

Threat Modeling

SecAppDev 2017

whoami

- Run Cigital's Architecture Analysis practice
 - 30+ years in software development in many different domains
 - 15+ years focusing on software security
-
- Executive Director of IEEE Computer Society CSD initiative

<http://cybersecurity.ieee.org/center-for-secure-design/>



@jimdelgrosso

About Me

- Andrew Lee-Thorp
- alee-thorp@cigital.com
- @Cigital - Android assessments and tool development, security developer, source code review, threat modelling
- > 10 years cutting code
- Occasional speaker

What Is Threat Modeling?

- A software design analysis capable of finding flaws
- A defect discovery technique that is part of your SSI
 - You do have an SSI, right?

The Defect Universe – Bugs and Flaws



Cross Site Scripting
Buffer Overflow



Weak/Missing/Wrong
Security Control

(Implementation) BUGS

(Design) FLAWS

Code Review

Penetration Testing

Threat Modeling

Threat Modeling Vocabulary

Asset

Likelihood

Security Control

Impact

Threat Agent

Mitigation

Attack Surface

Traceability Matrix

Threat

Threat Model Process

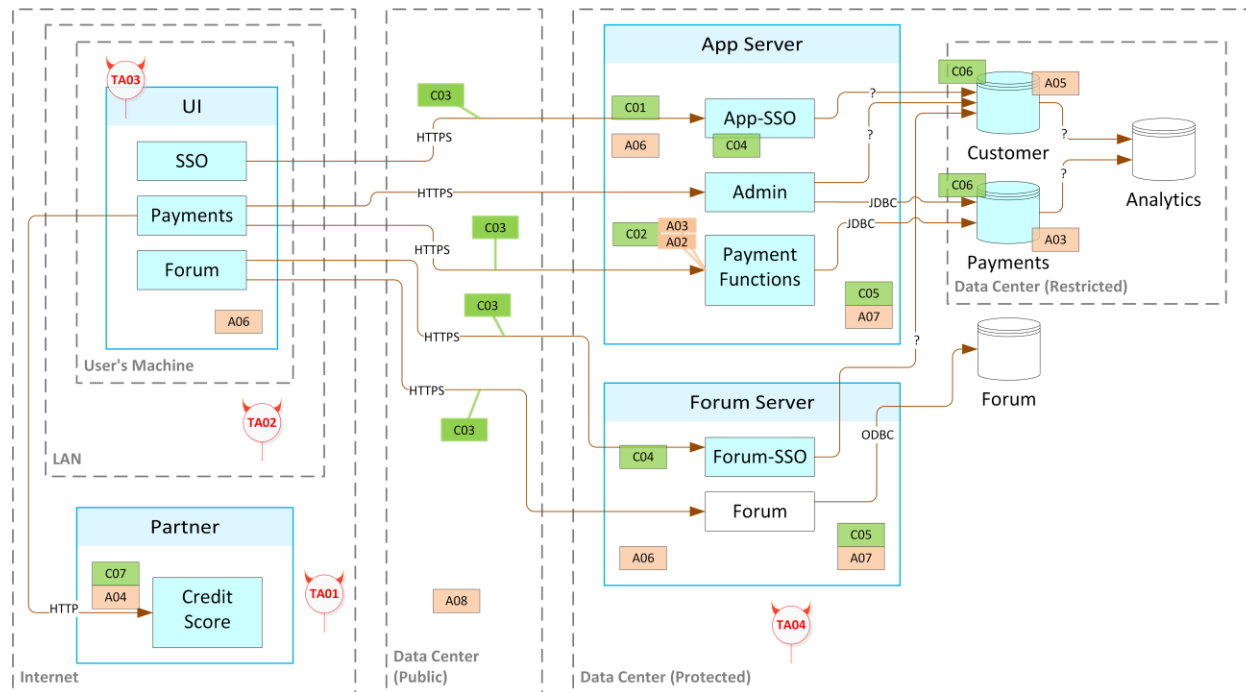
- Define scope and depth of analysis
- Gain understanding of what is being modeled
- Model the System
- Model the attack possibilities
- Interpret the threat model
- Create the Traceability Matrix

System Threat Models

System Threat Model

Characteristics of the System Threat Model include:

- Holistic view of application's security posture
- Considers both application and infrastructure
- Builds roadmap for additional security activities



Decompose and Model the System

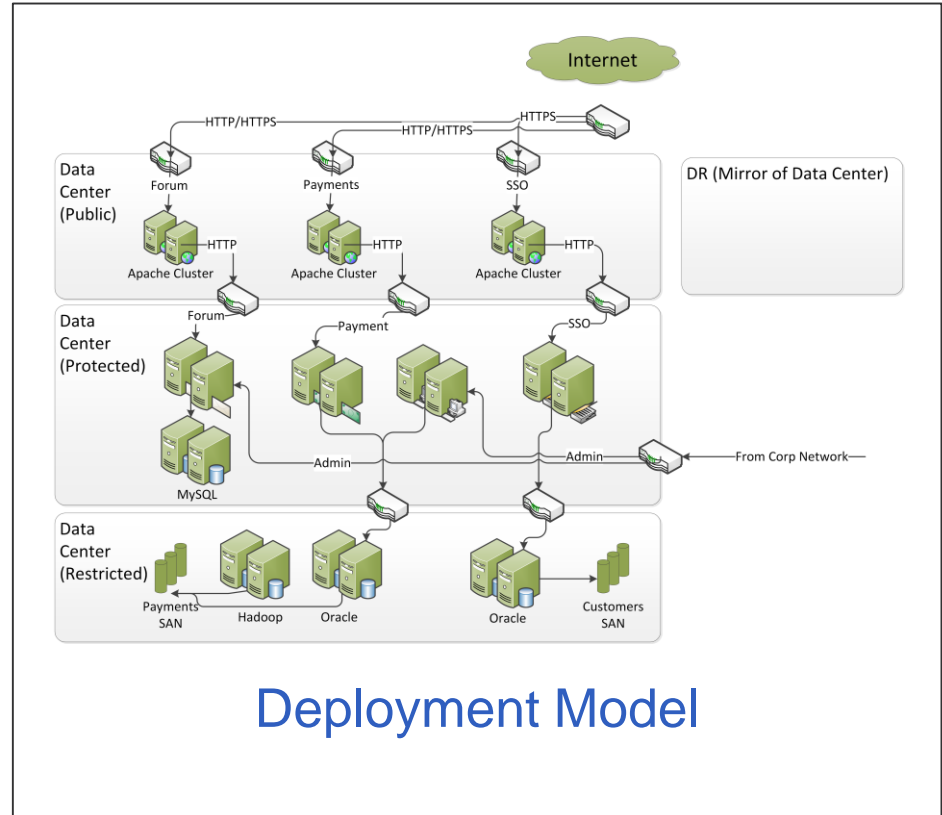
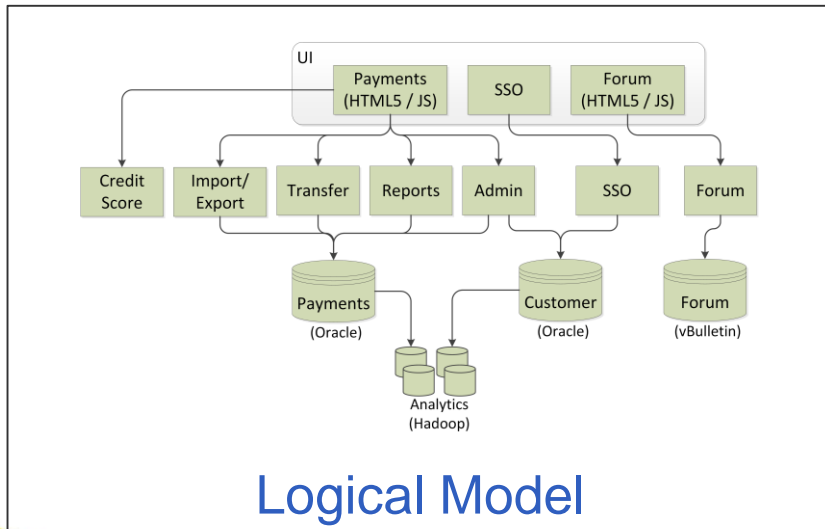
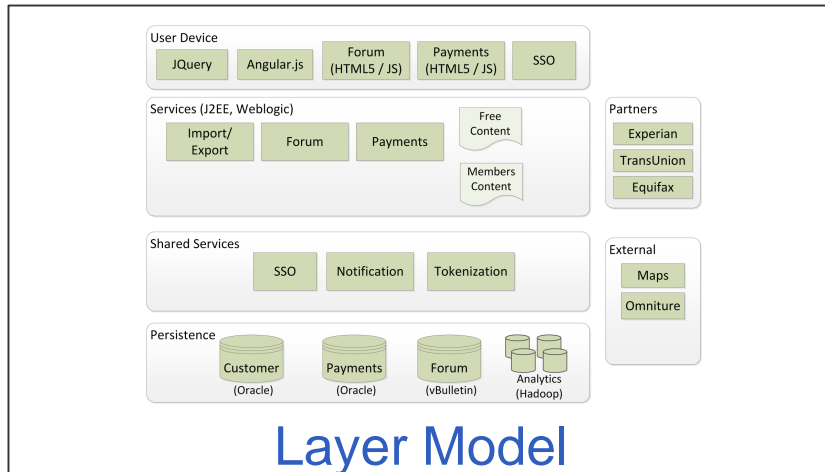
- Understand how the system works (before trying to break it)
 - Who uses the system?
 - What are the business goals?
 - What are the dependencies between systems?
 - What systems (components) does this system make use of
 - What systems (components) use this system
- Review (some) development documentation
- Interview members of various teams

Gain Understanding from Interviews

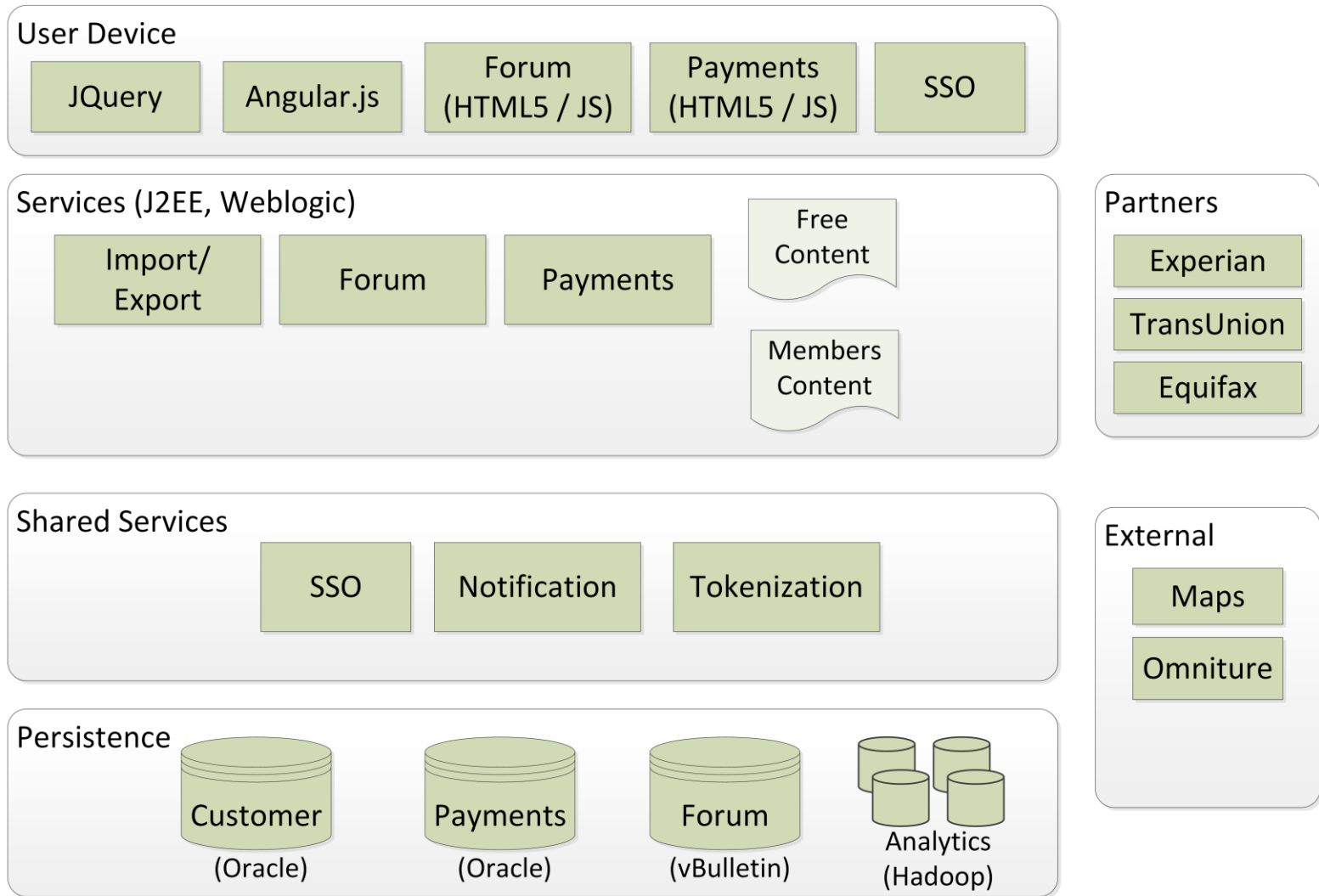
From the interview, you learn:

- Social-networking application; accepts payment
- Some content and features membership-only; some, free
- App is JavaEE app; uses WebLogic as JavaEE container
- Backend database is Oracle
 - Stores user's preferences
 - Produces some membership-only reports
- Web UI built using JQuery JavaScript library
- Web UI calls third-party REST services for user-specific content
- User connectivity and interface to backend services uses HTTPS

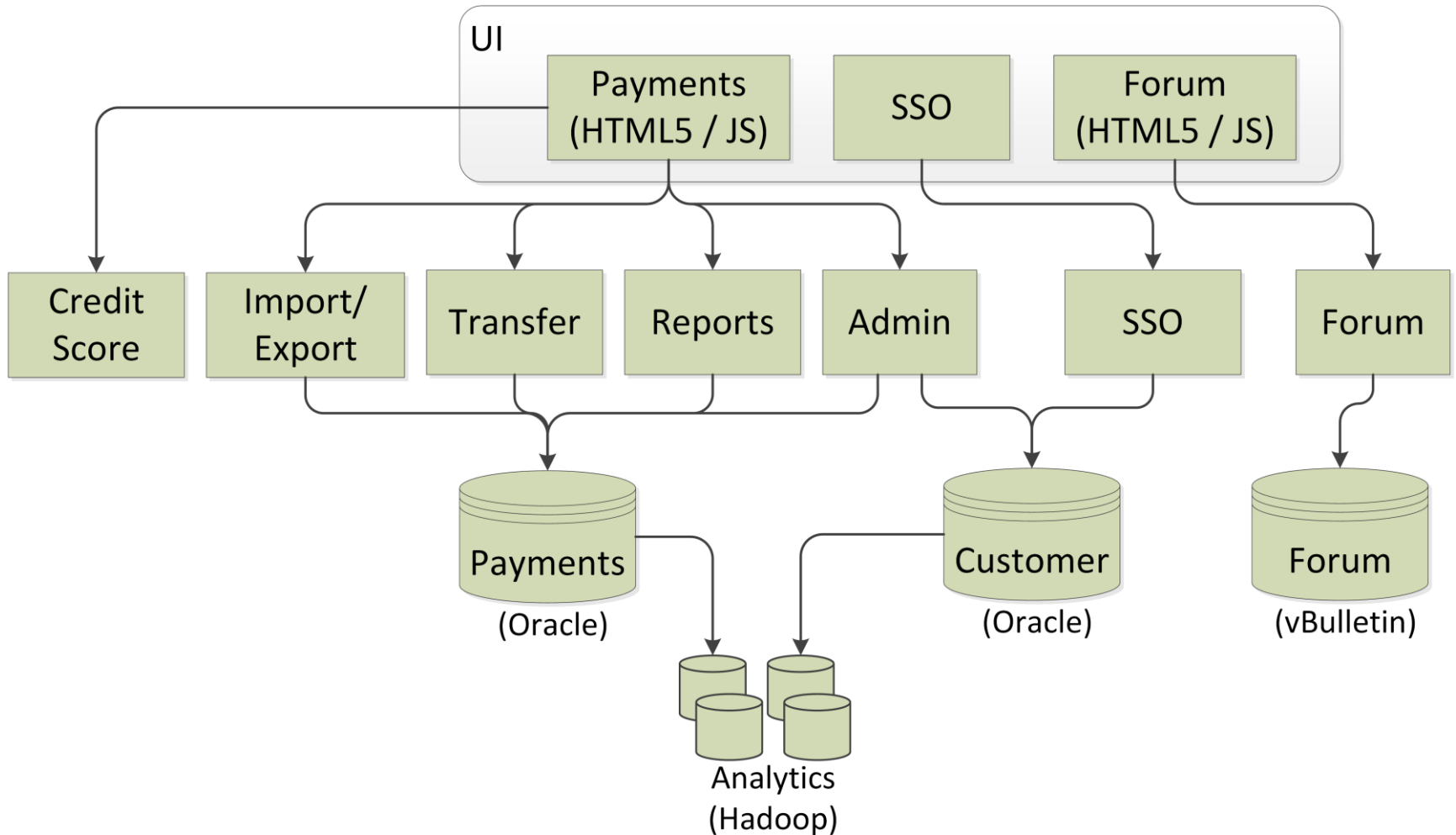
Diagrams from Development/Infrastructure Teams



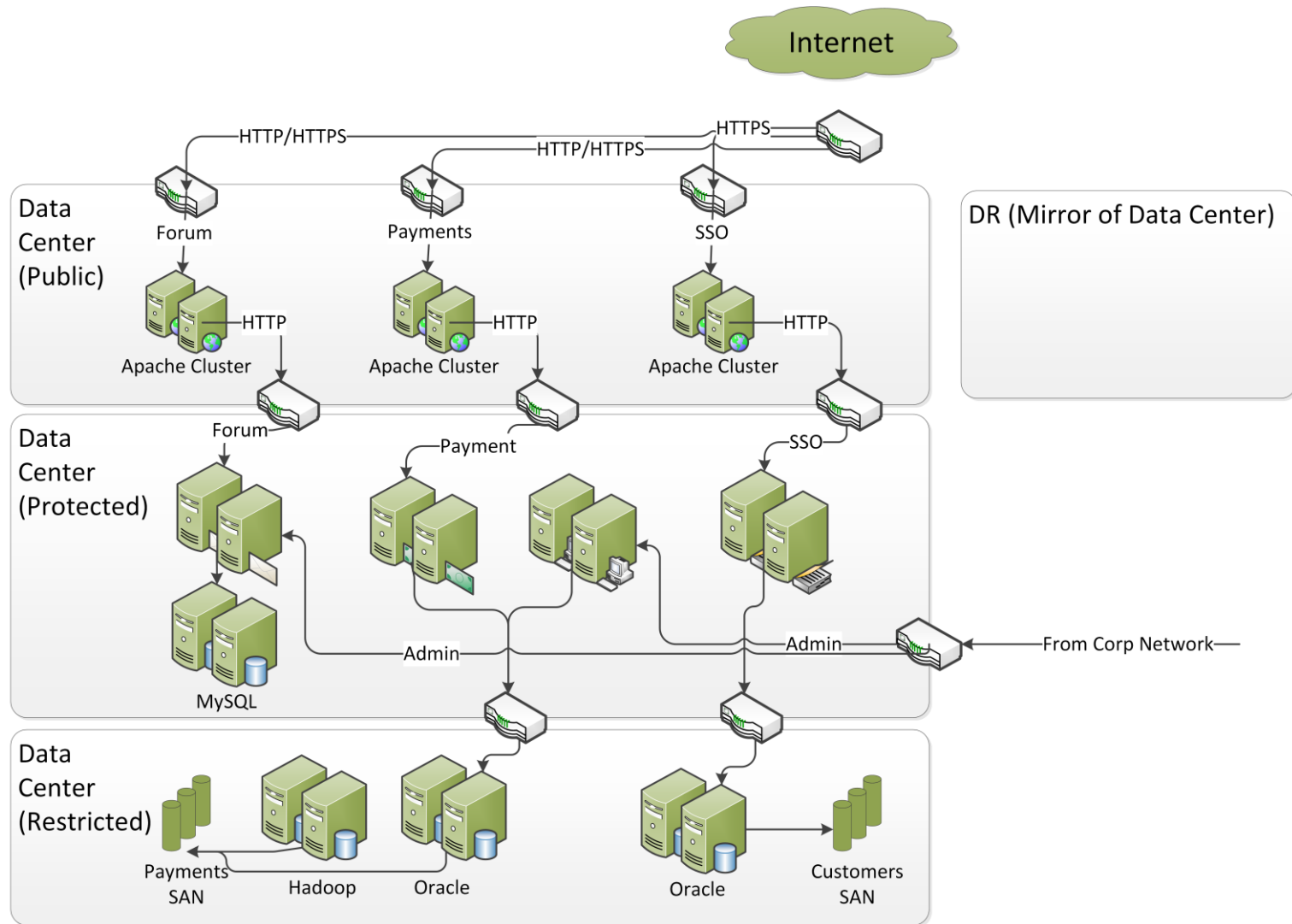
Layer Model (from Development)



Logical Model (from Development)



Deployment Model (from Infrastructure)



Modeling the System Structure

Based on the interviews and development/infrastructure diagrams, create a model that shows:

- Which components are in-scope for this “release”
- How control flows between these components
- How components and flows relate to host boundaries and network zones
- Application layer communication protocols that connect components

Model can use an existing diagram or one you create:

- For this class, we’ll create our own

Modeling the Attack Possibilities

To model the attack possibilities, continue to analyze the information we've collected in our interviews. And now add the related threat model elements:

Assets	Data and functionality that the system must protect
Security Controls	Mechanisms currently designed and implemented to protect the Assets
Threat Agents	Actors that want to harm the system

Juxtaposing the attack possibilities and the system creates the actual threat model. Interpreting the model produces a list of potential attacks.

Identifying Assets from Interviews

Information collected in development interviews:

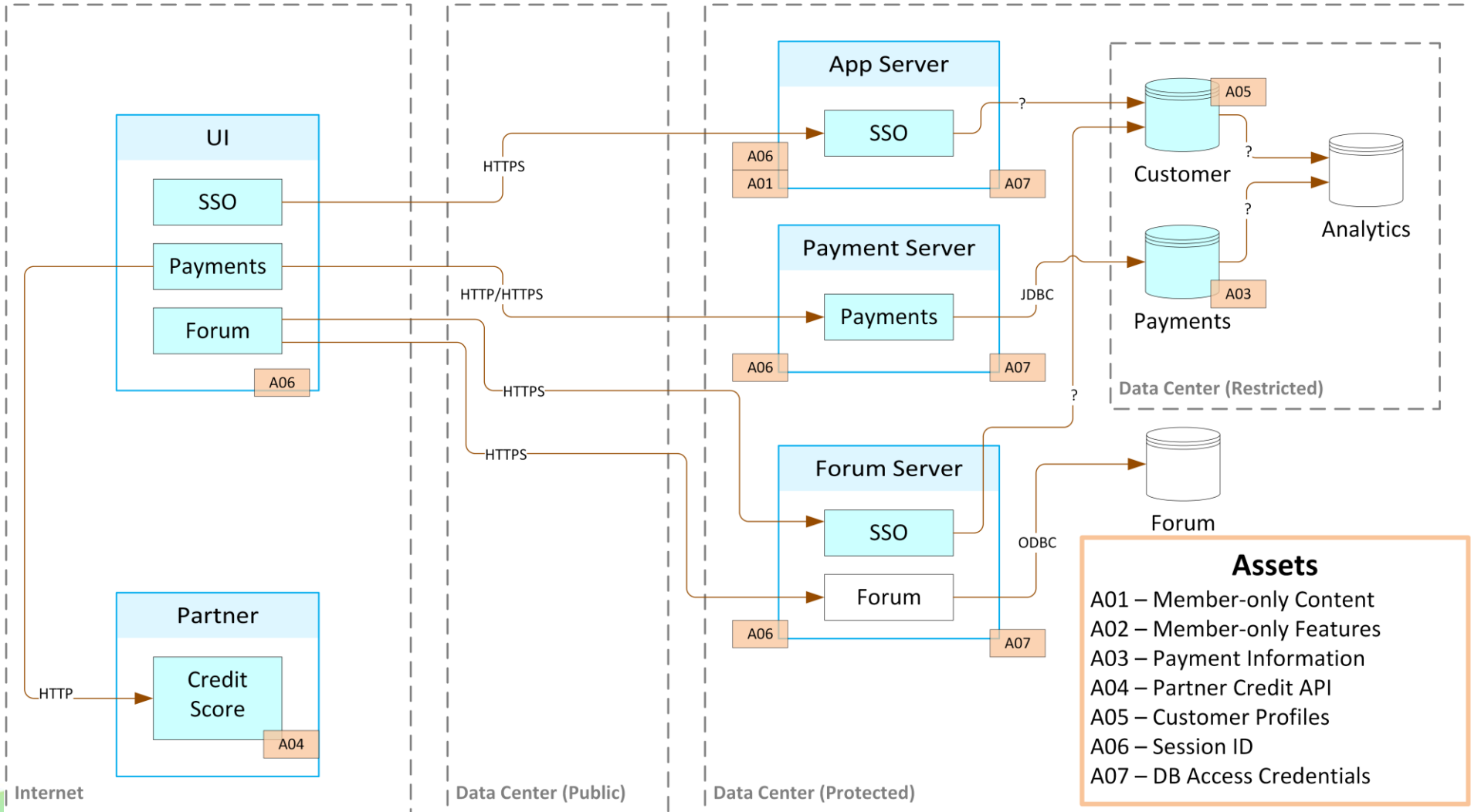
- Social-networking application; accepts payment
- Some content and features membership-only; some, free
- App is JavaEE app; uses WebLogic as JavaEE container
- Backend database is Oracle
 - Stores user's preferences
 - Produces some membership-only reports
- Web UI built using JQuery JavaScript library
- Web UI calls third-party REST services for user-specific content
- User connectivity and interface to backend services uses HTTPS

Identifying Assets from Interviews

Information collected in development interviews:

- Social-networking application; accepts payment
- Some content **[A01]** and features **[A02]** membership-only; some, free
- App is JavaEE app; uses WebLogic as JavaEE container
- Backend database is Oracle
 - Stores user's preferences **[A05]**
 - Produces some membership-only reports
- Web UI built using JQuery JavaScript library
- Web UI calls third-party REST services **[A04]** for user-specific content
- User connectivity and interface to backend services uses HTTPS

Model the Attack Possibilities: Assets



Identifying Controls from Interviews

Information collected in development interviews:

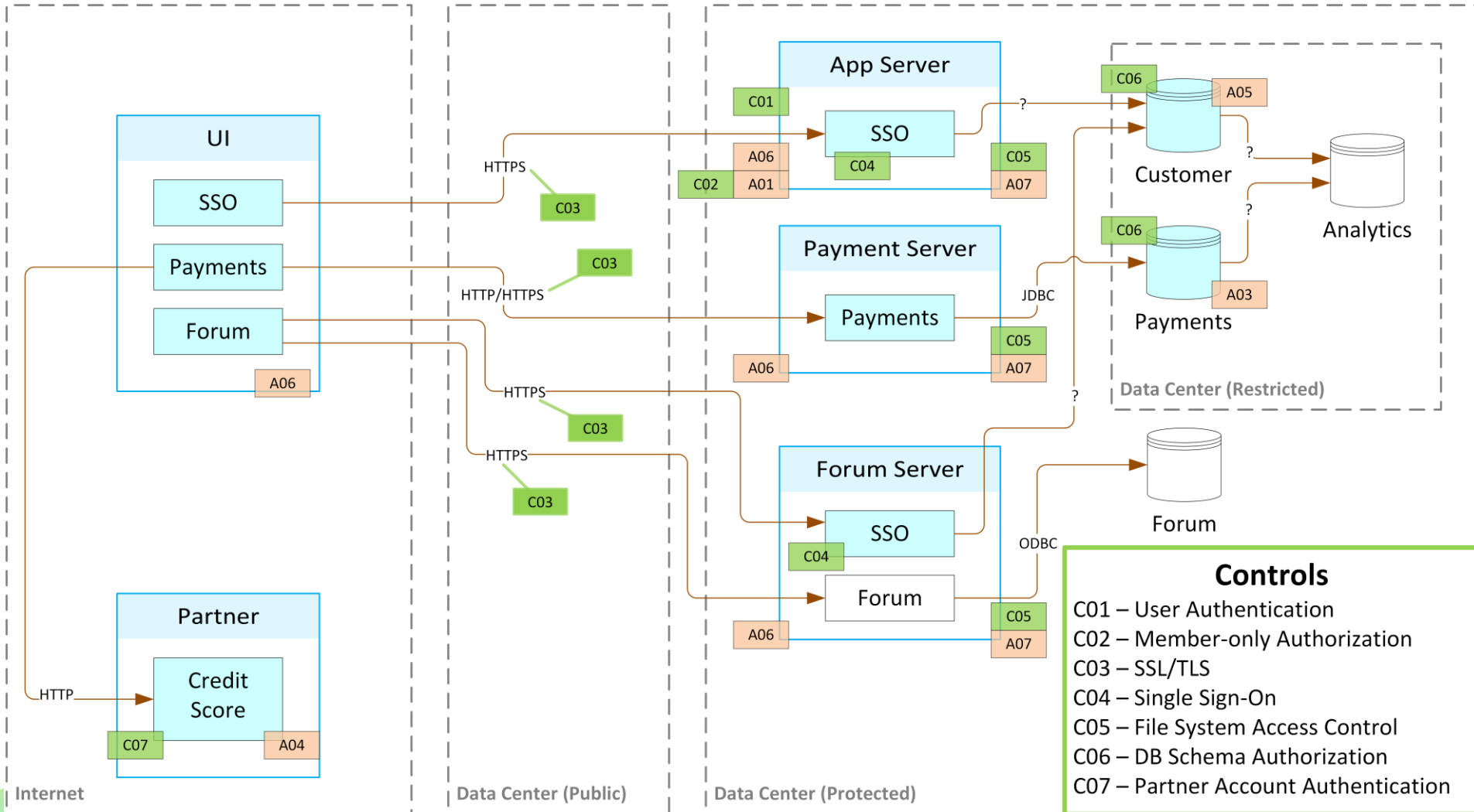
- Social-networking application; accepts payment
- Some content and features membership-only; some, free
- App is JavaEE app; uses WebLogic as JavaEE container
- Backend database is Oracle
 - Stores user's preferences
 - Produces some membership-only reports
- Web UI built using JQuery JavaScript library
- Web UI calls third-party REST services for user-specific content
- User connectivity and interface to backend services uses HTTPS

Identifying Controls from Interviews

Information collected in development interviews:

- Social-networking application; accepts payment
- Some content and features membership-only **[C01][C02]**; some, free
- App is JavaEE app; uses WebLogic as JavaEE container
- Web UI built using JQuery JavaScript library
- Web UI calls third-party REST services for user-specific content
- Backend database is Oracle
 - Stores user's preferences
 - Produces some membership-only reports
- User connectivity and interface to backend services uses HTTPS **[C03]**

Model the Attack Possibilities: Security Controls



Identify Threat Agents

Threat agents are primarily based on access. To identify threat agents:

- Start with the canonical threat agents for the software
- Associate the threat agent with system components they directly interact with
- Minimize the number of threat agents by treating them as equivalence classes
 - For example, assume a technically sophisticated threat agent and a script-kiddie are the same
- Assume that a threat agent can be motivated to attack the system
 - Consider motivation when evaluating likelihood

System TM Canonical Threat Agents

Most internet-based applications can start using canonical set of threat agents:

1. Unauthorized External, Internet-based Attacker
2. Unauthorized Internal/External (client-side), LAN-based Attacker
3. Authorized External, Malicious User
4. Authorized Internal, Malicious App/System Admin

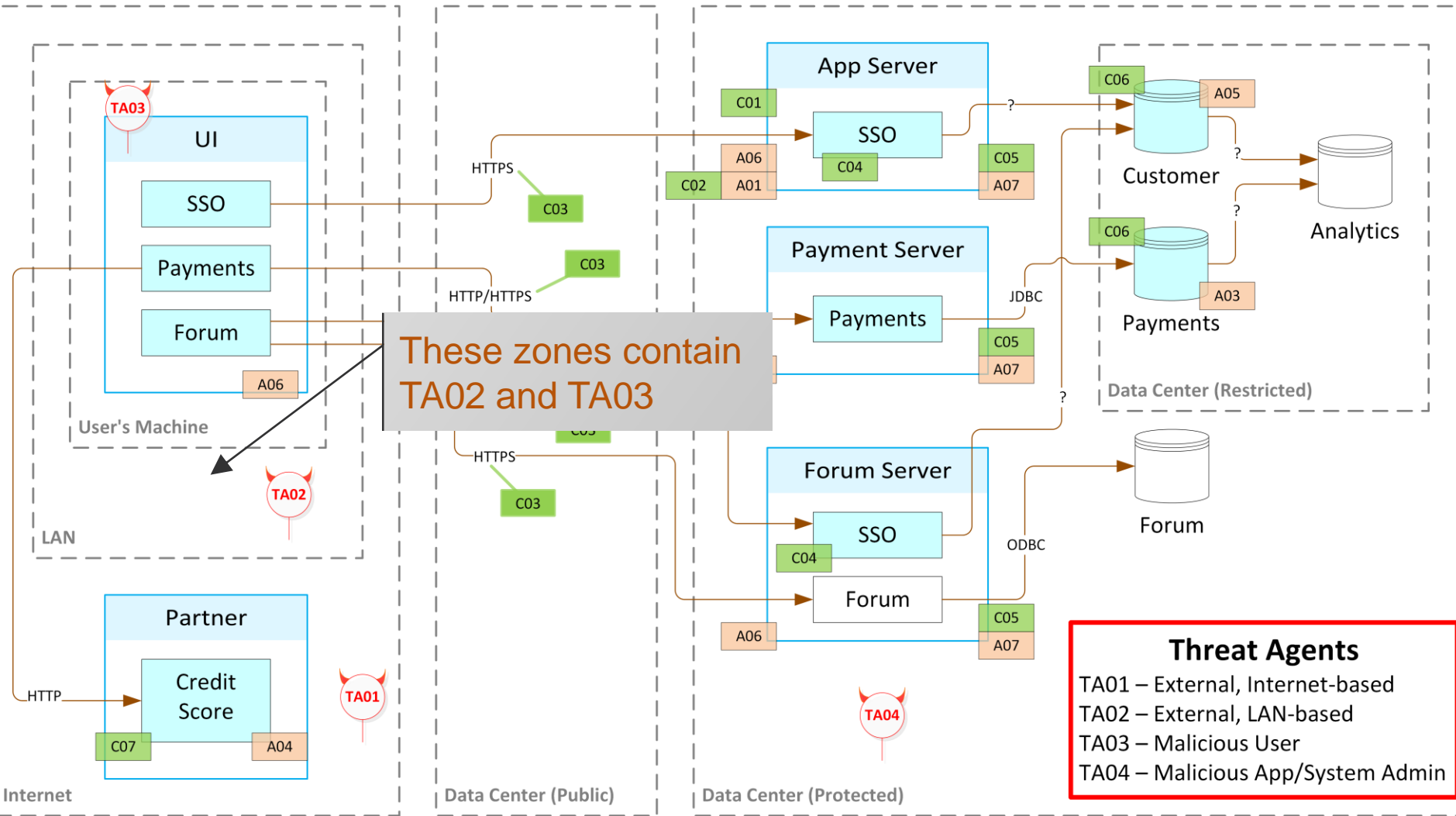
Cloud-hosted applications should account for:

5. Authorized Malicious Cloud Provider Admin

Mobile client applications should account for:

6. Malware on a Jailbroken/Rooted device

Model the Attack Possibilities: Threat Agents



Threat Agents

- TA01 – External, Internet-based
- TA02 – External, LAN-based
- TA03 – Malicious User
- TA04 – Malicious App/System Admin

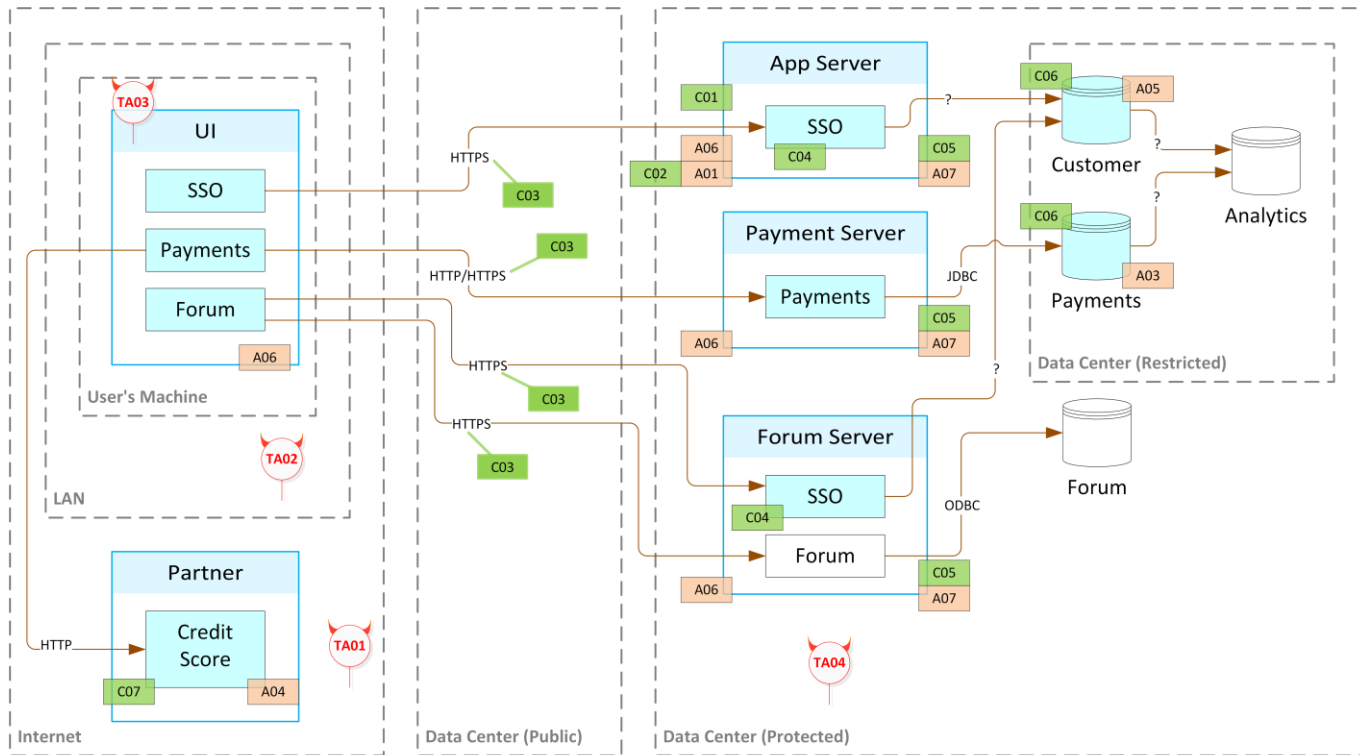
Interpret the Threat Model

To interpret the threat model, start with threat agent and follow flow-of-control paths to reach an asset:

- Is there any path where threat agent can reach asset without going through a control?
- For any security control along each of those paths:
 - What must threat agent do to defeat the control?
 - Can threat agent defeat the control?

Record missing or weak controls in the traceability matrix

Interpret the Threat Model (In-Class)



Assets

A01 – Member-only Content
 A02 – Member-only Features
 A03 – Payment Information
 A04 – Partner Credit API
 A05 – Customer Profiles
 A06 – Session ID
 A07 – DB Access Credentials

Threat Agents

TA01 – External, Internet-based
 TA02 – External, LAN-based
 TA03 – Malicious User
 TA04 – Malicious App/System Admin

Controls

C01 – User Authentication
 C02 – Member-only Authorization
 C03 – SSL/TLS
 C04 – Single Sign-On
 C05 – File System Access Control
 C06 – DB Schema Authorization
 C07 – Partner Account Authentication

System Threat Model Lab



System Threat Model Lab: Objectives

Lab objectives:

- Reinforce what you just learned
- Build a complete threat model with optional diagram for a fictitious system
- Work in independent groups
 - Even with a defined process, people come up with different threat models
 - The models converge over time but is not likely to happen right away

System Threat Model Lab: Model the System

To model the system:

- Receive and review all artifacts
- Review the interview notes made by your colleague
- Create a component diagram
 - OK to "flag" assets, controls, etc. in handouts
 - **Only draw a component diagram now!!**

Duration: 60 minutes (includes 15 min. to review)

System Threat Model Lab: Review System Models

Let's review the system models:

- How different was each group's interpretation of the system?
- What areas were identified where you need to get additional information?

System Threat Model Lab: Add Assets and Threat Agents

Base your work on ONLY the provided system model diagram!

Add attack possibilities to the model:

- Assets
- Threat agents

Duration: 30 minutes (includes 10 min. to review)

System Threat Model Lab: Add Security Controls

Base your work on **ONLY** the provided system model diagram!

Add attack possibilities to the model:

- Security controls
- Controls added should **ONLY** be based on documents received from client

Duration: 45 minutes (includes 20 min. to review)

QUESTIONS?