





PROJECTS CHAPTERS EVENTS ABOUT

Search OWASP.org

Cheat Sheets

 Watch 396

 Star 11,684

[Donate](#) [Join](#)

Threat Modeling Cheat Sheet

[← Third Party Javascript Management Cheat Sheet](#) [Transaction Authorization Cheat Sheet →](#)

Introduction

Objective of the Threat Modelling Control Cheat Sheet – To provide guidance to architects, designers and reviewers, on deriving threat models for applications.

Audience for this cheat sheet

1. Designers and Architects.
2. Threat Modeling SMEs or Security Assessors who are responsible for analyzing the security of the entire applications' components.

This cheat sheet provides guidance to assess existing apps as well as new apps. The instructions in here will help designer and architects address applications risks in an early stage of the development life cycle to help developers consider these risks while writing the code. It will also help assessors to look at risks from a comprehensive perspective.

Following the guidance in this cheat sheet, the assessors will list all possible risks and

The OWASP Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

- [AJAX Security](#)
- [Abuse Case](#)
- [Access Control](#)
- [Attack Surface Analysis](#)
- [Authentication](#)
- [Authorization Testing Automation](#)
- [Bean Validation](#)
- [C-Based Toolchain Hardening](#)
- [C-Based Toolchain Hardening](#)
- [Choosing and Using Security Questions](#)
- [Clickjacking Defense](#)
- [Content Security Policy](#)

then verifies whether there are enough security controls to protect against these risks. The assessor will then give better recommendations on how to mitigate these risks. It will help the assessor discover logical attacks. In general, the threat modeling will help designers, architects and assessors discover logical attacks.

Preparation

Understand Risk Management Basics in the context of Application Security

Understand the Relation between Risk, Threats, and Vulnerabilities.

Threat Modeling Terminologies

Information Asset, a body of knowledge that is organized and managed as a single entity. Like any other corporate asset, an organization's information assets have financial value.

Threat Agent, an individual or group that can manifest a threat. It is fundamental to identify who would want to exploit the assets of a company, and how they might use them against the company.

Attack Surface, the sum of the different points (the "attack vectors") where an unauthorized user (the "attacker") can try to enter data to or extract data from an environment.

Likelihood, the possibility of a a threat event occurring where a threat actor will exploit a weakness. The likelihood of threat events resulting in adverse impacts estimates the possibility that a threat event would result in an actual outcome. The combined analysis of both threat assessment vectors impacts established an overall threat likelihood.

Impact, the potential damage (physical, logical, monetary loss, etc) of a threat event.

Control a safeguard or countermeasure to avoid, detect, counteract, or minimize security

[Credential Stuffing Prevention](#)

[Cross-Site Request Forgery Prevention](#)

[Cross Site Scripting Prevention](#)

[Cryptographic Storage](#)

[DOM based XSS Prevention](#)

[Denial of Service](#)

[Deserialization](#)

[Docker Security](#)

[DotNet Security](#)

[Error Handling](#)

[Forgot Password](#)

[HTML5 Security](#)

[HTTP Strict Transport Security](#)

[Injection Prevention](#)

[Injection Prevention in Java](#)

[Input Validation](#)

[Insecure Direct Object Reference Prevention](#)

[JAAS](#)

[JSON Web Token for Java](#)

[Key Management](#)

[LDAP Injection Prevention](#)

[Logging](#)

[Mass Assignment](#)

[Microservices based Security Arch Doc](#)

[Multifactor Authentication](#)

[Nodejs security cheat sheet](#)

risks to information, computer systems, or other assets.

Mitigation A systematic reduction of risk or likelihood's impact to an asset.

Tractability Matrix, a grid that allows documentation and easy viewing of what is required for a system's security.

Define Objectives

Before starting the threat modeling process; it is important to identify business objectives of the applications, and identify security & compliance requirements. This is very important to be defined in advance to help to evaluate the impact of any vulnerability during the risk analysis process.

Identify application design

Understanding application design is a key activity to perform application threat modeling. It will enable the user of this cheat sheet to draw an accurate data flow diagram. Therefore, it will be easier to identify all possible risks. Moreover, the more the user of this cheat sheet understands application design, the better they will understand logical application attacks. The objective of the design document is to enumerate application components.

Review the application design document

If you are not performing threat modeling during the development (in the design phase) so you have to review the application design documents to understand the application structure and to help to generate the data flow diagram. If there are no available design documents so you have to create one. Move to the next section

Create design documents

There are many ways to generate design documents; the **4+1** view model is one of the

[OS Command Injection Defense](#)
[PHP Configuration](#)
[Password Storage](#)
[Pinning](#)
[Protect FileUpload Against Malicious File](#)
[Query Parameterization](#)
[REST Assessment](#)
[REST Security](#)
[Ruby on Rails](#)
[SAML Security](#)
[SQL Injection Prevention](#)
[Securing Cascading Style Sheets](#)
[Server Side Request Forgery Prevention](#)
[Session Management](#)
[TLS Cipher String](#)
[Third Party Javascript Management](#)
[Threat Modeling](#)
[Transaction Authorization](#)
[Transport Layer Protection](#)
[Unvalidated Redirects and Forwards](#)
[User Privacy Protection](#)
[Virtual Patching](#)
[Vulnerability Disclosure](#)
[Vulnerable Dependency Management](#)
[Web Service Security](#)
[XML External Entity Prevention](#)
[XML Security](#)

matured approaches to building your design document.

Reference to **4+1** view model of architecture [here](#).

Please note that the **4+1** is comprehensive, you may use any other design model during this phase.

The following subsections show the details about **4+1** approach and how this could help in the threat modeling process:

Logical View

Create a logical map of the Target of Evaluation.

Audience: Designers.

Area: Functional Requirements: describes the design's object model.

Related Artifacts: Design model

Implementation View

Audience: Programmers.

Area: Software components: describes the layers and subsystems of the application.

Related Artifacts: Implementation model, components

Please refer to the image in the appendix section for sample design for the implementation view.

Process View

Audience: Integrators.

Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.

Related Artifacts: (no specific artifact).

Deployment View

Upcoming Global Events

[OWASP Projects Summit, Feb 27-29th](#)

[Global AppSec Dublin June 15-19th](#)

[Global AppSec SF October 19th-23rd](#)

Create a physical map of the Target of Evaluation

Audience: Deployment managers.

Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.

Related Artifacts: Deployment model.

Use-Case View

Audience: All the stakeholders of the system, including the end-users.

Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.

Related Artifacts: Use-Case Model, Use-Case documents

Decompose and Model the System

Gain an understanding of how the system works to perform a threat model, it is important to understand how the system works and interacts with its ecosystem. To start with creating a high-level information flow diagram, like the following:

1. Identify the trusted boundaries of your system/application/module/ecosystem that you may want to start off with.
2. Add actors – internal and external
3. Define internal trusted boundaries. These can be the different security zones that have been designed
4. Relook at the actors you have identified in #2 for consistency
5. Add information flows
6. Identify the information elements and their classification as per your information classification policy
7. Where possible add assets to the identified information flows.

Define and Evaluate your Assets

Assets involved in the information flow should be defined and evaluated according to their value of confidentiality, integrity and availability.

Consider Data in transit and Data at rest

Data protection in transit is the protection of this data while it's traveling from network to network or being transferred from a local storage device to a cloud storage device – wherever data is moving, effective data protection measures for in-transit data are critical as data is often considered less secure while in motion.

While data at rest is sometimes considered to be less vulnerable than data in transit, attackers often find data at rest a more valuable target than data in motion.

The risk profile for data in transit or data at rest depends on the security measures that are in place to secure data in either state. Protecting sensitive data both in transit and at rest is imperative for modern enterprises as attackers find increasingly innovative ways to compromise systems and steal data.

Create an information flow diagram

Whiteboard Your Architecture

It is important to whiteboard system architecture by showing the major constraints and decisions in order to frame and start conversations. The value is actually twofold. If the architecture cannot be white-boarded, then it suggests that it is not well understood. If a clear and concise whiteboard diagram can be provided, others will understand it and it will be easier to communicate details.

Manage to present your DFD in the context of MVC

In this step, Data Flow Diagram should be divided in the context of Model, View,

Controller (MVC).

Use tools to draw your diagram

If you don't like to manually draw your DFD; there are several tools available that could be used:

Poirot

The Poirot tool isolates and diagnoses defects through fault modeling and simulation. Along with a carefully selected partitioning strategy, functional and sequential test pattern applications show success with circuits having a high degree of observability.

MS Threat modeling

A tool that helps in finding threats in the design phase of software projects.

Define Data Flow over your DFD

Define Data Flows over the organization Data Flow Diagram.

Define Trust Boundaries

Define any distinct boundaries (External boundaries and Internal boundaries) within which a system trusts all sub-systems (including data).

Define applications user roles and trust levels

Define access rights that the application will grant to external entities and internal entities.

Highlight Authorization per user role over the DFD

Highlight Authorization per user role, for example, defining app users' role, admins' role, anonymous visitors' role...etc.

Define Application Entry points

Define the interfaces through which potential attackers can interact with the application or

supply them with data.

Identify Threat Agents

Define all possible threats

Identify Possible Attackers threat agents that could exist within the Target of Evaluation. Use Means, Motive, and Opportunities to understand Threats posed by Attackers. Then associate threat agents with system components they can directly interact with.

Work on minimizing the number of threat agents by:

- Treating them as equivalent classes.
- Considering the attacker's motivation when evaluating likelihood.
- Consider insider Threats

The user of this cheat can depend on the following list of risks and threat libraries sources to define the possible threats an application might be facing:

1. Risks with [OWASP Top 10](#).
2. Testing Procedure with [OWASP ASVS](#).
3. Risks with [SANS Top 25](#).
4. Microsoft [STRIDE](#).

Map Threat agents to application Entry points

Map threat agents to the application entry point, whether it is a login process, a registration process or whatever it might be and consider insider Threats.

Draw attack vectors and attacks tree

During this phase conduct the following activities:

- Draw attack vectors and attacks tree.
- Identify Use Cases/Abuse Cases.
- Re-Define attack vectors to consider multi-step attacks.

Mapping Abuse Cases to Use Cases

TODO

Re-Define attack vectors

In most cases after defining the attack vectors, the compromised user role could lead to further attacks into the application. For example, assuming that an internet banking user credentials could be compromised, the user of this cheat sheet has to then redefine the attack vectors that could result from compromising the user's credentials and so on.

Write your Threat traceability matrix

Define the Impact and Probability for each threat

Enumerate Attacks posed by the most dangerous attacker in designated areas of the logical and physical maps of the target of evaluation.

Assume the attacker has a zero-day because he does. In this methodology, we assume compromise; because a zero-day will exist or already does exist (even if we don't know about it). This is about what can be done by skilled attackers, with much more time, money, motive and opportunity that we have.

Use risk management methodology to determine the risk behind the threat

Create risks in risk log for every identified threat or attack to any assets. A risk assessment methodology is followed in order to identify the risk level for each vulnerability and hence for each server.

Here we will highlight two risk methodology that could be used:

DREAD

DREAD, is about evaluating each existing vulnerability using a mathematical formula to retrieve the vulnerability's corresponding risk. The **DREAD** formula is divided into 5 main categories:

- **Damage** - how bad would an attack be?
- **Reproducibility** - how easy it is to reproduce the attack?
- **Exploitability** - how much work is it to launch the attack?
- **Affected users** - how many people will be impacted?
- **Discoverability** - how easy it is to discover the threat?

DREAD formula is:

Risk Value = (Damage + Affected users) x (Reproducibility + Exploitability + Discoverability).

Then the risk level is determined using defined thresholds below.

PASTA

PASTA, Attack Simulation & Threat Analysis (PASTA) is a complete methodology to perform application threat modeling. PASTA introduces a risk-centric methodology aimed at applying security countermeasures that are commensurate to the possible impact that could be sustained from defined threat models, vulnerabilities, weaknesses, and attack patterns.

PASTA introduces a complete risk analysis and evaluation procedures that you can follow to evaluate the risk for each of the identified threat. The main difference in using PASTA

Approach is that you should evaluate the impact early on in the analysis phase instead of

addressing the impact at the step of evaluating the risk.

The idea behind addressing the impact earlier in PASTA approach is that the audience that knows impact knows the consequences on a product or use case failures more than participants in the threat analysis phase.

Application security risk assessments are not enough because they are very binary and leverage a control framework basis for denoting risks. It is recommended to contextually look at threats impacts, probability and effectiveness of countermeasures that may be present.

$$R = (T * V * P * I) / \text{Countermeasures}$$

For more details [about PASTA](#).

Rank Risks

Using risk matrix rank risks from most severe to least severe based on Means, Motive & Opportunity. Below is a sample risk matrix table, depending on your risk approach you can define different risk ranking matrix:

- Risk Value: 01 to 12 → Risk Level: **Notice**
- Risk Value: 13 to 18 → Risk Level: **Low**
- Risk Value: 19 to 36 → Risk Level: **Medium**
- Risk Value: 37 to 54 → Risk Level: **High**

Determine countermeasures and mitigation.

Identify risk owners and agree on risk mitigation with risk owners and stakeholders.

Provide the needed controls in forms of code upgrades and configuration updates to reduce risks to acceptable levels.

Identify risk owners

For the assessors: After defining and analyzing the risks, the assessor should be working on the mitigation plan by firstly identifying risk owners which is the personnel that is responsible for mitigating the risk. i.e. one of the information security team or the development team.

For the designers or the architects: they should assign the risk mitigation to the development team to consider it while building the application.

Agree on risk mitigation with risk owners and stakeholders

TODO

Build your risk treatment strategy

- **Reduce:** building controls in the form of code upgrades, confirming a specific design for the application or building a specific configuration during the deployment phase to make sure that application risk is reduced.
- **Transfer:** For a specific component in the application the risk can be transferred to an outsourced third party to develop that component and making sure that the third party is doing the right testing for the component; or during the deployment phase, outsourcing a third party to do the deployment and transferring that risk to that third party.
- **Avoid:** an example of avoiding the risk is disabling a specific function in the application that is the source for that risk.
- **Accept:** if the risk is within acceptable criteria set earlier, in that case, the designer

risk owner can accept that risk.

For the assessor, this is considered as the last step in the assessment process. The following steps should be conducted by the risk owner, however, the assessor shall engage in 6.5 (Testing risk treatment) to verify the remediation.

Select appropriate controls to mitigate the risk

Selecting one of the controls to reduce the risk, either by upgrading the code, or building a specific configuration during the deployment phase and so on.

Test risk treatment to verify remediation

Mitigation controls will not vanish the risk completely, rather, it would just reduce the risk. In this case, the user of this cheat sheet should measure the value of the risk after applying the mitigation controls. The value of the risk should be reduced to the acceptable criteria set earlier.

Reduce risk in risk log for verified treated risk

After applying the mitigation and measuring the new risk value, the user of this cheat sheet should update the risk log to verify that risk has been reduced.

Periodically retest risk

TODO

Appendix

TODO: *Sample Design for Implementation View in 4+1 Model*

TODO: *Sample Design for Implementation View in 4+1 Model*



Spotlight: Software Improvement Group



SIG gives technology leaders the visibility they need to address current software problems and prevent future ones from ever happening. Drawing on proprietary methods and decades of expertise, SIG helps organizations fundamentally improve the security and performance of the enterprise applications that support every aspect of their businesses.

Corporate Supporters

hackerone



Usecure



CONTRAST
SECURITY



netsparker



SYNOPSIS[®]

[HOME](#) [PROJECTS](#) [CHAPTERS](#) [EVENTS](#) [ABOUT](#) [PRIVACY](#) [SITEMAP](#) [CONTACT](#)

Open Web Application Security Project, OWASP, Global AppSec, AppSec Days, AppSec California, SnowFROC, LASCON, and the OWASP logo are trademarks of the OWASP Foundation. Unless otherwise specified, all content on the site is Creative Commons Attribution-ShareAlike v4.0 unless otherwise noted and provided without warranty of service or accuracy. For more information, please refer to our [General Disclaimer](#). Copyright 2020, OWASP Foundation, Inc.