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

Diagram:
- Threat
- Asset under attack
- Component
- Attack vector
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**
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
You Are Here

Architectural Risk Analysis

- Security Requirements
- Abuse Cases
- Risk Analysis
- External Review
- Risk-Based Security Tests
- Code Review (Tools)
- Risk Analysis
- Penetration Testing
- Security Operations

- Requirements and Use Cases
- Architecture and Design
- Test Plans
- Code
- Tests and Test Results
- Feedback from the Field
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
Elements of a Threat Model

- Structural view
- Behavioral Views
- Threat Actors

- Assets
- Attack Vectors
- Privilege / ‘trust’
The most effective Threat Modeling Tool

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
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:
  - Spoofing
  - Tampering
  - Repudiation
  - Information Disclosure
  - Denial of Service
  - Escalation of Privilege

- Backed by DFDs
An Example DFD

Legend

- **Entity/Role**: [Graphical representation]
- **Data Flow**: [Graphical representation]
- **Process**: [Graphical representation]
- **Data Store**: [Graphical representation]
Attack Trees

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

From Bruce Schneier’s Blog
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)
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.
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
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
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
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
Avoid ‘the stack’

What does this diagram tell you about component interaction?
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
1.2 – Identify Application Attack Surface

[Diagram showing a network architecture with various components such as Web Container, Application Server, Host System, and Database Host. The diagram illustrates the flow of data between different tiers and components, including B2C, B2B, Internet, Cache, Store, LAN, LDAP, Middleware, and DB.]
Exercise: Attack Surfaces in your Mobile App
1.3 - Annotate with design patterns
### 1.4 – Consider Patterns’ responsibilities

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

DOM

Html

Spring

Java

Tomcat

Tomcat

Fix/TCP

Rest

MQ
Exercise: Find responsibilities

DOM

Filtering

Encoding

Authentication

Session Mgmt

Adapter

Java

JSF

Spring

Tomcat

Html

Rest

MQ

FIX/TCP

© 2010 Cigital Inc. All Rights Reserved. Proprietary and Confidential.
Exercise: Find responsibilities

- DOM
- Html
- Ordering
- Dispatch
- JSF
- Spring
- Java
- Tomcat
- Adapter
- Encoding
- Authentication
- Session Mgmt
- Rest
- MQ
- FIX/TCP
1.5 – Identify Frameworks

Showing frameworks indicates where important service contracts exist ‘up’ and ‘down’
1.6 – Identify Controls Explicitly
2 – Identifying Assets
2.1 Identify Critical Data Assets
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
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
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.”
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?
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 programmatic means to verify SSN in the application server
Exercise: What can do we do about it?
What kinds of users do these messages help?
Users often make bad decisions
Exercise

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

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious CSR</td>
<td>Use interface to harvest PII, SSN</td>
<td>Leverage functional design/use case</td>
<td>Mandate disclosure &amp; coverage</td>
<td>Have the CSR enter information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious CSR</td>
<td>Use interface to harvest PII, SSN</td>
<td>Leverage functional design/use case</td>
<td>Mandate disclosure &amp; coverage</td>
<td>Support self-service PW reset</td>
</tr>
</tbody>
</table>
Can you identify the critical structure from 1.6?
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
2.6 – Identify Equivalence-classes

What ‘intermediate objectives’ equate to assets?
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
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
### Motivating Insiders (tabular)

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

<table>
<thead>
<tr>
<th>Who</th>
<th>Capabilities</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rooted Phone User</td>
<td>• Smart phone (Droid, OSX, Palm, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Un-rooted Phone User</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB Admin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>What</td>
<td>How</td>
<td>Impact</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Unauthorized Internet User</td>
<td>Access to the system</td>
<td>Ability to scan, sniff network</td>
<td>Access to binaries</td>
</tr>
<tr>
<td>Authorized Public User</td>
<td>Access to binaries</td>
<td>Able to contribute code to your site (mash-up, open-source, etc.)</td>
<td></td>
</tr>
<tr>
<td>Authorized Partner User</td>
<td>Access</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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
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
3.2 – Identity Principal Resolution

Arrows indicate resolution of principal/assertion propagation
3.3 - Place Threats on Diagram

Threats
1 - Internet-based Attacker (unauthorized)
2 - Internet-based Attacker (authorized user credentials)
3 - Attacker with LAN credentials

System/Net Arch.
- Untrusted
- Hosting LAN
- Application Host
- Database Host
- High-trust (App.)

Threat Attack Vectors
- Social Engineering
- Phishing Attacks (XSRF)
- Man in the Middle (MIM)
- Access 'admin' /face directly
- SQL injection

Diagram showing the placement of threats on a network diagram.
3.4 – Show Authorization in Structure

Coloration shows authorization by role
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
Question:

- Who are we going to consider on our password reset?
  - We discussed Malicious CSR
  - How ‘bout a malicious customer?
  - Cingular
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?
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
4.3 – Think big for a moment...
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
Misuse and Abuse Cases

- Users embed HTML formatting commands to make profile attractive.
- Format commands work, but block directives cause formatting errors.
- Open Text Responses

- Mis-Actor
- Steal Identities
- Answer with JavaScript
- Edit Profile
- Edit Profile Basics
- Edit Demographic Data
- Edit Questionnaire Data
- Upload Photos
Analyzing Abuse Cases

Use Case View

Component View

- ProfileStructuredData
- ProfileQuestionnaire
- HTMLWhiteListFilter
- ProfilePhotos
- ScamSpamMonitor
Software Security Is Not Security Software

Diagram of user roles and security vulnerabilities.
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
## What, Impact

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public, unauthorized, Internet user</td>
<td>Directly request and gain access to another user’s info</td>
<td></td>
<td>PR Incident</td>
<td>Non-compliance</td>
</tr>
<tr>
<td>Public or partner, authorized user</td>
<td>Upload malicious content as part of normal workflow</td>
<td></td>
<td>Increase QSA assessment cost</td>
<td></td>
</tr>
</tbody>
</table>

### 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")
Example:

- Jimmy, Johnny, and Sally

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jealous friend</td>
<td>‘Phish’ credentials</td>
<td>&lt;Various&gt;</td>
<td>Campaign destroyed</td>
<td>???</td>
</tr>
</tbody>
</table>

© 2010 Cigital Inc. All Rights Reserved. Proprietary and Confidential.
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’

- Break encapsulation
- Data leakage
- Callbacks

Architecture Analyses
- Attack Resistance Analysis
- Ambiguity Analysis
- Underlying Framework Weakness Analysis

- Callback
- Privileged threads

- Lure component
  - Manipulate query
  - Access function
  - Impersonation
  - Escalation
ARA’s find ‘Flaw’s

Transactions

Why can’t I just grab ‘A’ and ‘B’?
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
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
2. Find Key Structural Components

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

**THIS is where the:**
- Technical meat is...
- The deep security domain knowledge is...
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:
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);
```
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
Know thy enemy & how they attack you (REDUX)

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

© 2010 Cigital Inc. All Rights Reserved. Proprietary and Confidential.
## Who owns the table?

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public, unauthorized, Internet user</td>
<td>Directly request and gain access to another user’s info</td>
<td>• Forceful browsing</td>
<td>PR Incident</td>
<td>• FD: 3.2: session mgmt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Failure to demand auth</td>
<td>Non-compliance</td>
<td>• SR: 2.3.4: URL, forms data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Session Fixation</td>
<td>Increase QSA assessment cost</td>
<td>• FD: 3.4: Controller design</td>
</tr>
<tr>
<td>Public or partner, authorized user</td>
<td>Upload malicious content as part of normal workflow</td>
<td>• Upload exceptional large file</td>
<td>SLA violation</td>
<td>• SD: 1.3: WebSeal integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use file as injection vector</td>
<td>Data loss/corruption</td>
<td>• SP: 1.3: Demanding Auth.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upload dual-type file (such as GIFAR)</td>
<td>Wholesale system breach</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Business Analyst

- **(Security) Architect**

- **Business Analyst**

© 2010 Cigital Inc. All Rights Reserved. Proprietary and Confidential.
Don’t worry about “left to right”

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>How</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
</table>
| Public, **UNAUTHORIZED**   | Directly request and gain access to another user’s info | • Forceful browsing  
  • Failure to demand auth  
  • Session Fixation  
  • **CSRF**           | **PR Incident**  
  **Non-compliance**  
  Increase QSA assessment cost  
  **Fraud**            | • FD:3.2: session mgmt  
  • SR:2.3.4: URL, forms data  
  • FD: 3.4: Controller design  
  • SD: 1.3: WebSeal integration  
  • SP:1.3: Demanding Auth.  |
| Internet user              |                                           |                              |                                  |                                       |
| Public or partner, **authorized user** | Upload malicious content as part of normal workflow | • Upload exceptional large file  
  • Use file as injection vector  
  • Upload dual-type file (such as GIFAR) | **SLA violation**  
  **Data loss/ corruption**  
  Wholesale system breach | • SP: 9.3: Virus scanning uploads  
  • FD: 6.1: Upload quota  
  • SP: 2.2: Filtering input  
  • SD: 6.3: Re-encoding files  
  • 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**
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:
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
Tips #3: Filling the Gaps...

- How do we design tests to fill this gap?
A security knowledge architecture
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**
Exploit Graphs

- Flowchart of:
  - Delivery
  - Increasing privilege
  - Gaining access
  - Subverting protections
  - Architectural failure
  - Attack Actualization

- Shows effective mitigation
- Compendium Chart

<table>
<thead>
<tr>
<th>Step #</th>
<th>Detail: How/What</th>
<th>Conditions</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Deliver Attack: (get attack code onto machine w/ Asset)</td>
<td>Client must have Internet access</td>
<td>Disable Javascript in browser. <strong>NOTE:</strong> doing so prevents other sites from working.</td>
</tr>
<tr>
<td>1.1</td>
<td>Trick user to point browser to Javascript.</td>
<td>Browser must have “run Javascript” enabled.</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Send victim email containing malicious Javascript.</td>
<td>User’s mail reader must interpret Javascript.</td>
<td>Disable Javascript execution in mail reader.</td>
</tr>
</tbody>
</table>
Steering Committee: Roles with TM Process

- Developers
- Corporate Security
- Security Group
- EA
- Industry Data

Incident Data
New Technology Standards
Policy
Controls
Thank you for your time.