**Unit 4: Threat Identification and Modelling**

There is a ready-made solution that provides a structured approach to application security—the secure development lifecycle (SDL). It is a set of development practices for strengthening security and compliance. For maximum benefit, these practices should be integrated into all stages of software development and maintenance.

What are the benefits of SDL?

The most important reasons to adopt SDL practices are:

Higher security. In SDL, continuous monitoring for vulnerabilities results in better application quality and mitigation of business risks.

Cost reduction. In SDL, early attention to flaws significantly reduces the effort required to detect and fix them.

Regulatory compliance. SDL encourages a conscientious attitude toward security-related laws and regulations. Ignoring them may result in fines and penalties, even if no sensitive data is lost.

Almost all software systems today face a variety of threats, and the number of threats grows as technology changes. [Malware that exploits software vulnerabilities grew 151 percent in the second quarter of 2018](https://www.computerweekly.com/news/252449265/WannaCry-and-NotPetya-inspiring-new-attacks), and [cyber-crime damage costs are estimated to reach $6 trillion annually by 2021](https://www.forbes.com/sites/cognitiveworld/2018/09/05/a-scoville-heat-scale-for-measuring-cybersecurity/#4a2bced43327). Threats can come from outside or within organizations, and they can have devastating consequences. Attacks can disable systems entirely or lead to the leaking of sensitive information, which would diminish consumer trust in the system provider. To prevent threats from taking advantage of system flaws, administrators can use [threat-modeling](https://en.wikipedia.org/wiki/Threat_model) methods to inform defensive measures. In this blog post, I summarize 12 available threat-modeling methods.

Threat-modeling methods are used to create

* an abstraction of the system
* profiles of potential attackers, including their goals and methods
* a catalog of potential threats that may arise

Many threat-modeling methods have been developed. They can be combined to create a more robust and well-rounded view of potential threats. Not all of them are comprehensive; some are abstract and others are people-centric. Some methods focus specifically on risk or privacy concerns.

Threat modeling should be performed early in the development cycle when potential issues can be caught early and remedied, preventing a much costlier fix down the line. Using threat modeling to think about security requirements can lead to proactive architectural decisions that help reduce threats from the start. Threat modeling can be particularly helpful in the area of [cyber-physical systems](https://en.wikipedia.org/wiki/Cyber-physical_system).

Cyber-physical systems integrate software technology into physical infrastructures, such as smart cars, smart cities, or smart grids. While innovative, cyber-physical systems are vulnerable to threats that manufacturers of traditional physical infrastructures may not consider. Performing threat modeling on cyber-physical systems with a variety of stakeholders can help catch threats across a wide spectrum of threat types.

The 12 threat-modeling methods summarized in this post come from a variety of sources and target different parts of the process. No one threat-modeling method is recommended over another; organizations should choose which method to use based on the specific needs of their project. I encourage readers interested in more detailed information about these methods to read our [SEI white paper](https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=524448) on the same topic.