TUTAMANTIC

HOWTO guide for Rapid Threat Model Prototyping

info@tutamantic.com
www.tutamantic.com

TUTAMANTIC

INTRODUCTION

What is Threat Modeling useful for?

Use threat modeling to shape your system design to your security requirements, enabling you to make good decisions on key architectural and engineering elements. Best time is at design-time, but it is also useful on existing systems.

Key Concepts of Threat Modeling

Use threat modeling to incrementally identify what security issues you know, what you do not know, and what you need to know next.

Your business requirements or user stories will scope your efforts and enable you to focus on the intent of the system in question.

A taxonomy & pattern-based approach lets you organize vulnerabilities in a more systematic and repeatable manner.

Benefit - This approach helps you leverage community knowledge and avoid reinventing solutions.

3 security architectural principles

Secure in Design

Use the Defense-In-Depth concept (figure 1) to organize your *Zones Of Trust* on your system processes and datastores.

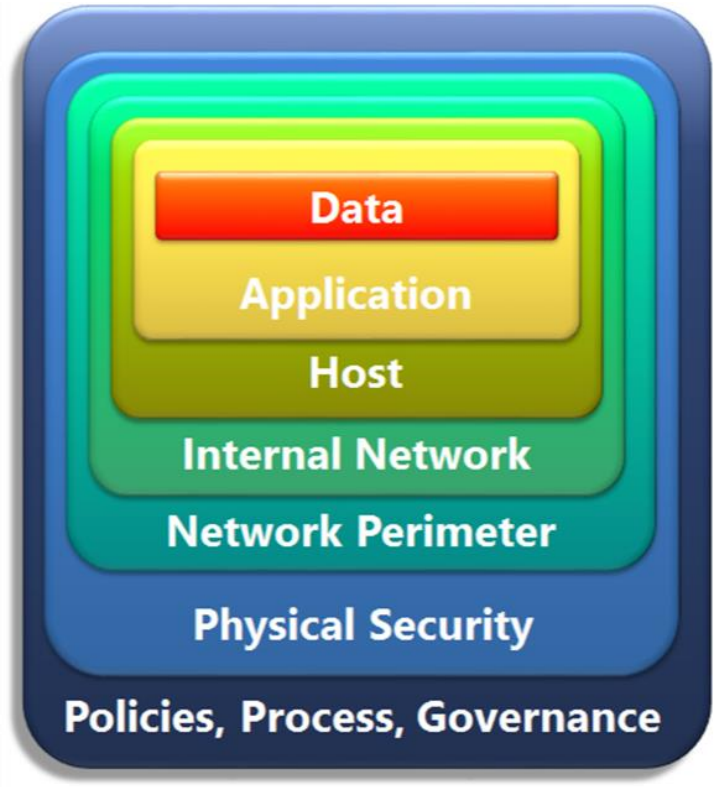


Figure 1

Secure by Default

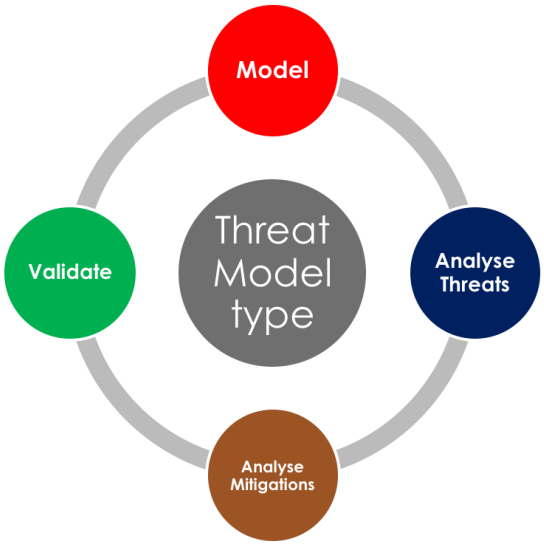
- Require the *Identity* (authentication) for anything that interacts with the design.
- Use the concept of *Least Privilege* (authorization/access control) to define smallest permission sets on these interactors.

Secure in Deployment

Determine the underlying system and dependencies

- Enumerate the end-points being used for access.
- Define the protocols to be used for inter and intra system communication.
- Determine the underlying network topology.

Rapid Threat Model Prototyping step-by-step



The **Rapid Threat Model Prototyping** (RTMP) methodology follows traditional threat modeling steps but is different in some very distinct ways. It uses the Agile architecture framework and software prototyping concepts to provide a system that follows the Pareto rule (80% of threat model outcomes can be done with 20% of the effort of normal threat model analysis) and provides *Just-In-Time* (JIT) analysis.

RTMP is designed to find access control issues the fastest, because *Privilege Escalation is the most dangerous part* of an attack kill chain. Poor access control (authorization) gives rise to Injection attacks, Repudiation attacks, and Information disclosure attacks.

Follow these steps and you will be able to produce threat models consistently and quickly using the least amount of information necessary at each step.

Where to do Threat Modeling in Agile/DevOps

Much of the threat model work should be done during Sprint 0. The development team (including project architects and security SMEs) need to use the Business Requirements and User Stories to model the system.

They then can add the first iteration of security metadata to the model to find initial threats. This is the Agile Architecture *Model Storming* practice. The team will add just enough information to kick off Sprint 1, fulfilling the *JIT* principle for Agile Architecture.

Each Sprint *N*, the team will check with the architect (or designer) to see if the underlying model has changed, which will automatically change the threat model because no metadata will exist for the new version. This is the Agile Architecture *Single-Source-of-Information* practice. The team will add just enough extra security metadata for the threat model to cover that sprint (JIT principle).

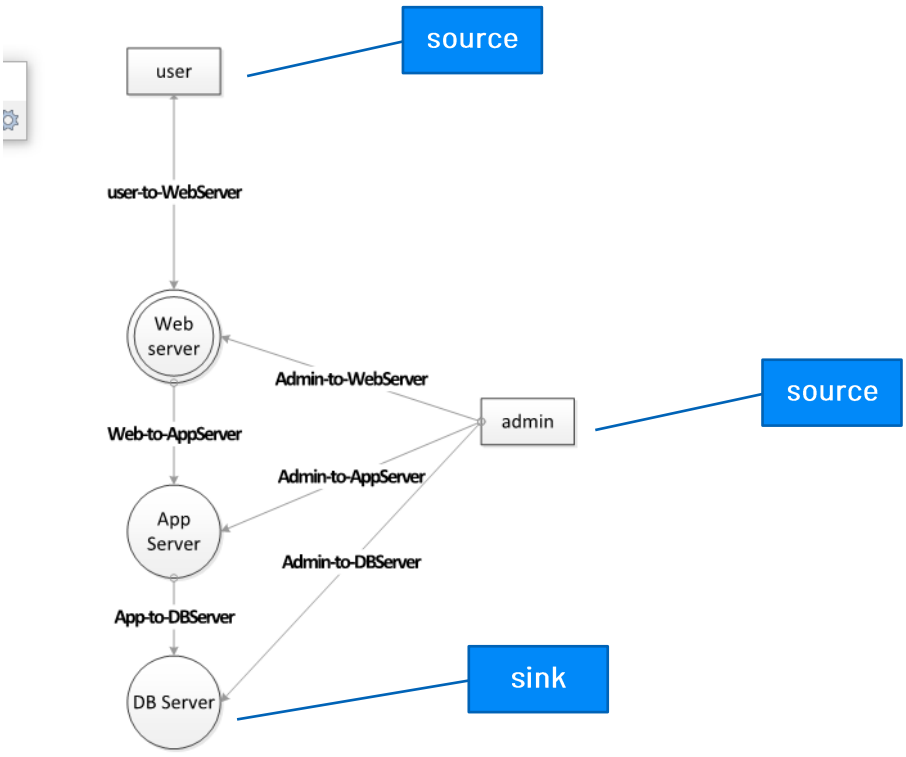
The team will find threat information that they can use to drive the implementation of secure engineering patterns & practices. This combines the Agile Architecture *Test-Driven-Design* and *Look-Ahead-Modeling* practices. A threat taxonomy (CWE, CAPEC) or mnemonic (STRIDE) category should describe the threat information. The model should be able to describe more detail of the threats based on the threat categories and model capabilities. The team should not need to go into detail with the threats.

The development team will discuss and apply one (at minimum) or more specific security mitigations (controls) for each discovered threat category. Some threats may be risk accepted if they don't break business requirements. All active threats are backlogged and prioritized, including risk-accepted threats. The baseline *Team Service Line Agreement* should *always* cover all HIGH threats in-sprint.

First step – Model the system

The best way to model the system is to use an existing diagram of the system. If you have no existing diagram, create a context or process diagram from the Business Requirements or User Stories. Use a whiteboard or a drawing tool (eg. Draw.io, Visio, Lucidcharts, etc.) and plan to capture the security metadata as part of the model.

1. The model should demonstrate the basic flows and processes.

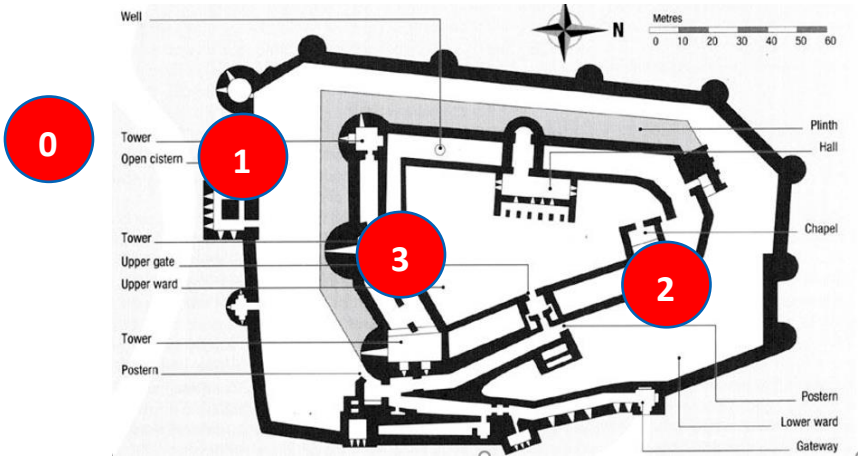


2. Make sure you understand the sources (where the flows start) and the sinks.
3. The sinks generally are where the system's assets "hit the disk" or are stored.
4. The flows should show the direction from the calling process to the called process.
5. The team should put the minimum amount of elements necessary for the model to make it reasonably complete (80% complete).

Designate Zones of Trust

The main concept for RTMP is Zones of Trust. They are similar to Trust Boundaries in classic threat modeling but they provide a stronger underpinning for threat analysis because RTMP has analysis rules that are driven by the Zones of Trust.

Elements in the same Zone of Trust have similar requirements for the protection of information, access permissions, and auditing/logging. Communication is possible within Zones of Trust and between zone layers. Different security issues become apparent for different zone set-ups.



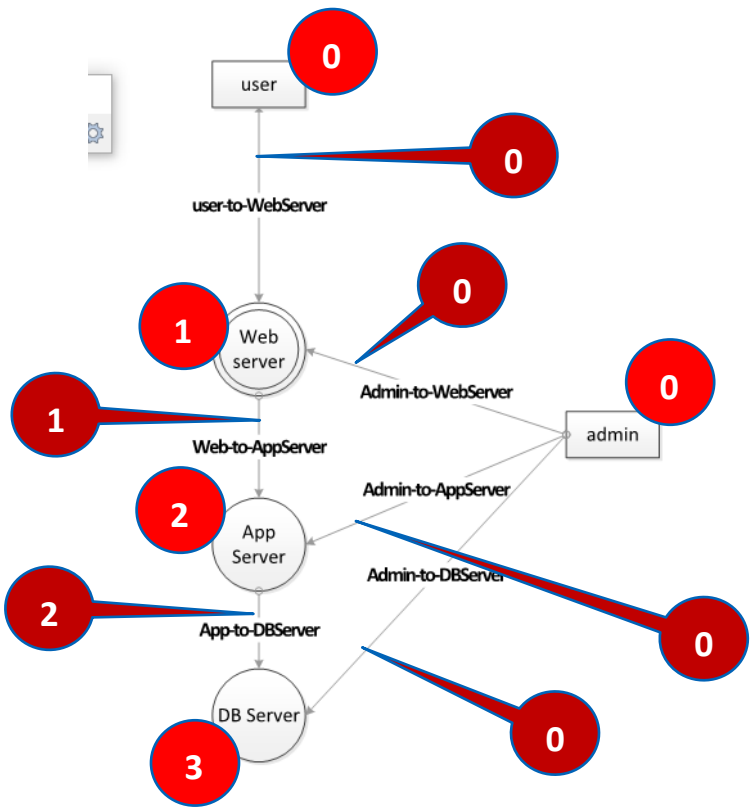
Zones of Trust are numerical rankings of all the elements in the threat model. A higher zone indicates a more critical element in that particular model. The rankings are relative to each model.

Define the processes where access requires additional privileges or role membership. Assign a numerical Zone rank to each process and data/process-flow.

Rules for assigning Zones

1. Zone 0 – designates elements that the modelled system does not control (e.g. anonymous or external users, other systems). These elements are *never* in the system's control.
    a. For example, a modelled system may interact with other *internal* systems and those systems would still be designated Zone 0.
2. Zone 1 - defines a boundary zone which is *designed* to communicate directly with non-controlled elements (e.g. web server in a DMZ).
3. Zones **greater** than 1 – different logical zones of the system, based on higher levels of criticality to the system. Access to these zones could require special access controls, tighter security overview and/or more auditing.
    a. You may find certain circumstances where a Zone 0 element connects directly to an element that is greater than Zone 1.
    b. Connections that are greater than a difference of 1 are more critical.
    c. As a general rule, elements that represent data "hitting the disk" (getting stored in a database for example) should be designated as the highest zones in a model. They are top targets for malicious parties.
4. *Flows* should take the origin *process*'s zone rank.
5. Sources are generally considered to be outside the direct control of the modeled system and should be the *lowest* zone of the model.
6. Sinks are defined as the end of a data/process flow and should be the *highest* zone of the model.

The following illustration shows Zones of Trust applied to a sample model :



Second step – (Threats) RTMP default analysis rules

RTMP uses 6 rules based on the Zones of Trust to create default threat values for the model. The STRIDE mnemonic is used as a reference list for creating these default values. STRIDE is easy to understand and covers a huge range of possible threat scenarios. It also follows the Pareto rule by allowing the user to add just the STRIDE designation while the model provides the context for the listed threat.

STRIDE Framework* for finding threats

Threat	Property we want
Spoofing	Authentication
Tampering	Integrity
Repudiation	**AUTHENTICATION + INTEGRITY**
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

* Framework, not classification scheme. STRIDE is a good framework, bad taxonomy

STRIDE contains data-oriented and transactional elements. Data-oriented security is covered by the "CIA Triad" and outlines 3 basic principles that secure systems should do. These ensure the Confidentiality, Integrity, and Availability of data or communications. The STRIDE threats of Information Disclosure, Tampering and Denial of Service map to the CIA Triad.

Information Disclosure threats exist when malicious parties can get "too much information" because the system is unable to enforce confidential states.

Tampering threats happen when the system is subject to attacks that alter data or communications, thus invalidating the integrity of the data.

Denial of Service is a security threat that maliciously reduces Quality-of-Service so it is no longer performant, predictable or available.

Transactional security covers whether an interactor to the system has identity and is authorized to fulfill the intended communications. Once the communications take place, a secure system should prevent any interactors from denying (aka repudiating) those communications.

Spoofing threats exist because of poor or nonexistent identity verification and authentication. The system is unable to verify the authenticity of the element interacting with it.

Elevation of Privilege is the *most* dangerous security threat. Poor or nonexistent access control (aka authorization) will give a malicious party the opportunity to do any of the other attacks that STRIDE represents. RTMP looks for Elevation of Privilege threats first.

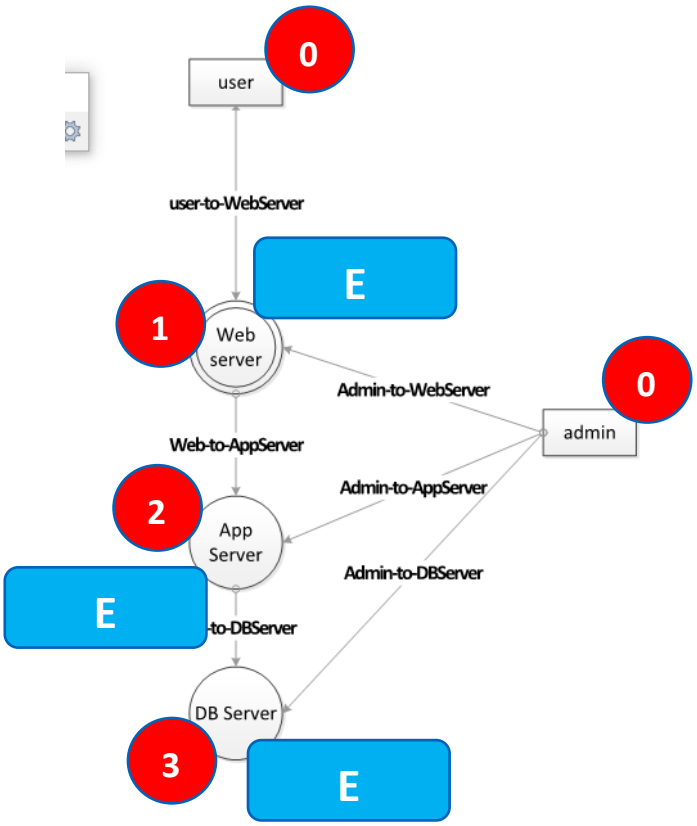
Repudiation threats occur when data integrity cannot be proven and when authentication linked to that data cannot be verified. A malicious party could therefore deny association due to lack of proof.

More information on STRIDE can be found here:
https://en.wikipedia.org/wiki/STRIDE_(security)

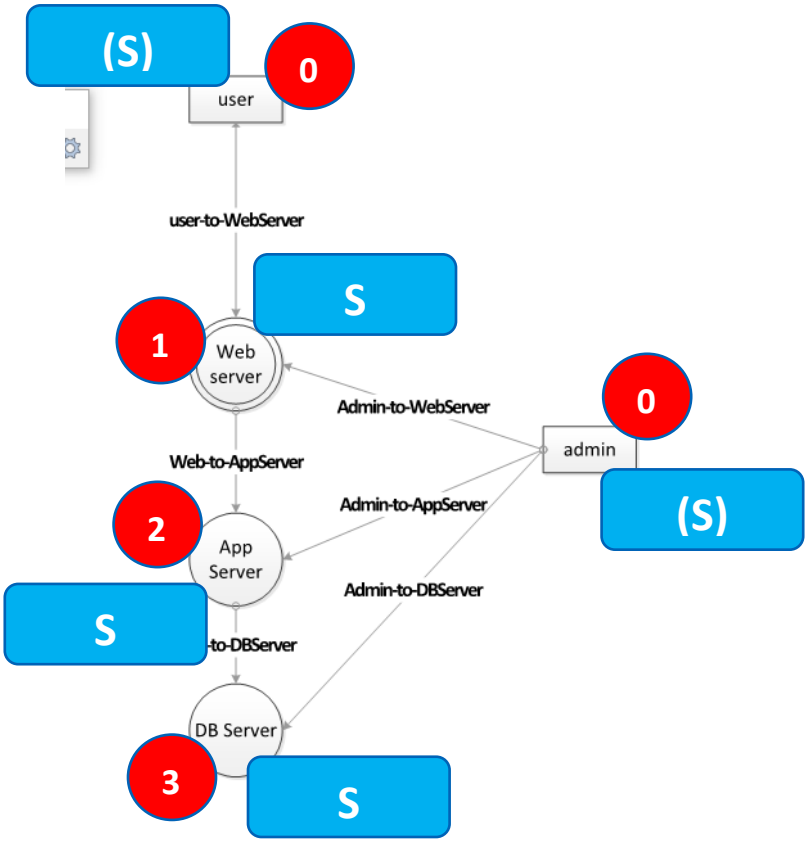
Rules for assigning default STRIDE values

[Do these rules in order]

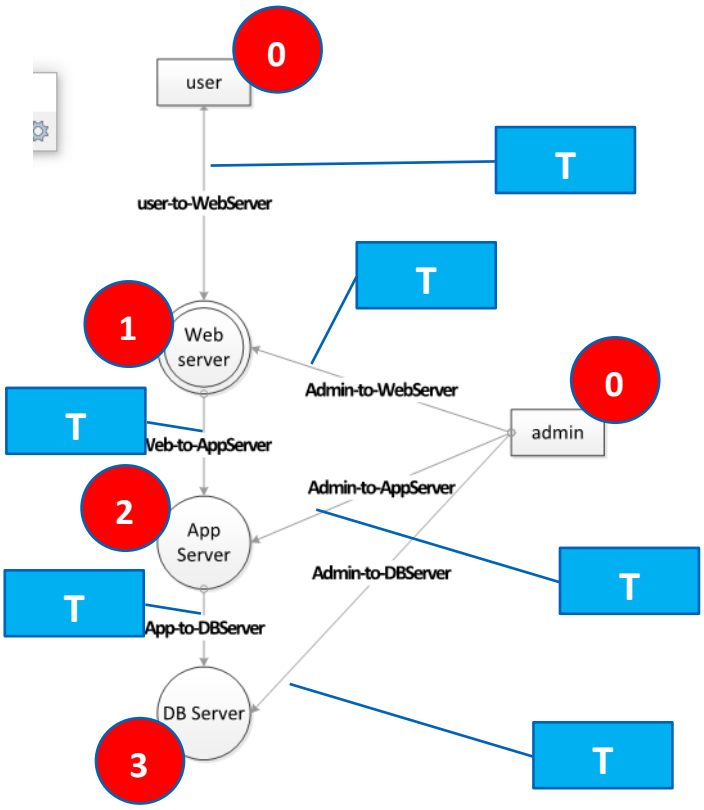
1. Elevation of Privilege Rule – Do this rule first. Place these on destination *processes* where there is a positive difference between the Zone of an origin process and the Zone of a destination process.
    a. Reason - A positive difference means you are modeling a more critical zone which probably needs more strict access control.
    b. It is most important to discover these threats first.



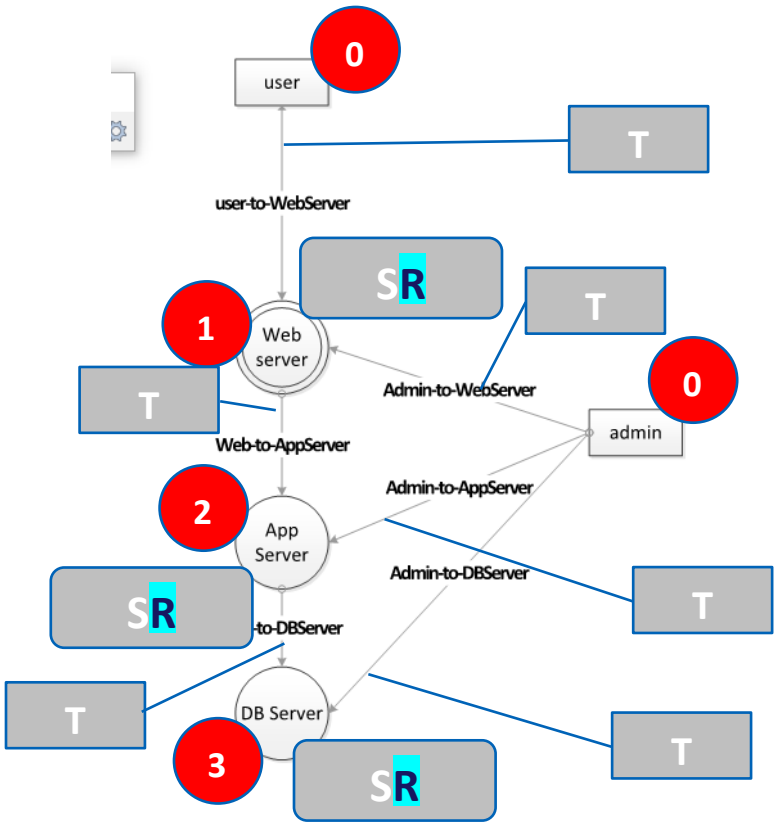
2. Spoofing Rule – Place these on destination processes where the origin process is Zone 0 or outside the control of the current model. You can also place these on the origin process. It's one of the few threats that you may want to provide guidance for the users of the origin process.
    a. Reason - You want to identify where the modelled system needs to enforce identity and this is at the entry points to the system.



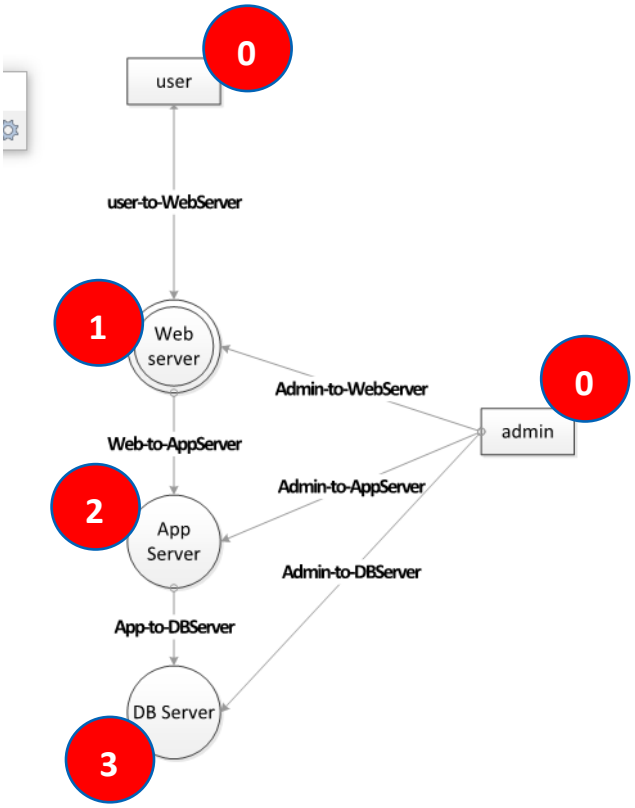
3. Tampering Rule – Put these on connecting *flows* where there is a positive difference between the Zone of an origin process and the Zone of a destination process.
    a. Reason - This is mainly a data issue, so there is a tampering potential on the data or communication between processes that should require access control permissions.



4. Repudiation Rule - Place these on destination processes where there is Tampering on the flow and Spoofing on the process.
    a. Reason – The action of repudiating a communication happens when the communication is susceptible to being changed (tampered) and it has poor identity management (authentication).

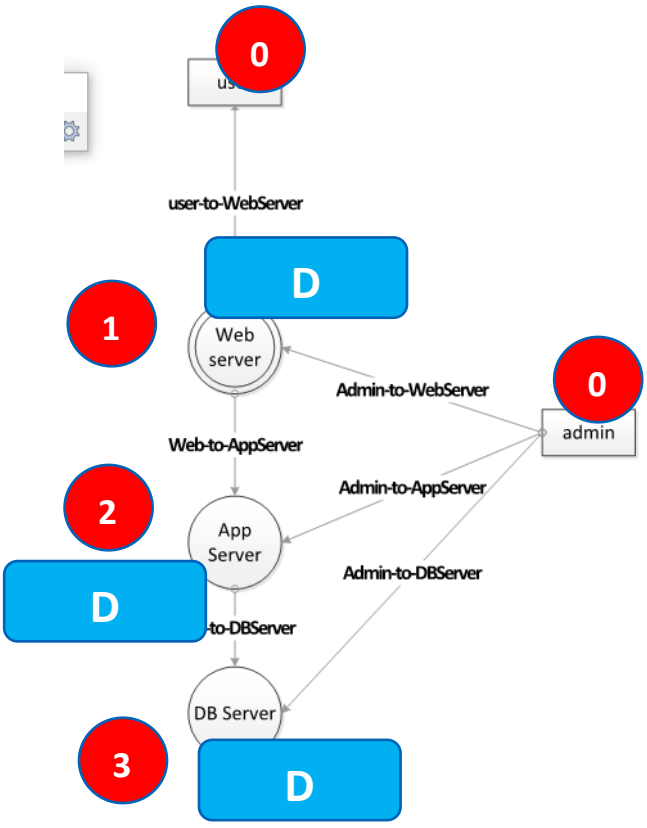


5. Information Disclosure Rule – Place these on destination *processes* where there is a negative difference between the Zone of an origin process and the Zone of a destination process.
    a. Reason – Communication (information) coming from a more critical Zone of Trust is potentially more sensitive and should be protected in a less critical zone.



(in this case there are no Information Disclosure findings from the rules)

1. Denial of Service Rule - Place these on destination processes where the origin process is Zone 0 or outside the control of the current model.
    a. Reason - You want to identify where the modelled system communicates with systems that are outside its control.



Third step – Mitigations

The model should now have a number of STRIDE threat categories that are allocated to a process or flow. Each of these categories needs to have at least 1 mitigation mapped to it.

The following table lists the elements of the OWASP Top 10 (OT10, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project), mapped to STRIDE elements. Each OT10 element has 3 STRIDE elements associated, in order of relevance. *Use the table to map potential mitigations based on OT10.*

OWASP TOP 10	STRIDE
A1 - Injection	1. Tampering 2. Elevation of Privilege 3. Denial of Service
A2 - Broken Authentication	1. Spoofing 2. Repudiation 3. Information Disclosure
A3 - Sensitive Data Exposure	1. Information Disclosure 2. Spoofing 3. Elevation of Privilege
A4 - XML External Entities (XXE)	1. Tampering 2. Elevation of Privilege 3. Information Disclosure
A5 - Broken Access Control	1. Elevation of Privilege 2. Repudiation 3. Tampering
A6 - Security Misconfiguration	1. Elevation of Privilege 2. Spoofing 3. Information Disclosure
A7 - Cross-Site Scripting (XSS)	1. Tampering 2. Elevation of Privilege 3. Repudiation
A8 - Insecure Deserialization	1. Tampering 2. Spoofing 3. Elevation of Privilege
A9 - Using Components with Known Vulnerabilities	1. Elevation of Privilege 2. Spoofing 3. Denial of Service
A10 - Insufficient Logging & Monitoring	1. Repudiation 2. Information Disclosure 3. Tampering

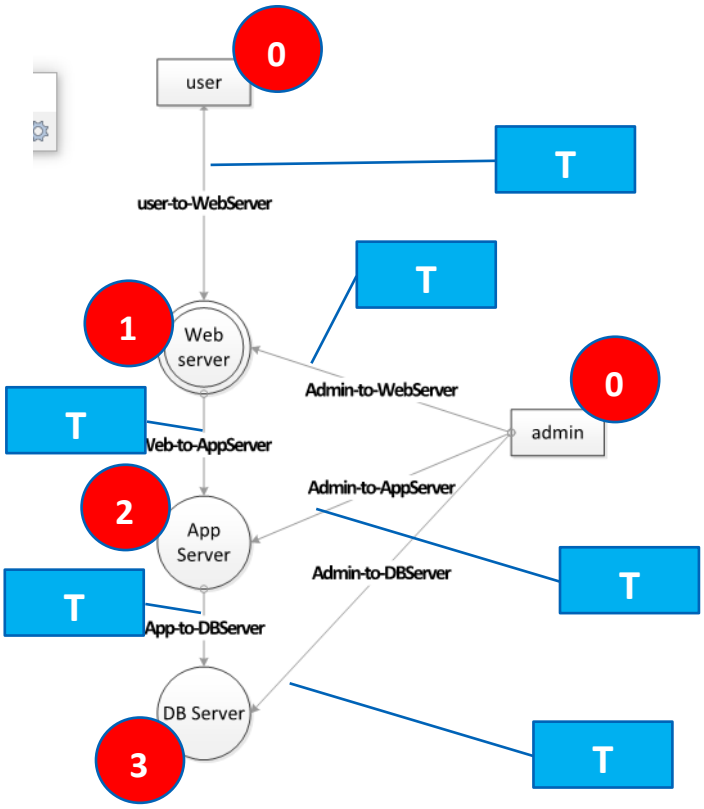
For example, all high-order Tampering issues are:

- A1 - Injection
- A4 - XML External Entities (XXE)
- A7 - Cross-Site Scripting (XSS)
- A8 - Insecure Deserialization

There are no medium-order Tampering issues.

All low-order Tampering issues are:

- A5 - Broken Access Control
- A10 - Insufficient Logging & Monitoring



The following automated Tampering issues could be mapped to the following OT10 elements and their corresponding mitigations:

mitigate against this OWASP TOP 10	Model element
A1 - Injection A7 - Cross-Site Scripting (XSS)	User-to-Webserver
A5 - Broken Access Control A8 - Insecure Deserialization	Web-to-AppServer
A1 - Injection	App-to-DBServer
A7 - Cross-Site Scripting (XSS)	Admin-to-WebServer
A5 - Broken Access Control	Admin-to-AppServer
A1 - Injection	Admin-to-DBServer

- A1 – Injection [use parameterized data]
- A5 - Broken Access Control [enforce SSH, OAuth2 or another authorization protocol]
- A7 - Cross-Site Scripting (XSS) [validate inputs by length and type, sanitize with HTML encoding]
- A8 - Insecure Deserialization [use serialization mediums that only permit primitive data types]

Fourth step – Validate

The threat model has two artefacts that are the result of the analysis:

- A list of threats
- A list of proposed mitigations

Threats

Each threat will require a test to be written to confirm it. This test can be part of the tests that the development team is writing for in-sprint activities or it can be given as a test plan to an internal or external security testing group.

If the test is good, the threat exists and the mitigation was not properly implemented.

Mitigations

A code review is done to verify if each mitigation in the threat model has been created and implemented in the design. The development team needs to focus on areas of code that have been identified in the threat model as weak.

Use static analysis code review tools for a more effective assessment process.

This concludes the HOW-TO guide for Rapid Threat Model Prototyping. Please provide any feedback you think may improve this guide and the product to info@tutamantic.com.