



build | integrate | secure

# Threat Modeling

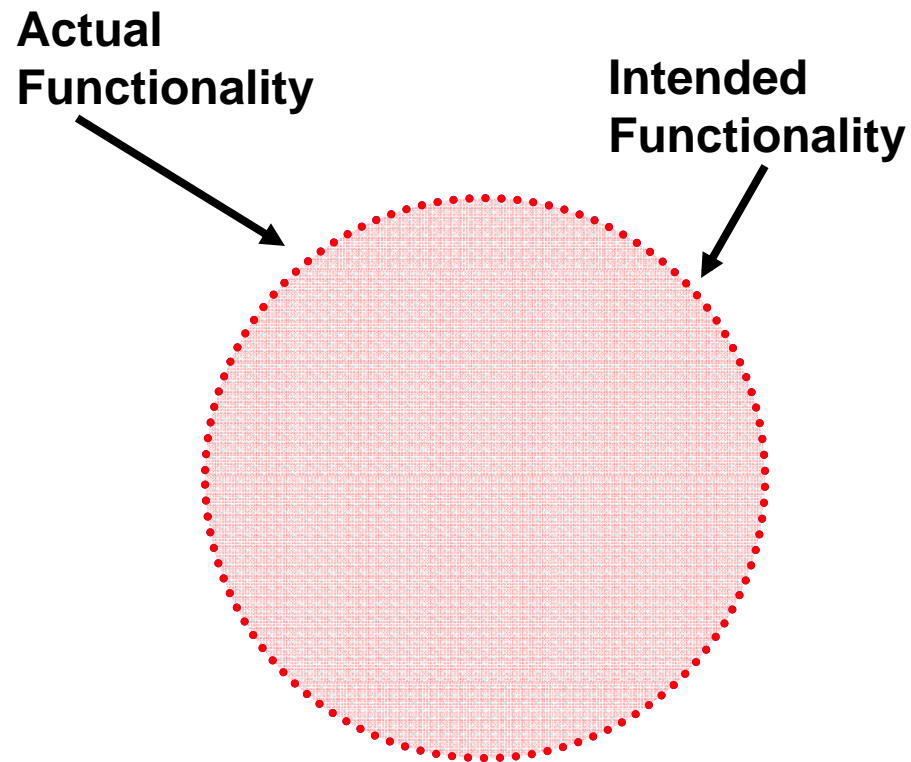
Categorizing the nature and severity of application vulnerabilities.

John Dickson  
Cap Diebel

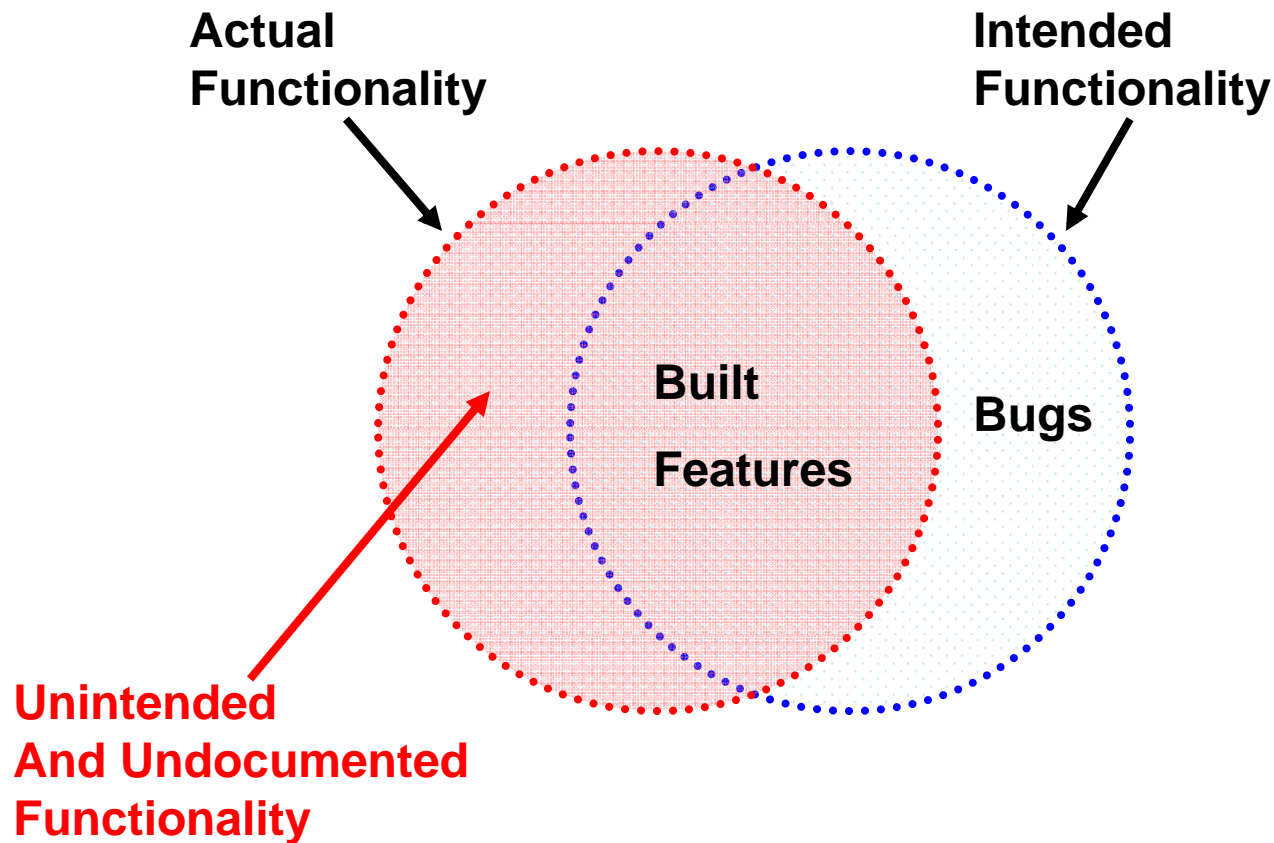
# Security Assessment Dilemmas

- How do we rate the severity of a vulnerability beyond low, medium, and high?
- When we find a large variety of vulnerabilities at once, how do we easily differentiate them from one-another?
- How can we help development teams with remediation?
- How do we ensure vulnerabilities don't reappear when development teams change?

## Software Implementation – Perfect World



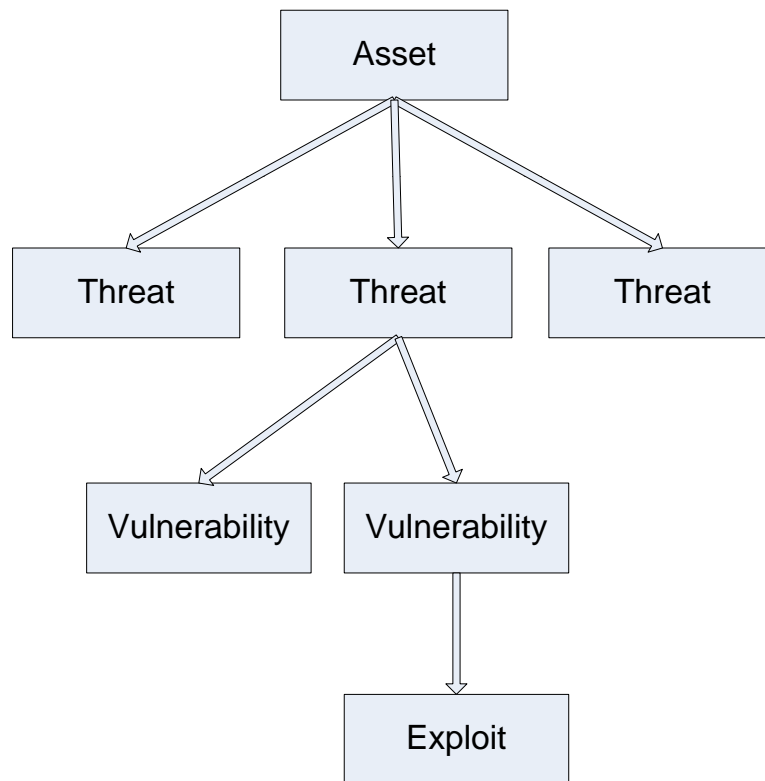
## Software Implementation – Real World



# What is Threat Modeling?

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

# What Threat Modeling Covers

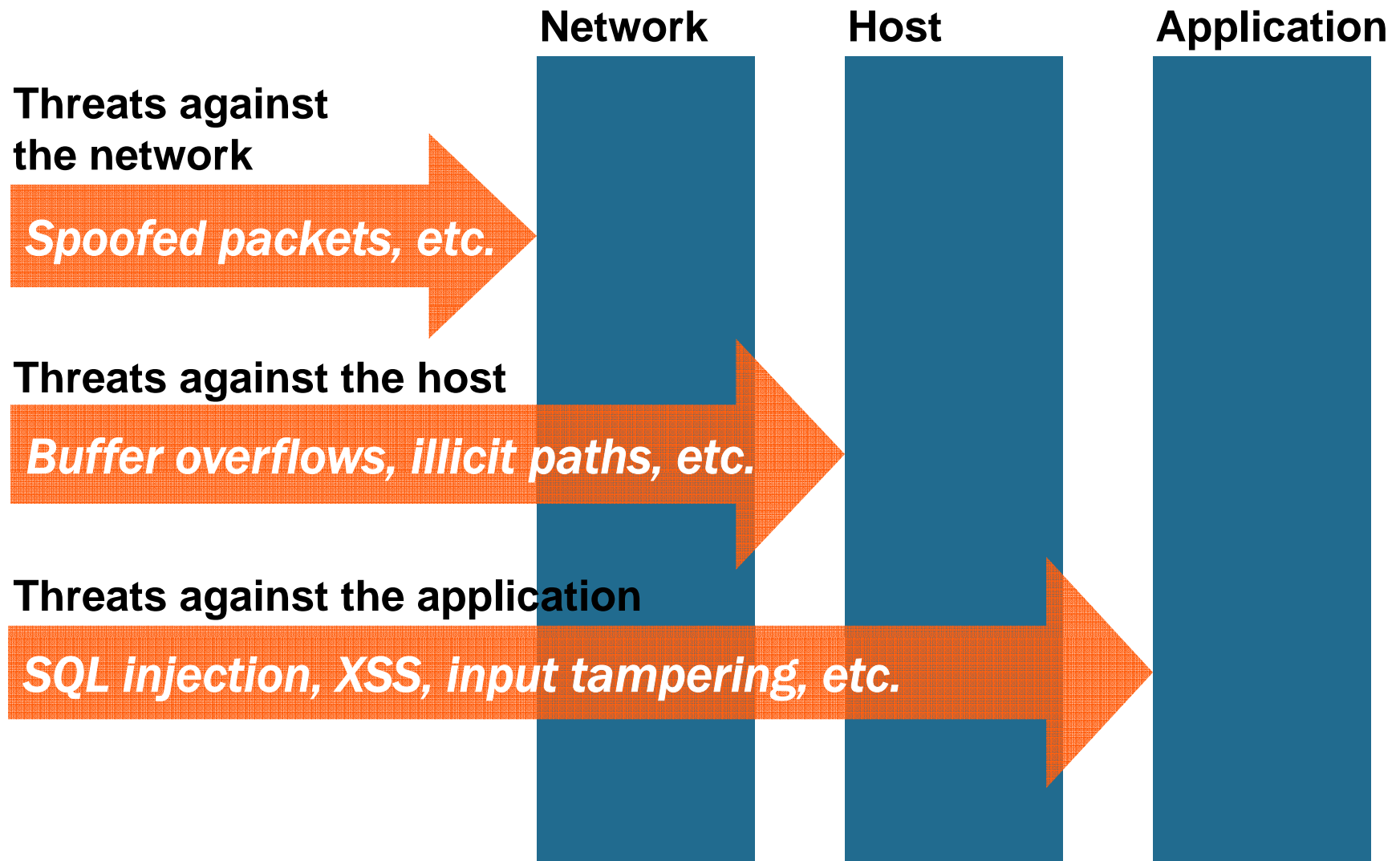


What valuable data and equipment should be secured?  
Example: Database Server

What could an adversary do to damage the asset?  
Example: Trick the application into destroying data.

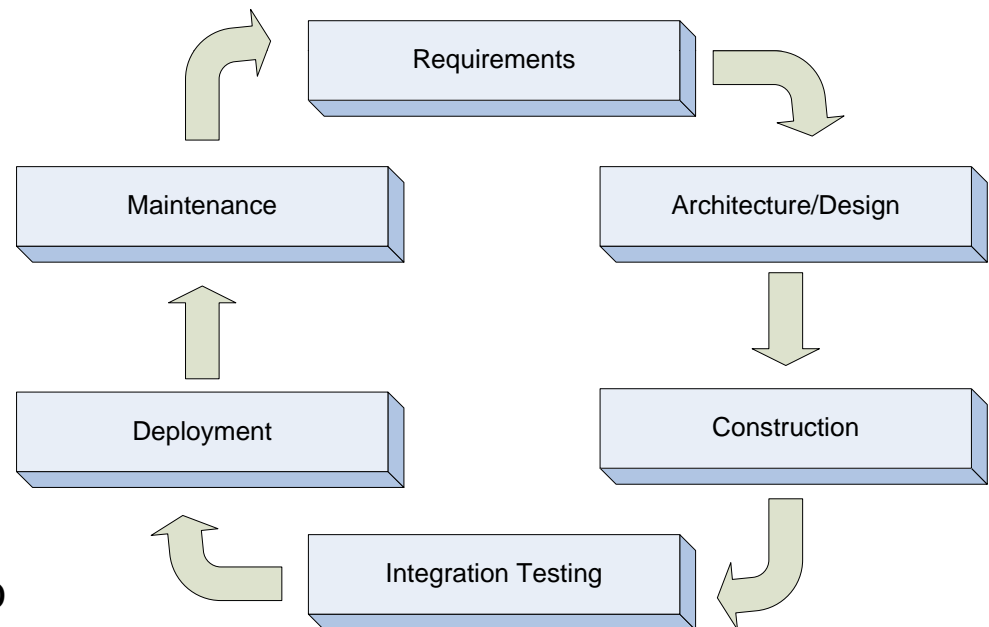
What flaws in the application allow an adversary to realize a threat?  
Example: SQL Injection flaw

What action could an adversary take to exploit a vulnerability?  
Example: Inject malicious SQL code in a request.



# 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 possible threats.



# 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

# 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)

# 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

|                              |   |
|------------------------------|---|
| <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<br>Elevation of privilege   |

# Threat Description (Cont)

|                            |  |
|----------------------------|--|
| <b>Mitigated?</b>          | No   |
| <b>Known mitigation</b>    | Stored Procedures<br>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 allows 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:<br>“select 1 from logins where username = “ + username +<br>“and password = “ + password |
| <b>STRIDE classification</b>   | Tampering with data<br>Elevation of privilege   |
| <b>DREAD Rating</b>            | 3   |
| <b>Corresponding threat ID</b> | 1   |
| <b>Bug</b>                     | 1   |

## Rating Vulnerabilities With DREAD

An acronym created by Microsoft to rate the severity of a vulnerability. 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?           |

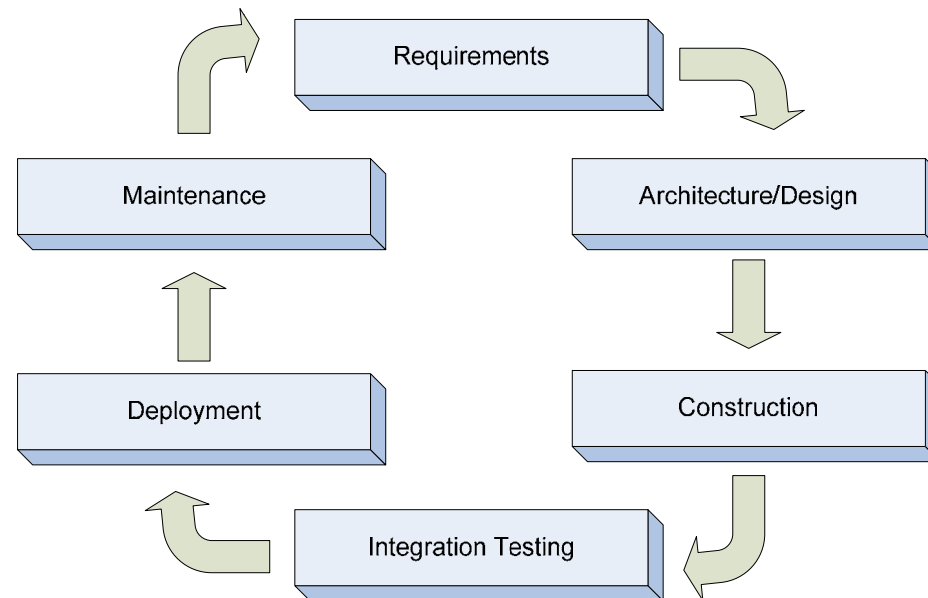
The final DREAD rating is the average of all scores.

## 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                  |

# How to Use the Threat Model

Where does it fit in the software development lifecycle?



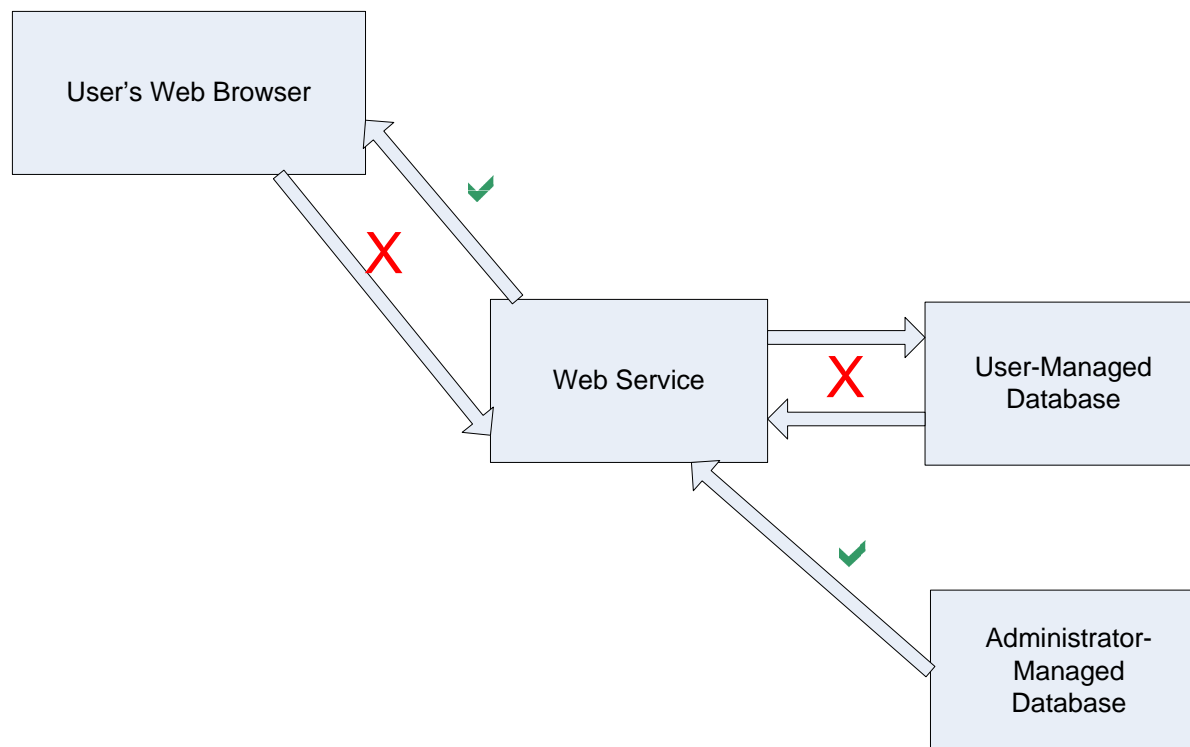
# Requirements

- 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: A programmer builds system administration tools into the same web application accessed by normal users.

# Design

- System assets are derived from the design.
- When listing possible threats, 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 injected SQL or HTML.

```
private static final 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.

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

Preventing the web server from displaying directory listings does not matter if an attacker can reach web-accessible ftp logs.

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



# Maintenance

- Threat models become a part of future application QA efforts.
- 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 Dickson**

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

**Cap Diebel**

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

3463 Magic Dr, Ste 315

San Antonio, TX 78229-2992

Phone: (210) 572-4402