Michael Anthony Borchardt

Web Application Security: Threat Modeling

March 9th, 2024

Objective and Scope

The purpose of this Web Application Security document is to better communicate the idea behind threat modeling, with context to web applications. In today's online environment, there are many threats a web application administrator may come into contact with while overseeing operations.

Foremost, this document is intended for personnel overseeing, administrating, or developing websites, web applications, or web services. This document is also intended for novice users, who may be curious about threat modeling. I do my best to avoid language that is too technical.

1. Introduction.. 3
2. What is Threat Modeling?.. 3
3. Three Steps of Threat Modeling... 3
4. Starting a Threat Model.. 4

1. Introduction

The materials presented within the contents of this document have been collected from several sources across the Internet. These organizations include, but are not limited to, the Open Web Application Security Project (OWASP), the National Institute of Standards and Technology (NIST), the SANS Institute, and other recognized sources of industry best practices.

2. What Is Threat Modeling?

First, the main idea behind threat modeling is to identify vulnerabilities and risks in a web application before an initial production launch. Concerning web applications, threat modeling is mitigating vulnerabilities and risks associated with connecting to an online environment. When threat modeling web applications, the best threat models are generated from assuming the perspective of the attacking party.

There are many bad actors to be encountered when connecting an application to a public domain. While many of the risks associated with a public domain can be mitigated through proper threat modeling, there will, inevitably, be some risk that cannot be mitigated.

3. Three Steps of Threat Modeling.

There is a three-step process concerning threat modeling web applications. Foremost, the first step involves decomposing the application. While many applications are too complex to quickly decompose, this task may require a team of individuals. I understand decomposing the application may be perceived as vague and complex terminology. However, I understand decomposing the application as figuring out how the application interacts with the public environment.

Decomposing the application includes a number of activities, which I would consider essential. First, identify entry points to better understand where a potential attacker could interact with the application. Next, identify assets associated with the application that the attacker would be interested in. For example, some attackers may be interested in data held by an application's storage container.

The second step involved in constructing an appropriate threat model is to determine and rank threats. However, this may appear to be a difficult task. When ranking threats, an organization must consider which assets are most valuable. The most valuable assets have the highest-ranking threats.

The third step in constructing a threat model is to determine the countermeasures needed to mitigate the highest-ranking threats. Oftentimes, an organization will consider a cost-benefit analysis methodology before implementing mitigation strategies. How valuable is the asset they're trying to protect, and is the cost of protecting the asset worth the money? These are questions best left to higher-ranking administrative staff members.

Depending on the web application, website, or web service, business continuity may not be entirely dependent upon the up-time of the public domain. In some scenarios, the business is a public web application, and business continuity is entirely dependent upon the use of the service. In such scenarios, guaranteeing server up-time is of mission-critical importance.

Good threat modeling makes use of appropriate countermeasures, as a preventative effort, to stop bad actors from impacting web application functionality and continuity.

4. Starting a Threat Model.

In order to begin threat modeling, a tester must make a list of an organization’s assets. While I perceive a well-prepared organization as having this information ready for a tester, I wouldn’t ever expect this much. Being prepared to conduct a thorough analysis of an organization’s assets is part of the job.

Furthermore, the easiest way to gather this information is by asking for appropriate documentation, reviewing the documentation, and interviewing senior officials with the organization. Although this may seem like tedious work, this is an important part of mapping a threat model.

In most scenarios involving web applications, protecting the data used by the web application is of the highest priority. Some web applications make use of private medical information, financial information, or sensitive consumer data. This is all data that most organizations are expected to keep secure, as defined by regulatory standards set forth by local, state, and federal governments.

As I’ve already stated, decomposing the application is a process by which we better understand how attackers could interact with the application. For example, a poorly configured form element connected to an SQL database is common. This is an avenue by which the attacker could gain access to assets held by the application, and this is also a common strategy to manipulating data help by the application. Data is an asset most organizations care to protect access to.

I would categorize data stored by the web application as a primary asset. Also, I would consider a poor configuration on a HTML form element connected to an SQL database as a serious threat, as this is a very simple vulnerability that a novice
bad actor could exploit.

So, as is understood, we’ve started with compiling a short list of the most important assets, which is the web application’s data. Next, we’ve identified ways that an attacker could gain access to this data. Our third step to mitigating this risk to our asset would be implementing a countermeasure, which wouldn’t be too difficult.

In the scenario we’re using, appropriately configuring the HTML form element to interact with the end-user and database in a safe way would suffice as a countermeasure. However, we must be sure to test the form element with a few SQL injection attacks to be confident that we’ve implemented the form element appropriately.