



build | integrate | secure

# Threat Modeling

Categorizing the nature and severity of system vulnerabilities

John B. Dickson, CISSP

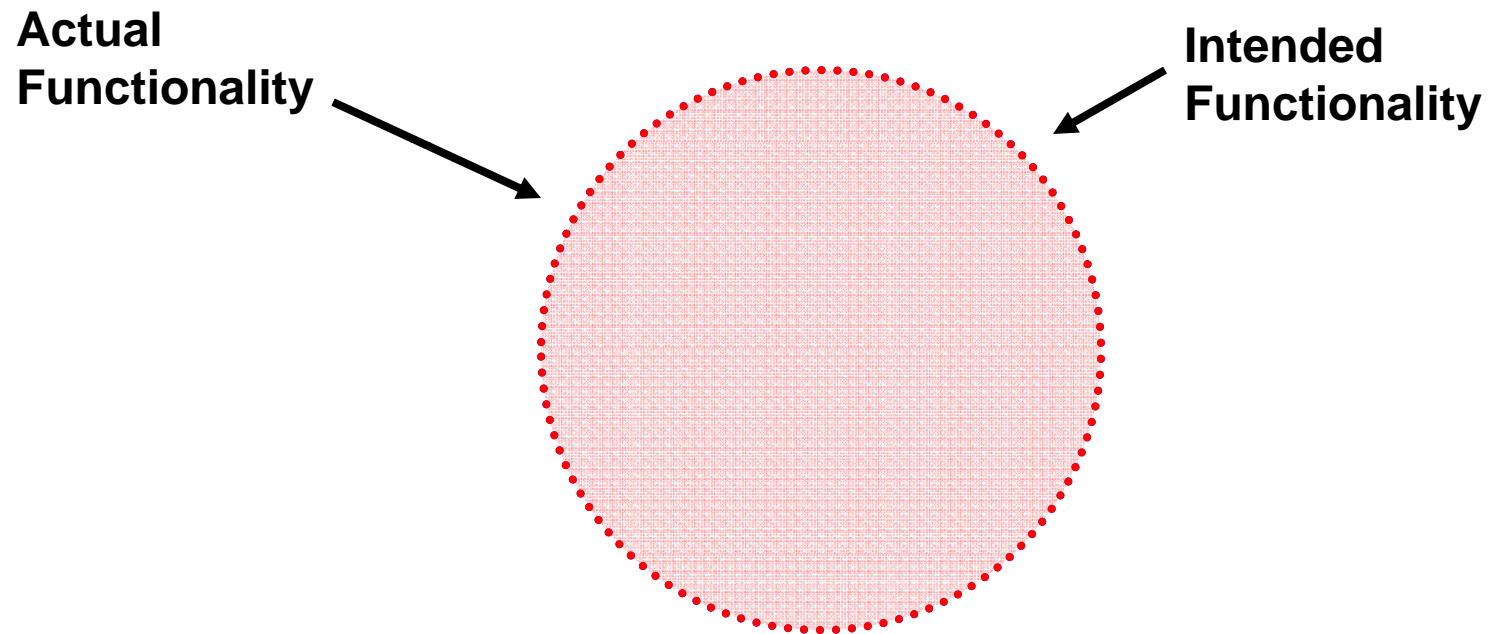
# What is Threat Modeling?

- Structured approach to identifying, quantifying, and addressing threats.
- Threat modeling allows system security staff to communicate the potential damage of security flaws and prioritize remediation efforts.

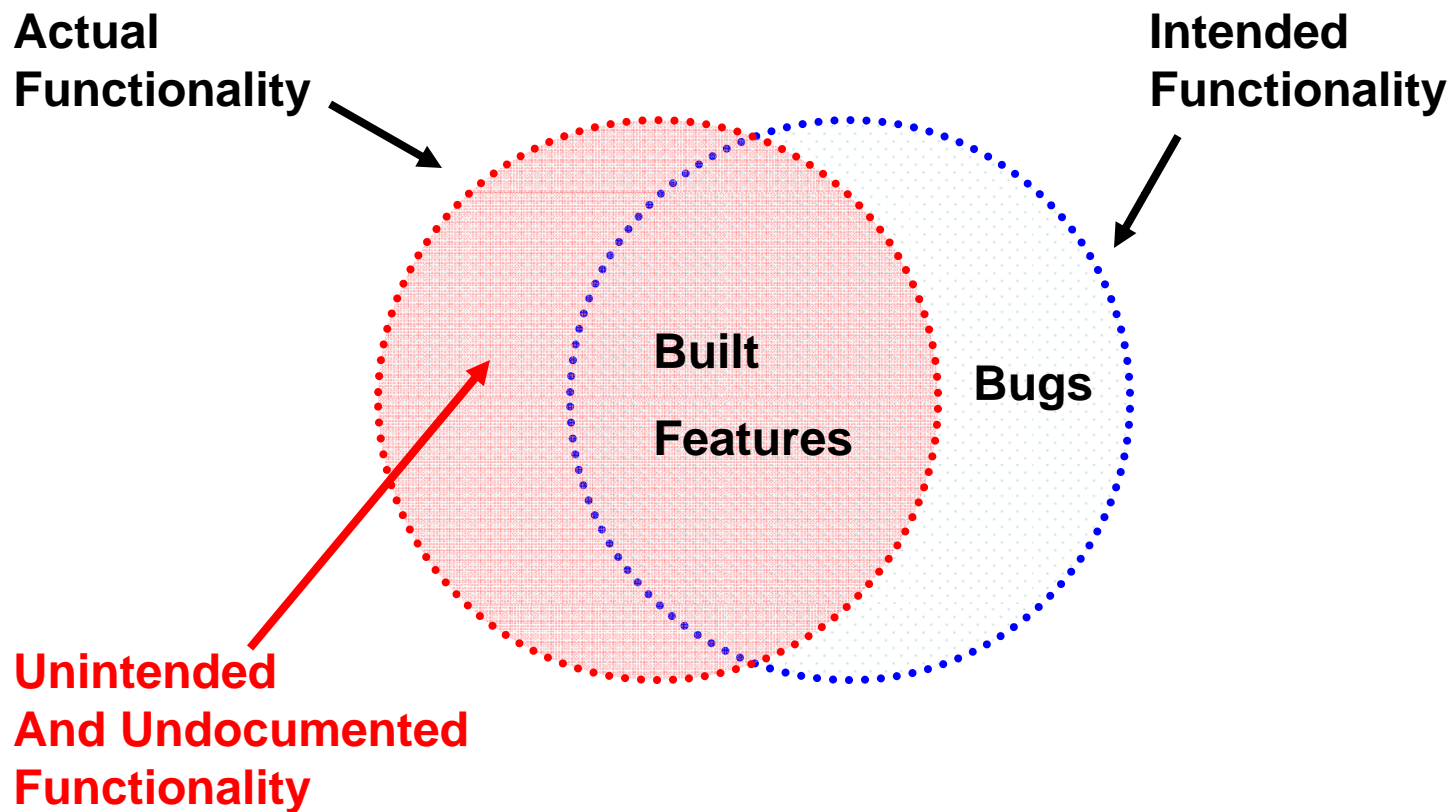
# What Threat Modeling Covers

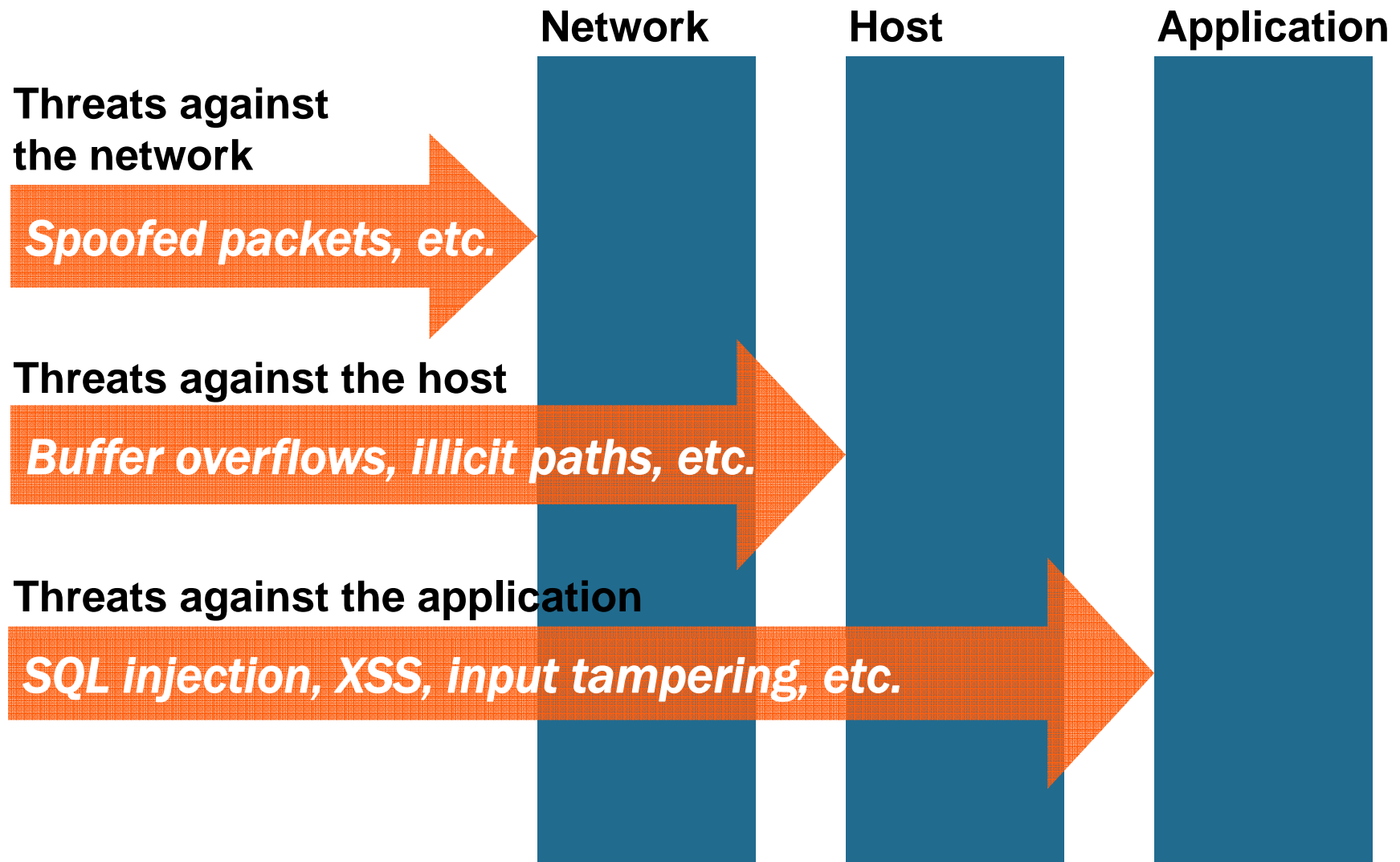
- Assets – What valuable data and equipment should be secured?
- Threats – What may an attacker do to the system?
- Vulnerabilities – What flaws in the system allow an attacker to realize a threat?

## Software Implementation – Perfect World



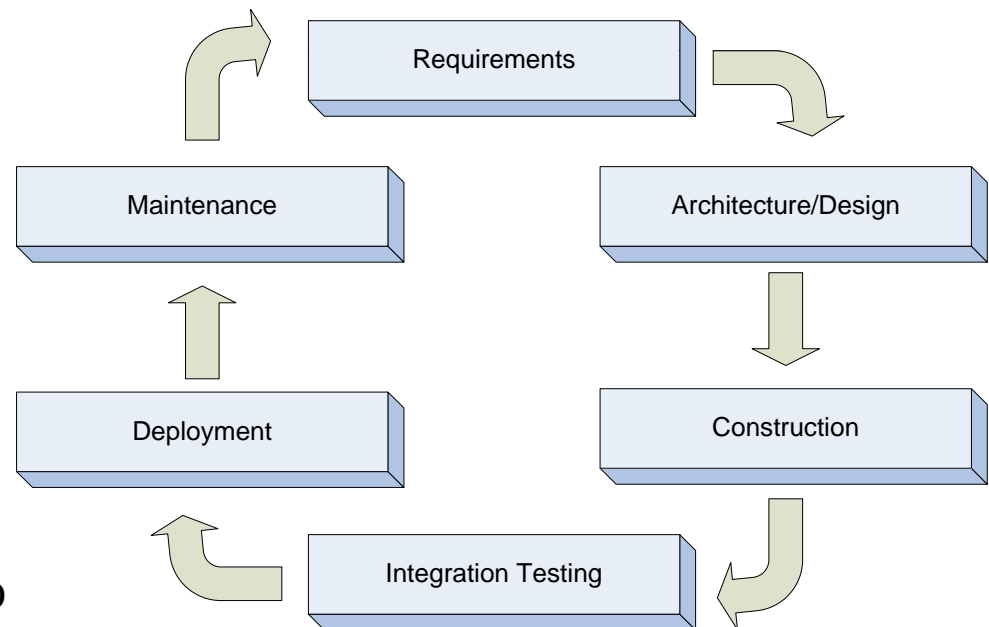
## Software Implementation – Real World



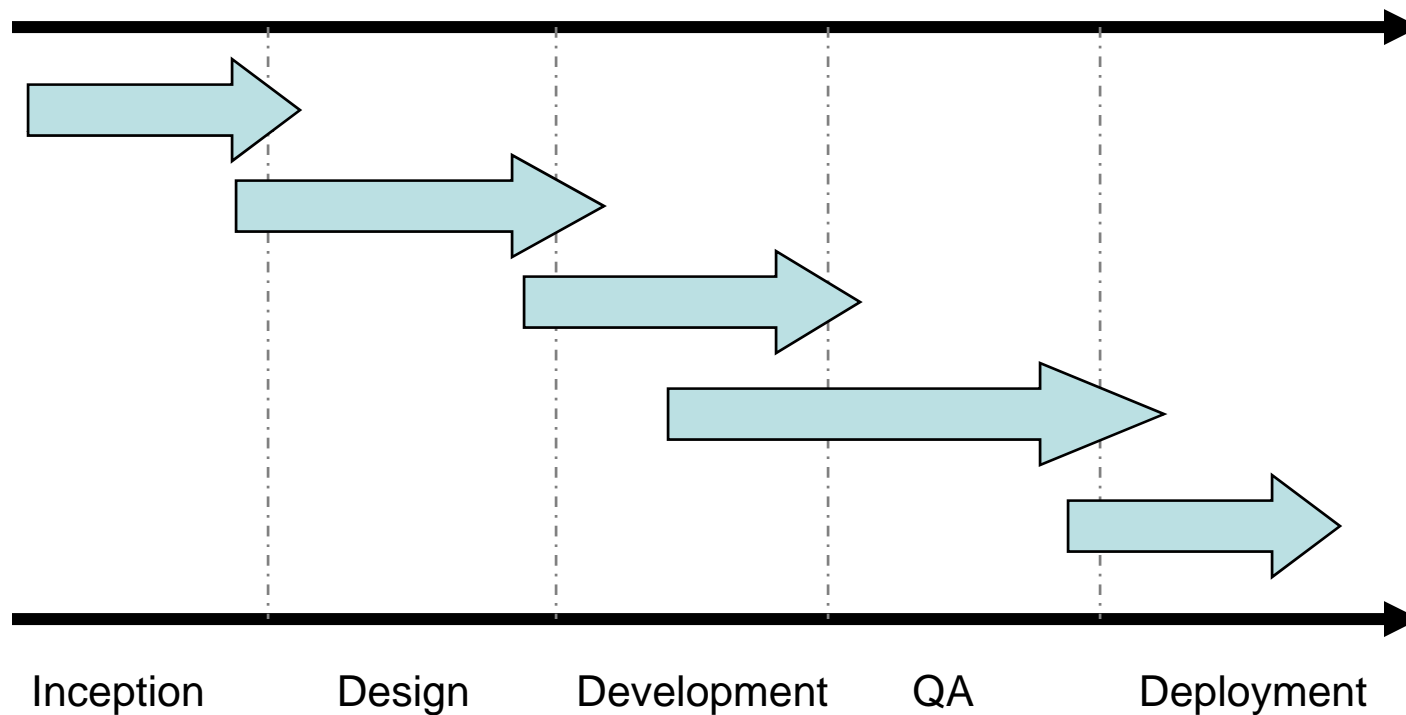


# Who Does Threat Models and When

- Ideally, threat models are created during system design prior to any deployment.
- In practice, threat models are often created for existing systems, making it part of maintenance.
- System designers with security experience are best equipped to identify the threats.



# Security Integration Points within the SDLC



Source: Gartner (February 2006)

# Steps to Threat Modeling

- Identify the Assets
- Describe the Architecture
- Break Down the Applications
- Identify the Threats
- Document and Classify the Threats
- Rate the Threats

# Identify Assets

- Entry and exit points
- System assets and resources
- Trust levels (access categories)

# An Example

TekComCorp has a data collection web application that allows users to login in and enter or change personal data.

The following information was collected on the application:

Architecture:      Web Application - ASP.Net  
                         Database - SQL Server 2000

Assets:              User Login Credentials  
                         User Personal Information  
                         Administrative Resources  
                         System Hardware

# Microsoft Threat Reporting Template

**ID** – Unique ID # of the threat

**Name** – Brief name of the asset threat

**Description** – Detailed description of threat and its importance.

**STRIDE** – How can we classify this threat?

**Mitigated** – Is the application safe from this threat?

**Known Mitigation** – How can we protect against the threat?

**Investigation Notes** – What do we know about this threat so far?

**Entry Points** – What possible means does an adversary have?

**Assets** – What assets could be damaged?

**Threat Tree** – How can we visualize the threat? (Optional)

# Threat Description

<b>ID</b>	1
<b>Name</b>	Login Subversion
<b>Description</b>	An adversary tries to inject SQL commands through a request into the application to circumvent the login process.
<b>STRIDE classification</b>	Tampering with data Elevation of privilege

# Categorizing Threats With STRIDE

**An acronym created by Microsoft to help categorize threats.**

**S**poofing Identity

**T**ampering with Data

**R**epudiation

**I**nformation Disclosure

**D**enial of Service

**E**levation of Privilege

# Threat Description (Cont)

<b>Mitigated?</b>	No
<b>Known mitigation</b>	Stored Procedures Parameterized Queries
<b>Investigation notes</b>	The database calls to in the application were reviewed and string concatenation was used on the login query.
<b>Entry points</b>	(1.1) Login Page
<b>Assets</b>	(1.2) Access to backend database
<b>Threat tree</b>	None

## The Difference Between a Threat and a Vulnerability

- A vulnerability is a valid way for an attacker to utilize a threat.
- A vulnerability is an actual exploit, while a threat is a theoretical exploit.

Example: An adversary gaining privileged access to a sensitive database is a threat. An administrator account “admin” with the password “admin” is a vulnerability.

# Vulnerability Description

<b>ID</b>	1
<b>Name</b>	Login SQL injection
<b>Description</b>	The login query concatenates the username and login parameter to a SELECT statement: “select 1 from logins where username = “ + username + “and password = “ + password
<b>STRIDE classification</b>	Tampering with data Elevation of privilege
<b>DREAD Rating</b>	3
<b>Corresponding threat ID</b>	1
<b>Bug</b>	1

# Rating Threats With DREAD

An acronym created by Microsoft to rate the severity of a threat. Each quality is rated based on a scale developed for each project. On most projects a scale of 1 – 3 is sufficient.

<b>D</b> amage Potential –	How bad can an exploit hurt?
<b>R</b> eproducibility –	How reliably can the flaw be exploited?
<b>E</b> xploitability –	How easy is the flaw to exploit?
<b>A</b> ffected Users –	How many users can be impacted by an exploit?
<b>D</b> iscoverability –	How “visible” is the vulnerability?

The final DREAD rating is the average of all scores.

## Rating Threats cont.

	High (3)	Medium (2)	Low (1)
Damage potential	Attacker can retrieve extremely sensitive data and corrupt or destroy data	Attacker can retrieve sensitive data but do little else	Attacker can only retrieve data that has little or no potential for harm
Reproducibility	Works every time; does not require a timing window or specific extreme cases	Timing-dependent; works only within a time window	Rarely works
Exploitability	Just about anyone could do it	Attacker must be somewhat knowledgeable and skilled	Attacker must be VERY knowledgeable and skilled
Affected users	Most or all users	Some users	Few if any users
Discoverability	Attacker can easily discover the vulnerability	Attacker might discover the vulnerability	Attacker will have to dig to discover the vulnerability

# Threat Description

<b>ID</b>	2
<b>Name</b>	Adversary injects malicious JavaScript or HTML code in the web page.
<b>Description</b>	An adversary tries to inject malicious JavaScript into web pages to capture session information.
<b>STRIDE classification</b>	Tampering with data Information Disclosure

# Threat Description (Cont)

**Mitigated?**

No

**Known mitigation**

HTML Encoding  
Script Filtering

**Investigation notes**

The page allows entry of JavaScript code that will then be executed when an administrator views the record.

**Entry points**

(1.2) Information Update page

**Assets**

(1.2) Access to backend database  
(1.3) User information

**Threat tree**

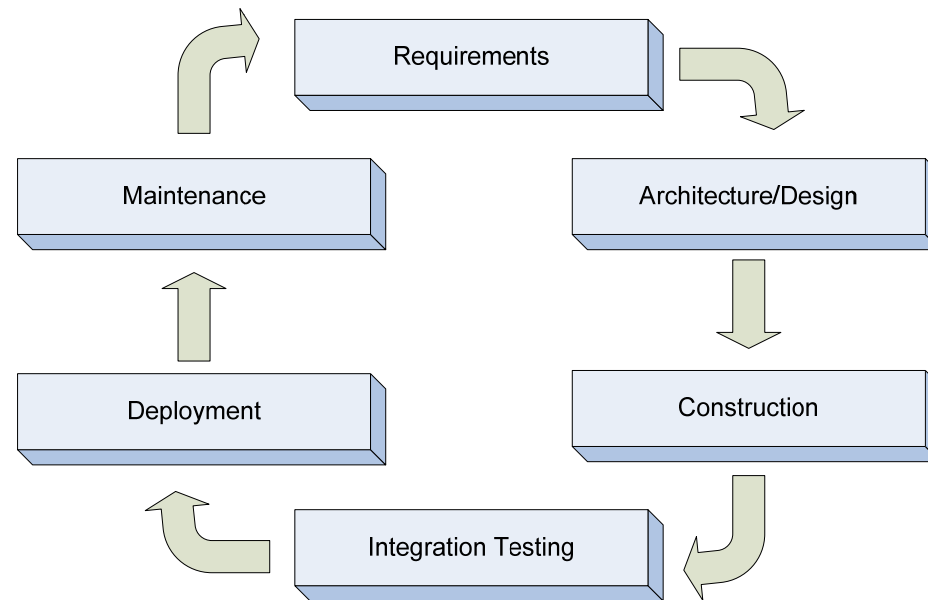
None

# Vulnerability Description

<b>ID</b>	2
<b>Name</b>	Cross Site Scripting (XSS)
<b>Description</b>	JavaScript can be saved into the address field when editing personal information. If an administrator views this page, the JavaScript will run.
<b>STRIDE classification</b>	Tampering Information Disclosure
<b>DREAD Rating</b>	2
<b>Corresponding threat ID</b>	2
<b>Bug</b>	2

# How to Use the Threat Model

Where does it fit in the system development lifecycle?



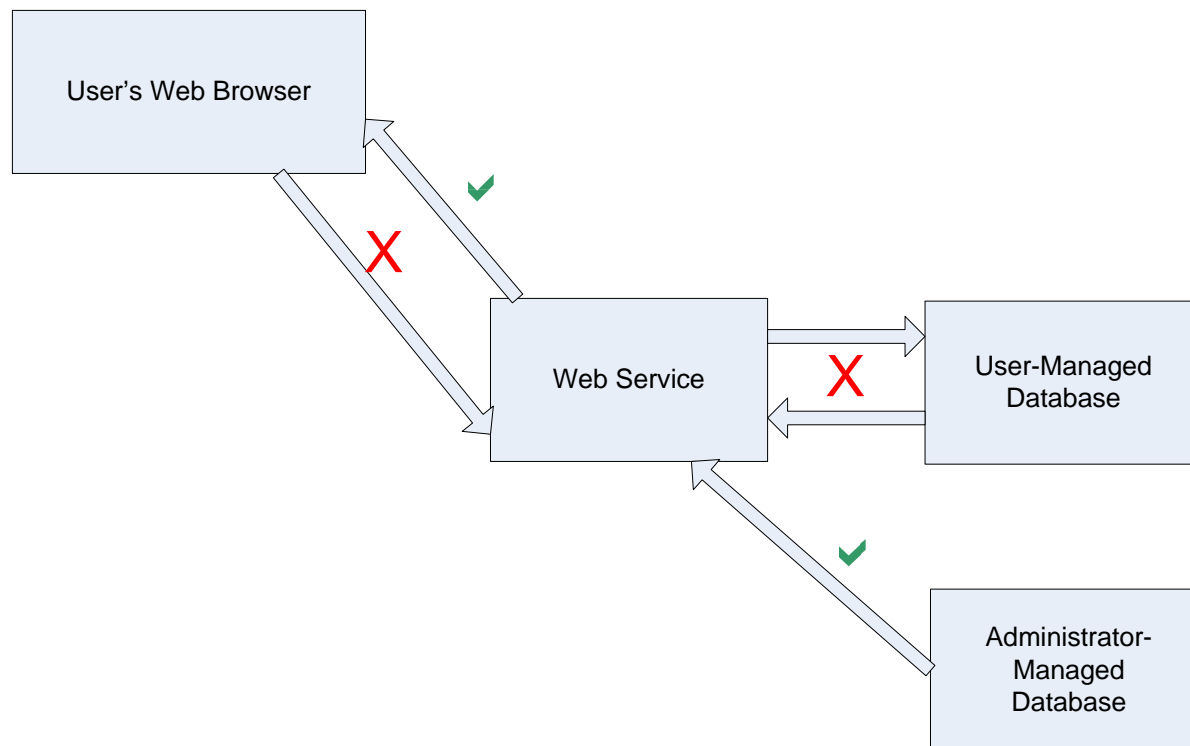
# Requirements

- The requirements specification defines all functionality the application will have.
- The application should never be able to do anything NOT specified in the requirements.

Example: Requirements do not specify the level of authentication required for money transfers from account to account over the web.

# Design

- System assets are derived from the design.
- Identify what data passed among components can and cannot be trusted.



# Construction

- Always be mindful to keep users and assets protected from unreliable data.
- As a general rule, any data passed into a server page or function cannot be trusted not to have SQL or HTML injection.

```
private String EXTRACT_ORDER = "SELECT * FROM dtaOrderInformation WHERE ";
```

```
public static string RemoveOrderItem(string userID, string cartID, string itemID) {
```

```
    ...
```

```
}
```

```
private static string ProcessOrder(string userID, string cartID) {
```

```
    ...
```

```
}
```

# Integration Testing

- Test every identified threat prior to deployment.
- Report vulnerabilities by their impact on defined assets.
- Threat reporting should note strengths as well as weaknesses.

## User Trust Asset: 8 threats, 1 unmitigated

An attacker can lock accounts, requiring administrators to unlock them, but they otherwise cannot attack a user through his or her browser.

Security is relatively strong, but needs attention for next release.

## Order Database Asset: 3 threats, 2 unmitigated

An attacker cannot gain administrative access, but can read any user's information and destroy data.

Alarmingly unsecure. May require delay of deployment.

# Deployment

- Ensure that IT systems do not defeat application safeguards. IT infrastructure should be tested against threats as well.

Example: Preventing command-line injection in the application means little if an adversary has Telnet access to the server.

A sturdy gate does no good if there is a hole in the wall.



# Maintenance

- All system changes should be evaluated for impact on system assets.
- Effected assets should undergo threat tests as part of the normal change integration process.

# Denim Group Contacts

**John B. Dickson, CISSP**

[john@denimgroup.com](mailto:john@denimgroup.com)

3463 Magic Dr, Ste 315

San Antonio, TX 78229-2992

Phone: (210) 572-4402