
This lecture provides reference material for the book entitled “The Art of Software Security Testing” by Wysopal et al. © 2007

This lecture material is copyrighted by Laurie Williams (2007). However, you are encouraged to download, forward, copy, print, or distribute it, provided you do so in its entirety (including this notice) and do not sell or otherwise exploit it for commercial purposes.

For PowerPoint version of the slides, contact Laurie Williams at williams@csc.ncsu.edu

Microsoft STRIDE (six) threat categories

- Spoofing identity – pose as another user
- Tampering with data – malicious modification of data
- Repudiation – can the action (prohibited action) be traced?
- Information disclosure – disclose of information to individuals who aren’t supposed to have it
- Denial of service – deny access to valid users (e.g. consume all the CPU time)
- Elevation of privilege – unprivileged user gains privileged access (becomes part of the trusted system)

- Need to learn to THINK LIKE AN ATTACKER!

Example Security Testing Techniques

- **Spoofing Identity**
  - Attempt to force the application to use no authentication; is there an option to allows this, which a non-administrator can use?
  - Can you view a valid user's credentials on the wire or in persistent storage?
  - Can “security tokens” (e.g. a cookie) be replayed to bypass an authentication stage?

- **Tampering with the data**
  - Is it possible to tamper with than rehash the data?
  - Create invalid hashes and digital signatures to verify they are checked correctly.

- **Repudiation**
  - Do conditions exist that prevent logging or auditing?
  - Is it possible to create requests that create incorrect data in an event log?

Example Security Testing Techniques II

- **Information Disclosure**
  - Attempt to access data that can be accessed only by more privileged users.
  - Make the application fail in a way that discloses useful information to an attacker (for example, error messages)
  - Kill the process and then perform disk scavenging, looking for sensitive data written to disk.

- **Denial of Service (Dos)**
  - Flood a process with so much data it stops responding to valid requests.
  - Does malformed data crash the process?

- **Elevation of Privilege**
  - Can you execute data as code
  - Can an elevated process be forced to load a command shell, which in turn will execute with elevated privileges?
**Risk-based security testing**

- “Focus testing on areas where difficulty of attack is least and the impact is highest.”
  - Chris Wysopal [intro to Chapter 4]

---

**Misuse case vs. Threat model**

- **Misuse case:** You’re the bad guy who doesn’t know anything about how the system is developed. You’re using your skill to get in.

- **Threat model:** You are on the (extended) development team and understand who the system was developed. You’re analyzing and evaluating security threats based on an understanding of the application’s design.
Threat modeling

- Security-based analysis that helps people determine the highest level risks posed to the product and how threats manifest themselves
- Determine which threats require mitigation and how to mitigate the threats
- Find complex, multi-step bugs that are not likely to be found in other ways
- Also, understand application better, learn application (new person)
- Also, good for developers that build upon your application
- Also, find bugs in general
- Also, good for developing test cases!! Every threat in the threat model must have a test plan outlining one or more tests.


Threat Modeling: Four Steps

- Identify threat paths
- Identify threats
- Identify vulnerabilities
- Rank/prioritize the vulnerabilities
Step One: Identify Threat Paths

1. Identify and rank the risk of the different user access categories

<table>
<thead>
<tr>
<th>Risk</th>
<th>Access Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very high</td>
<td>Anonymous remote user</td>
</tr>
<tr>
<td>High</td>
<td>Authenticated remote user with file manipulation capability</td>
</tr>
<tr>
<td>Medium</td>
<td>Authenticated remote user</td>
</tr>
<tr>
<td>Low</td>
<td>Local user with execute privileges</td>
</tr>
<tr>
<td>Very low</td>
<td>Administrative local user</td>
</tr>
</tbody>
</table>

2. Develop a data flow diagram . . . .

Data Flow Diagram (DFD): Symbols

- Represents a task that the driver performs.
- Represents an entity that is external to the driver, such as a user, user process, or operating system component.
- Shows the flow of data between components.
- Represents a data store: a file, a device register, a data structure, and so on.
- Represents a boundary between driver code and external entities.

http://www.microsoft.com/whdc/driver/security/threatmodel.mspx
Figure 4-3 IM sample data flow diagram

Figure 4-4 IM sample data flow diagram with trust boundaries
Another example . . .

For each threat path, the next step is to go a level deeper to identify the processing that is performed along the threat path and enumerate the individual threats to that processing. Starting with the highest-risk threat path, the processing performed along the path is analyzed:

- What processing does the component perform?
- How does it determine identity?
- Does it trust data or other components?
- What data does it modify?
- What external connections does it have?
Credentials data cold be malformed, causing processing errors in the Verify Credentials/Authenticate process. Could lead to:
• Corruption of credential data
• Remote execution of code
• Denial of service (DoS) to the Authentication process
• Accepts data from anonymous users and passes to SQL server

Registration data cold be malformed, causing processing errors in the Manage Account process. Could lead to:
• Corruption of user information
• Remote execution of code
• Denial of service (DoS) to the Registration process
• Accepts data from anonymous users and passes to SQL server
Initialization Credentials can be malformed, causing processing errors in the IM Chat process. Could lead to:
- Corruption of chat log
- Remote execution of code
- Denial of service (DoS) to the Chat Engine process
- Malformed data being sent to another IM user

Step Three: Identifying Vulnerabilities

- Now you know the threats . . .
- A threat becomes a vulnerability when the designers fail to build any security features into the application that mitigates the threat.
- Some of the security mitigations to look for are data validation testing, resource monitoring, and access control for critical functions.
- The vulnerability hunt can branch in several directions: detailed security design review, security code review, or security testing.
Step Four: Rank/prioritize the vulnerabilities

- Use the DREAD model to rank a threat’s severity:
  - **Damage potential**: The extent of the damage if a vulnerability is exploited.
  - **Reproducibility**: How often an attempt at exploiting a vulnerability works.
  - **Exploitability**: How much effort is required? Is authentication required?
  - **Affected users**: How widespread could the exploit become?
  - **Discoverability**: The likelihood that the researcher or hacker will find it.

---

Microsoft Security Risk Management

**DREAD**

<table>
<thead>
<tr>
<th>Letter</th>
<th>Risk (1)</th>
<th>Impact (2)</th>
<th>Likelihood (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Damage potential</td>
<td>Damage: The extent of the damage if a vulnerability is exploited.</td>
<td>Likelihood: The likelihood that the researcher or hacker will find it.</td>
</tr>
<tr>
<td>R</td>
<td>Reproducibility</td>
<td>Reproducibility: How often an attempt at exploiting a vulnerability works.</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Exploitability</td>
<td>Exploitability: How much effort is required? Is authentication required?</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Affected users</td>
<td>Affected users: How widespread could the exploit become?</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Discoverability</td>
<td>Discoverability: The likelihood that the researcher or hacker will find it.</td>
<td></td>
</tr>
</tbody>
</table>

Use DREAD to rank your vulnerabilities

http://msdn.microsoft.com/security/securecode/threatmodeling
Other References