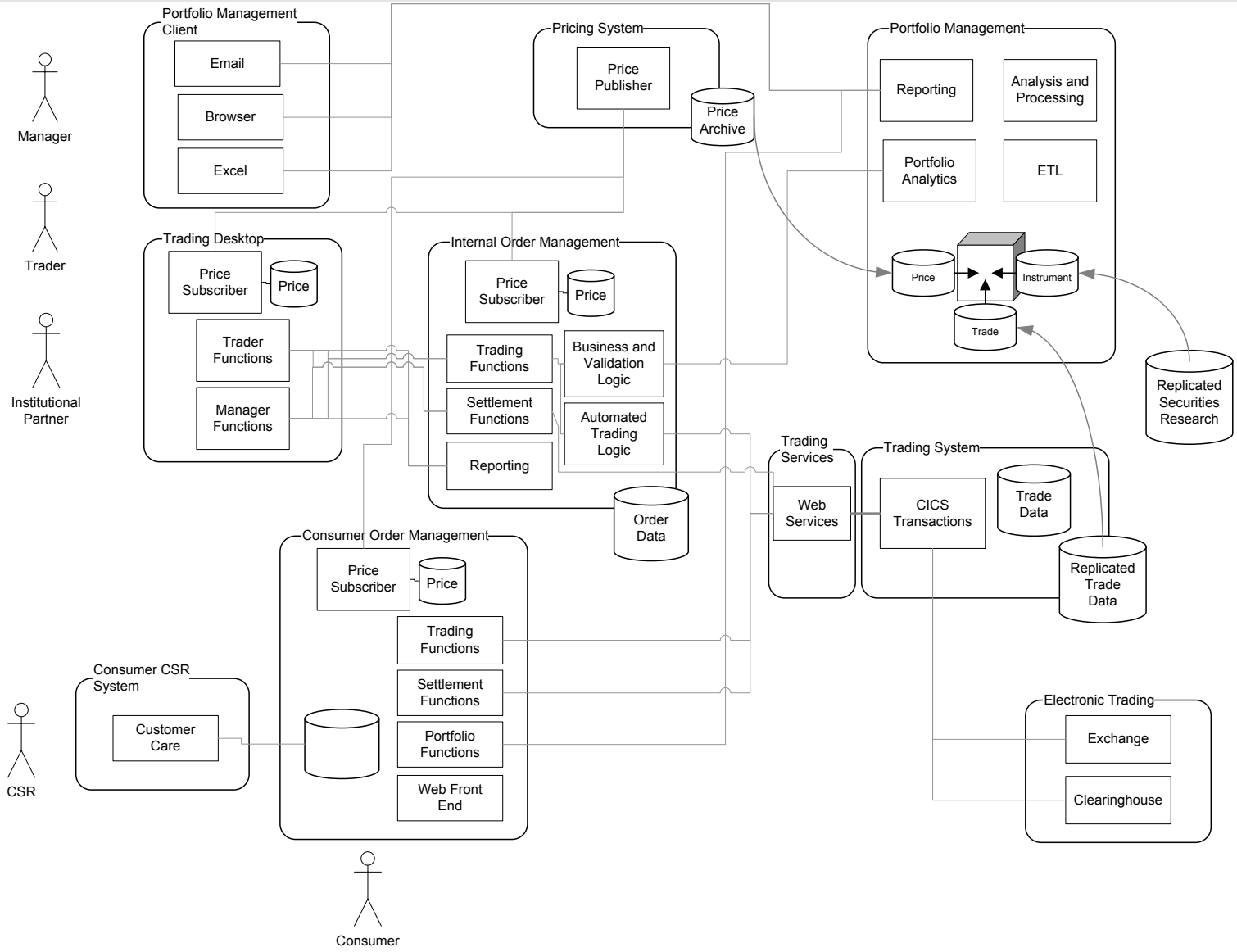

Can you identify the critical structure from 1.6?

cigital

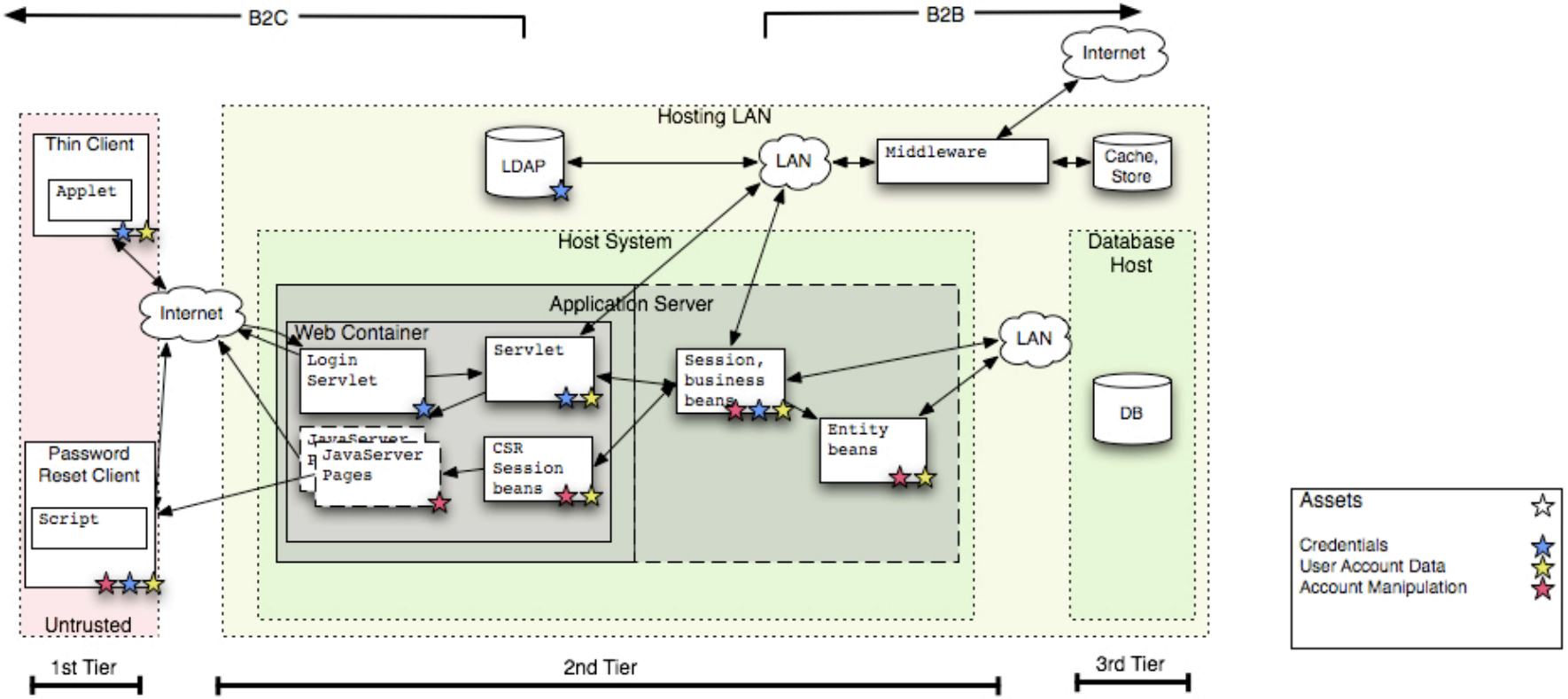
# Discussion

- What assets exist on mobile devices?

# Exercise: Identify Critical Data Assets



# 2.5 - Identify ‘Intermediate’ Asset Objectives




## Identify

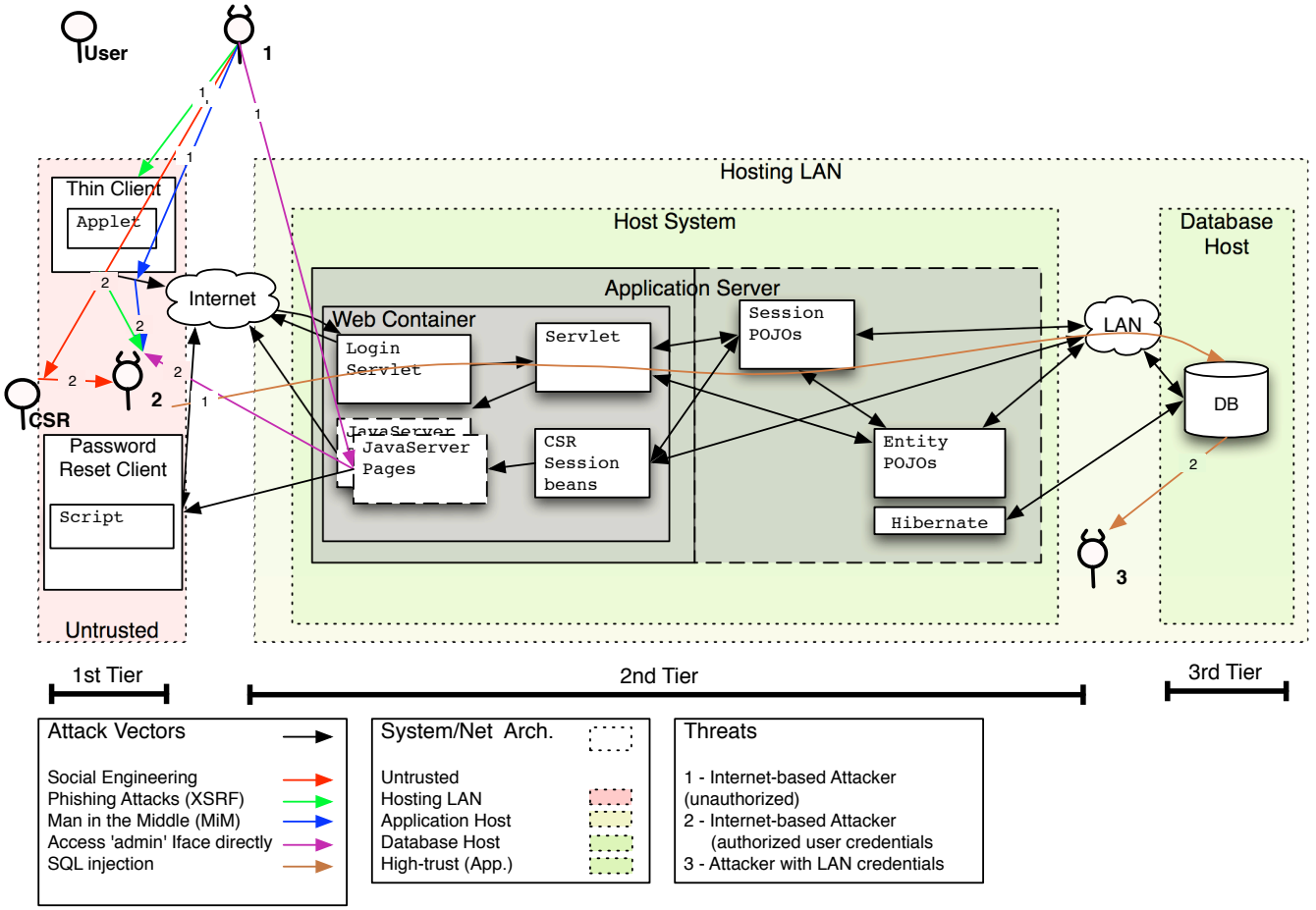
- Sensitive data
- Privileged function

## Look out for:

- Proxies, facades, etc.
- Services: ws-, beans, etc.
- UI vs. implementation
- Aggressive caching schemes

cigital

# 2.6 – Identify Equivalence-classes



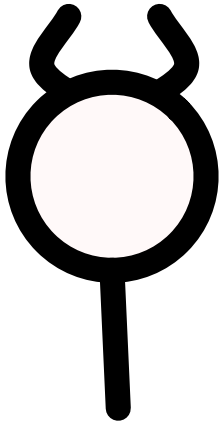

What 'intermediate objectives' equate to assets?

cigital

# 3 – Identify Threat Agents

# Threat

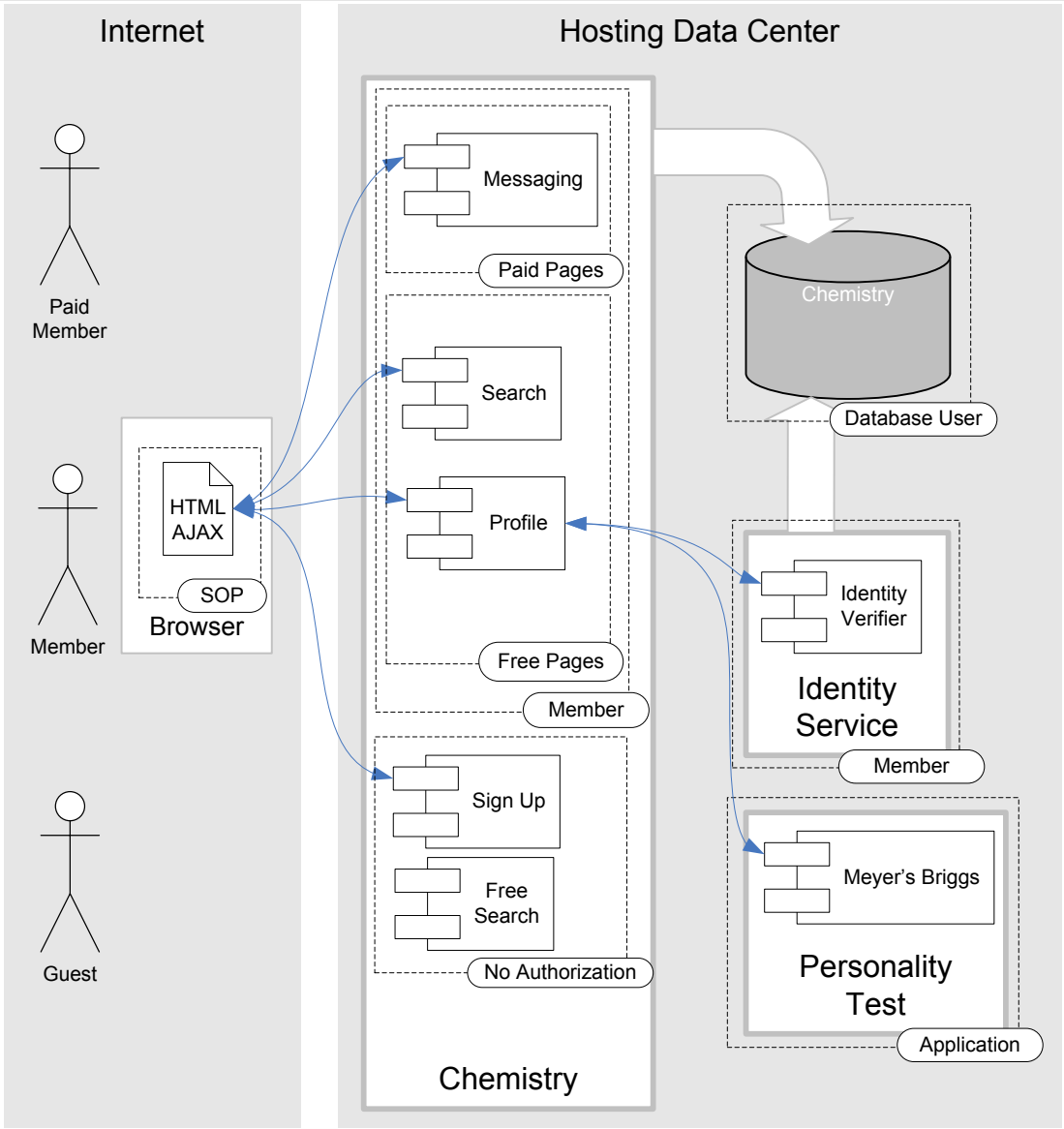
- Capability
  - Access to the system
  - Able to reverse engineer binaries
  - Able to sniff the network
- Skill Level
  - Experienced hacker
  - Script kiddie
  - Insiders
- Resources and Tools
  - Simple manual execution
  - Distributed bot army
  - Well-funded organization
  - Access to private information



- Threats help
  - Encourage thorough thought about how intentions for misuse
  - Determine “out of bounds” scenarios

cigital

# Model Threats by Modeling Users

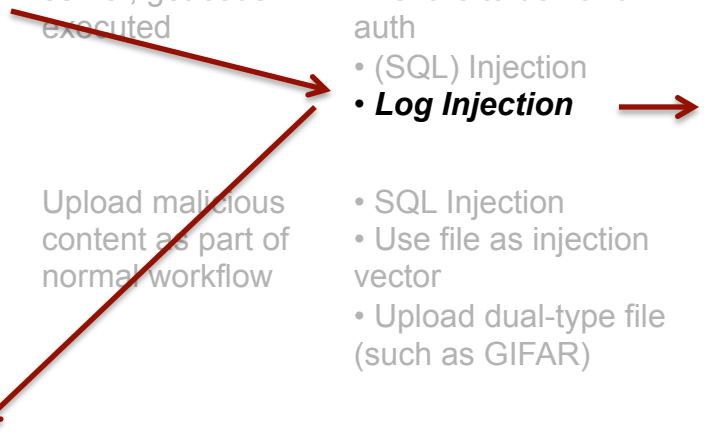



- Threats are users that have malicious intent
- Like users they have capabilities within the system
- Threats have a goal that usually involves subverting a security control or finding a "loophole" in the system

cigital

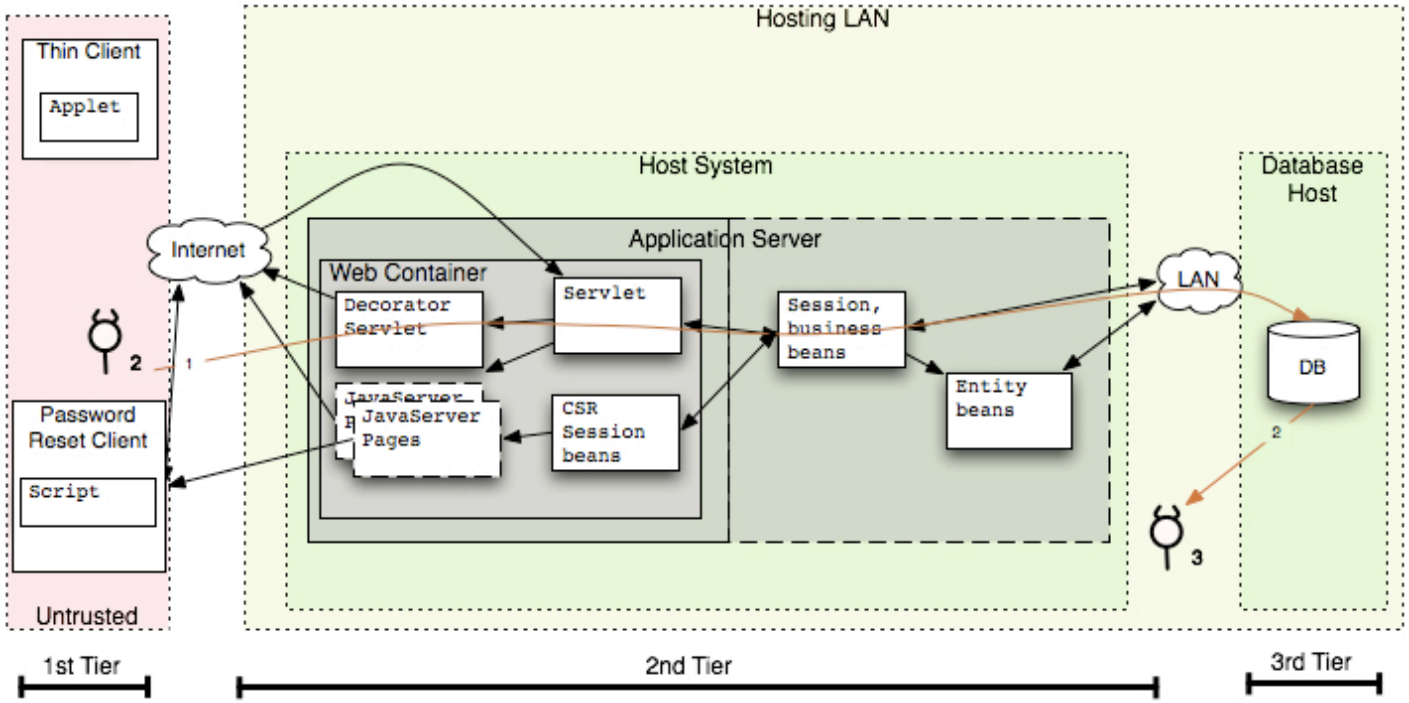
# Motivating Insiders (tabular)

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, ***UNAUTHORIZED***, Internet user | Get code onto server, get code executed | • Failure to demand auth<br>• (SQL) Injection<br>• ***Log Injection*** | PR Incident<br>*Non-compliance*<br>Breakdown in Monitoring Control | |
| Authorized User | Upload malicious content as part of normal workflow | • SQL Injection<br>• Use file as injection vector<br>• Upload dual-type file (such as GIFAR) | | |
| ***Database user*** | | | | |



cigital

# Targeting Assets, in General




- Remember:
  - Focus on common, simple attacks
  - Escalation to admin/LAN credential possible where credential stores reside in site database
  - 'insiders' need not be

cigital

# Exercise: Mobile Threats

| Who | Capabilities | What | How |
|---|---|---|---|
| Rooted Phone User | • Smart phone (Droid, OSX, Palm, etc.) | | |
| Un-rooted Phone User | | | |
| DB Admin | | | |
| CSR | | | |
| Developer | | | |

# WHO

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Unauthorized Internet User | | | | |
| Authorized Public User | | | | |
| Authorized Partner User | | | | |

- Access
  - Access to the system
  - Ability to scan, sniff network
  - Access to binaries
  - Able to contribute code to your site (mash-up, open-source, etc.)
- Skill Level
  - Experienced hacker
  - Script kiddy
  - Insiders
- Resource-level and Tools
  - Simple manual execution
  - Standard penetration-testing tools
  - Distributed botnet
  - Well-funded organization

cigital

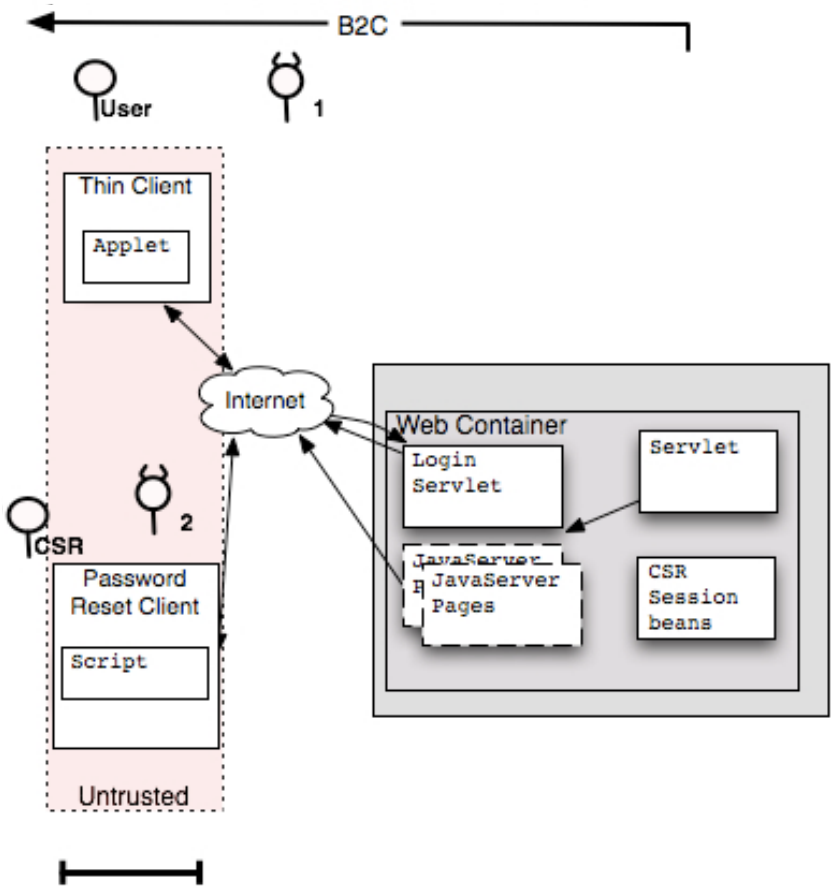
# 3.1 - Anchor Threats in Use Cases

Consider attack surface

- Actors become Threats
- Use becomes misuse

Convert Actors to mis-actors

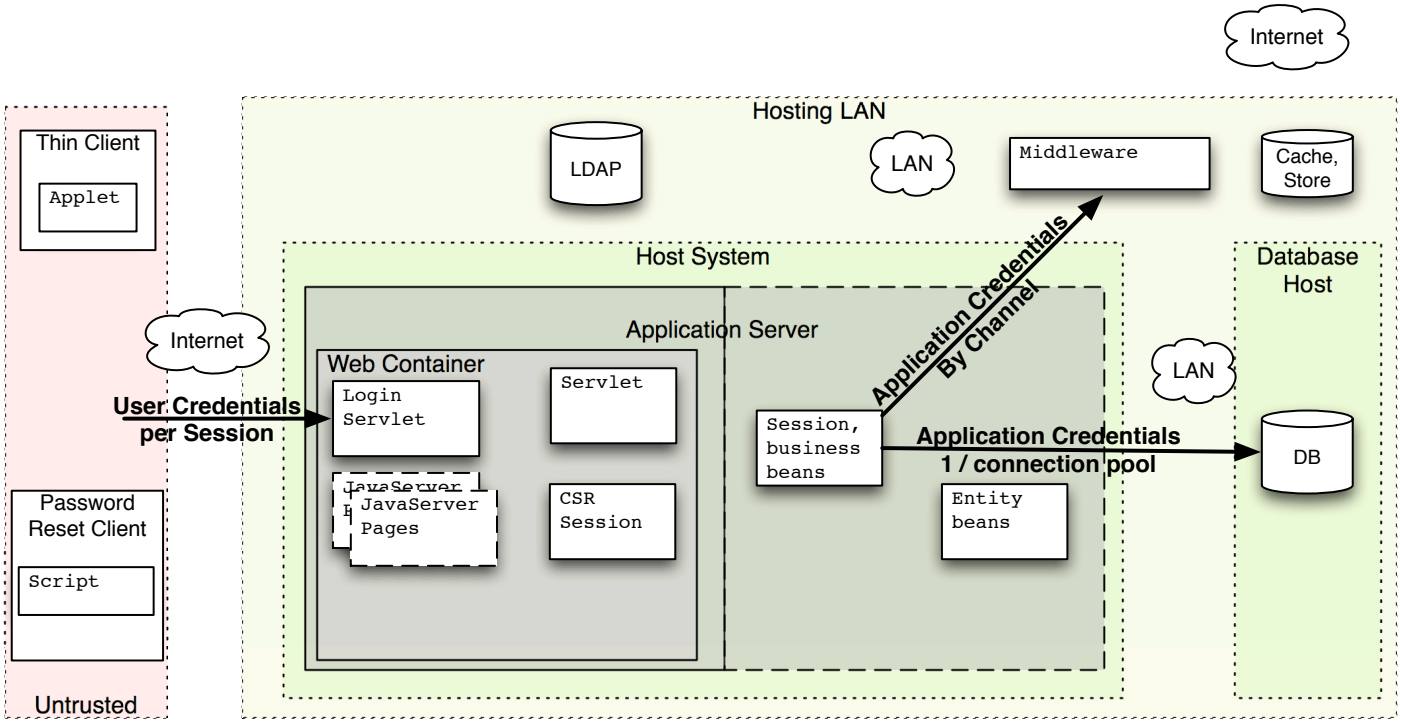
- Abuse – Make actors behave stupidly
  - Error conditions
  - Alternative flows
  - Fuzz testing
  - Boundary/value testing
- Misuse – Make actors deviant/evil
  - Societe Generale
  - Think like an attacker



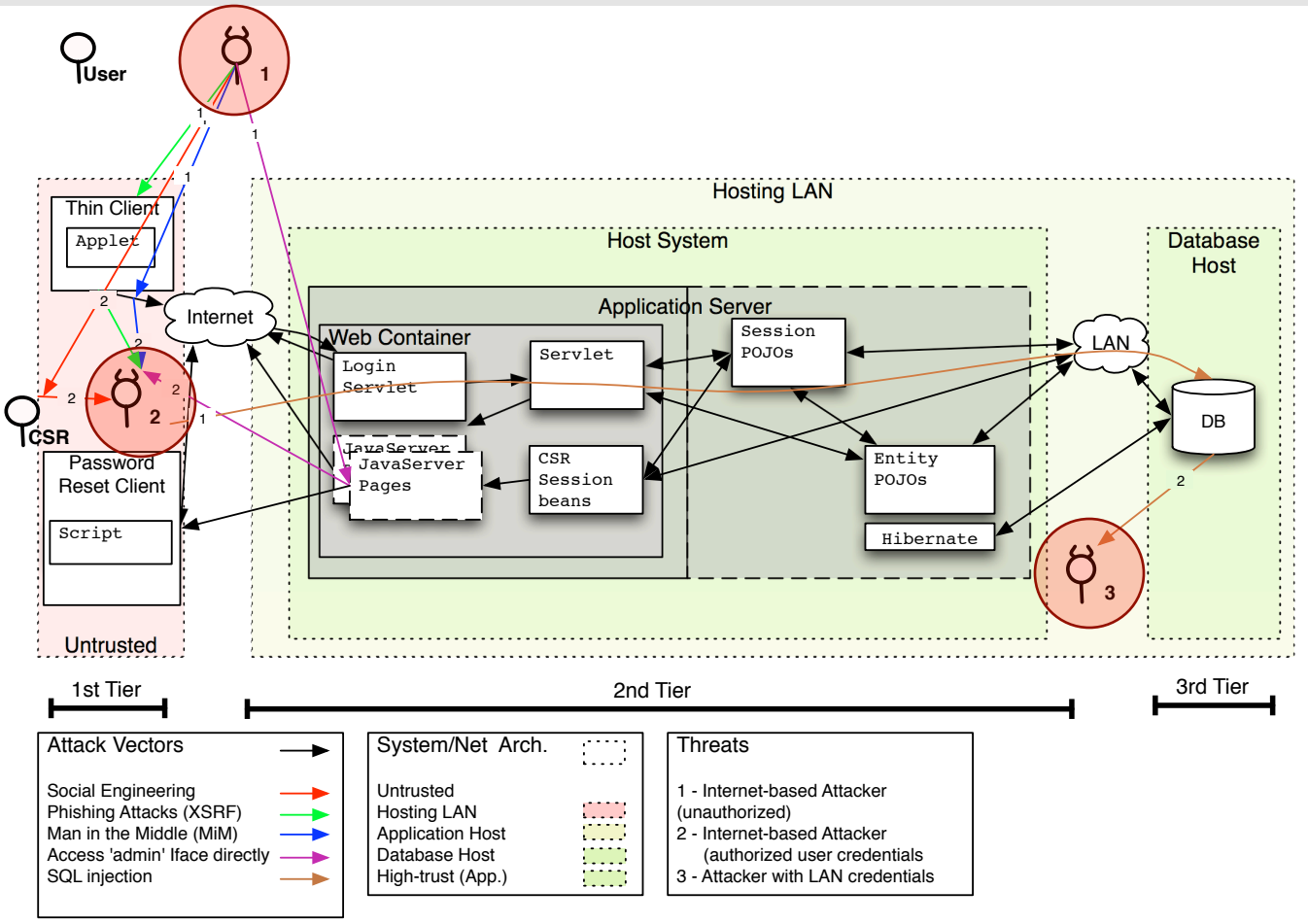

cigital

# 3.2 – Identity Principal Resolution

## Arrows indicate resolution of principal/assertion propagation



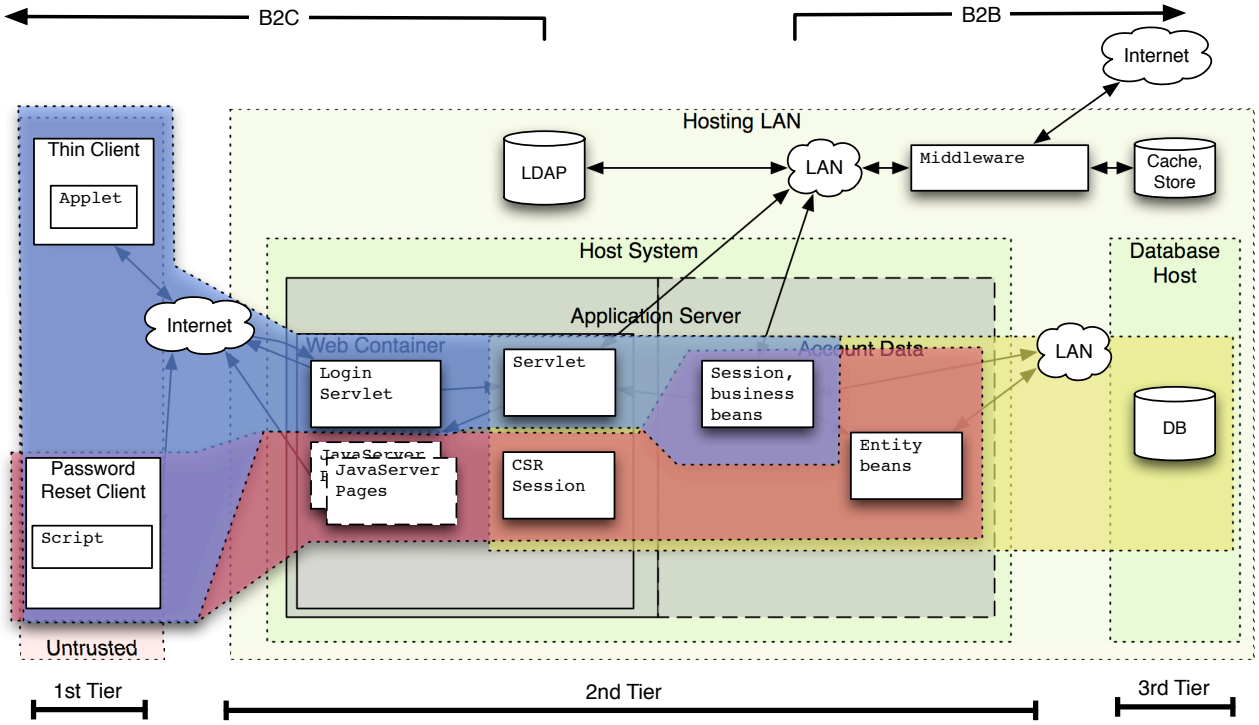
# 3.3 - Place Threats on Diagram




cigital

# 3.4 – Show Authorization in Structure

## Coloration shows authorization by role




cigital

# Who

- Update your information when:
  - You determine privilege escalation is possible(*)
  - M&A
  - New partner arrangements
  - New business models/architectures
  - Privileged roles change
  - HR events

cigital

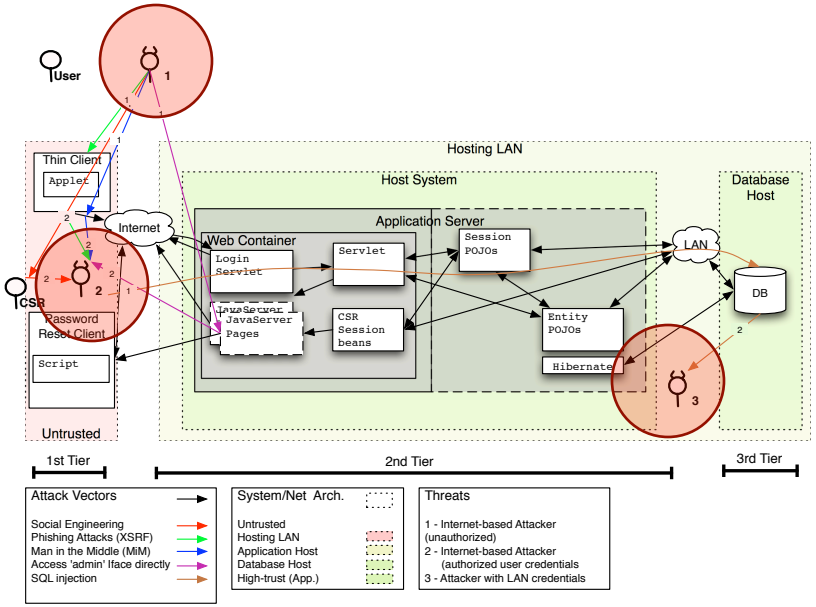
# Question:

- Who are we going to consider on our password reset?
  - We discussed Malicious CSR
  - How ‘bout a malicious customer?
    - Cingular

cigital

# 4 – Enumerate Goal Impacts

# 4.1 – Assign Threats Malicious intent




- What is each Threat's motivation?
- What would drive escalation?
- Why would each try beyond the first control/hurdle?

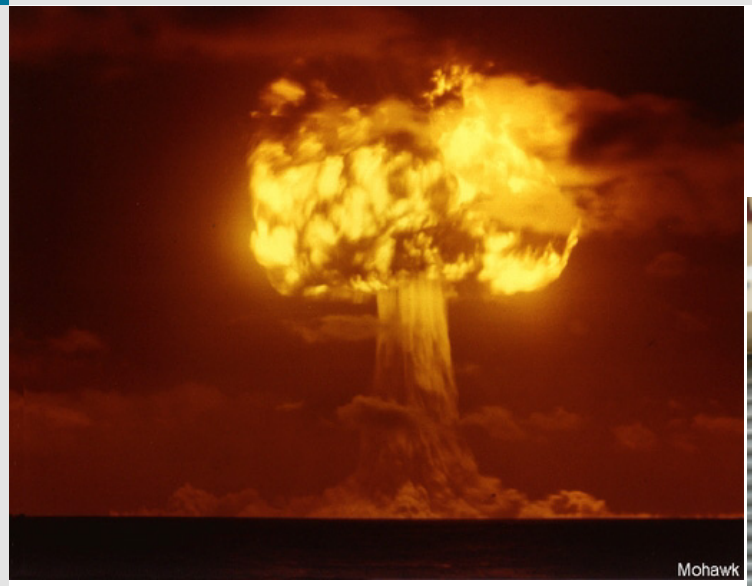
cigital

# 4.2 - Instantiate Doomsday Attacks

- Prohibitive regulatory/compliance fines
- Revocation of operating license
- Expensive litigation, injunction, or similar
- Failure to comply to MSA, SLA, or QoS
- Loss of essential business credibility
- Dramatic loss to revenue, stock value, market share, etc.
- Catastrophic PR incident
- Levied penalties (increase in processing fee)
- Company or org. dissolved
- Loss of strategic advantage
- Loss of customers

cigital

# 4.3 – Think big for a moment…





cigital

# 4.4 After thinking big, think $$$

- Licensing
- 99.99% Recouped

# 5 – Document Misuse

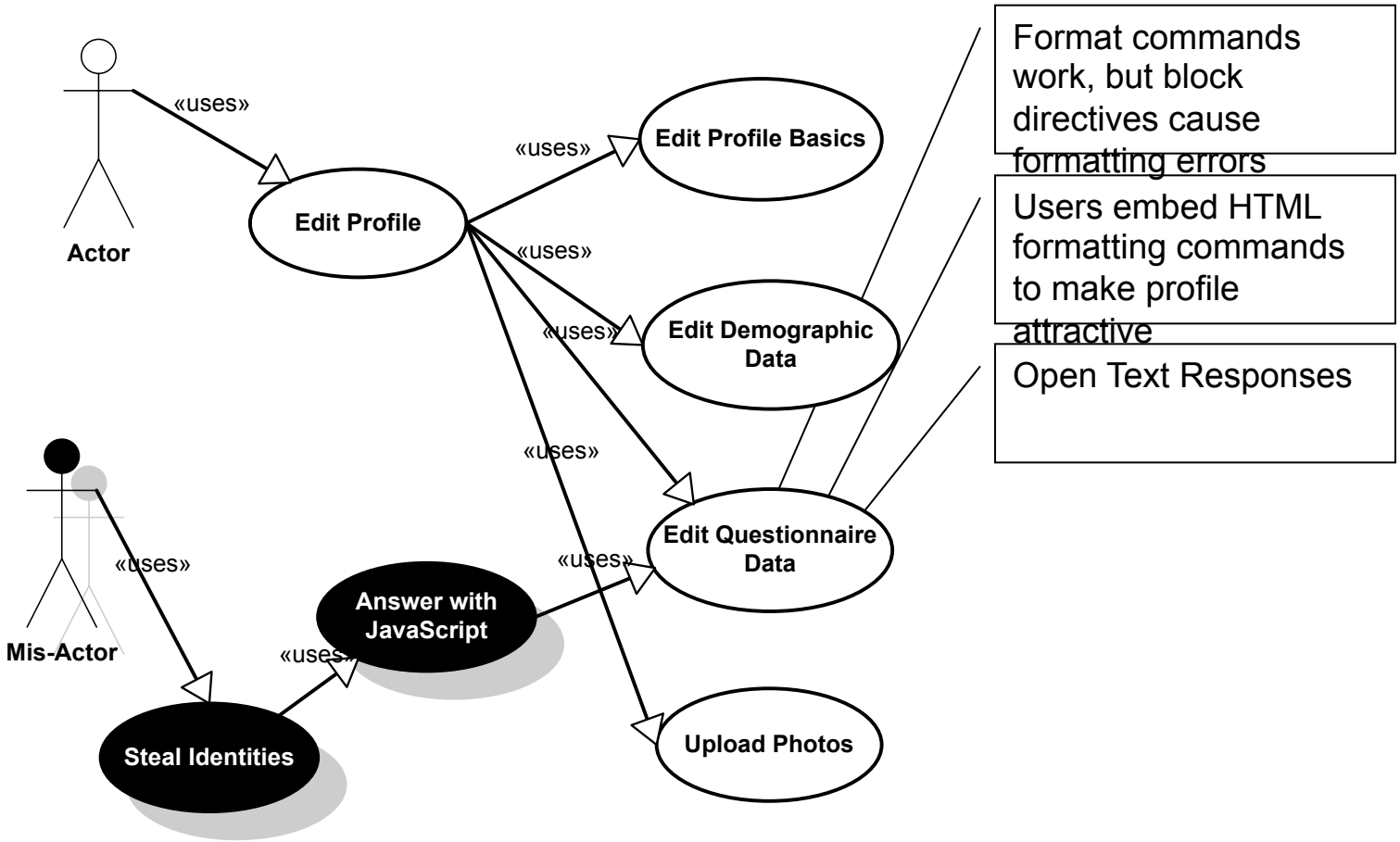
# 5.1 - Add in Misuse Cases

Convert Actors to Threats

- Abuse – Make actors behave stupidly
  - Error conditions
  - Alternative flows
  - Fuzz testing
  - Boundary/value testing
- Misuse – Make actors deviant/evil
  - Societe Generale
  - Think like an attacker

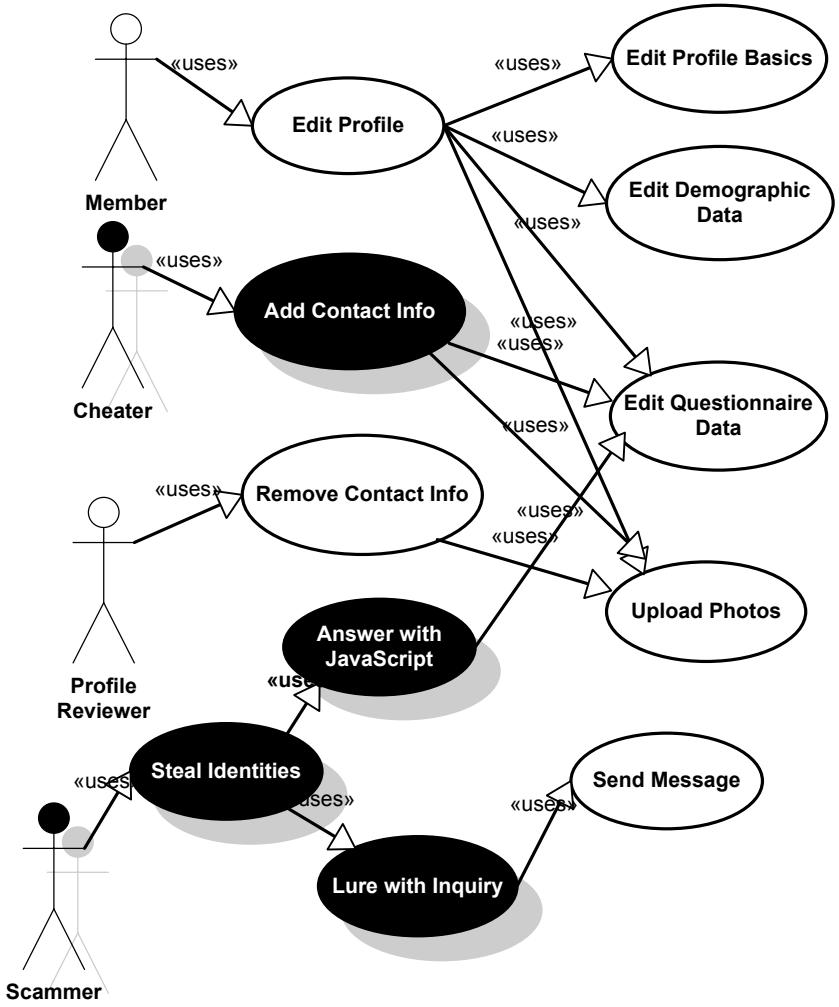
cigital

# Misuse and Abuse Cases

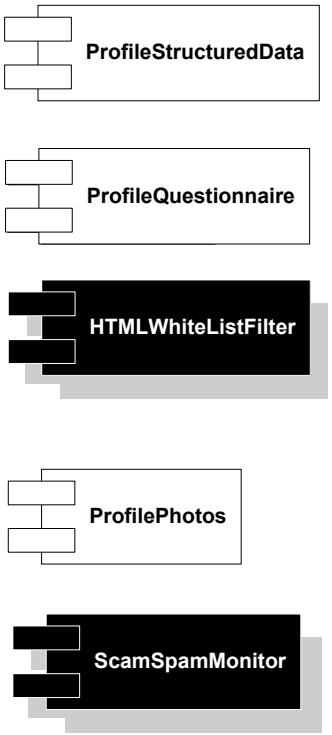


# Analyzing Abuse Cases

## Use Case View

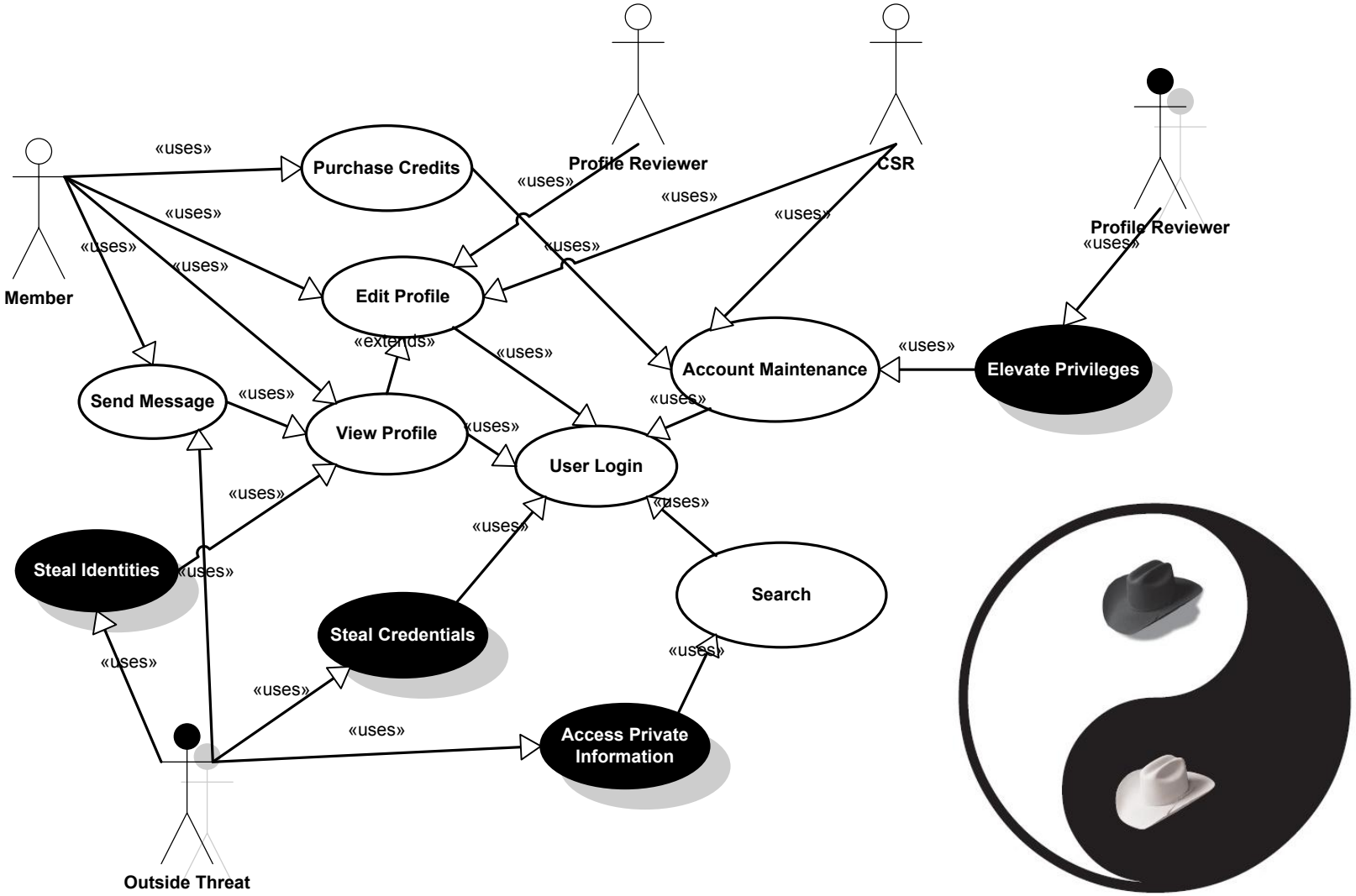



## Component View

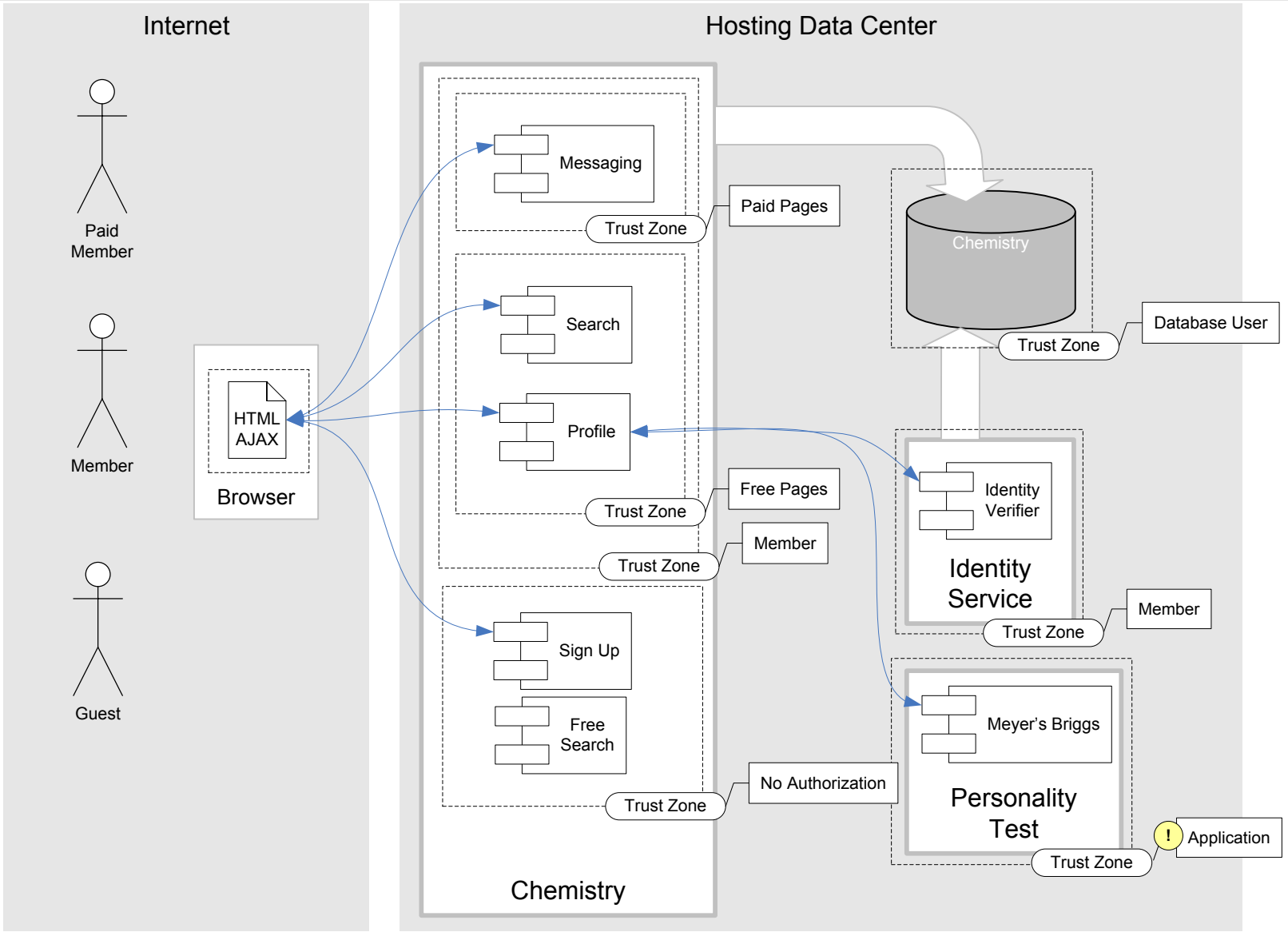



cigital

# Software Security Is Not Security Software



# Exercise: Document Misuse



# Tie-back to Threat Modeling

# What

- When use changes, revisit the what
- When a new *how* is discovered, consider *what* is opened
- Prioritize based on impact

- This will act as your testing rules of engagement

cigital

# What, Impact

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, unauthorized, Internet user | Directly request and gain access to another user's info | | PR Incident<br>*Non-compliance*<br>Increase QSA assessment cost | |
| Public or partner, authorized user | Upload malicious content as part of normal workflow | | SLA violation<br>*Data loss/ corruption*<br>*Wholesale system breach* | |

Roles

- BU Stakeholders can help with 'business logic'
- Architects can help with interaction with application arch-type

Avoid

- Get technology-specific about your misuses ("Using TamperData...")
- Require 'security fu' for misuse ("XSS the form...")
- Devolve into a Data Sensitivity classification
- Specify requirements ("Use SSL, with digest authentication")

cigital

# Example:

- Jimmy, Johnny, and Sally

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Jealous friend | 'Phish' credentials | <Various> | Campaign destroyed | ??? |

cigital


# Architectural Risk Analysis

## Recognizing Insecure Design

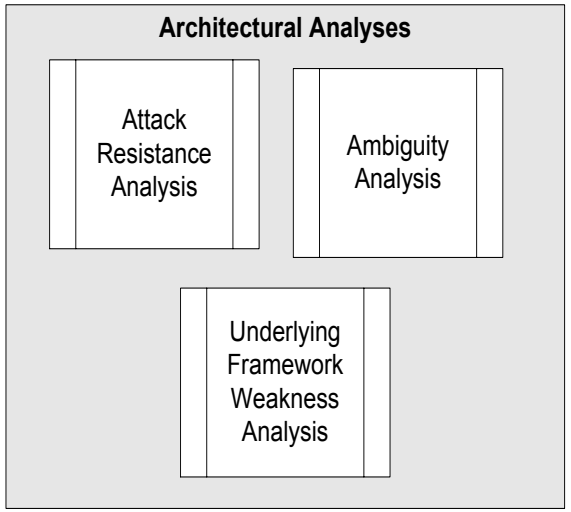
*"Techniques to ensure that your software architecture exhibits all of the required non-functional, emergent properties in concert with one another."*

John Steven

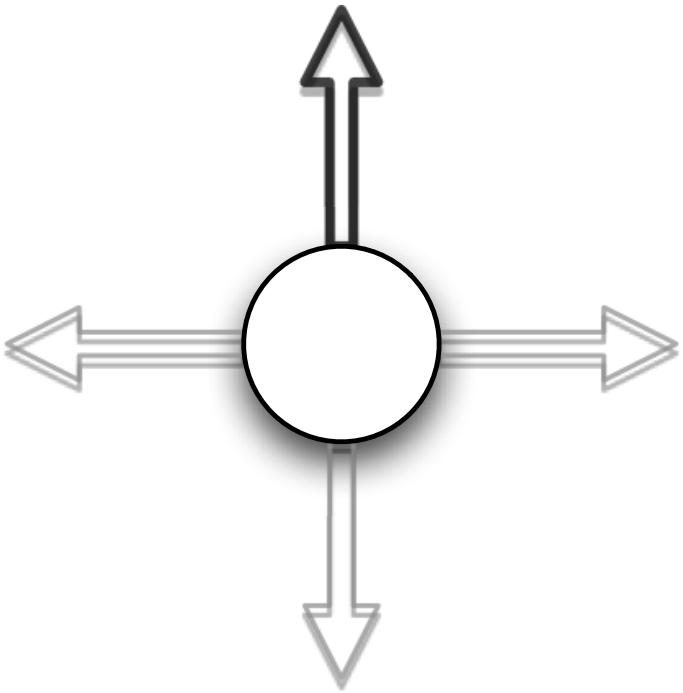
Software Security Principal

Cigital, Inc.

# 7.3 – Do ‘ARA’




Callback
Privileged threads



Break encapsulation
Data leakage
Callbacks

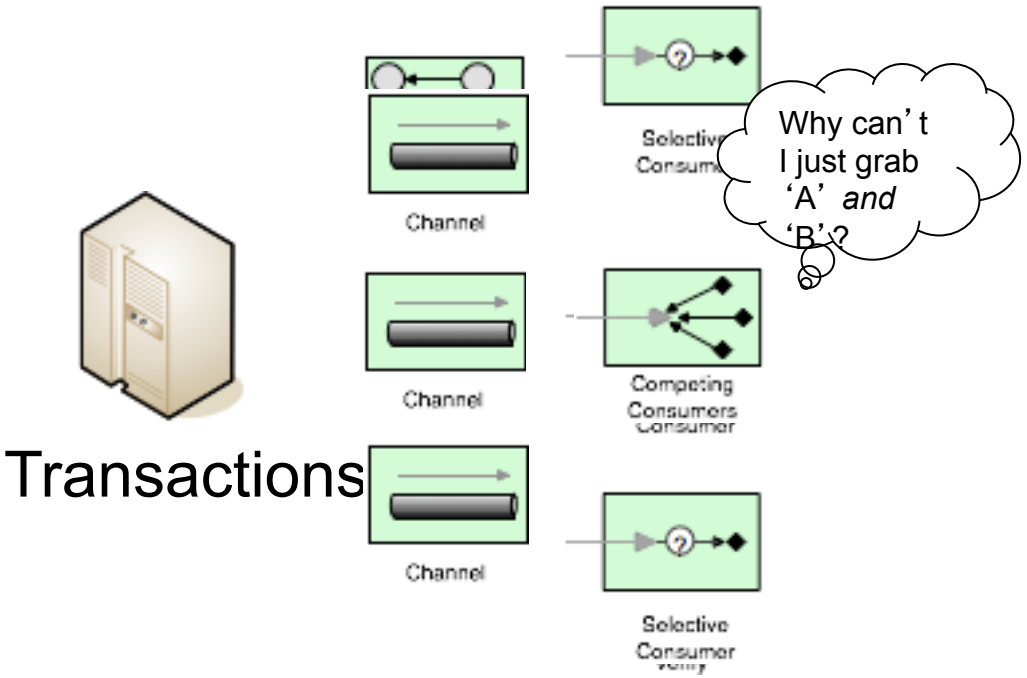
Lure component
- Manipulate query
- Access function

Impersonation
Escalation

X

cigital

# ARA's find 'Flaw's



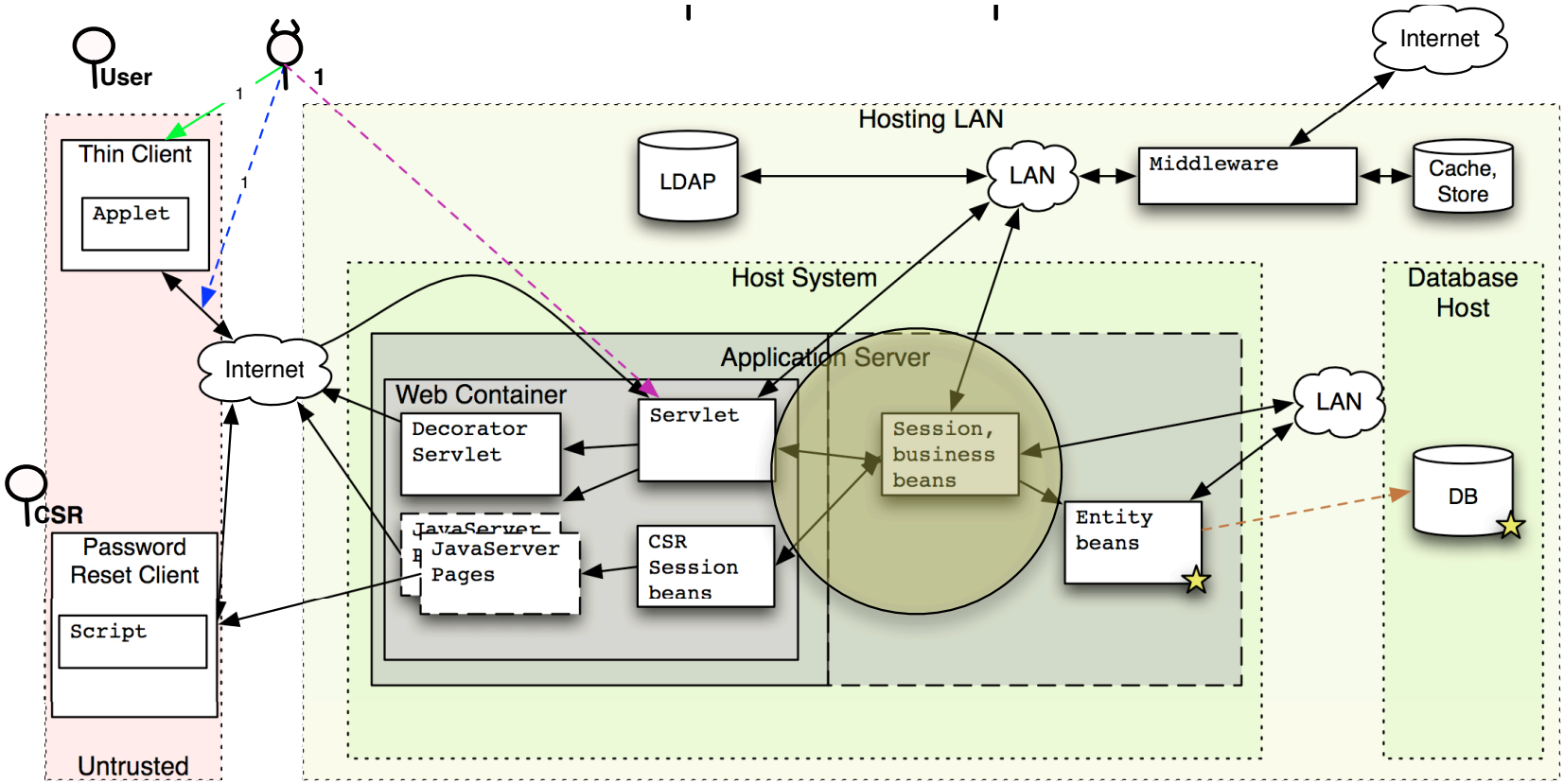
# ARA Is About Identifying Flaws

FLAWS - Design

- Misuse of cryptography
- Duplicated data or code
- Lack of consistent input validation
- Missing authorization checks
- Insecure or lack of auditing
- Lack of authentication or session management on APIs
- Missing compartmentalization

cigital

# Augment with ‘Goal-oriented’ Vectors



# 1. Enumerate Potential Failures in design elements

Ask: is each element:

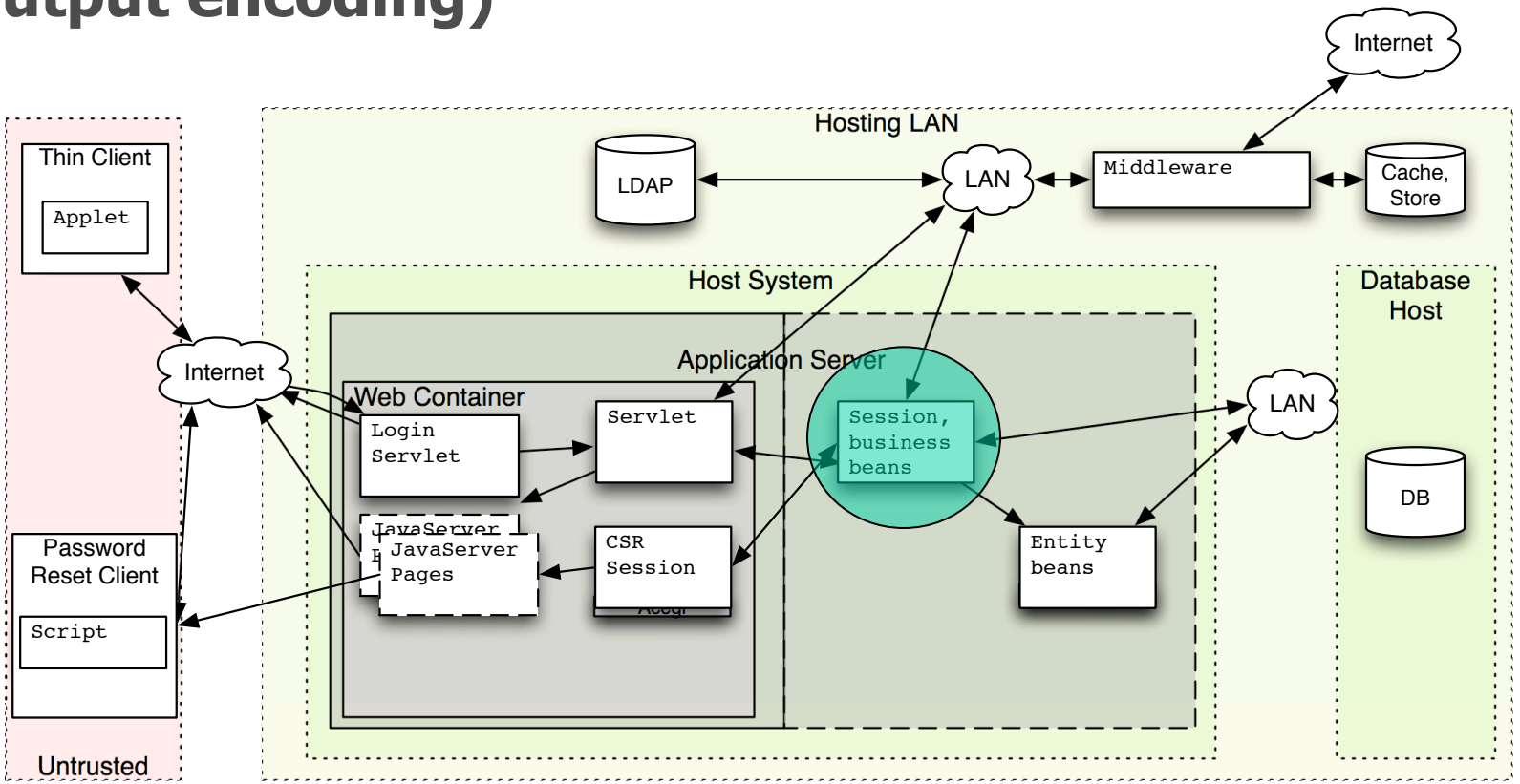
- Control absent?
- Used ineffectively
  - What's the effect of digesting a password?
  - Does code signing prevent malicious code?
  - What does SSL (w/o) certs provide?
- Implemented correctly?
- Present, but unused

Jeff Williams has suggested this framework for security controls for some time

cigital

# 2. Find Key Structural Components

**Component diagrams show critical choke points for security controls (input validation, authentication, output encoding)**



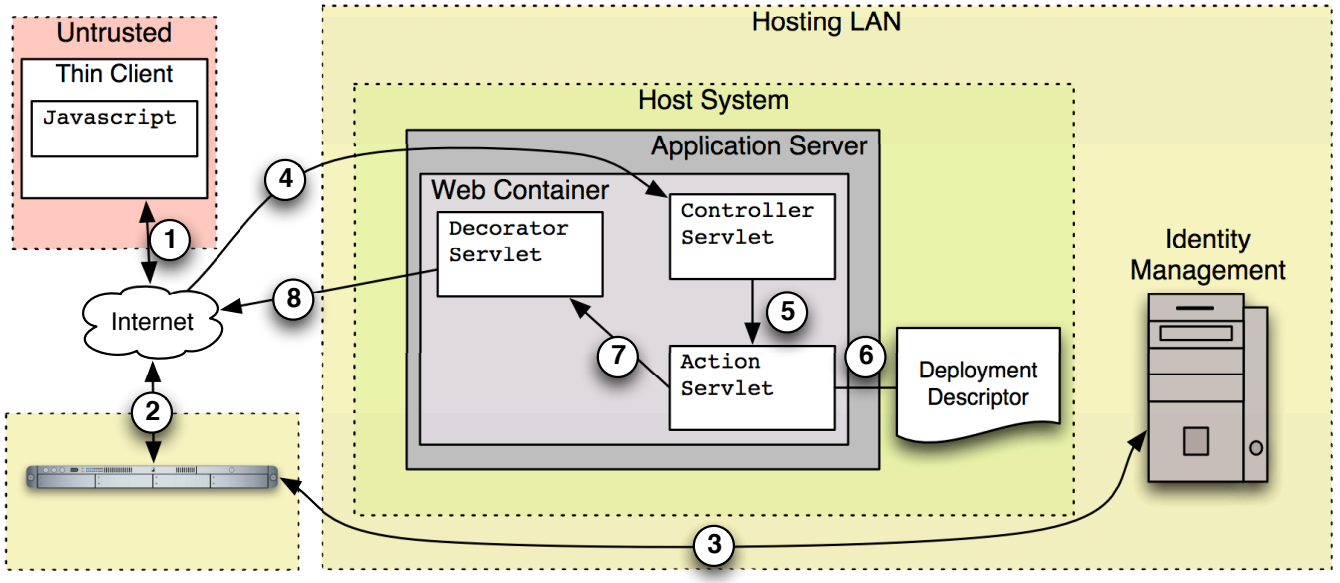

cigital

# Critical Functionality Pointers

- Based on idiom/paradigm
- Control Patterns
  - Command Patterns
  - Inversion of Control containers
  - Session Management and other flow-drivers
- Underlying frameworks
  - Callbacks
  - Plugins
  - Frameworks
- Security features

cigital

# Exercise: What do you expect



# Tie-back to Threat Modeling

# Pass tech.-specific KM by REFERENCE

- Do not duplicate technical resources in your T.M., that's a later step.

- Reference:
  - Code review guide:
    - http://www.owasp.org/index.php/Code_Review_Guide_Frontispiece
  - Testing guide:
    - http://www.owasp.org/index.php/Category:OWASP_Testing_Project

cigital

# How

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, unauthorized, Internet user | Directly request and gain access to another user's info | • Forceful browsing<br>• Failure to demand auth<br>• Session Fixation | PR Incident<br>*Non-compliance*<br>Increase QSA assessment cost | • FD:3.2: session mgmt<br>• SR:2.3.4: URL, forms data<br>• FD: 3.4: Controller design<br>• SD: 1.3: WebSeal integration<br>• SP:1.3: Demanding Auth. |
| Public or partner, authorized user | Upload malicious content as part of normal workflow | • Upload exceptional large file<br>• Use file as injection vector<br>• Upload dual-type file (such as GIFAR) | SLA violation<br>*Data loss/ corruption*<br>*Wholesale system breach* | • SP: 9.3: Virus scanning uploads<br>• FD: 6.1: Upload quota<br>• SP: 2.2: Filtering input<br>• SD: 6.3: Re-encoding files<br>• SR: 6.5: Spec for valid file types |

- THIS is where the:
  - Technical meat is…
  - The deep security domain knowledge is…

cigital

# Pilfer tech.-specific security standards

```
action.setComment(node.getTextContent());


name == "ITEM_PRICE"){
ot_price){
  transaction.setOrderPrice(
  Double.parseDouble(node.getTextContent()));
  got_price = true;
e {
  Logger.log("Input Validation Error: ITEM_PRICE set t
  transaction.abort();




name == "ITEM_NO"){
ot_item_number){
  transaction.setItemNo(
        Integer.parseInt(node.getTextContent());
e {
  Logger.log("Input Validation Error: ITEM_NO set twic
  transaction.abort();
```

## As well as explanation

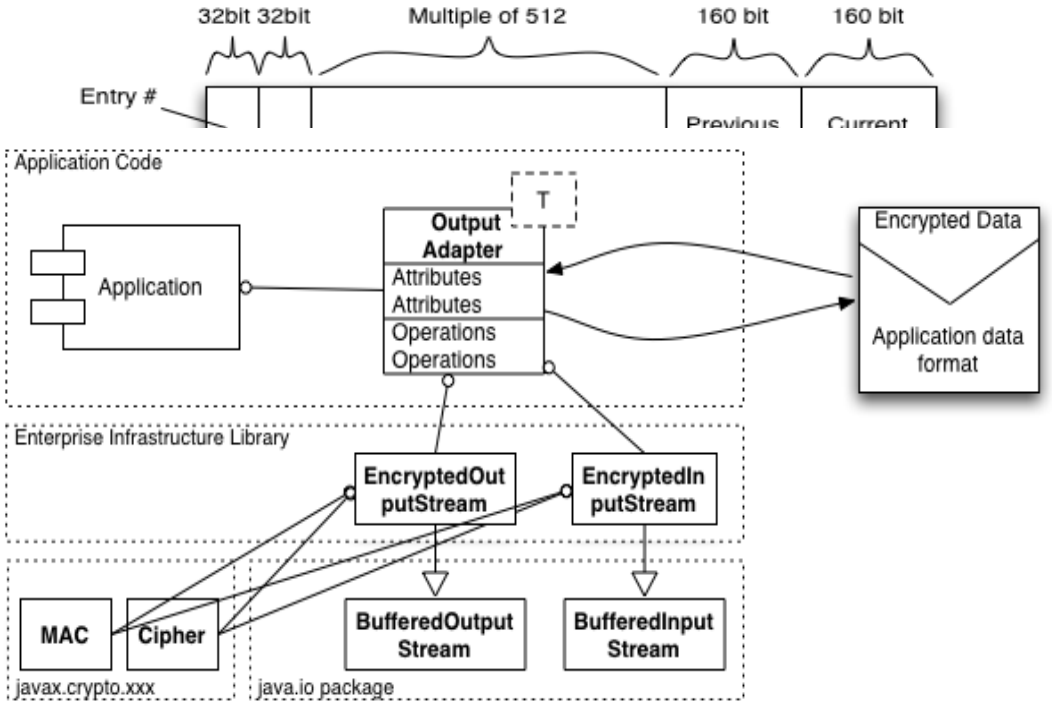
A particular kind of input filtering avoids this class of potential XML injection attacks. Specifically, parser code should look for inappropriately duplicated tag elements and treat any such as errors that cause an abort in processing. In other cases, continuing having failed silently may be preferable. In either case, the error should be logged.

Only particular elements fall prey to such semantics, such as invariants like price. The example below demonstrates simple protection of elements with an expected cardinality of one with a conditional guard:

cigital

# Advice can extend into design

- Sketch high-level design
- Outline low-level design, implementation
- Includes instructive code snippets:
  - Technology specific
  - J2EE version specific
- Describes both:
  - "must use"
  - "avoid"




***"Here's how to use 'out of the box' APIs securely, for logging…"***

```
// Create a new JCA PasswordCredential based on username and password
credential = new PasswordCredential(callerName, callerPassword.toCharArray());

// Associate JCA managed connection with PasswordCredential
credential.setManagedConnectionFactory(conn);
```

cigital

# How

- Update your information when:
  - When research shows you attacks have been facilitated/automated
    - Metasploit variety, SSL attack APIs, etc.
  - Incident response shares data
  - Industry research shows attacks

cigital

# Know thy enemy & how they attack you (REDUX)

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, unauthorized, Internet user | Directly request and gain access to another user's info | • Forceful browsing<br>• Failure to demand auth<br>• Session Fixation | PR Incident<br>*Non-compliance*<br>Increase QSA assessment cost | • FD:3.2: session mgmt<br>• SR:2.3.4: URL, forms data<br>• FD: 3.4: Controller design<br>• SD: 1.3: WebSeal integration<br>• SP:1.3: Demanding Auth. |
| Public or partner, authorized user | Upload malicious content as part of normal workflow | • Upload exceptional large file<br>• Use file as injection vector<br>• Upload dual-type file (such as GIFAR) | SLA violation<br>*Data loss/ corruption*<br>*Wholesale system breach* | • SP: 9.3: Virus scanning uploads<br>• FD: 6.1: Upload quota<br>• SP: 2.2: Filtering input<br>• SD: 6.3: Re-encoding files<br>• SR: 6.5: Spec for valid file types |

- Who: Skill, Motivation, Access
- What: Technology-agnostic conceptual
- How: The specific tactics that might make attack successful
- Impact: the cost of successful attack
- Mitigation: traceability into elements designed to resist, identify, or prevent attack

cigital

# Who owns the table?

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, unauthorized, Internet user | Directly request and gain access to another user's info | • Forceful browsing<br>• Failure to demand auth<br>• Session Fixation | PR Incident<br>*Non-compliance*<br>Increase QSA assessment cost | • FD:3.2: session mgmt<br>• SR:2.3.4: URL, forms data<br>• FD: 3.4: Controller design<br>• SD: 1.3: WebSeal integration<br>• SP:1.3: Demanding Auth. |
| Public or partner, authorized user | Upload malicious content as part of normal workflow | • Upload exceptional large file<br>• Use file as injection vector<br>• Upload dual-type file (such as GIFAR) | SLA violation<br>*Data loss/ corruption*<br>*Wholesale system breach* | • SP: 9.3: Virus scanning uploads<br>• FD: 6.1: Upload quota<br>• SP: 2.2: Filtering input<br>• SD: 6.3: Re-encoding files<br>• SR: 6.5: Spec for valid file types |

Business Analyst

Business Analyst

(Security) Architect

(Security) Architect

cigital

# Don't worry about "left to right"

| Who | What | How | Impact | Mitigation |
|---|---|---|---|---|
| Public, ***UNAUTHORIZED***, Internet user | Directly request and gain access to another user's info | • Forceful browsing<br>• Failure to demand auth<br>• Session Fixation<br>• ***CSRF*** | PR Incident<br>*Non-compliance*<br>Increase QSA assessment cost<br>***Fraud*** | • FD:3.2: session mgmt<br>• SR:2.3.4: URL, forms data<br>• FD: 3.4: Controller design<br>• SD: 1.3: WebSeal integration<br>• SP:1.3: Demanding Auth. |
| Public or partner, ***authorized user*** | Upload malicious content as part of normal workflow | • Upload exceptional large file<br>• Use file as injection vector<br>• Upload dual-type file (such as GIFAR) | SLA violation<br>*Data loss/ corruption*<br>*Wholesale system breach* | • SP: 9.3: Virus scanning uploads<br>• FD: 6.1: Upload quota<br>• SP: 2.2: Filtering input<br>• SD: 6.3: Re-encoding files<br>• SR: 6.5: Spec for valid file types |

- When testing finds an attack:
  - First, decide if its *impact* warrants further exploration
  - Are additional impacts possible?
  - Consider *what* conceptual goals the attack supports
  - Then consider *who* could launch the attack against the application
- After analysis converges, iterate secure design

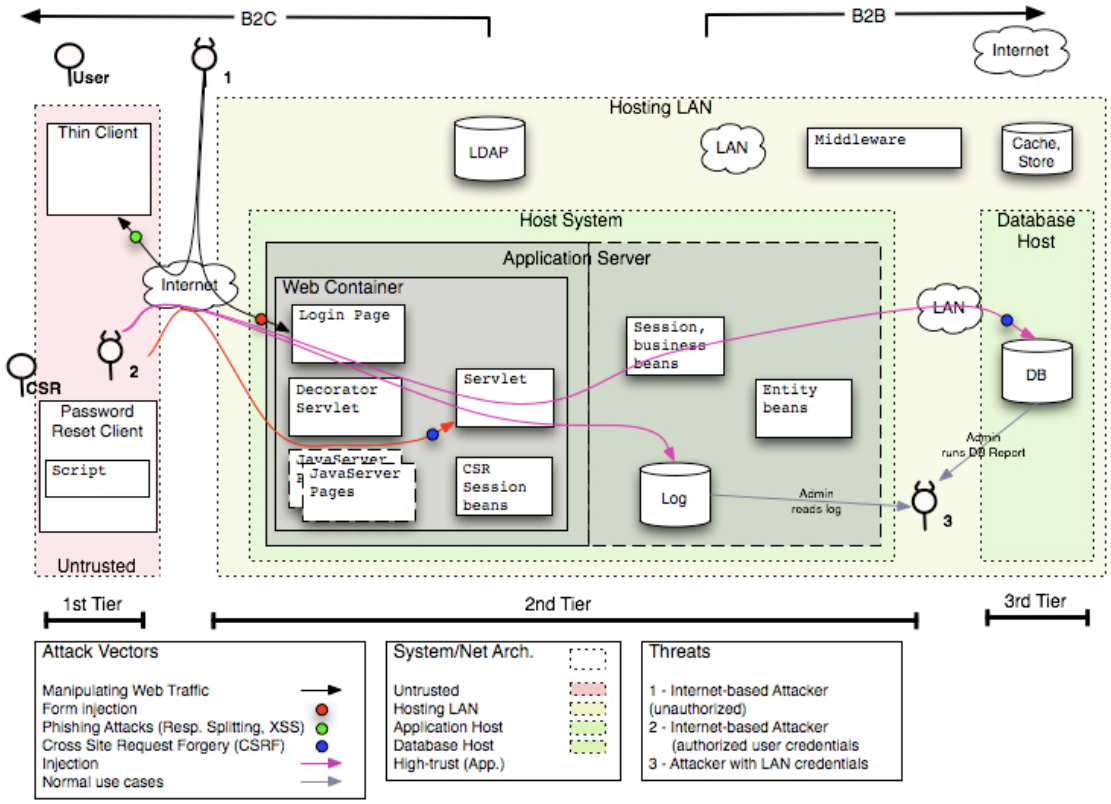
cigital

# Take Homes

- Base Threat Model in *software architecture*
- When specific *use (cases)* and high-level architecture are defined:
  - Inventory roles, entitlements, if one doesn't exist
  - Inventory assets: sensitive data, privileged components
- Enumerate initial *attack vectors*
  - Use common 'low-hanging' fruit
- Elaborate more attacks
  - Find opportunities for privilege escalation
  - Layer attacks to target or 'hop' to assets
  - Fill in gaps by 'inventing' attacks
- Use Threat Modeling to drive security testing:

cigital

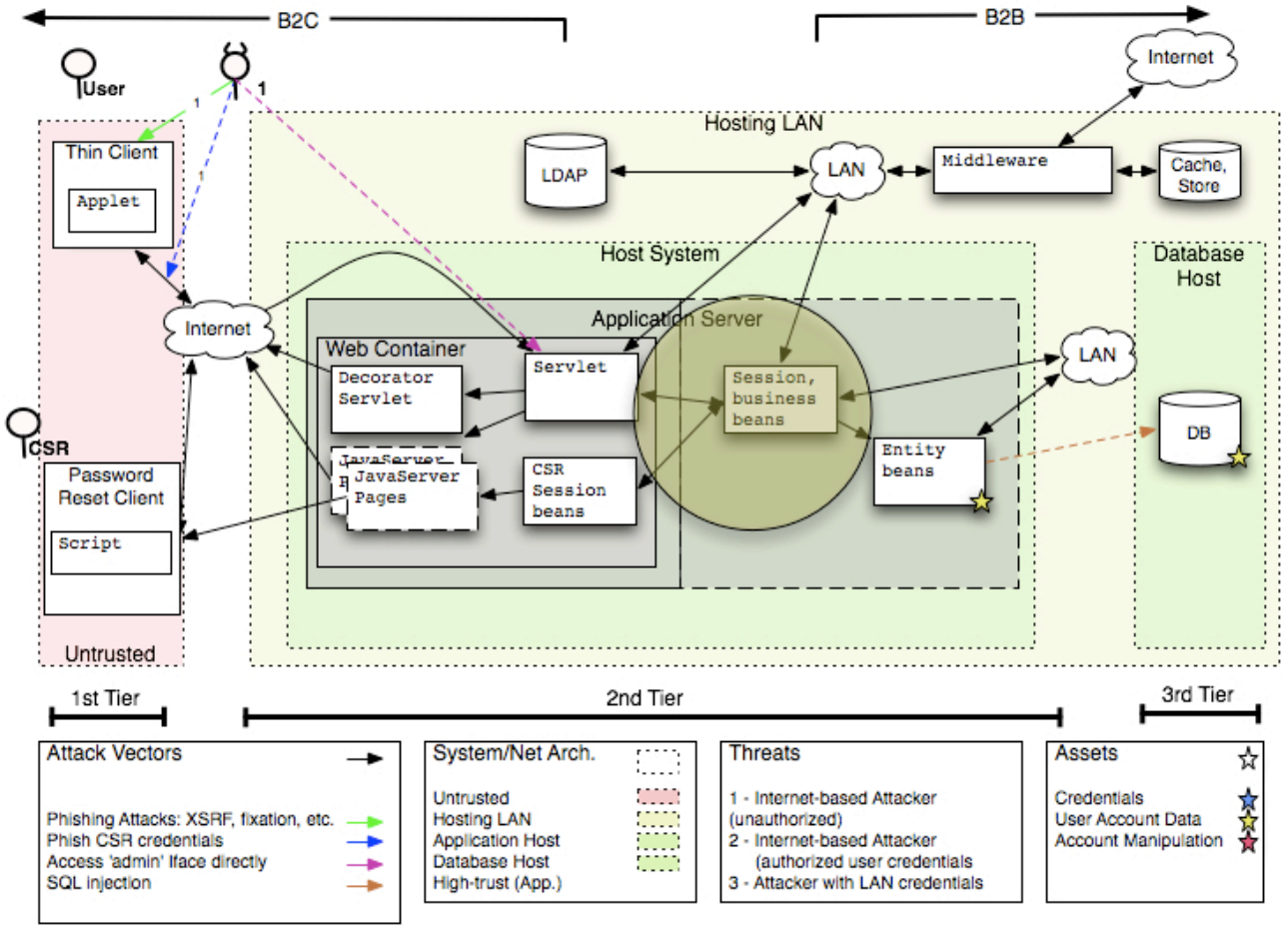
# Tips #2: Target Using Layered Attacks




- Bootstrap later attacks with those that ‘deliver’
  - Use one layer to exploit another (net, app)
  - Combine attacks to reach desired target

cigital

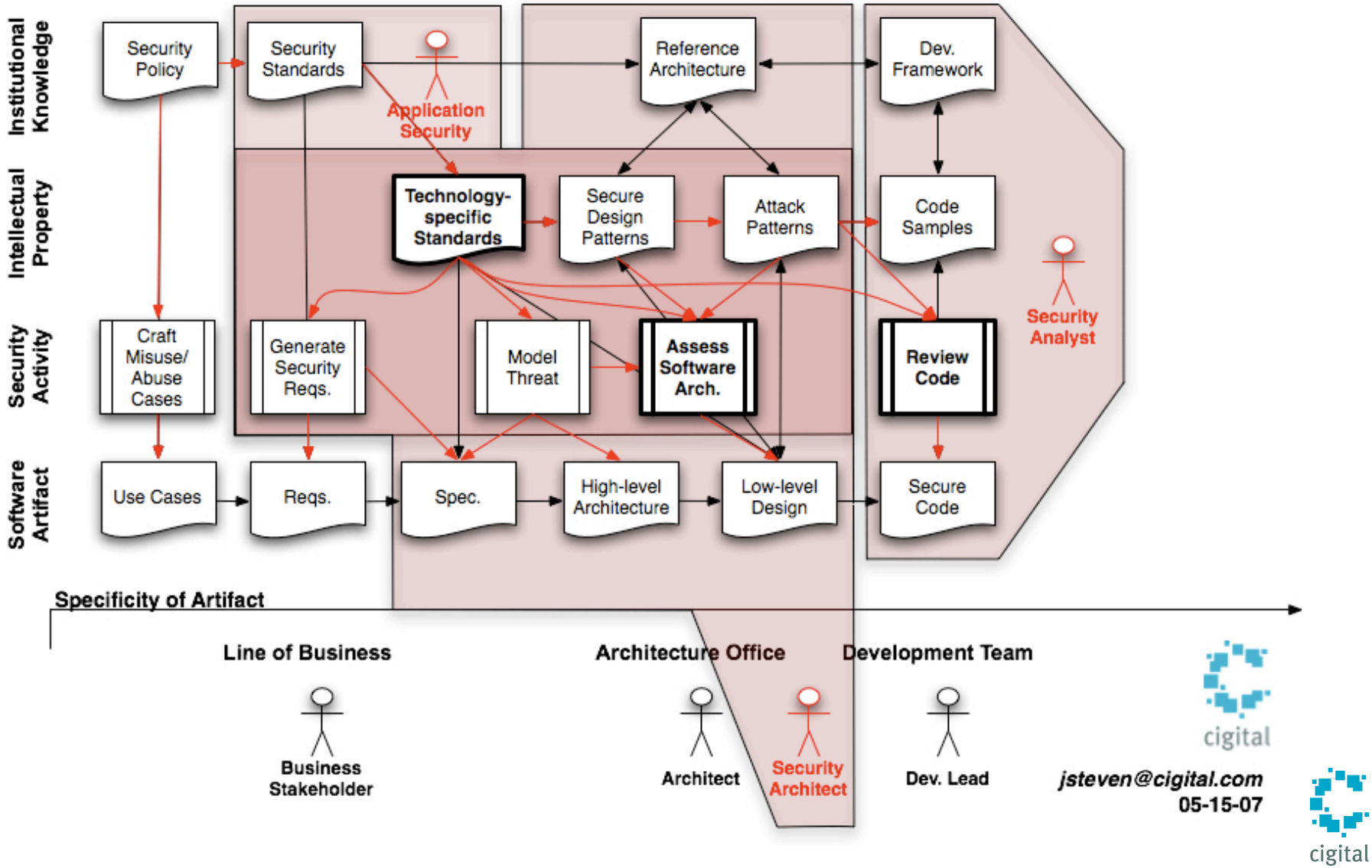
# Tips #3: Filling the Gaps…



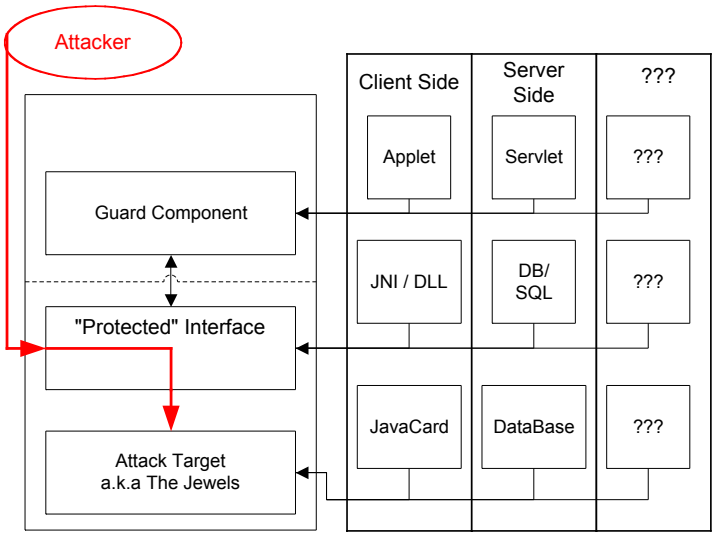

- How do we design tests to fill this gap?

cigital

# A security knowledge architecture



# Attack Patterns




### *1.1 Attack Pattern*

- **Description**

  General Indication

  General Recipes for Exploit

  General Protection Schemes

- **Known Instances:** A brief description of:

1. Who and on what contract the instance was identified
2. A brief description of why the vulnerability is applicable to the pattern
3. Mapping from general scenario to components within the instance

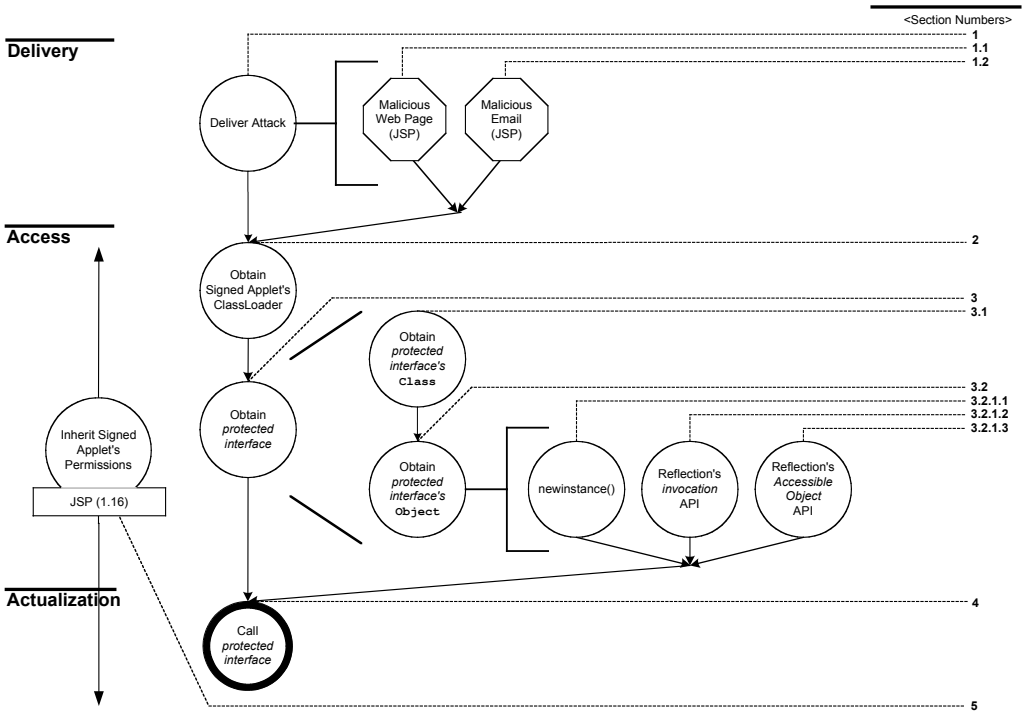
**Indications:** Signs of weakness consultants look for to determine whether a particular security vulnerability scenario is likely to apply to a system. Indications are specific concrete properties of the software which are easily detectable.

- **Protection Schemes**

cigital

# Exploit Graphs

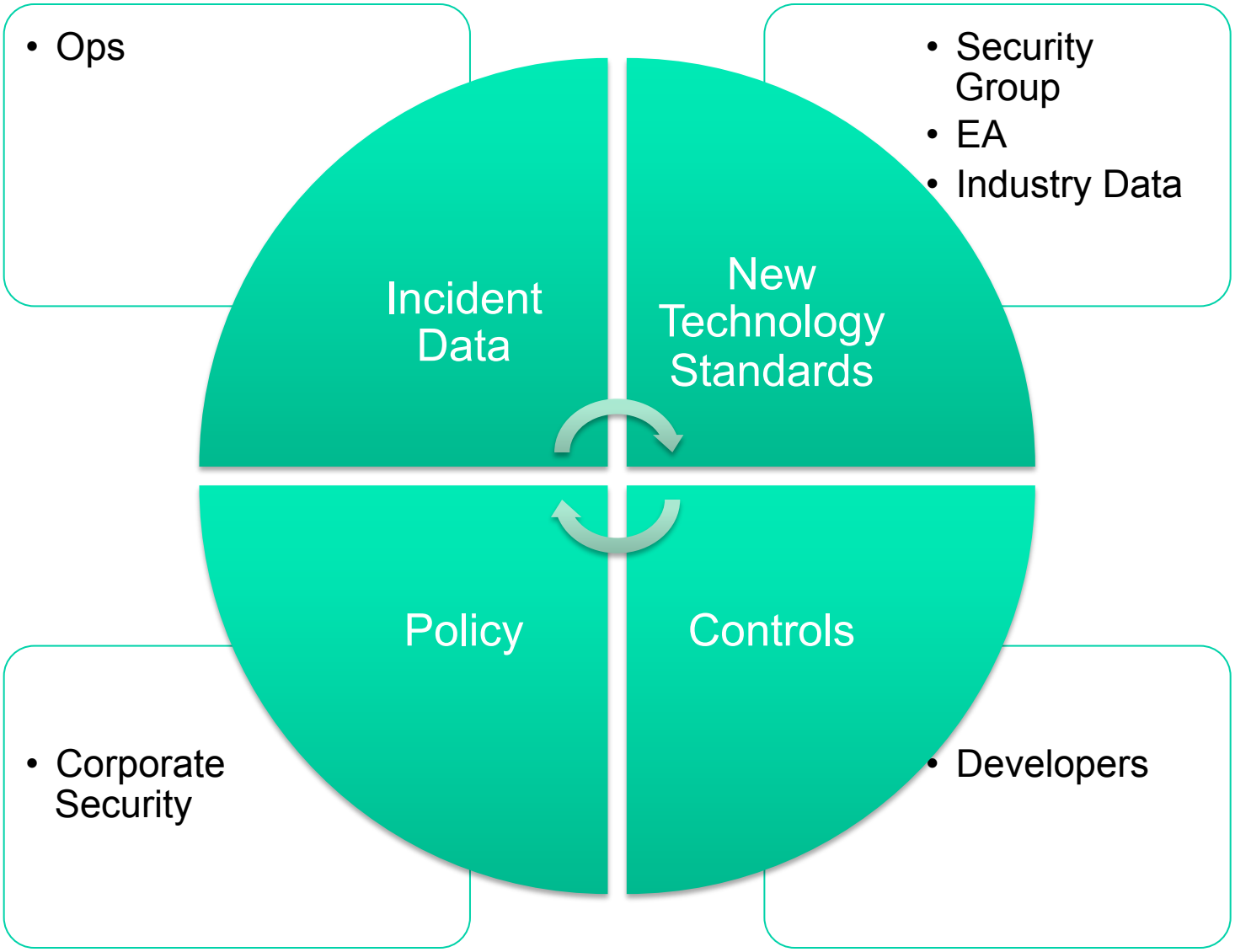
- Flowchart of:
  - Delivery
  - Increasing privilege
  - Gaining access
  - Subverting protections
  - Architectural failure
  - Attack Actualization
- Shows effective mitigation
- Compendium Chart




| Step # | | Detail: How/What | Conditions | Protection |
|---|---|---|---|---|
| Delivery | 1 | Deliver Attack: (get attack code onto machine w/ Asset) | Client must have Internet access | |
| | 1.1 | Trick user to point browser to Javascript. | Browser must have "run Javascript" enabled. | Disable Javascript in browser. **NOTE:** doing so prevents other sites from working. |
| | 1.2 | Send victim email containing malicious Javascript. | User's mail reader must interpret Javascript. | Disable Javascript execution in mail reader. |

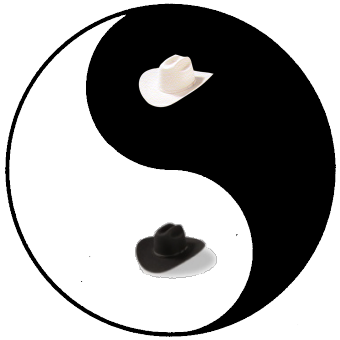
cigital

# Steering Committee: Roles with TM Process




cigital

# Thank you for your time.



cigital