A Domain-specific Modeling Framework for Attack Surface Modeling

Tithnara Nicolas Sun[1], Bastien Drouot[1], Fahad R. Golra[1], Joël Champeau[1], Sylvain Guerin[1], Luka Le Roux[1], Raúl Mazo[1,2], Ciprian Teodorov[1], Lionel Van Aertryck[3] and Bernard L'Hostis[3]

[1]*Lab STICC UMR6285, ENSTA Bretagne, Brest, France*

[2]*GIDITIC, Universidad EAFIT, Medellin, Colombia*

[3]*DGA-MI, Bruz, France*

{*first_author, second_author*}*@ensta-bretagne.org, rest_authors@ensta-bretagne.fr, DGA_authors@intradef.gouv.fr*

Keywords: Attack Surface Modeling, Model Federation, DSL, Modeling, Cyber Security.

Abstract: Cybersecurity is becoming vital as industries are gradually moving from automating physical processes to a higher level automation using cyber physical systems (CPS) and internet of things (IoT). In this context, security is becoming a continuous process that runs in parallel to other processes during the complete life cycle of a system. Traditional threat analysis methods use design models alongside threat models as an input for security analysis, hence missing the life-cycle-based dynamicity required by the security concern. In this paper, we argue for an attacker-aware systems modeling language that exposes the systems attack surfaces. For this purpose, we have designed Pimca, a domain specific modeling language geared towards capturing the attacker point of view of the system. This study introduces the formalism along with the Pimca workbench, a framework designed to ease the development and manipulation of the Pimca models. Finally, we present two relevant use cases, serving as a preliminary validation of our approach.

1 INTRODUCTION

As industry is moving towards automation through software and communication technologies, cyber security is coming into focus, more than ever before. The systems modeling industry addresses this need through initiatives like threat modeling (Farrell et al., 2019). Threat modeling techniques, like attack trees, are used alongside system models to perform security analysis. The outcomes of such analysis are then serialized as different artifacts. Despite the advances in the domain of cyber security, not all security analysis techniques are fully automated and a considerable amount of human involvement is required in analyzing the security status of an architecture (Siponen and Willison, 2009). For example, in security audits like Systems Security Engineering Capability Maturity Model (SSE-CMM), the security guidelines are manually validated by reviewers. In such situations, a domain specific language (DSL) that highlights the attack surface, *i.e.* the sum of different points where an attacker can interact on the system (Manadhata and Wing, 2011), improves the communication between stakeholders, the comprehension, and the rationalization of the system architecture.

In this article, we present Pimca framework alongside its DSL for highlighting the security concerns of a system. Pimca can be seen as a security-focused systems modeling language, which captures the coarse-grain architecture of large-scale systems (like CPS, enterprise systems and system of systems) to exhibit the security concerns (*e.g.* attack surfaces, attack scenarios, *etc.*) while abstracting away the internal architectural details. The Pimca DSL was originally developed by the Directorate General of Armaments (DGA), French ministry armed forces. In this context, it was used to facilitate preemptive attack surface analysis, attack impact analysis, attack routing analysis. Apart from the DSL, this study introduces an open-source security modeling framework[1], based on Pimca, that offers *(i)* A graphical modeling language *(ii)* A methodology, and *(iii)* The associated tools focused on the following objectives:

1. *System Modeling with the Intent of Highlighting the Attack Surface.* A language that incorporates the basic system analysis and the understanding of system functions to model the system from the perspective of cyber threat analysis (CTA).
2. *Security Concerns Modeling using a Graphical Language, Geared towards Automation.* This

[1]http://downloads.openflexo.org/CTA

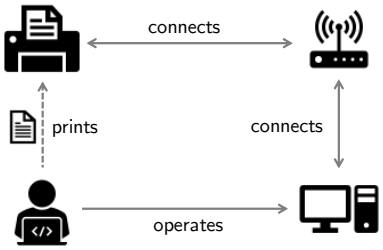

Figure 1: Running example.

helps support both manual and automated analysis. Furthermore, a human-readable model facilitates communication between multiple stakeholders.

3. *Analysis Independent Modeling.* Analysis techniques are often tightly coupled with certain DSLs, resulting in a plethora of languages. With the separation of concerns, we look forward to enabling different security analysis based on the same modeling language.

Section 2 details the Pimca modeling language. Section 3 introduces the tooling and implementation of the proposed framework. Section 4 illustrates the capabilities of the framework on two use-cases from the CPS and enterprise network domains. The related works are over-viewed in Section 5 before concluding the paper in Section 6.

2 PIMCA DSL

This section introduces Pimca, a systems DSL conceived for highlighting the attack surface during CTA. To facilitate the comprehension, let us consider a simple use-case of a technician who wants to print a document on a printer. The topology of the system comprises of a workstation and a printer, both connected to a local area network (LAN), as shown in Fig. 1. To print the document, the technician operates his workstation and sends data through the LAN. Data is then transmitted to the printer, which performs the printing task. This example highlights several aspects of systems architecture. For example, the interaction of the *cyber* component (”network”) with a *physical* agent (”technician”) and the use *concrete relations* (”operates” and ”connects”) and *conceptual relations* (”prints”).

The Pimca model captures the structure of the system-under-study. An excerpt[2] of the metamodel is presented in Fig. 2. A system is viewed as a composition of macro structures in the context of cyber physical systems, industrial control systems or systems of systems. These systems are composed of interacting and interdependent elements (Lee et al., 2015), abstracted as a *knowledgeComponent*. A *knowledgeComponent* is uniquely identifiable and has a defined address which can be either physical, virtual or generic. A *knowledgeComponent* is an abstract metaclass with three concrete subclasses: *machinery*, *resource* and *relation*.

A ***Machinery*** defines the active elements of the system. Instances of *machinery* have the ability to interact with other elements of the system via relations; *i.e.*, they can trigger actions and react to them. This concept is used to capture the active elements during CTA. When considering attack possibilities, one should consider actions and effects triggered by the active elements as potential attack steps.

A machinery has a *dimensionType i.e.*, physical, virtual or compound. A dimension defines the space in which the machineries interact *i.e.*, a machinery with physical dimension has a tangible reality and can only interact directly with other *”physical machineries”*. Similarly, a machinery with virtual dimension has an intangible reality and can only interact directly with other *”virtual machineries”*. A machinery with *compound* dimension bridges the gap between the physical and virtual dimensions. This distinction is useful for CTA because an adversary starts an interaction with the system either from a *virtual* or a *physical* node. To progress further in a different dimension, an adversary is compelled to pass through a *”compound machinery”*.

The *machinery* is further specialized to represent *performers*, *interfaces*, *checkpoints* and *networks*.

- *Performer* represents a human element. This concept is mandatory for CTA because the human elements have a unique behavior compared to automated elements. They expose a particular weakness of the system *e.g.* social engineering. Inherently, the performers have physical existence, so they are modeled as *physical machineries*. In our running example, the operator working in the system is a *performer*. The *performer* is further specialized to *attacker*. An *attacker* represents a human adversary or a group of human adversaries.
- *Interface* represents a concept bridge between two dimensions, physical and virtual. This concept is vital in CTA because it characterizes the attack surface as highlighted by (Theisen et al., 2018) and (Manadhata and Wing, 2011). Interfaces are the only concepts in the Pimca language that can be of compound dimension *i.e.*, they can access both physical and virtual spaces simultaneously.

[2]The complete metamodel is now open-source and accessible online at https://github.com/fgolra/Pimca

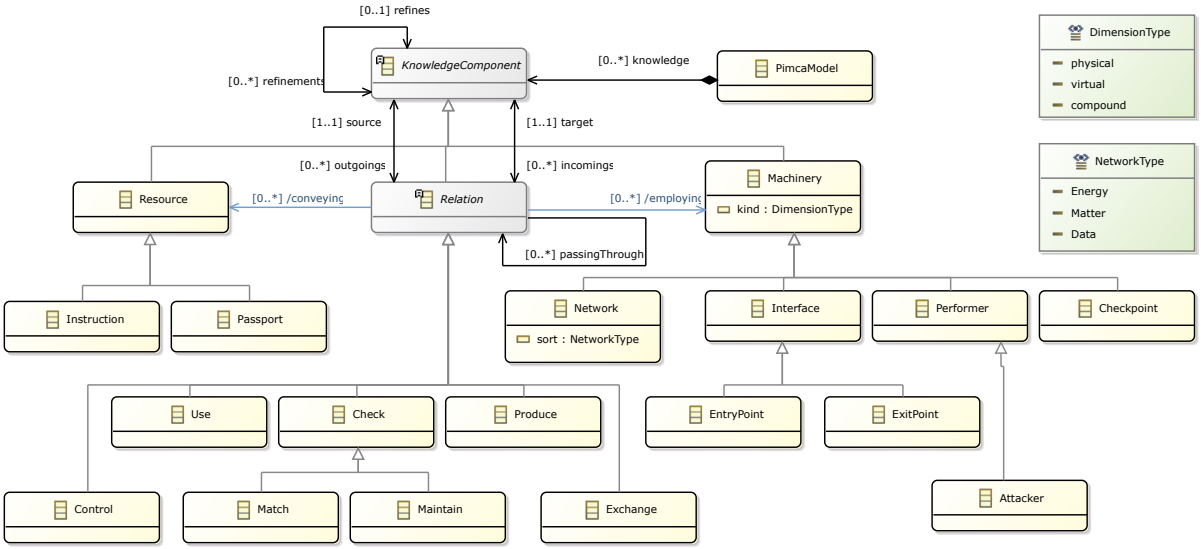

Figure 2: Pimca metamodel.

However Pimca model also allows for *interfaces* to be defined between two dimensions of the same type. To show directionality, the interfaces are specialized into *entryPoint* and *exitPoint*. From a virtual system perspective, a keyboard is an *entryPoint* (a *physical* to *virtual* bridge) while a screen is an *exitPoint* (a *virtual* to *physical* bridge). The workstation can be modeled as an *interface* between the physical space of the technician and the virtual space of the LAN network.

- *Checkpoint* represents an element checking for a specific *resource* before granting passage. In CTA, this concept highlights *machineries* that will hinder the *attacker's* progression. *Checkpoints* are either *physical* or *virtual* machineries. For example, an ID checkpoint at the entrance of an industrial site is a *physical checkpoint*. A network firewall is a *virtual checkpoint*.

- *Network* represents exchange channels. In CTA, this concept highlights *machineries* that allow reaching linked *knowledgeComponents*. Accessing a particular *network* is a pivotal step for attackers, because it offers access to other *machineries* by nature. In a *physical network*, the entities interact at the *physical* level and may only be reached by *physical* means. In a *virtual network*, the entities interact at the *virtual* level and may only be reached by *virtual* means. The nature of the exchange is defined by the *networkType* which is either *energy*, *matter* or *data*. For example, an electric grid is a *physical energy network*, while a LAN is a *virtual data network*.

A ***Resource*** defines passive elements of the system. As opposed to *machineries*, *resources* do not act on their own. They are only recipient/targets of the actions performed by the *machineries*. For example, *resources* may range from the water running in a pumping system to the electricity supplied to pumps of said system. This concept is the counterpart of *machineries* as it represents valuable assets in the system.

While *resources* can be defined as is, we also define two specialized resources *i.e.*, *passport* and *instruction*. These resources are defined as follows:

- *Passport* represents a *resource* directly linked to a specific *checkpoint*. The use of a *passport* is mandatory to get through the *checkpoint*. This concept is required for CTA since it captures the keys (for example IDs, SSH keys, *etc.*) required to pass through checkpoints.

- *Instruction* represents the resources like commands, inputs or orders used by a *machinery* to restrict its behavior. This concept symbolizes the fact that the behavior of a machinery may constitute a target in itself for the attacker. If *instructions* are represented in the system, then they can be tampered with. For example, the LAN security policy is an instance of *instructions*. An automaton description that models the behavior of a *machinery* is also an instance of *instructions*.

A ***Relation*** models the complex interactions between the *knowledgeComponents*. A *relation* has a single *source* and a single *target* of any type including other *relations*. The relation may be *employing* intermediary *machineries* on which it depends. In the example, the "print" *relation* depends on the network and the workstation. Another relation can be *con-*

veying a resource. In the example, the document to be printed is linked with the "print" relation using a *conveying* attribute. Another attribute of a relation, *passingThrough* links it to other relations on which it depends. In the running example, the "print" *relation* is not possible without the following relations: "operates", "connected to" and "connects". Thus, a *relation* refers to any numbers of *machineries* via the *employing* attribute, to any numbers of *resources* via the *conveying* attribute and to any numbers of *relations* via the *passingThrough* attribute. A *relation* is an abstract type and has multiple concrete types: *exchange*, *control*, *use*, *produce* and *check*, which are defined as follows:

- *Exchange* is a bidirectional *relation* between two machineries. In the example the workstation *machinery exchanges* data with the LAN *network*.
- *Control*, similarly to *Exchange*, relates two machineries, with the source influencing (dominating) the behavior of the target. This relation emphasizes the criticality of the source machinery *i.e.*, successfully corrupting the source, results in gaining control over the target actions. In the example the operator (*performer*) *controls* his workstation (*machinery*).
- *Use* relation states that the source *machinery* needs a *knowledgeComponent* to perform its actions. This *relation* highlights the reliance of the *source* on the *target* to properly function *i.e.*, successfully depriving the *source* of the *target* results in disrupting behavior. For example, a computer (*machinery*) needs (*uses*) electricity (*resource*).
- In *Produce* relation, opposed to *use*, the source produces the target *knowledgeComponent*. This highlights the reliance of the target on the source to exist, successfully corrupting the *source* results in corrupting or disrupting the production of the *target*. For example, a power plant (*machinery*) is producing the electricity (*resource*). Shutting down the power plant stops the electricity.
- *Check* relation, defined between a *machinery* and a *knowledgeComponent*, represents a "sensorial" relation representing that the *source machinery* has partial knowledge of the *target* state. Tampering with the *target* might be detected by the *source*. For example, a sensor *checks* the level of water tanks. If the water level raises, the sensor measures it. This relation is specialized as:
  - *Maintain* is a *check relation* between a *performer* and a *KnowledgeComponent*, such that the human source is responsible to keep the target node in proper condition. In this case, a performer may also perform operations on the target node. It can be seen as a conditional writing access from the *source* to the *target* in addition to the reading access. This *relation* is crucial in CTA because it highlights the resilience of the target node as long as the source is operational. In our example, an operator can *maintain* the printer (machinery). If the printer malfunctions, the technician acts to repair it.
  - *Match* is the specialization of the *check* relation, between a *checkpoint* and a *passport*, capturing that a specific passport is required to pass through a checkpoint. This *relation* is a key feature because *checkpoints* are the roadblocks an adversary encounters when attacking the system, disabled only by the right *passports*.

With these concepts and relations we are able to model the macro structures of a system. Moreover, the versatile nature of the interface (machinery) enables capturing the attack surface of the cyber system, thus satisfying our first objective.

3 PIMCA FRAMEWORK

This section overviews the Pimca workbench design and the proposed modeling methodology.

Workbench Overview: A Pimca model is intended for conducting various security analysis like attack surface analysis, impact analysis, attack coverage analysis, *etc.* and also the development of threat modeling views like attack modeling and vulnerability modeling. The heterogeneity of both the targeted systems and the analysis types pushes for a model federation based approach (Golra et al., 2016). Model federation allows binding the models of different paradigms through semantically rich references. Openflexo[3] is an open-source model federation framework. To link different paradigms, OpenFlexo proposes reusable COTS technology adapters that bridge the paradigm gaps (EMF, JDBC, Pdf, *etc.*).

The Pimca workbench, illustrated in Fig. 3, uses the diagramming adapter and the FML core language of Openflexo, which provides a satisfactory solution for our second requirement (the creation of a graphical language editor for Pimca). The graphical workbench is composed of multiple panels as follows: The project structure explorer *(1)* contains the Pimca XML model and the diagrams exportable in several graphical formats. Each model can be viewed through the modeling editor *(3)*. The Pimca model explorer *(2)* lists all the instances of *machinery*, *resource* and

[3]https://www.openflexo.org/

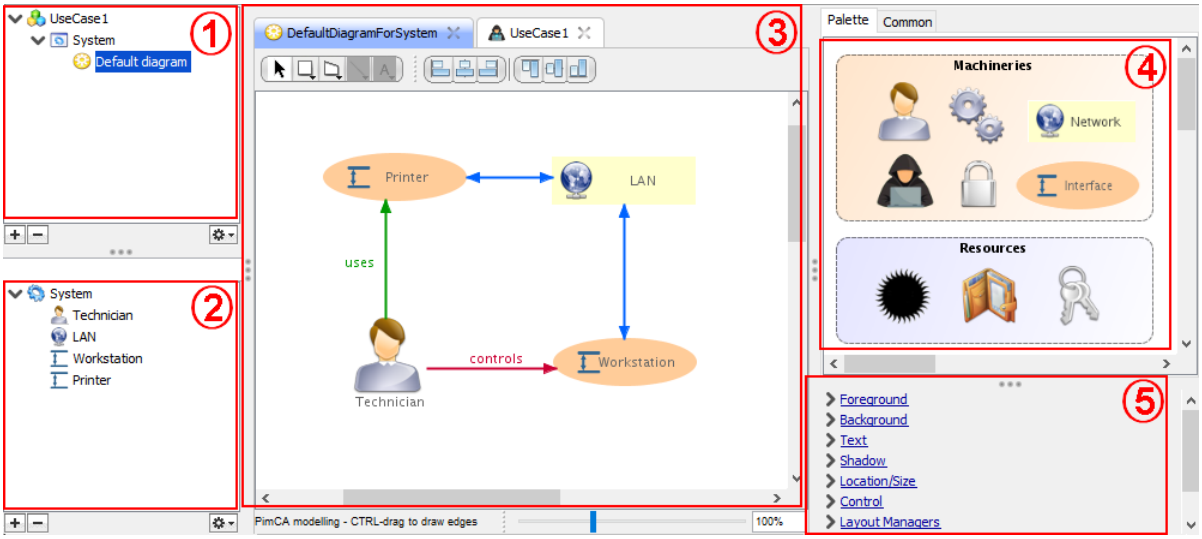

Figure 3: Running example implementation in Openflexo technology.

relation used in a given model. One can edit the properties of individual instances through this panel. The model explorer can also be used to instantiate, to delete, to show, and/or hide the language concepts. The modeling editor shows the graphical representations associated to each Pimca model. We can explore the different levels of granularity of the model for a composite node, we can zoom in to view the sub-nodes and the relationships between them. New models are designed and edited in this panel. The concept palette *(4)* exposes the available modeling concepts and relations, enabling their instantiation via drag-and-drop operations. The model inspector *(5)* allows inspecting and editing the graphical properties of individual elements and of the model as a whole.

Modeling Methodology: A Pimca model evolves iteratively along with the designer comprehension of the targeted system. If a system design document is present (say a SysML specification) the process is streamlined. In this case, the initial step of the process consists of identifying and instantiating the active (*machineries*) elements and their functional connections (*exchanges*). The following step consists of identifying the passive objects (*resources*) along with their relations to the active elements (*use/produce*). Then the other relations are instantiated according to the conceptual understanding of the systems function. In case, the system is unknown, the system discovery phase can be interleaved with the Pimca modeling phase. As such, as the level of comprehension increases the Pimca model can be iteratively refined. In this situation, the Pimca model can be seen as a blackboard capturing the systems architecture from a cyber-security perspective. In this situation, the model federation environment exposed by our workbench could be leveraged for manipulating the heterogeneous information sources.

Currently, the architecture model can be a SysML model or any other EMF model. The architecture model, the Pimca model and the diagram are all federated in the Pimca workbench and the environment ensures that they remain synchronized. With this modeling style, on one hand, we facilitate both manual and automated analysis (objective 2) and on the other hand, we make the framework extensible for real-life use. For example, if needed, one can connect a vulnerability database to our proposed framework, using a database technology adapter.

4 PRELIMINARY EVALUATION

We evaluate the effectiveness of our approach on two use cases: (i) a CPS water pumping station inspired by (Rocchetto and Tippenhauer, 2016) and (ii) a company enterprise network.

4.1 UC1 - Water Pumping Station

System Description: The water pumping station system features a water tank, various actuators, sensors and a Programmable Logical Controller (PLC). Water flows in the system from the environment, through a motorized valve controlled by PLC. The water flows out of the system from the tank through a pumping mechanism also controlled by PLC. There is a manual valve between the tank and the pump. The water level is monitored by the PLC through a sensor.

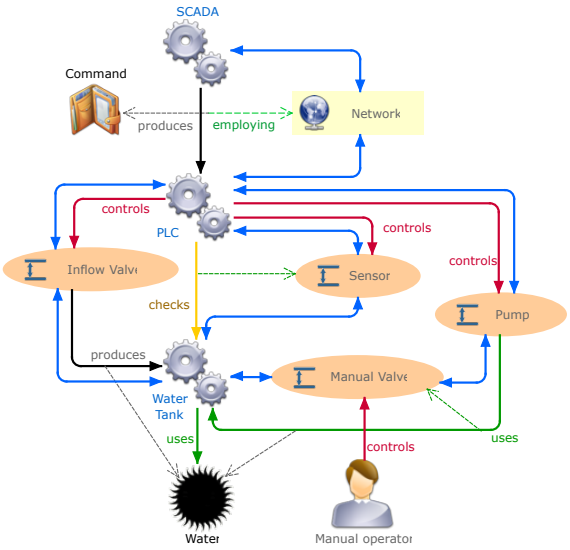

Figure 4: Use case 1: Water Pumping Station.

The PLC follows the commands entered through a keyboard. The commands describe that (i) the motorized valve must be activated and the pump must be shut down when the water reaches a low threshold, and (ii) the motorized valve must be closed and the pump must be activated when the water reaches a high threshold. PLC is connected to the plant network which also contains a supervisory control and data acquisition (SCADA). The ***Pimca model*** representing the model entities and their relations is shown in Fig. 4.

Attack Surface: Suppose the attacker's goal is to target the water running in the system. The Pimca model shows three relations that can be used to target the water: (i) *uses* relation that points to water, (ii) *produces* relation between inflow valve and water tank that *conveys* water, and (iii) *uses* relation between the pump and the water tank that *conveys* water. Furthermore, we can infer that the inflow valve, the tank, the manual valve and the pump are possible intermediary targets, because the three target *relations* depend on them. Informally, tracing back the *relations* pointing towards a node highlights the possibilities of reaching the node *i.e.* the "direct attack surface" of the node.

We can extend this reasoning further. The inflow valve that *produces* the water is *controlled* by PLC. The tank is *checked* by PLC *employing* the sensor. The manual valve is *controlled* by the operator. The pump is *controlled* by PLC. Again, the Pimca model shows the four *relations* to target the highlighted nodes: the operator *controlling* the manual valve, PLC *controlling* the inflow valve, PLC *using* the sensor on the tank to *check* it and PLC *controlling* the pump. These *relations* rely on three specific nodes: PLC, the sensor and the operator. We can repeat the process again. The sensor is *controlled* by PLC, which we already identified as a potential attack surface node. The operator has no *relation* pointing towards it. SCADA *produces* a command for PLC *employing* the network. Now, we can see the potential targets for the nodes we have already highlighted.

To sum up, we can deduce the attack surface of the system depending on the capabilities of the attacker. For example, if the attacker has social engineering capabilities and has access to the network, the attack surface extends to the operator node and the *control* relation between the operator and the manual valve. From their network access, the command also becomes part of the attack surface on the system.

Discussion: In this use case, we designed the Pimca model to represent the system architecture using the methodology presented in Section 3 and carried out a manual CTA on it. This reasoning exposes the different ways to target the water in the system by deductive reasoning and tracing back the *relations* pointing towards the nodes. We exploit the *use*, *produce* and *control relations* through the physical and virtual dimensions. To conclude, our systems modeling approach allows attack surface deduction on this use case.

4.2 UC2-Corporate Network

System Description: The corporate network system is composed of a web server and an internal network protected by a firewall. The web server provides a service that is directly accessible from the internet and it has access to the company's internal network. The firewall prevents external connections from reaching the internal network. In this use case, a computer and an active directory are connected in the internal network. The computer (PC1) is used by the administrator of the active directory. The active directory service handles the user accounts on a computer using Windows OS. This kind of service is a critical target for an attacker because it is vital to the system's security. The ***Pimca model*** showing the model entities and their relations is shown in Fig. 5. Please note that we chose not to represent *employing & passing through* attributes because they are deductible in this particular analysis.

In our scenario, the attacker has *control* over the web server and the web server *uses* the web server key (WS-key). Because the attacker *controls* the web server, we can infer that (s)he can also *use* the WS-key *employing* the web server. This *use* relation is *passing through* the *control* relation with the web server and the *use* relation with WS-key.

Attack surface: Let us consider that the attacker's

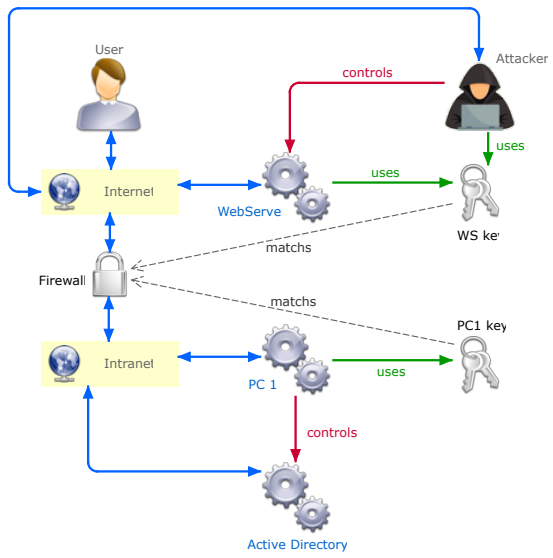

Figure 5: Use case 2: Corporate Network.

goal is to target the active directory in the internal network and that the attacker has a fixed IP. We can see that the firewall prevents access from the internet to the company's network without a *matching* key. So if the attacker is connected to the internet, the attack surface on the system is restricted to the user and the web server. If however the attacker can *use* the matching *passport* with the firewall, the attack surface expands. In this case, the internal network can be reached and consequently, PC1 and the active directory become part of the attack surface.

To sum up, we can restrict the attack surface of the system depending on the reach of the attacker. The reach can be deduced with *checkpoint* nodes that prevent the attacker from going further. For example, unless the attacker has the matching *passport*, the attack surface remains restricted to the internet.

Discussion: This use case shows our methodology in a context of an enterprise system. We carried out a CTA, complementary to the previous use case, in which we reduce the attack surface depending on the attacker's location and ability to get through a *checkpoint*. To conclude, our systems modeling approach allows attack surface refinement on this use case.

5 RELATED WORKS

Security analysis approaches vary from automated and formal approaches like Dolev-Yao (Dolev and Yao, 1981) to manual and informal approaches like SSE-CMM security audits. The Dolev-Yao attacker model is a widely-used formal model for security analysis on cryptographic protocol, and has been extended to cyber physical systems (Schaller et al., 2009; Steinmetzer et al., 2015; Rocchetto and Tippenhauer, 2016). Even though such formal approaches enable automatic security analysis, it requires a throughout understanding of the system behavior to create a realistic attacker model. An other example is Dagger (Peterson, 2016), a mission-driven semi-automated security modeling approach. We argue that a separation of concerns between the specification and analysis offers the flexibility needed for analysis independent CTA. Pimca is specifically designed for systems modeling for security analysis. Once the system is modeled, security analysis techniques can use this model (Drouot and Champeau, 2019). This allows security analysts to tailor their analysis to specific security needs such as attack surface evaluation or attack scenarios exploration.

STRIDE is a proactive security analysis paradigm (Kohnfelder and Garg, 1999) used in the context of software systems. Efforts have been made to extend STRIDE to non-software systems. In particular, Khan et al. introduce a STRIDE-based methodology adapted to cyber physical systems (Khan et al., 2017). However, (Farrell et al., 2019) share the limitations of STRIDE for cyber physical systems (CPS) noting the difficulties to use its concepts for non-software nodes. In comparison to STRIDE, our framework is not restricted to information systems and can be used to model the physical systems as well, *i.e.*, CPS and system of systems. In addition, contrary to STRIDE, the Pimca framework highlights the attack surface of the modeled system.

Moving Target Defense (MTD) is a security solution that exploits the inherent asymmetry between proactive attackers and the reactive defense (Jajodia et al., 2011) through dynamical changes of the system configuration (Zhuang et al., 2014; Xu et al., 2014). We argue that MTD fails to capture network-unrelated vulnerabilities of the system such as the human factor and operational-level vulnerabilities. Furthermore, MTD is has clear limitation for attack surface modeling on complex and dynamic architectures. On the contrary, our approach aims to capture and characterize specific relations between system elements. With the specification of *relations*, *resources* and *machineries*, Pimca framework can be used to model the system operational functions, while at the same time highlighting the attack surface.

6 CONCLUSION

We present a systems modeling framework, Pimca, with a domain-specific modeling language, methodol-

ogy and associated tools in the context of cyber threat analysis. The proposed modeling language satisfies the intention of highlighting the attack surfaces in a system model. The framework includes a methodology for the development of Pimca models. The associated tools in the framework allow developing Pimca models and integrating then with other security analysis artifacts for the development of a security solution. The approach was evaluated using two use cases, which emphasized the system modeling along with the attack surface deduction and refinement enabled by the Pimca DSL and workbench. In future, we plan to extend the Pimca DSL with behavioral primitives which will enable dynamic attack scenario enactment based on vulnerability databases and realistic system configurations.

ACKNOWLEDGMENTS

We thank the French ministry of the armed forces and the DGA for funding this research.

REFERENCES

Dolev, D. and Yao, A. C. (1981). On the security of public key protocols. In *22nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357. IEEE Computer Society.

Drouot, B. and Champeau, J. (2019). Model federation based on role modeling. In *7th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2019*, pages 72–83.

Farrell, M., Bradbury, M., Fisher, M., Dennis, L., Dixon, C., Yuan, H., and Maple, C. (2019). *Using Threat Analysis Techniques to Guide Formal Verification: A Case Study of Cooperative Awareness Messages*, pages 471–490. Springer.

Golra, F. R., Beugnard, A., Dagnat, F., Guerin, S., and Guychard, C. (2016). Addressing modularity for heterogeneous multi-model systems using model federation. In *Companion Proceedings of the 15th International Conference on Modularity*, MODULARITY Companion 2016, pages 206–211. ACM.

Jajodia, S., Ghosh, A. K., Swarup, V., Wang, C., and Wang, X. S. (2011). *Moving target defense: creating asymmetric uncertainty for cyber threats*, volume 54. Springer Science & Business Media.

Khan, R., McLaughlin, K., Laverty, D., and Sezer, S. (2017). STRIDE-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6.

Kohnfelder, L. and Garg, P. (1999). The threats to our products. *Microsoft Interface, Microsoft Corporation*, page 33.

Lee, J., Bagheri, B., and Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23.

Manadhata, P. K. and Wing, J. M. (2011). An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386.

Peterson, E. (2016). Dagger: Modeling and visualization for mission impact situation awareness. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 25–30. IEEE.

Rocchetto, M. and Tippenhauer, N. O. (2016). CPDY: Extending the Dolev-Yao attacker with physical-layer interactions. *Lecture Notes in Computer Science*, pages 175—192.

Schaller, P., Schmidt, B., Basin, D., and Capkun, S. (2009). Modeling and verifying physical properties of security protocols for wireless networks. In *22nd IEEE Computer Security Foundations Symposium*, pages 109–123.

Siponen, M. and Willison, R. (2009). Information security management standards: Problems and solutions. *Information & Management*, 46(5):267 – 270.

Steinmetzer, D., Schulz, M., and Hollick, M. (2015). Lock-picking physical layer key exchange: Weak adversary models invite the thief. In *8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, pages 1:1–1:11. ACM.

Theisen, C., Munaiah, N., Al-Zyoud, M., Carver, J. C., Meneely, A., and Williams, L. (2018). Attack surface definitions: A systematic literature review. *Information and Software Technology*, 104:94–103.

Xu, J., Guo, P., Zhao, M., Erbacher, R. F., Zhu, M., and Liu, P. (2014). Comparing different moving target defense techniques. In *First ACM Workshop on Moving Target Defense*, pages 97–107. ACM.

Zhuang, R., DeLoach, S. A., and Ou, X. (2014). Towards a theory of moving target defense. In *First ACM Workshop on Moving Target Defense*, pages 31–40. ACM.