Security Property Modeling

Hiba Hnaini[1], Luka Le Roux, Joel Champeau and Ciprian Teodorov
Lab STICC, SL Department, ENSTA Bretagne, Brest, France

Keywords: Cyber-security, Modeling, Attacker, Methodology, Formal Methods, Model-checking, Property Specification, Case Study.

Abstract: With the increasing number of cyber-attacks on cyber-physical systems, many security precautions and solutions have been suggested. However, most of these solutions aim to prevent the access of an adversary to the system. Though, with the increasing number of elements used in a system, and thus vulnerabilities, it is essential to study the risks introduced to the system to make the system itself efficient enough to react to the attacks once an attacker has obtained access. Analyzing and discovering the risks is the first step to making the system more resilient.
This paper proposes a methodology that combines the qualitative risk analysis with formal methods (model checking) to identify the risks that were not recognized during testing or functional modeling phases.
To examine this methodology, a car reservation system is modeled with an attacker, and then its security properties are verified using UPPAAL model checking tool. As a result, some risks were identified and tested for the possibility of them occurring and their effects on the system.

1 INTRODUCTION

The rapid advancement and progress of the Information Technology sector over the past years has made it possible to integrate technology in almost all domains. Some of these domains require systems with solid coordination between hardware and software or physical and cyber elements, known as cyber-physical systems. They are present in fields such as transportation, health care, and infrastructure. Due to their impacts on the real world these systems form an appealing target to cyber attackers.

A cyber-physical system is composed of many elements, each with its own weaknesses or vulnerabilities, all of which are inherited by the system, thus leading to a lot of opportunities for potential attacks. To address this issue, several approaches have been proposed (Wardell et al., 2016).

A multiple protective layer or "Defense in Depth" strategy has been proposed to handle such threats (Wardell et al., 2016). Regardless of the essential role this strategy has in system protection, it only provides preventive measures that help control the access to the system and its components. Previous work around the use of formal methods in cyber-security focuses on uncovering critical vulnerabilities in a system (Mitchell and Bau, 2011; Yu et al., 2016; Wardell et al., 2016; Kalaie et al., 2015). However, we believe it is important to use formal methods to unveil the risks as well as the vulnerabilities. The qualitative risk analysis approach may be more challenging, but it helps render the system more resilient if an adversary is to gain access to the system.

This paper proposes a methodology that uses formal methods (model checking) and qualitative risk analysis to identify the risks on the system after the adversary has acquired access.

The proposed methodology is applied to a car reservation system to discover the risks present in the system. The case study presented in this article shows the effectiveness of this methodology over other ones.

The paper is structured as follows. Section 2 overviews some related works. Section 3 introduces a motivating example from the automotive domain. Section 4 presents the proposed security modeling methodology. Section 5 discusses the results obtained on the car reservation system. Section 6 concludes this study emphasizing open research questions.

2 RELATED WORK

This study is linked to work done on qualitative risk analysis of cyber-physical systems using formal methods based on model checking.

Hnaini, H., Roux, L., Champeau, J. and Teodorov, C.
Security Property Modeling.
DOI: 10.5220/0010388206940701
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 694-701
ISBN: 978-989-758-491-6

A risk in cyber-security is known as a potential loss or damage of an asset due to a threat exploiting a vulnerability. Where an asset is a resource to be protected, a threat is anything that can exploit a weakness and cause damage to an asset, and a vulnerability is a weakness or gap in the security or protection of the system(Kbar, 2008).

A system with no threat is not vulnerable. The vulnerabilities of a system appear only in the existence of a threat. Also, a secured system or a system that is not vulnerable to an activity representing a threat is not threatened. Then, a threat is not considered as a genuine threat if the system is immune or not vulnerable to its actions(Taveras, 2019).

Thus, a risk is the coexistence of a vulnerability and a threat(Taveras, 2019). To reveal risks in a vulnerable system, a threat needs to be introduced.

Then, to test the possible impact of the risk, risk assessment is used. There are two types of risk assessment, qualitative and quantitative. In the quantitative method, risk characterization produces numerical estimates of the risk. Whereas in the qualitative approach, the estimates are non-numerical. There are two primary functions of qualitative risk assessment: ”risk identification” and ”risk characterization and analysis”. The first function, risk identification, is to find, recognize, and describe the risks in natural language or a narrative manner. The second function, qualitative risk analysis, is the risk characterization that generates a non-numerical evaluation of the risk (US Army Corps of Engineers, 2018).

A way of evaluating risks is by using formal methods. Formal methods are built on consistent mathematical fundamentals. Formal verification is “the act of proving or disproving the correctness of underlying system algorithms concerning certain formal specifications using formal methods of mathematics.”.(Akhtar, 2014)

One type of formal methods is model-checking, in which a system’s state space is extensively processed. Model-checking tackles finite-state systems, but in complex systems, state-explosion can cause a limitation. However, errors that are undetected using testing and simulation are discovered by model checking.(Akhtar, 2014)

As shown in figure 1 (Christel Baier, 2008), model-checking necessities the existence of a system model derived from the actual system and formal properties deduced from the system requirements, known as safety properties. Then, the properties are evaluated on the system model to give two outcomes: satisfied or violated.

A property is a statement that describes what the system should or should not do. It translates a requirement from natural or informal language into a formal expression, whereas a model describes how the system behaves.

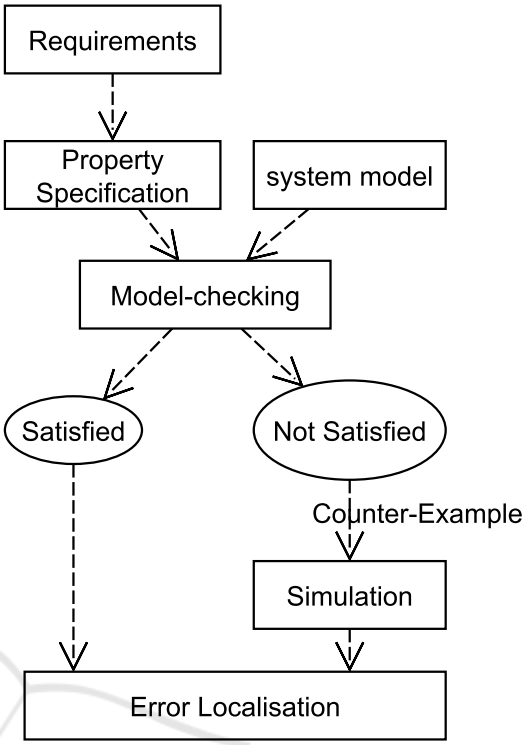

Figure 1: Model-Checking Representation.

A model-checker examines all pertinent states in a system model to validate if they meet a property. A counter-example of the property is given if it is not satisfied, showing how the undesired state is reached (As seen in figure 1)(Christel Baier, 2008).

Model-checking has many domains of application, such as verifying military systems, aircraft reservation systems, spacecraft, industrial control systems, and storm surge barriers, as mentioned in(Christel Baier, 2008).

There are different approaches to model a system, and thus the meaning of the defined properties changes accordingly. For example, a property in a safety-critical system, such as an airplane, a car, or a medical facility ensures the safety of the user where a malfunction can cause a tragedy(Hsiung et al., 2007). However, a property in network protocols security intends to check if the protocol is secure against security threats. (Basin et al., 2018)

There are also different purposes for using properties. The most common one is to verify that the system satisfies all the requirements. Another is to find the problems in a model of an algorithm, system, protocol, etc..., and propose solutions that handle these problems one by one(Lamport, 2000). However,

these are not our objectives as we are defining properties to find security risks in the existing model, similar to (Mitchell and Bau, 2011; Yu et al., 2016; Wardell et al., 2016; Kalaie et al., 2015).

We can describe our work as close to (Mitchell and Bau, 2011; Yu et al., 2016). However, our work assumes that the adversary has all the knowledge to access the system and studies the attack's effects. Unlike (Mitchell and Bau, 2011), that experiments if the attacker can gain access based on protocol analysis. (Yu et al., 2016) is very specific to the online shopping business, whereas our methodology is more general.

There are many works around model checking and cybersecurity analysis (Mitchell and Bau, 2011; Yu et al., 2016; Wardell et al., 2016; Kalaie et al., 2015). However, most of them focus on model-checking techniques and tools, but not on properties modeling related to the qualitative risk analysis method.

Through our research, we have found many papers related to qualitative risk analysis. We have also found papers addressing formal methods. However, to our knowledge, none of the papers found addressed both at the same time. And thus, this paper presents a methodology that targets using formal methods in qualitative risk analysis, with emphasis on properties modeling.

3 MOTIVATING EXAMPLE

This section introduces the case study used throughout this paper to illustrate the proposed approach from its concepts to expected outcomes. The Car Reservation Embedded System (CRES) assumes a provider company (that owns the cars) and users (who are equipped with a personal badge). Users can book a car from the company through a website, then unlock, drive, and return that car using their badge. To make it possible, the cars are equipped with a system that knows the bookings, controls the access to the car, and can be interacted with using a badge.

From a cyber-security perspective, such a system is a desirable target for malicious adversaries and exposes a large attack surface. In this paper, we focus on one car, its embedded system, and the environment. In particular, we suppose that the online booking website is secure.

This paper focuses on security analysis given a behavioral model and abstract specification of the adversary goal and capabilities. Those are considered entry-points to our contributions. Sub-section 3.1 provides a more detailed introduction to the system under study. Sub-section 3.2 introduces its modeled behaviors.

3.1 System Presentation

To discover the functionalities and behaviors of the system, we have studied a car reservation embedded system. Its original specifications include desired nominal scenarios and (C) source code.

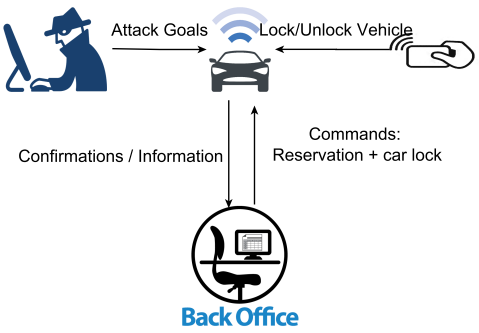

Figure 2: System Architecture.

The entire system is composed of a system or computer embedded in the car, a server or back office, and an ID badge for each user.

The reservation is first made and added to the computer in the car. Then, the booking is started. The user can now lock the car and resume the session later or request to end the reservation if the trip has ended. If the session is not ended, the user can unlock the car again to continue the session.

The embedded system interacts with one component at a time, either the server or the ID badge.

The driver uses the ID badge or user to lock and unlock the vehicle, whereas the owning reservation company uses the server to control the car.

These components (the embedded system in the car, the server, and the badge) are connected through many technologies and protocols. This signifies that the system inherits all the vulnerabilities and weaknesses of these technologies and protocols. Adversaries can exploit them to target or access the system.

When applying the methodology discussed in section 4, an interesting target in our system under study (Car Reservations System) would be, for example, a car lock. The feared events would be that the attacker might want to keep the car locked or unlocked according to the desired outcome.

3.2 System Model

As a first step, the system under study is modeled by identifying the behaviors.

To obtain the formal model of the system, a functional model is derived from the textual specification of of the system, as explained in (Mitchell and Bau,

2011). A high-level of abstraction is considered since the aim of the formal model is to check the behavioral functions of the system.

Figure 3 illustrates the behavior of the software deployed on the car in the Car Reservation System. The figure shows the different states the system can have, with the transitions from and to each state. The states, represented by the circle, are the system's conditions after or before a transition. One state the system can have is idle. It is the initial state of the system that shows that there is no ongoing process or reservation. Another state is "Car Unlocked" which is reached after receiving a Car Unlock or Start Reservation message. This state shows that there is an ongoing reservation or that the car is currently unlocked and that the engine can now be started. The other states are Reserved, Car Locked, Ongoing Session, Res End Requested, and Verify END Session.

The transitions are the messages exchanged between the different components of the system to command it to act in a certain way. The transition messages with a question mark (?) mean that the system is waiting to receive this message from another component (Back Office, Mobile App, Badge, etc.). The other transition messages with an exclamation mark (!) represent the ones sent by the system to another component after receiving the expected message. The other labels are guards to verify some system conditions.

The first step is for the server to send an Add Reservation request to the system embedded in the car. Then, the reservation is initiated by the Start Reservation request, where the car is unlocked, and a driving session is started. The car at this stage can be locked using a Lock Car request for an intermediate stop. Then, the car is unlocked, using the Unlock Car request, and the session resumes. The session can be finished using the END RES Request. Then, the car is locked using the Lock Car request, and it goes back to idle after verifying the conditions of the car lock and the network connection. Also, after each request, a confirmation message is sent to the server.

Figure 4 shows an abstract model of the server which, based on the requests previously define, communicates with the system embedded in the car.

We show another model in Figure 5. It represents using an ID badge to control the car lock using the Lock Car and Unlock Car requests. Then, the entire model, with its different components, is verified to ensure the well functioning of the system.

Some of these functional requirements are:

- If the car is unlocked by the server or the ID, the state in the server should also be car unlocked.

- If the car is locked by the server or the ID, the state in the server should also be car locked.

- If the car is in the Ongoing Session state, the state in the server should be car unlocked (In service of the user).

- If the car is in the reservation end requested state, the state in the server should be car unlocked.

- If the car is in the Verify end session state, the state in the server should be car locked.

Using the methodology presented in Figure 1, these functional requirements have been encoded in queries and verified on this model. However, this level of specification lacks the conditions necessary for identifying and formalizing the security requirements. To achieve this, we identify the main methodological tasks needed to formalize the security requirements, and we integrated them into the model-checking methodology. The experimental section illustrates these in the car-sharing system.

4 SECURITY MODELING METHODOLOGY: CONTRIBUTION

We adopt the Qualitative Risk Analysis approach in our modeling methodology. The model comprises 3 principal components, the system, the attacker, and the properties.

The aim of having a model with these three components is to produce the co-existence of a vulnerable system and a threat to reveal the risks.

The attacker model is considered being a threat. It is used to uncover the risk in the system, which is assumed to be vulnerable. Then, the properties which represent the risks are checked on the entire model. This serves to use model-checking in qualitative risk analysis. This is represented in figure 6, where an attacker model is added to figure 1. The figure shows how the attacker is integrated into model checking. The attacker model is composed of the system model using an attack composition operator, and the attacker interests are combined with the system requirements to specify the properties. Then, the combined properties, requirements, and attacker interests are verified on the composed model: system and attacker. This results in identifying the genuineness of the risks and if there is a possibility of them happening.

4.1 Attacker Model

After the system is modeled and verified, the vulnerabilities, that form potential targets of the attacker in

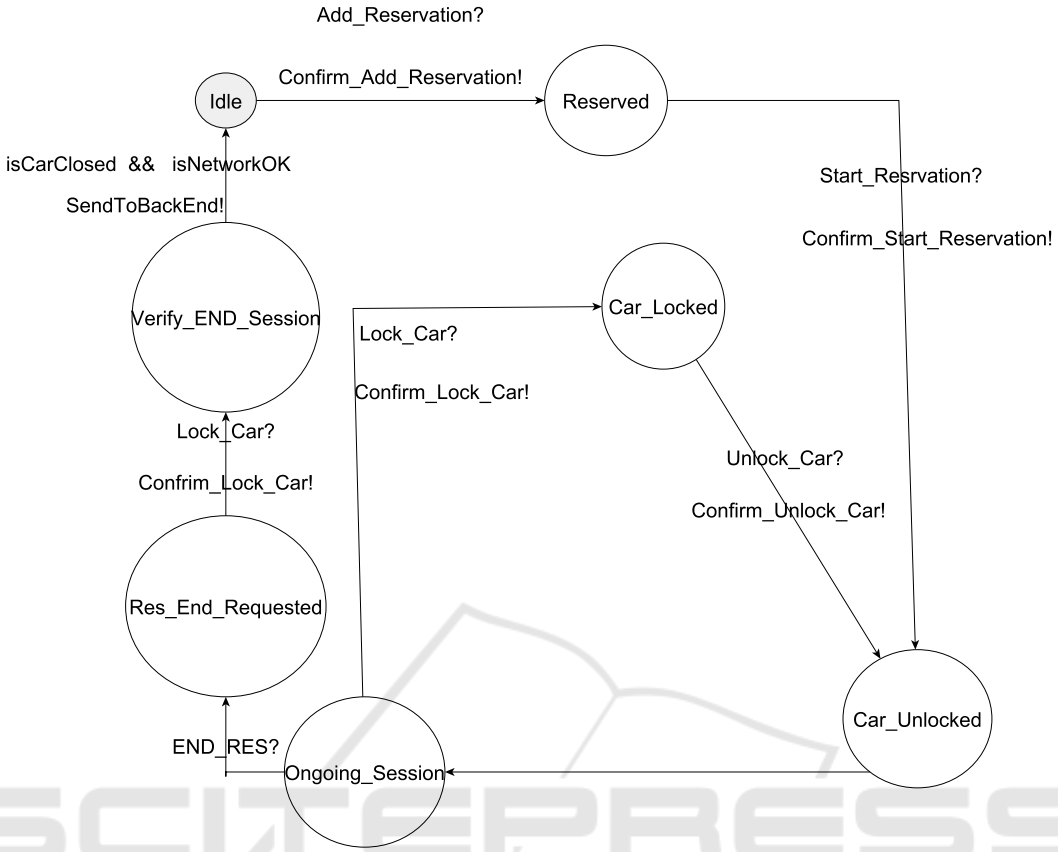

Figure 3: System Model (General Representation).

the system, are identified. By considering the adversary's perspective as in (Mitchell and Bau, 2011) and (Yu et al., 2016), it is made possible to recognize some behaviors that form appealing targets for the attacker. The attacker is then modeled, representing the threat to the system to exploit these vulnerabilities. This step aims to reveal the risks on the system when an attacker attacks.

An attacker is modeled assuming that they have initial knowledge of the messages transmitted between the different components of the system. Thus, the adversary is given the ability to manipulate system states similarly to the legitimate system.

The attacker can be given all capabilities to manipulate the system at any point in the execution. This type of attack model can help discover all potential attack scenarios (Figure 7). This method aims to as many risks as possible in the system by introducing an attacker that carries most of the threats that can threaten the system. Also, this method is employed because there is no ultimate way of modeling a single attack scenario. And this is because there is no way to be sure of the intentions of the attacker.

However, in larger systems, this can cause a state explosion problem, so there are too many states and scenarios. This makes large models impossible to explore exhaustively. A solution to this problem would be to break down the attacks into several scenarios where the attacker is not given full capabilities, but only the ones enough to achieve a certain attack scenario.

This can also be resolved using the Partially Bounded Context-Aware Verification technique discussed in (Roux and Teodorov, 2019). This approach aims to decompose the state-space and the verification problem using environment-based guides that can help select or remove attack scenarios.

Another approach is to use the scenarios to limit the capabilities of the attacker and observe the effects on the system. Or to use the scenarios to observe how much damage an attacker with minimal capabilities can cause. This approach helps identify the resistance a system has to a different type of attacks represented by different scenarios.

Another important aspect to be modeled in the context of model checking and qualitative risk analysis is the properties. They allow to verify the behavior

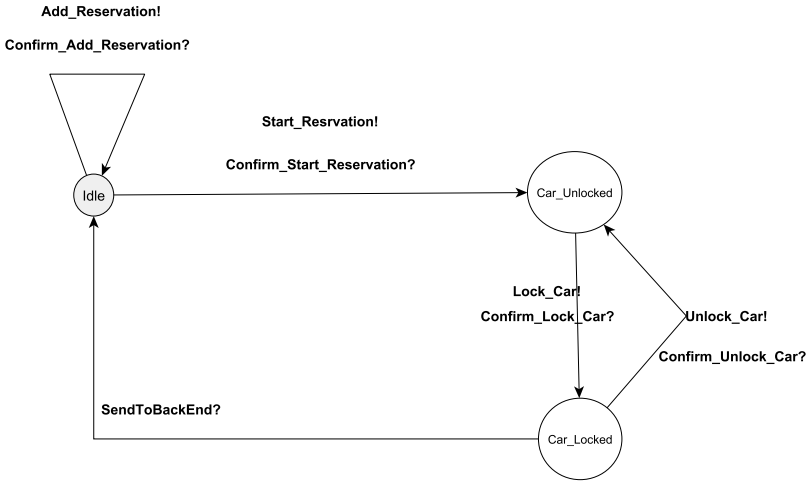

Figure 4: Server Model (General Representation).

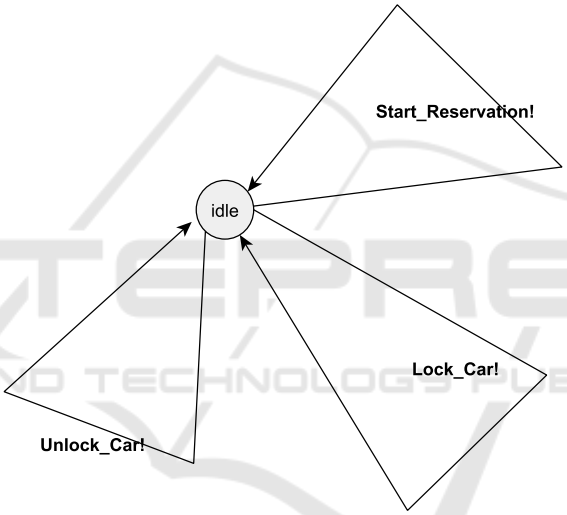

Figure 5: ID badge Model (General Representation).

or functioning of the system in it is a normal case or when it is under an attack. Section 4 discusses how the properties are modeled and the link between them and risk identification.

4.2 Security Properties

After determining the potential targets, the attacker's goals, or the feared events as defined in the EBIOS methodology(Fortat et al., 2015), are translated into properties. Each property represents one feared event. Verifying the properties helps determine if the attack issued on the system has been successful or not and whether the system is resilient to this attack. The properties check if a certain attack on the system reached the feared event.

Thus, properties represent risks on the system. A system would want to prevent the risk, but an adversary would want the opposite, as in Table 1.

According to which perspective is considered, there are two ways to write the security properties.

The first way is to take the system's point of view and create properties that ensure the well functioning of the system during or after an attack. After the model is simulated and the property is checked, if the property verification is successful, the system is resilient to this attack and its risks. However, if the property verification fails, the system is faulty, or the attack issued was successful in changing the original behavior of the system and thus reaching the risk.

Another way to write the properties is to consider the attacker's perspective. Here, properties ensure that the goals of the attack or risks are reached. In

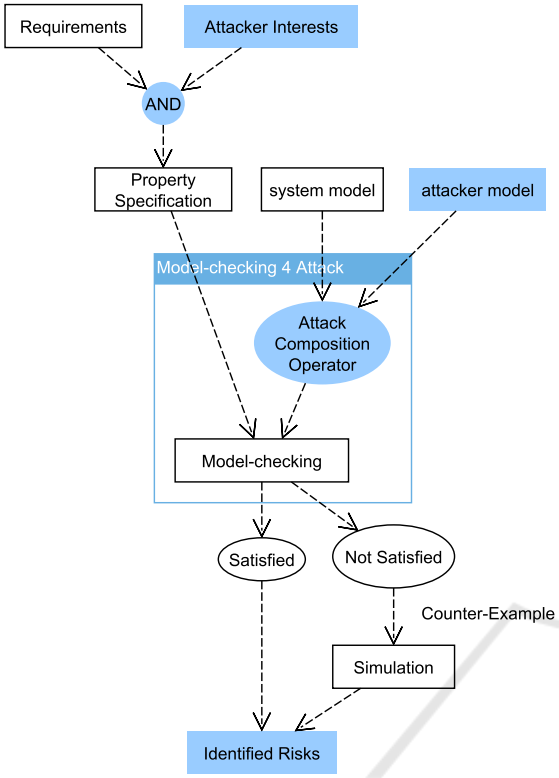

Figure 6: Model-Checking with Attacker.

contrary to the previous case, where the system's perspective was considered, the successful verification of a property shows that the system is faulty, or that the attack has successfully reached its goals of altering the system's behavior. This means that the attack has introduced a risk to the system. However, the failed verification of the property implies that the attack was unsuccessful, and the system is resilient to the attack the property describes. Thus, there is no risk caused by this threat.

Table 1 summarizes the security property verification from both perspectives (system and attacker).

From a system's perspective, (in table 1) it is to be made sure that the potential targets or assets are not put at risk by the attacks or the threats.

However, from an adversary's perspective, it is the opposite. The properties, in this case, are written to verify that the threats on the targeted assets were successful in creating a risk by controlling, altering, blocking... the target or the entire system.

To cover all the risks or properties, the methodology can adopt both perspectives.

5 APPLICATION TO THE MOTIVATING EXAMPLE

In this case-study, the attacker is given all possible capabilities to manipulate the system at any point in the simulation. We have adopted this method because our aim of modeling the attacker is not to abstract generic behaviors based on classified attacks, but to reveal all the risks on the system. This is represented in Figure 7 where the attacker can send all messages that can be received by the car's computer model (Figure 3).

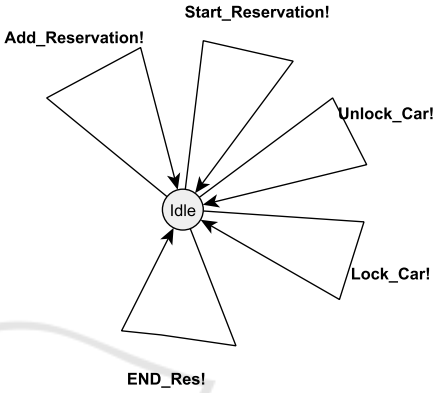

Figure 7: Attacker Model (General Representation).

In our model, we are considering the system's point of view to construct the properties since we aim to verify the well functioning and the resilience of the system during or after an attack.

Considering the previously given example of a potential target or targeted asset, the car lock, the following property can be deduced:

- If the car is unlocked, it should be able to go back to idle and be booked again.

This means that the attacker, representing the threat, should not be able to keep the car unlocked after the session has ended. Keeping a car unlocked would cause the risk that the car becomes out of service and cannot be booked again.

Then, the property is translated from natural language into a query to be verified upon simulation of the model.

6 CONCLUSION

The use of many components and protocols in one system increases the number of vulnerabilities. This is because the system inherits all of the vulnerabilities of each element and protocol added. Even though researchers have developed a multiple-protective-layer strategy and other security protocols, these solutions

Table 1: Security Property Modeling.

*	System Perspective (+)	Attacker Perspective (-)
Property Successful (+)	Threat causes no risk (+)	Threat causes risk(-)
Property Failed (-)	Threat causes risk(-)	Threat causes no risk (+)

have focused on preventing the adversary from gaining access to the system.

However, with the many vulnerabilities present in a system, one must assume that the attacker is to gain access at some point. And thus, in the presence of a vulnerability and a threat, the attack will introduce a risk on the system by changing its normal behavior.

This paper has introduced a methodology that employs formal methods and model checking to help discover these risks through qualitative risk analysis, where uncovering the risks is the first step to making a system itself more resilient to attacks. This methodology was applied to a study case composed of a Car Reservation System. As a result, we identified risks in the system, such as the attacker could keep the car locked or unlocked.

In the study case, the adversary is given full capabilities to attack the system. As for future work, these capabilities are to be broken down into scenarios. This approach helps determine the extent of damage or risks an attacker with minimal or different capabilities can cause. Also, scenarios that attack all the elements of the model are to be added.

ACKNOWLEDGMENTS

This project has been supported by the French Directorate General of Armaments (DGA), the European Regional Development Fund (ERDF) of the EU, the Brittany Region, the Departmental Council of Finistère and Brest Métropole as part of the Cyber-SSI project within the framework of the Brittany 2015-2020 State-Region Contract (CPER).

REFERENCES

Akhtar, N. (2014). Requirements, formal verification and model transformations of an agent-based system: A case study. *Computer Engineering and Intelligent Systems - IISTE*, 5:1–16.

Basin, D., Cremers, C., and Meadows, C. (2018). *Model Checking Security Protocols*, pages 727–762. Springer International Publishing, Cham.

Christel Baier, J.-P. K. (2008). *Principles of Model Checking*. International series of monographs on physics. The MIT Press.

Fortat, F., Laurent, M., and Simatic, M. (2015). Games based on active nfc objects: Model and security requirements. In *2015 International Workshop on Network and Systems Support for Games (NetGames)*, pages 1–3.

Hsiung, P.-A., Chen, Y.-R., and Lin, Y.-H. (2007). Model checking safety-critical systems using safecharts. *Computers, IEEE Transactions on*, 56:692–705.

Kalaie, A., Bhatia, S., Koutsoukos, X., Stouffer, K., Tang, C., and Candell, R. (2015). Towards a systematic threat modeling approach for cyber-physical systems. volume 2015.

Kbar, G. (2008). Security risk analysis for asset in relation to vulnerability, probability of threats and attacks. In *2008 International Conference on Innovations in Information Technology*, pages 668–672.

Lamport, L. (2000). The mutual exclusion problem part ii: Statement and solutions. *Journal of the ACM*.

Mitchell, J. C. and Bau, J. (2011). Security modeling and analysis. *IEEE Security & Privacy*, 9(03):18–25.

Roux, L. and Teodorov, C. (2019). *Partially Bounded Context-Aware Verification*, pages 532–548.

Taveras, P. (2019). Cyber risk management, procedures and considerations to address the threats of a cyber attack.

US Army Corps of Engineers (2018). Risk assessment - quantitative methods. In *Corps Risk Analysis Online Training Modules*, pages 1–49.

Wardell, D. C., Mills, R. F., Peterson, G. L., and Oxley, M. E. (2016). A method for revealing and addressing security vulnerabilities in cyber-physical systems by modeling malicious agent interactions with formal verification. *Procedia Computer Science*, 95:24 – 31. Complex Adaptive Systems Los Angeles, CA November 2-4, 2016.

Yu, W., Yan, C. G., Ding, Z., Jiang, C., and Zhou, M. (2016). Modeling and verification of online shopping business processes by considering malicious behavior patterns. *IEEE Transactions on Automation Science and Engineering*, 13(2):647–662.