Technical Disclosure Commons

Defensive Publications Series

September 30, 2019

Automated Interactive Threat Analysis of IT Architectures

Tim Hemel

Automated Interactive Threat Analysis of IT Architectures

Abstract

Threat modeling or architectural risk analysis (ARA) is a process to find cybersecurity threats in an IT system by analyzing its architecture. Because of the large number of possible threats to consider in such an analysis, it helps to automate it. Automatic application of threat analysis rules gives more consistent results and reduces the dependency on expert knowledge in threat modeling, but asks the threat modeler to annotate the architecture with relevant information. The threat modeler will however still have to know what information to provide. In this disclosure we describe a system that interactively asks the threat modeler to supply such information, based on the analysis rules. This reduces the dependency on expert knowledge even further.

Keywords

Threat modeling, architectural risk analysis (ARA), cybersecurity, IT security engineering.

Background

Threat modeling or architectural risk analysis (ARA) is a process of finding and mitigating cybersecurity threats to an IT system by analyzing its architecture. Shostack describes many ways to threat model in great detail in [1].

Most threat modeling techniques describe an architecture with a Data Flow Diagram (DFD) [2]. A DFD consists of components that exchange data via so-called data flows. Common elements in a DFD are processes (that actively process data), data stores (in which we store and from which we retrieve data), external entities (elements located outside of the system, but still interacting with it), and data flows (that describe how data flows from one component to another).

In threat modeling, we extend a DFD with so-called trust zones, separated by trust boundaries [3].

Hernan et al. describe a threat modeling process using the STRIDE framework [4]. They create a DFD with trust boundaries and ask questions about these elements, guided by the STRIDE framework, which describes six types of attacks [5]. Relating attacks from STRIDE with DFD elements leads to questions such as “do we have information disclosure in this data store?”.

STRIDE by itself is very abstract, and Hernan et al. mention attack trees as a way to refine attacks from STRIDE into more specific threats [4]. The authors also mention attack patterns as “the manifestation of [threats] in the context of some specific technology” [4]. In other words, attack patterns describe threats, but for a specific technology.

STRIDE per element [6], [7] (and later STRIDE per interaction [1]) takes the STRIDE model and defines rules that guide us which STRIDE attack types to consider per DFD element (or interaction between elements). Applying STRIDE per element even to a relatively small DFD results in a large number of applicable threats to consider, and that is why automating the threat modeling process can help. Automatically finding all threats to consider is more consistent and

less tedious.

The first generally available version of the Microsoft Threat Modeling tool lets you create a data flow diagram, and will automatically generate applicable threats using STRIDE per element [8]. Later versions of the tool let you refine the element types in a DFD, by creating so-called stencils, to which you assign properties. For example, a ‘web server’ stencil has certain properties, that make it easier to determine whether web server specific threats apply to a DFD element [9].

When implementing threat modeling, developers at EMC realized that a DFD alone did not capture enough information, and introduced annotated DFDs to capture specific properties of the DFD elements. Also, STRIDE was found too abstract and required that the threat modeler had enough attack knowledge. They made the approach more scalable and less knowledge-intensive by moving away from STRIDE and applying a collection of more specific threats, called a threat library [10]. The author also mentions the implementation of a tool to automate this process: “Given an annotated dataflow diagram, the tool will use a set of analysis rules to generate a list of potential threats that the development team can review to confirm applicability and plan mitigation.” [10]

Description of the Innovation

The automated threat modeling approach reduces the dependency on expert knowledge, but still has a problem. While it is not hard to assign general properties to DFD elements, such as whether it is a web server or an email server, more obscure properties exist that are nevertheless relevant for determining threat applicability. If threat modelers do not know about the existence or relevance of such properties, they are unlikely to assign those properties to elements, resulting in undiscovered threats.

The threat analysis rules however contain knowledge about all relevant properties. Therefore, the threat analysis engine can detect if the threat modeler has defined relevant properties for a DFD element, and if not, ask the threat modeler to define those properties for the DFD element in question. The threat modeler can then redo the analysis, now with more properties defined, and discover more threats.

A threat modeler may need to do this more than once: the relevance of one particular property may depend on the value of another property. For example, in the first analysis, the threat analysis tool may not ask about property xyz, because it does not know about property abc, on which xyz depends. The tool will ask the threat modeler to define property abc and redo the analysis, after which the tool will determine that it needs to know about property xyz and will ask about that property if it has not been defined.

This iterative analysis in which the system asks the operator for information makes the threat modeling process interactive, and while it asks extra effort from the threat modeler, it greatly reduces the dependency on expert knowledge and therefore makes the process easier.

Disclosures

This document discloses the following:

1. A system that implements:
    - Reading a specification of an IT-architecture, in the form of a data flow diagram, annotated with properties.
    - Analyzing such an architecture using a threat library and a set of analysis rules that determine whether a threat from the threat library applies to the data flow diagram's elements.
    - Reporting the applicable threats for an architecture.
2. The system from disclosure 1, wherein if the analysis of the applicability of a threat is undecidable because of missing properties, the system will notify the operator which properties are missing and ask the operator to add the missing properties.
3. The system from disclosure 2 that also visualizes the data flow diagram and optionally its properties.
4. The system from disclosure 2, wherein the specification of an IT-architecture can be created and modified with an editor (graphical or otherwise) that lets the operator create or modify the data flow diagram and the annotated properties.
5. The system from disclosure 3, wherein the specification of an IT-architecture can be created and modified with an editor (graphical or otherwise) that lets the operator create or modify the data flow diagram and the annotated properties.

References

[1] A. Shostack, *Threat modeling: Designing for security*. Wiley, 2014.

[2] Wikipedia, "Data-flow Diagram." [Online]. Available: https://en.wikipedia.org/wiki/Data-flow_diagram. [Accessed: 23-Sep-2019].

[3] C. Cornutt, "Core Concepts: Trust Boundaries." [Online]. Available: https://www.microsoft.com/security/blog/2009/08/27/the-threats-to-our-products/. [Accessed: 23-Sep-2019].

[4] S. H. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Uncover Security Design Flaws Using The STRIDE Approach." [Online]. Available: https://web.archive.org/web/20081017133440/http://msdn.microsoft.com/en-us/magazine/cc163519.aspx. [Accessed: 23-Sep-2019].

[5] A. Shostack, "The Threats to our Products." [Online]. Available: https://www.microsoft.com/security/blog/2009/08/27/the-threats-to-our-products/. [Accessed: 23-Sep-2019].

[6] L. Osterman, "Threat Modeling Again, STRIDE per Element." [Online]. Available: https://blogs.msdn.microsoft.com/larryosterman/2007/09/10/threat-modeling-again-stride-per-element/. [Accessed: 23-Sep-2019].

[7] Microsoft, "The STRIDE per Element Chart." [Online]. Available: https://www.microsoft.com/security/blog/2007/10/29/the-stride-per-element-chart/. [Accessed:

23-Sep-2019].

[8] A. Shostack, “Getting Started With The SDL Threat Modeling Tool.” [Online]. Available: http://web.archive.org/web/20081226161906/http://msdn.microsoft.com/en-us/magazine/2009.01.securitybriefs.aspx. [Accessed: 23-Sep-2019].

[9] Microsoft, “Threat Modeling Tool feature overview.” [Online]. Available: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-feature-overview. [Accessed: 23-Sep-2019].

[10] D. Dhillon, “Developer-driven threat modeling: Lessons learned in the trenches,” *IEEE Security & Privacy*, vol. 9, no. 4, pp. 41–47, 2011.