Using Threat Modeling To Find Design Flaws
Introduction

Jim DelGrosso

• 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
Bugs And Flaws
The Defect Universe

Cross Site Scripting
Buffer Overflow

(Implementation) BUGS

Code Review
Penetration Testing

Weak/Missing/Wrong
Security Control

(Design) FLAWS

Architecture Analysis
Examples Of Bugs And Flaws

Implementation BUGS
- SQL Injection
- XML/XPath/* Injection
- Cross-Site Scripting
- Buffer Overflow
- Unsafe system calls
- Predictable Identifiers
- Hardcoding secrets in code

Design FLAWS
- Misuse of cryptography
- Broad trust between components
- Client-side trust
- Broken or illogical access control (RBAC over tiers)
- Missing defense for replay attacks
- Insecure auditing
Threat Modeling
What Is Threat Modeling?

• A software design analysis capable of finding flaws
Threat Modeling Vocabulary

- Asset
- Security Control
- Threat Agent
- Attack Surface

- Likelihood
- Impact
- Mitigation
- Threat
Threat Model Process

Define scope and depth of analysis

Gain understanding of what is being modeled

Model the threat structure

Interpret the threat model
Different Threat Models

System Threat Model

Protocol/API Threat Model
Description Of Application

- Social networking payment application
- Some content is free and there is membership-only content
- Some features are free and others are membership-only
- The app itself is a J2EE app and uses WebLogic as the J2EE container
- Web UI is built using JQuery JavaScript library
- The backend database is Oracle 11g
  - Database stores user’s preferences
  - Produces some membership-only reports
- This Web UI calls third-party REST services for user-specific content
- User connectivity uses HTTPS and so do interfaces to backend services
Model Diagrams

Layer Model

Logical Model

Deployment Model
Modeling The System Structure

Based on application description and diagrams, create a model that captures:

• The components of the system that are in-scope for this “release”
• How control flows between the in-scope components
• How those components and flows relate to the host boundaries and network zones
• The application layer communication protocols connecting the components

This model can use an existing model diagram or one you create
Components come from the Logical & Layer Models

Protocols come from the Deployment Model

Machine boundaries come from the Deployment Model

Forum is out of scope.

Network zones come from the Deployment Model
Modeling The Threat Structure

Analyze the information we’ve collected and add the threat related elements

<table>
<thead>
<tr>
<th>Assets</th>
<th>The data and functions that the system must protect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Controls</td>
<td>The mechanisms currently designed and implemented to protect the Assets</td>
</tr>
<tr>
<td>Threat Agents</td>
<td>The actors that want to harm the system</td>
</tr>
</tbody>
</table>
Identify **Assets**

- Social networking payment application
- Some content is free and there is membership-only content
- Some features are free and others are membership-only
- The app itself is a J2EE app and uses WebLogic as the J2EE container
- Web UI is built using JQuery JavaScript library
- The backend database is Oracle 11g
  - Database stores user’s preferences
  - Produces some membership-only reports
- This Web UI calls third-party REST services for user-specific content
- User connectivity uses HTTPS and so do interfaces to backend services
Identify Assets

• Social networking payment application
• Some content is free and there is membership-only content [A01]
• Some features are free and others are membership-only [A02]
• The app itself is a J2EE app and uses WebLogic as the J2EE container
• Web UI is built using JQuery JavaScript library
• The backend database [A03] is Oracle 11g
  – Database stores user’s preferences
  – Produces some membership-only reports
• This Web UI calls third-party REST services [A04] for user-specific content
• User connectivity uses HTTPS and so does interface to backend services
Identify Controls

- Social networking payment application
- Some content is free and there is membership-only content
- Some features are free and others are membership-only
- The app itself is a J2EE app and uses WebLogic as the J2EE container
- Web UI is built using Jquery JavaScript library
- The backend database is Oracle 11g
  - Database stores user’s preferences
  - Produces some membership-only reports
- This Web UI calls third-party REST services for user-specific content
- User connectivity uses HTTPS and so do interfaces to backend services
Identify Controls

• Social networking payment application
• Some content is free and there is membership-only [C01] [C02] content
• Some features are free and others are membership-only [C01] [C02]
• The app itself is a J2EE app and uses WebLogic as the J2EE container
• Web UI is built using JQuery JavaScript library
• The backend database is Oracle 11g
  – Database stores user’s preferences
  – Produces some membership-only reports
• This Web UI calls third-party REST services for user-specific content
• User connectivity uses HTTPS [C03] and so does interface to backend services
Model The Threat Structure - Controls

Controls

C01 – User Authentication
C02 – Member-only Authorization
C03 – SSL/TLS
C04 – Single Sign-On
C05 – DB System User
C06 – DB Schema Authorization
C07 – Partner Account Authentication
C08 – File System Access Control
Identify Threat Agents

• Threat Agents are primarily based on access.

• Start with the Canonical Threat Agents for the software.

• Associate the Threat Agent with system components they can directly interact with.

• Minimize the number of Threat Agents, by treating them as equivalence classes. For example, assume a technically sophisticated attacker and a script-kiddie are the same.

• Assume that an attacker can be motivated to attack the system. Consider motivation when evaluating Likelihood.
Canonical Threat Agents

Most Internet-based applications can start with a canonical set of Threat Agents:
• External, Internet-based Attacker
• External (client-side), LAN-based Attacker
• External, Malicious User
• Internal, Malicious App/System Admin

Cloud-hosted applications should account for:
• Malicious cloud provider Admin

Mobile client applications should account for:
• Attacker with a jail-broken/rooted device
Model The Threat Structure – Threat Agents

These zones are part of TA02 and TA03

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

• Additional Threat Agents are business or application specific

• Additional Threat Agents increases the depth of the threat model, but also adds time to the analysis
Reminder – Different Types Of Threat Models

System Threat Model

Protocol/API Threat Model