1 Summary

Securing a software product or service is often a complex task. One of the first steps in it is to discover threats facing the software, and such activity is known as 'threat modelling'. In these notes we discuss different approaches to do threat modelling, and focus on using Data Flow Diagrams as a way to identify threats. We also look at CIA and STRIDE models for evaluating threats.

2 Lecture notes

In software security industry, the words 'threat' and 'risk' often mean the same thing. In some sources, however, 'threats' are defined as people (threat actors) who impose danger. We assume that 'threats' mean risks and dangers (not people) that should be understood by modelling them. Thus, 'Threat Modelling' and 'Architectural Risk Analysis' mean the same thing as well.

We distinguish between Business level and Technical level of risk analysis. It can happen that a threat at the technical level is not actually a threat to the business, and some actual business threats may not be visible on the technical level. Business level risk analysis can be done as a preliminary step before technical risk analysis. It means looking at the threats to the business that are related to software and translating them into 'negative requirements' - i.e. use cases where software SHOULD NOT do certain things.

One of the ways to visualize threats is to put them to a 'threat tree' (also called 'attack tree'). Nodes of the tree are the reasons for threats and their consequences (which could be reasons for other threats), and arrows are drawn from threats to consequences. In practice, when such trees are created to describe actual software systems, they turn into graphs and become very confusing.

For technical risk analysis, there are two useful approaches to finding threats. One of them is to create a 'data flow diagram' (DFD), where we identify 'black boxes' that process data and draw arrows between them that denote data flows. Those boxes can be networks, physical hosts, virtual hosts, processes, files, etc. The process of creating a DFD can be iterative, so we start with a few 'black boxes' and expand them on each iteration until we get to a sufficient level of detail to make assumptions about. It is also important to identify security
domains among those 'black boxes'. One way to do that is to put some boxes within other ones like Matroska dolls to indicate that the compromise of the outer box leads to the compromise of all inner boxes. People like administrators, users and potential attackers can also be added to the diagram. An example Data Flow Diagram can be found in Figure 1.

When the entities in the system interact in a way which is more complex than the HTTP-like request-response paradigm, drawing a Message Sequence Chart (MSC) can be helpful for understanding sophisticated data flows.

After creating the data flow diagram, it becomes relatively easy to identify the Attack Surface: it consists of all points in the diagram where the (possibly) attacker-controlled data is handled by some box. Most communication protocols today follow a stack model, where higher-layer protocols are build on top of lower-layer ones, making a 'protocol stack'. Often different software handles each layer, and it is important to distinguish whether the data is actually processed or simply passed along at particular points on the DFD. For example, in a virtualized environment, the host operating system handles packets only up to IP or TCP layer, if they are destined for a guest OS. As a result, the attack surface of the host OS in this case is limited to TCP/IP and lower layers' drivers, since the upper-layer data is simply passed through to the guest OS. The points in the DFD where data is actually processed are called 'termination points'. Such points can be highlighted as large dots on the diagram and should be extensively tested for security and robustness on the actual system.

Once the data flows and termination points are known, we may start evaluating the threats associated with them. One classic approach is to consider the three security services, known as the CIA triad: Confidentiality, Integrity and Availability. Confidentiality means that the information is only provided to the intended recipient and is safe from prying eyes. Sometimes it is not necessary if e. g. the information is intended to be public. Integrity requires that the data stays accurate and does not get modified (either maliciously or erroneously) when it’s not intended to be modified. Availability means that the product or service is available, i. e. it is able to provide the intended functionality for the users. For each of the data flows and stores, we may consider if it is necessary to provide the above-mentioned security services for the given data flow and whether a failure to provide those services poses a risk to the system.

Another, newer method for describing threats is proposed by Microsoft as a part of the Security Development Lifecycle and is called STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. We may try to determine if any of these hazards pose a risk to the system when applied to each data flow.

Finally, when possible threats are identified and described, we need to determine for each threat whether it is worth to take actions to mitigate it. It may often happen that the probability of certain attacks is very small or their impact is so little that it is not profitable for the business to take mitigation steps.