Updating icsLang/sasLang to coreLang 1.0.0

2024

Introduction

When performing security evaluations of large IT systems it is helpful to use models of the systems to look at specific scenarios or run automatic simulations. In the topic of threat modeling, we refer to these models as threat instance models. With this models one can automatically run simulations of a possible attack to generate the most likely path of an attacker through the system. Considering that the IT systems are often large and complex, one challenge lies in how to create the models.

One solution of how to more easily create the threat models is to use a threat modeling language. In the language a developer defines which components, attacks and defences that may exist in a system. It is then possible for a user to use the language with the predefined constraints of the system to more easily create a model. Creating a threat modeling language can be a difficult task since it requires a great understanding of the system. It is therefore common to reuse threat modeling language and build on top of them [1]. One such language is coreLang [2]. In this paper we describe the efforts to update the threat modeling language icsLang [1] from compatibility with coreLang version 0.4.0 to coreLang version 1.0.0. The final section describes how sasLang, which is based on icsLang, was subsequently updated from compatibility with icsLang version 0.0.1 to icsLang version 0.0.2.

Updating icsLang

Previously, icsLang was based on coreLang 0.4.0. When coreLang 0.5.0 was introduced, there was a major shift where the **Hardware** asset was introduced instead of the **System** asset. Due to this, icsLang had to be re-designed and the choice was made to wait until version 1.0.0 of coreLang before updating icsLang with this change. There were three different solutions discussed[2], and in this section we will described with solution was

[1]https://github.com/mal-lang/icsLang

[2]https://github.com/mal-lang/icsLang/issues/13

chosen and how the changes were made. The choice was made to extend **IcsSystem** from **Hardware** instead of **System** and rename the asset to **IcsHardware**. We do not need the option two of dividing the hardware from industrial processes, since sasLang already has **Equipment** defined for this reason. Option three, having **IcsHardware** below sensors and actuators is not applicable to us. icsLang is built on MITRE ATTACK ICS [3], the lowest level of components defined are Field I/O, which are the sensors and actuators. We therefore assume that there are no vulnerabilities below the level of sensors and actuators.

After renaming the assets and extending from the new **Hardware**, there were several errors reported from the compiler. First, *nypassAccesscontrol* is gone, we don't replace it with *attemptUseVulnerabilityFromSoftwareFullAccess* since the removed *bypassAccessControl* led to *fullAccess*, we skip the intermediate and put *fullAccess* instead. Then, we changed the attack step *readContainedInformationAndData* on **Signal** to lead to *read* of information and contained data instead of *attemptAccess.* The attack step *attemptAccess* was removed for **Information** and **Data** and this concept is only on networks for coreLang 1.0.0. The attack step was removed because access was deemed too vague compared to read, delete and write. Lastly, we are not adding the *connect* attacksteps from old **System** to the new **IcsHardware** since these attack steps have been replaced with other attack steps. For example, *connect* previously led to amongst others *attemptUseVulnerability.* Instead there is now *attemptUseVulnerabilityFromPhysicalAccess* and similar more specific attack steps. We have not detailed what was not updated. For example, the supplychain attacks and unsafe user activities are similar and therefore not described.

Updating sasLang

There were few changes needed for sasLang because it did not alter many of icsLang's components, but rather added more concepts. Besides renaming of assets to hardware to system, there was one change made to the asset **ConnectionRule**. **AccessPoint**, extends the coreLang asset **ConnectionRule** and **ConnectionRule** no longer has the attack step *attemptConnectToApplications*, it is now either inspected or not inspected payload. We assume that this data is not inspected in sasLang. Inspected payloads adds some limitations to the attacker.

References

[1] S. Hacks, S. Katsikeas, E. Rencelj Ling, W. Xiong, J. Pfeiffer, and A. Wortmann, "Towards a systematic method for developing meta attack language instances," in *Enterprise, Business-Process and Information*

Systems Modeling, A. Augusto, A. Gill, D. Bork, S. Nurcan, I. Reinhartz-Berger, and R. Schmidt, Eds. Cham: Springer International Publishing, 2022, pp. 139–154.

[2] S. Katsikeas, A. Buhaiu, M. Ekstedt, Z. Afzal, S. Hacks, and P. Mukherjee, "Development and validation of corelang: A threat modeling language for the ict domain," *Computers & Security*, vol. 146, p. 104057, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824003626

[3] MITRE ATT&CK, "Mitre att&ck ics matrix," 2024. [Online]. Available: https://attack.mitre.org/matrices/ics/