

## ***Risk-based Security Testing: Prioritizing Security Testing with Threat Modeling***



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](mailto:williams@csc.ncsu.edu).

NC STATE UNIVERSITY

## ***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!**

NC STATE UNIVERSITY

Howard, M. and Leblanc, D. *Writing Secure Code*, Microsoft Press, 2003.

## Example Security Testing Techniques

- **Spoofing Identity**
  - Attempt to force the application to use no authentication; is there an option to allow 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 and 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?

---

NC STATE UNIVERSITY

Howard, M. and Leblanc, D. *Writing Secure Code*, Microsoft Press, 2003.

## 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?

---

NC STATE UNIVERSITY

Howard, M. and Leblanc, D. *Writing Secure Code*, Microsoft Press, 2003.

## *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 4-1

#### Access Categories

Risk	Access Category
Very high	Anonymous remote user
High	Authenticated remote user with file manipulation capability
Medium	Authenticated remote user
Low	Local user with execute privileges
Very low	Administrative local user

### 2. Develop a data flow diagram . . . .

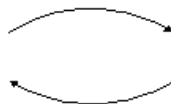
## Data Flow Diagram (DFD): Symbols



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.



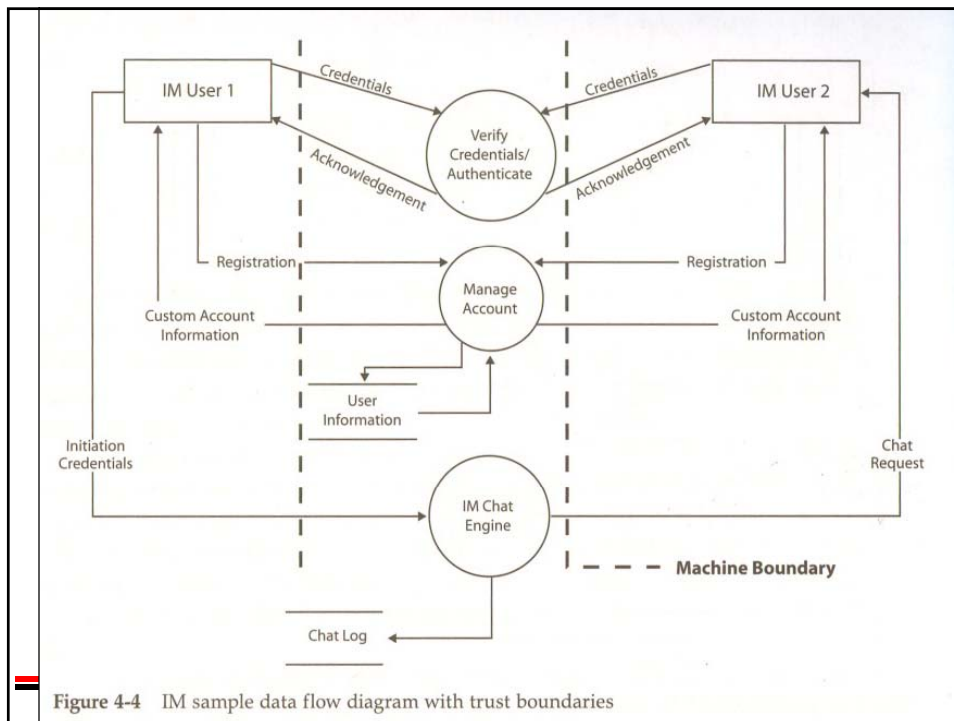
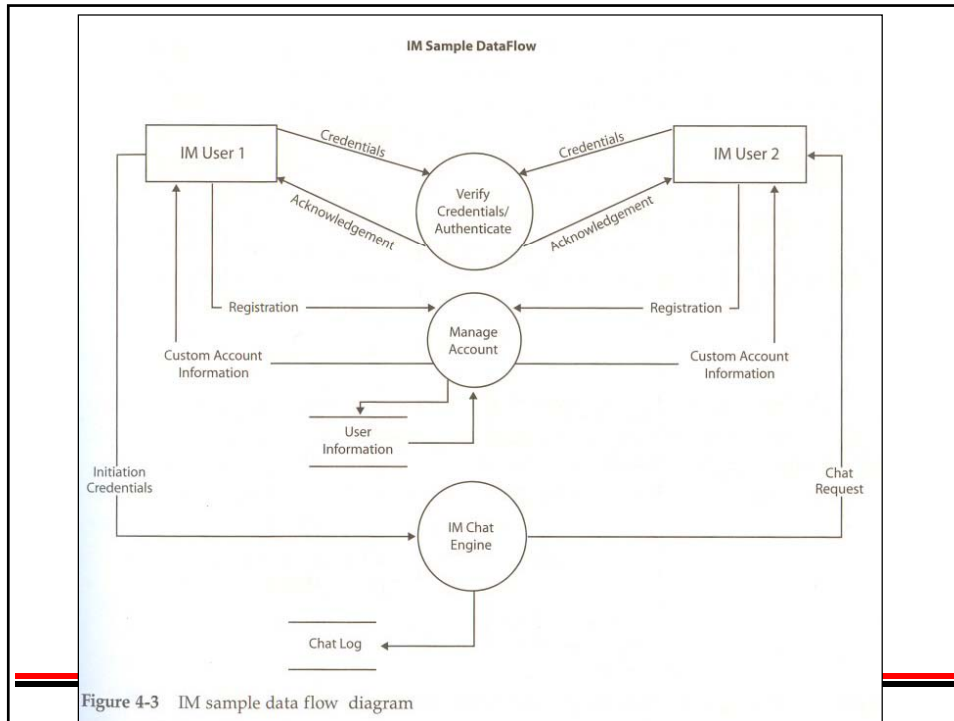
Shows the flow of data between components.



Represents a data store: a file, a device register, a data structure, and so on.



Represents a boundary between driver code and external entities.



## Another example . . .

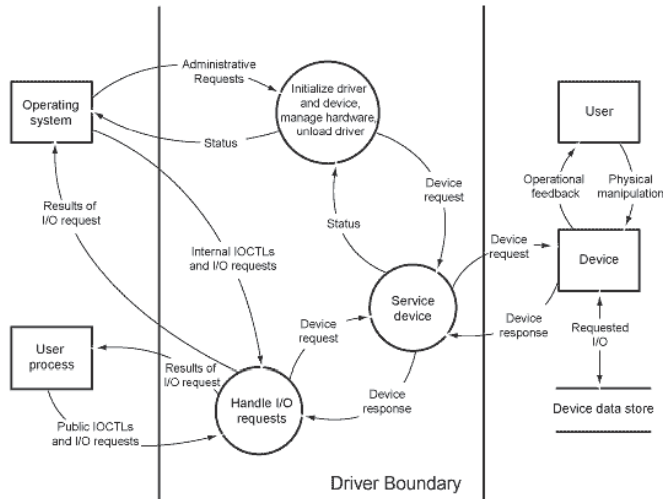
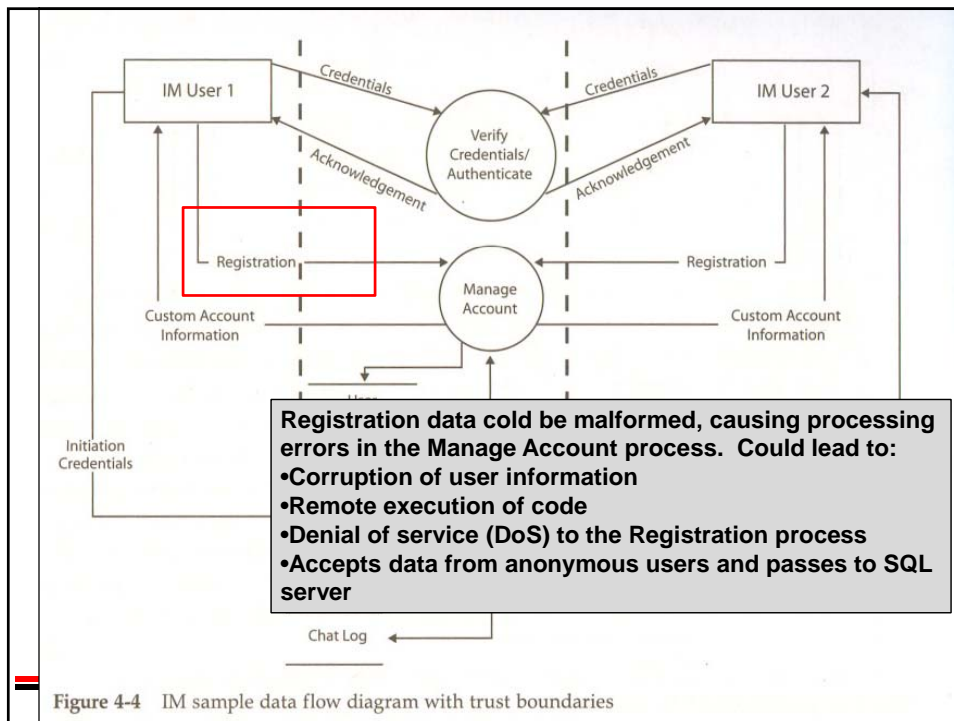
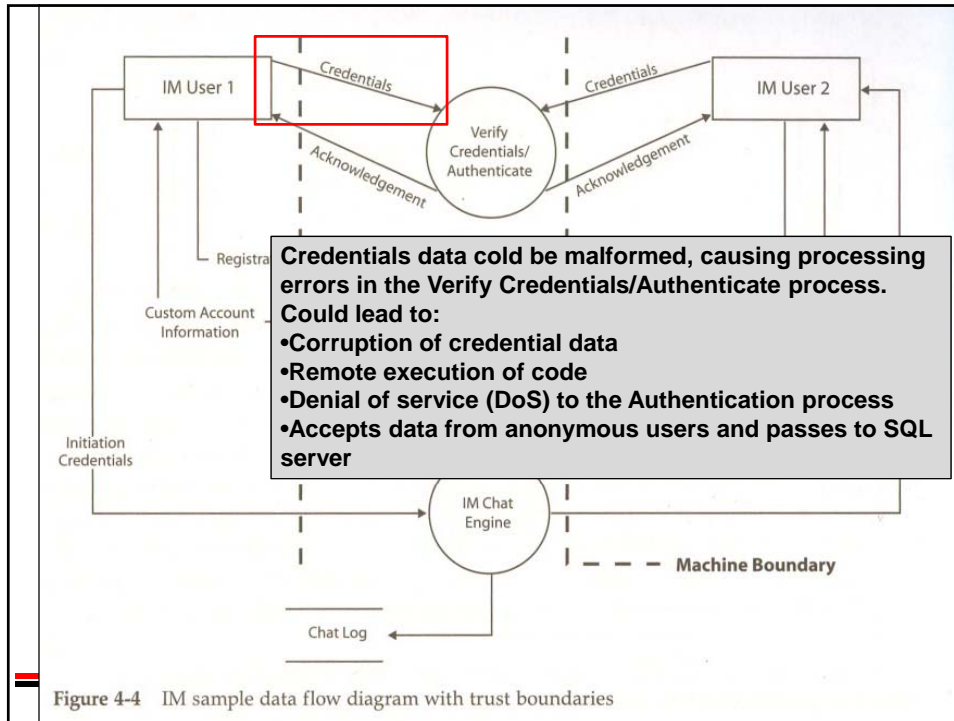
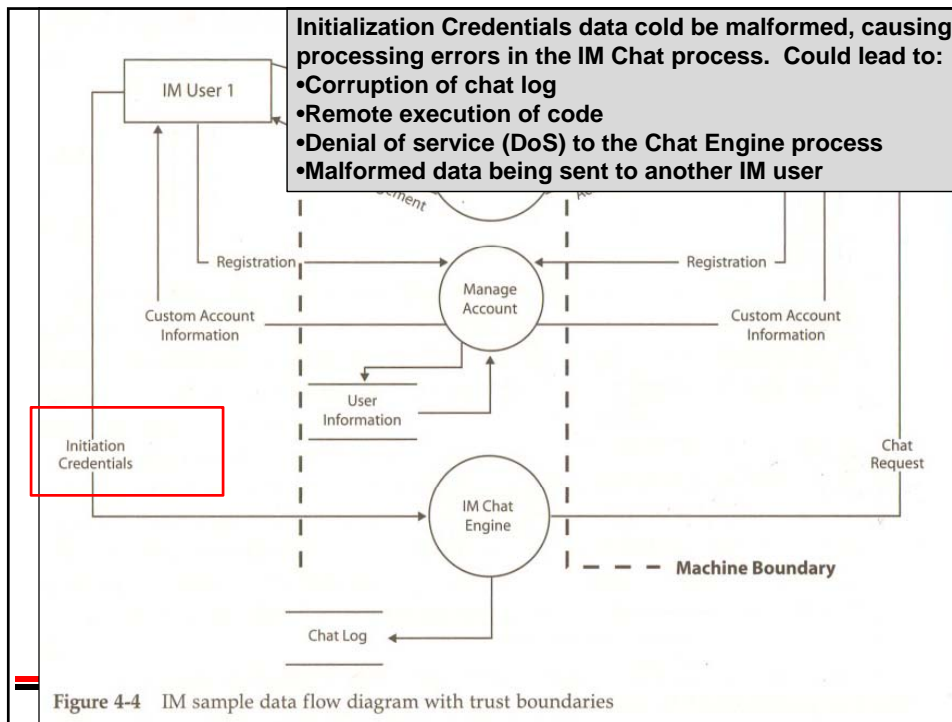


Figure 1. Sample data flow diagram for hypothetical kernel-mode driver.

## Step Two: Identify Threats

- 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?





## 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 becomes?
  - **Discoverability:** The likelihood that the researcher or hacker will find it

## Microsoft Security Risk Management DREAD

Table 3.6 Thread Rating Table

Rating	High (3)	Medium (2)	Low (1)
D Damage potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
R Reproducibility	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
E Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
A Affected users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
D Discoverability	Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.	The bug is obscure, and it is unlikely that users will work out damage potential.

After you ask the above questions, count the values (1-3) for a given threat. The result can fall in the range of 5-15. Then you can treat threats with overall ratings of 12-15 as High risk, 8-11 as Medium risk, and 5-7 as Low risk.

Use DREAD to rank your vulnerabilities

## **Other References**

- **Graff, Mark G. and van Wyk, Kenneth R., *Secure Coding: Principles and Practices*, O'Reilly, 2003.**
- **Howard, M. and LeBlanc, D., *Writing Secure Code*, Microsoft Press, 2003.**
- **Thompson, H., *Why Security Testing is Hard*, IEEE Security and Privacy, July/Aug 2003, pp. 83-86.**
- **Whittaker, J., *How to Break Software Security*, Addison Wesley, 2004.**