Threat Modeling
A Brief History and Unified Approach at Intuit
About Us

- Tin Zaw
  - Staff Software Engineer
  - Intuit, Inc

- Scott Matsumoto
  - Principal Consultant
  - Cigital, Inc
• Background
  – What Threat Modeling means (to us)
  – What our TM program looks like
• System Threat Modeling
• Lessons and Takeaways
• Protocol Threat Modeling
• Concluding Remarks
• Questions
BACKGROUND
Threat Modeling

Threat Modeling is a software design analysis that looks for security weaknesses by juxtaposing software design views against a set of attackers.

- It identifies secure-design weaknesses
  - Missing security controls
  - Weak or inappropriate security controls
  - Potential vulnerabilities
- Finds weaknesses that cannot be found by other techniques
- It is not a replacement for Pen-Testing or Secure Code Review
• Create a single approach from three disparate approaches in practice:
  – STRIDE
  – Homegrown1
  – Homegrown2
• Develop a common process so one security engineers can share work
• Develop a common process that would allow the business units do the threat models
• Create a shared vision of the security concerns with development early in the SDLC
1. Definition of the process
2. Formal training for security and the business units
3. Introduction of metrics
4. Scaling the new approach through tools and other automation
• Define scope and depth of analysis
• Gain understanding of what is being threat modeled
  – Use existing development artifacts
• Model the threat structure
  – Identify Assets, Security Controls and Attackers
  – Juxtapose threat structure and software model
• Interpret the threat model
  – Produce the list of threats
• Create the threat table for reporting the threats
  – Rank the risk of the threat
  – Propose mitigations
Different Types Of Threat Models

**System Threat Model**
- A holistic view of an application’s security posture
- Best view the security between the application and infrastructure
- Good for building a roadmap for additional security activities

**Protocol Threat Model**
- Analysis of message structure and interaction between components
- Bridges the gap between a System Threat Model and Code Review
SYSTEM THREAT MODELING
Create a Simplified System Model

Components come from the Logical & Layer Models

Machine boundaries come from the Deployment Model

Protocols come from the Deployment Model

Forum is out of scope.

Network zones come from the Deployment Model
We continue to analyze the information we’ve collected in our interviews and now add the threat related elements.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assets</td>
<td>The data and functions that the system must protect</td>
</tr>
<tr>
<td>Security Controls</td>
<td>The mechanisms currently designed and implemented to protect the Assets</td>
</tr>
<tr>
<td>Attacker Profiles</td>
<td>The actors what want to harm the system</td>
</tr>
</tbody>
</table>

Juxtaposing the Threat Structure and the System creates the actual Threat Model. Interpreting the model produces a list of potential threats.
Model The Threat Structure - Assets

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
Model The Threat Structure – Security 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
Model The Threat Structure – Attacker Profiles

Attacker Profiles
AP01 – External, Internet-based
AP02 – External, LAN-based
AP03 – Malicious User
AP04 – Malicious App/System Admin
• Using the model, start with an Attacker and follow the flow-of-control paths in the system to reach an Asset

  – Is there any path where Attacker can reach Asset without going through a Control?

  – For any Security Control along each of those paths:
    • What must the Attacker do to defeat the Control?
    • Can Attacker defeat the Control?

• Record missing or weak controls in the Threat Table
Create the Threat Table

• Collect Threats into the Threat Table.

• Each entry in the threat table:
  – Identifies the threat
  – Calculates the risk based on the attacker profile and the existing controls
  – Proposes mitigations to development to reduce the risk to an acceptable level.
    • Mitigations should be practical and implementable
    • Important to create a “shared vision” with the development team
Lessons and Experiences from the Intuit Journey

• Classroom Observations
• Threat Modeling Adoption Challenges
• Threat Modeling Program Wins
Classroom Observations

• Self-contained nature of the course
  – Come as you are, no need to do any preparation

• In-class group exercises
  – Problem solving by a group of different backgrounds – architects, Ops, testers, coders

• Mindset difference
  – Developers think of Assets first, Security Engineers think of Attackers first
Threat Modeling Adoption Challenges

• The term “Threat Modeling”
  – Foreign language to some, and most have their own interpretation of what it is

• Time commitment
  – Time for entire day of training (esp. when the subject does not seem directly related to their job)
  – Time required to Threat Model

• Getting Development to lead TM exercise
  – So far, TMs get done when Security leads. Need to change
• Instilling the security mindset
  – Not just for architects and developers. Everyone benefits from the training

• Alignment of security goals
  – “You agreed that it was an asset, so you need to protect it.”

• Flaws discovered exclusively by Threat Modeling
  – Design flaws, logic flaws

• Consistent language, diagrams and document format
  – Among product teams and between product and security teams, company wide
PROTOCOL THREAT MODELING
Protocol is an agreed-upon sequence of interactions between two or more components
  – Including data/message formats in these interactions

Why Threat Model a protocol?
  – Analyzing the details of the interactions between components will reveal weaknesses that a System Threat Model will not. For example:
    • Interleaving threats
    • Relay threats
• Two types of canonical attacker profiles
  – Malicious Client – the component action as the client in the interaction*
  – Man-in-the-Middle – any attacker who is able to observe / tamper with the messages in the interaction

• Some protocols may involve more than two entities (e.g. OAuth)
• Role of “client” and “server” may change depending on the interaction.
Model the Protocol - Assets

Assets are the data in the messages

**Assets**
- A01 – User ID
- A02 – Public Key
- A03 – Password
- A04 – Session ID

Client

1. Initialize (UserID)
2. Public Key Pk
3. Authenticate (Encrypt_Pk(Password))

Server

A01
A02
A03
A04

Assets are the data in the messages
Modeling the Protocol - Controls

Assets
A01 – User ID
A02 – Public Key
A03 – Password
A04 – Session ID

Controls
C01 – Encryption
Modeling the Protocol – Attacker Profile

**Assets**
- A01 – User ID
- A02 – Public Key
- A03 – Password
- A04 – Session ID

**Controls**
- C01 – Encryption

**Attacker Profiles**
- AP01 – Malicious Client
- AP02 – Man in the Middle

**Diagram**

Client

1. AP01
2. AP02

Server

- Initialize (UserID)
- Public Key Pk
- Authenticate (Encrypt\_pk(Password))
- SessionID
### Populate the Threat Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>Description</th>
<th>Attacker Profile</th>
<th>Asset</th>
<th>Control</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enumerate list of UserIDs</td>
<td>Call Initialize and interpret the response messages to determine valid UserIDs.</td>
<td>AP01</td>
<td>A01</td>
<td></td>
<td>Change the Initialize call to not take the UserID. Pass both the UserID and password in the same message and provide the same error message for all failures.</td>
</tr>
<tr>
<td>2</td>
<td>Hijack user session</td>
<td>AP02 eavesdrops on interaction 2, captures the SessionKey and hijacks the user’s session.</td>
<td>AP02</td>
<td>A03</td>
<td></td>
<td>Encrypt the session ID using a shared, session-specific key.</td>
</tr>
<tr>
<td>3</td>
<td>Login as UserID by replaying the encrypted password</td>
<td>Eavesdrop on the interaction 2, capture the encrypted password, and then login as UserID replaying the encrypted password.</td>
<td>AP02</td>
<td>A04</td>
<td></td>
<td>Add a randomly generated nonce that’s encrypted as part of the message.</td>
</tr>
<tr>
<td>4</td>
<td>Steal user passwords</td>
<td>Interpose between the client and server passing the client the attacker’s private key.</td>
<td>AP02</td>
<td>A02</td>
<td></td>
<td>Send the public key in a CA-certified X.509 certificate.</td>
</tr>
</tbody>
</table>
CONCLUDING REMARKS
• Next steps at Intuit
  – Develop metrics to measure return on time investment
  – Scale the training for different locations, product teams
  – Have Product teams build the system diagrams

• Threat Model is valuable
  – Different risk levels have different security needs
  – Categorize projects into low, medium or high risk levels
  – There are other ways to secure a project

• Training is the key to success of the program
• Security team has to take the lead
• Protocol Threat Models – a big win
  – Finer grained analysis even without code
  – Great for initialization sequences and any message flow needing crypto
• Threat modeling in partnership with development helps create the shared vision for proper remediation throughout the SDLC
• Build threat models around existing development artifacts rather than re-creating the wheel
QUESTIONS
THANK YOU FOR YOUR TIME