



Car Reservation Software Model

Lab STICC/P4S

Hiba Hnaini, Joel Champeau, Luka Le Roux, Ciprean Tedorov

October 2020

Outline

- Context
- Formal Validation & Related works
- Tool Used – UPPAAL
- Scenario to model
- System Modeling
 - System Simple Model
 - Back Office Model
 - RFID Model
- Risk Analysis
- Results
- Conclusion

Context

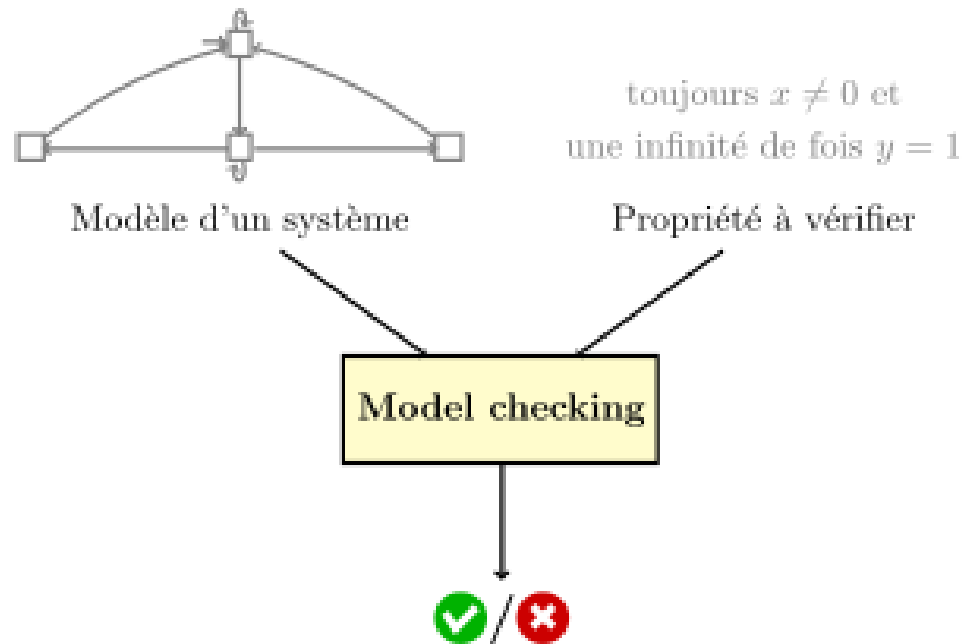
- A model of the software implemented in a embedded computer was created based on the scenarios
- The source code of the embedded system was also inspected
- The prototype was confirmed with the creators of the system
- For Lab STICC: How can formal models ease the cybersecurity analysis of connected vehicles?

Formal Verification & Model Checking

- Formal verification is the act of proving or disproving the correctness of a system with respect to a certain formal specification regarding system properties.
- Model Checking:
 - automatic verification technique
 - input:
 - a model of a system
 - a logical formula which describes the desired properties
 - model checking algorithm (output): checks if the model satisfies the formula if the property is not satisfied, a counterexample is produced

Formal Verification & Model Checking

- Natural language to formal specification (model and properties in formal logic)

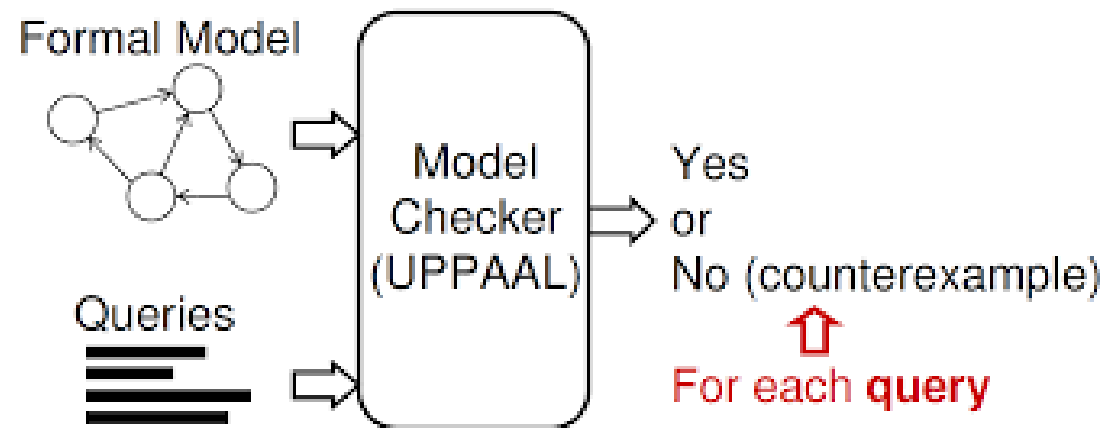


Related works

- Some work has been done on model-checking of cybersecurity
 - Finding vulnerabilities [\[1\]](#)
 - Checking network protocols (access to the system)[\[2\]](#)
 - Not focusing on properties[\[1\]](#) [\[2\]](#) [\[3\]](#)
- Our work
 - Assumes that the attacker already has access
 - Studies the risks on the system(Risk Analysis)
 - Focuses on defining properties

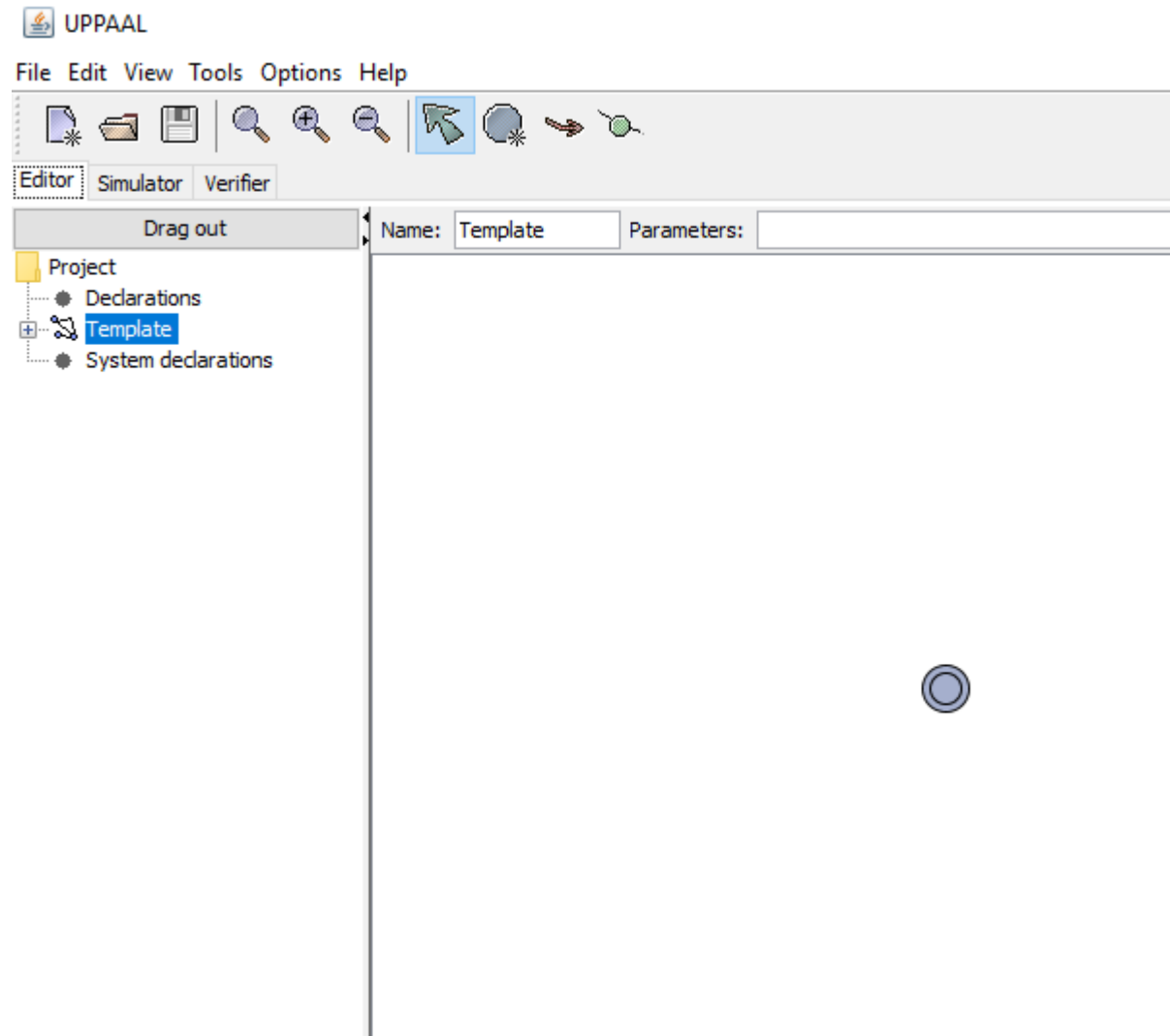
Tool Used - UPPAAL

- Uppaal is an integrated tool environment for modeling, validation and verification of real-time systems



UPPAAL Editor

- To create the templates of the different components and define the system, functions, and variables



UPPAAL Simulator

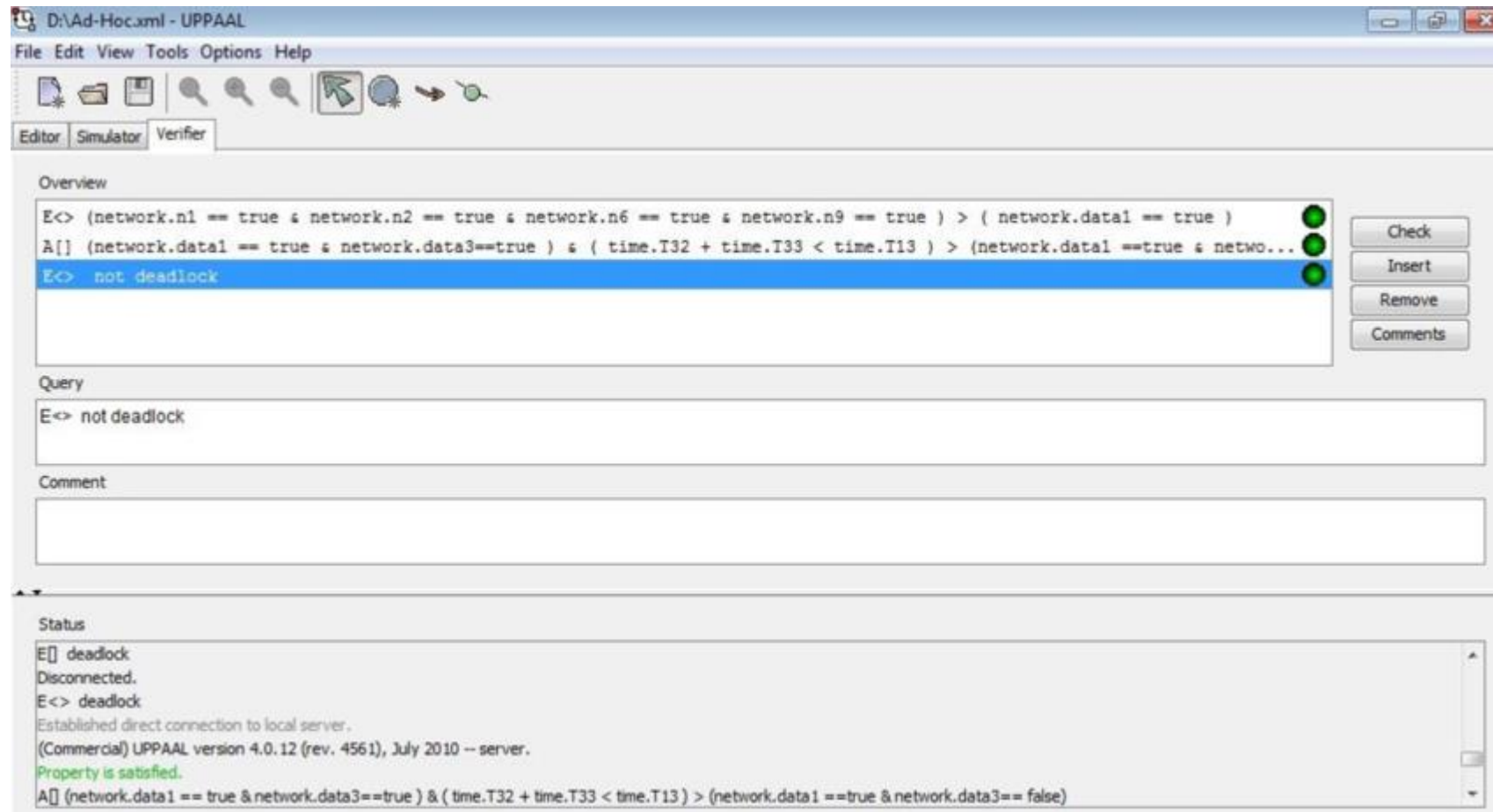
- To simulate a specific or random scenario while observing the variables, the components, and the sequence diagram

The screenshot displays the UPPAAL Simulator interface. The main window title is `/home/trobert/Téléchargements/tpUppaal/tournequet_cours.xml - UPPAAL`. The interface is divided into several panels:

- Editor/Simulator/Verifier:** Contains a menu bar (File, Edit, View, Tools, Options, Help) and a toolbar with various icons.
- Drag out (Left):** Shows "Enabled Transitions" with a list: `push: Process2 --> Process1` and `token: Process2 --> Process1`. Below this is a red instruction: "Select Next transition for current State" and buttons for "Next" and "Reset".
- Simulation Trace:** A list of events: `(locked, -)`, `token: Process2 --> Process1`, `(unlocked, -)`, `push: Process2 --> Process1`, `(locked, -)`, `token: Process2 --> Process1`, `(unlocked, -)`, `push: Process2 --> Process1`, and `(locked, -)`. A blue instruction "Execution Trace overview" is present.
- Trace File:** A text input field.
- Navigation:** Buttons for "Prev", "Next", "Replay", "Open", "Save", and "Random".
- Speed Control:** A slider from "Slow" to "Fast".
- Drag out (Middle):** Shows current state variables: `Process1.tokens = 1` and `Process1.H ∈ [0,30]`. An orange instruction "Variables values in current state" is overlaid.
- System Diagram:** A network of automata with two processes: **Process1** and **Process2**. Process1 has states `locked` and `unlocked`. Transitions include `push?`, `token? H=0,token++`, `H<=30 token?`, and `H<=30 push? reinit_tokens()`. Process2 has a state `token!` and a transition `push!`. The system is labeled "System : network of automata".
- Sequence Diagram:** A diagram showing the interaction between Process1 and Process2. States are `locked` and `unlocked`. Messages are `token` and `push`. The current state is highlighted in green. A green instruction "Sequence diagram with current state highlighted" is overlaid.

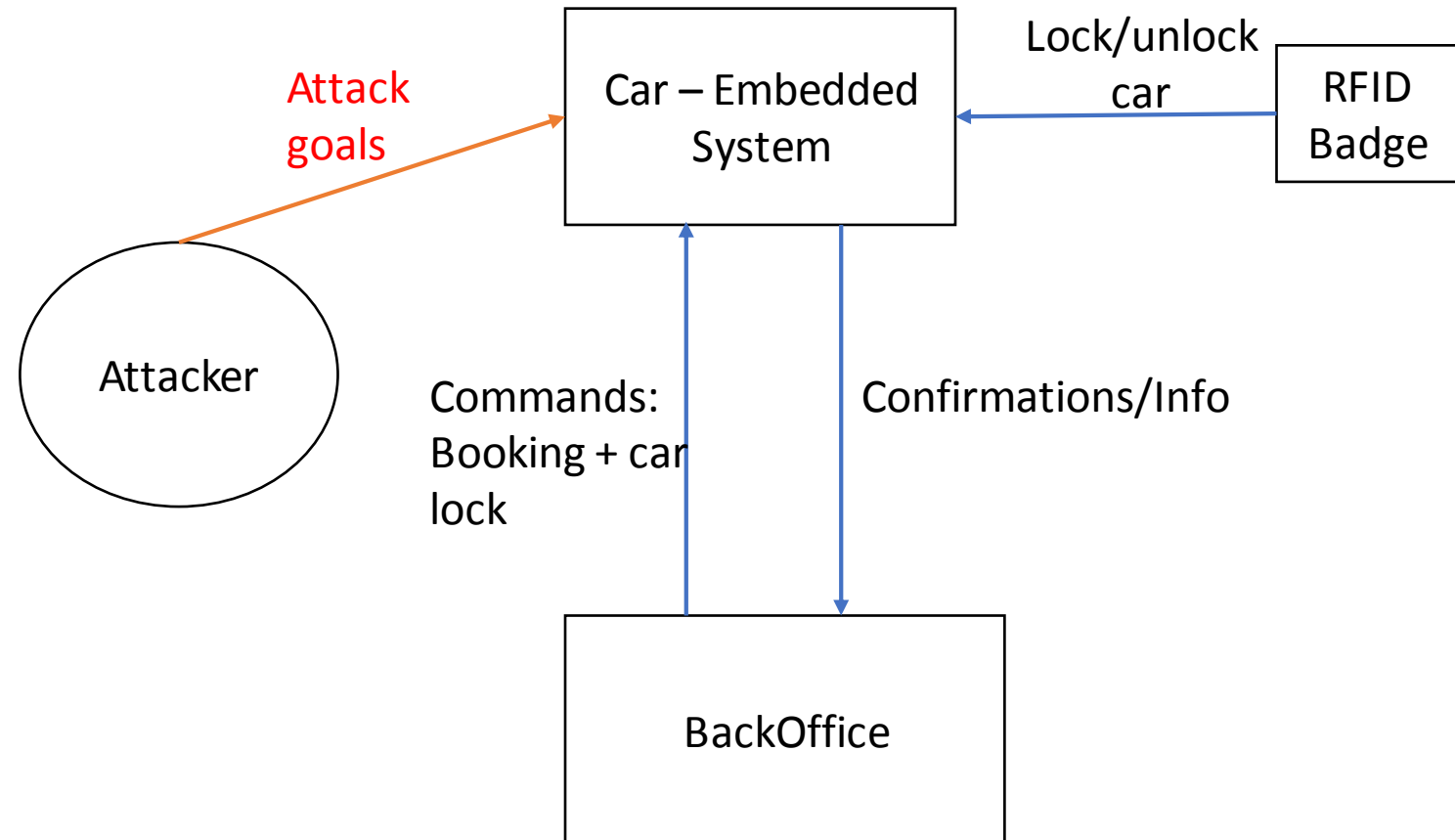
UPPAAL Verifier

- To verify the system properties or specifications written in the form of queries



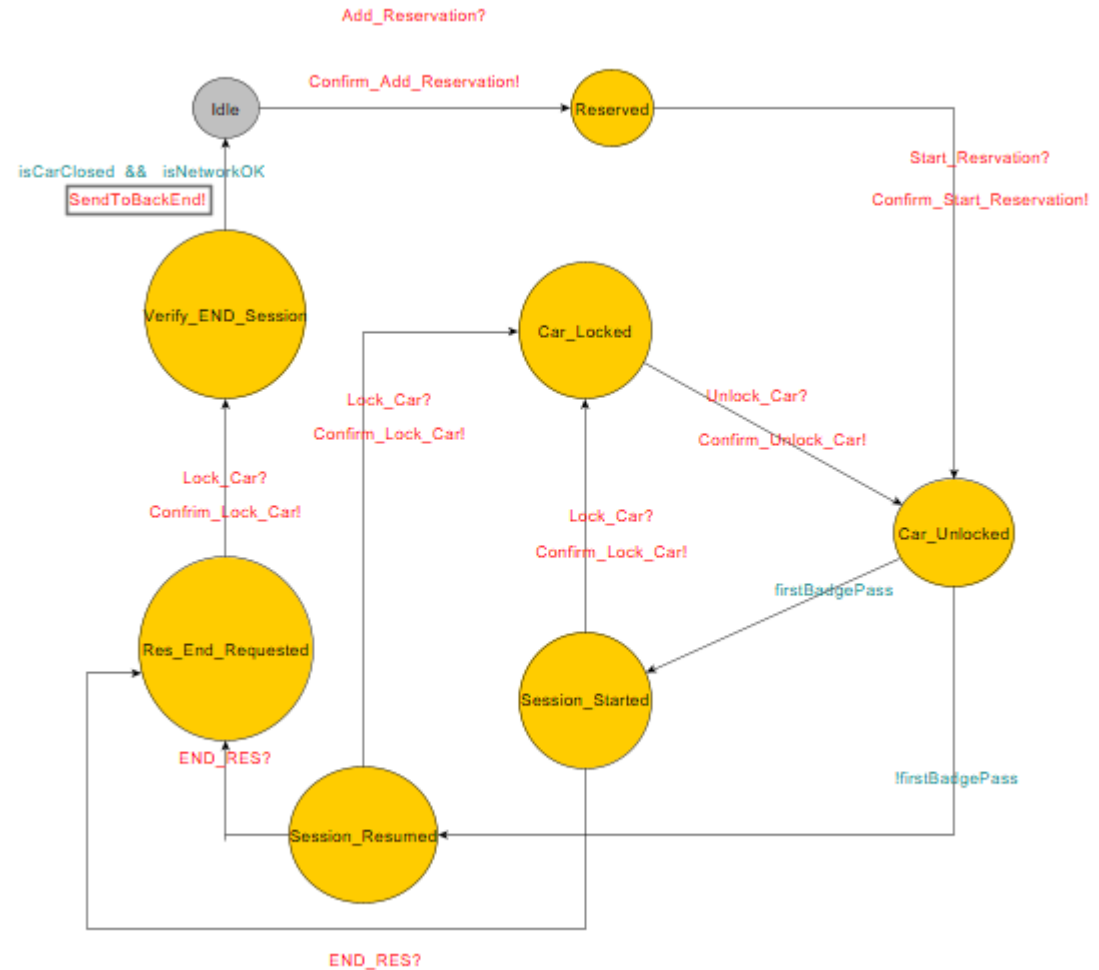
System Architecture

- Car
- BO
- RFID
- Attacker

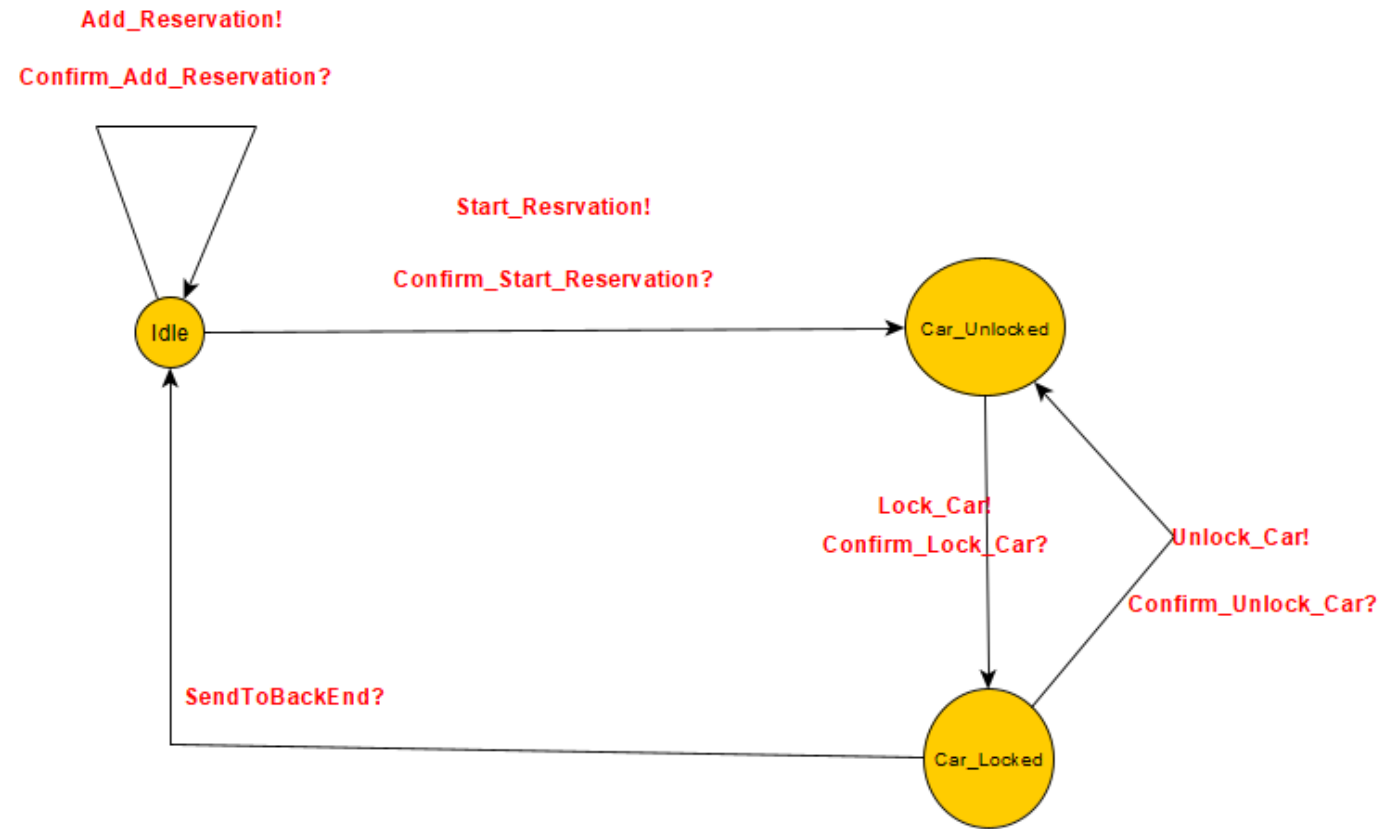


System Abstract Model

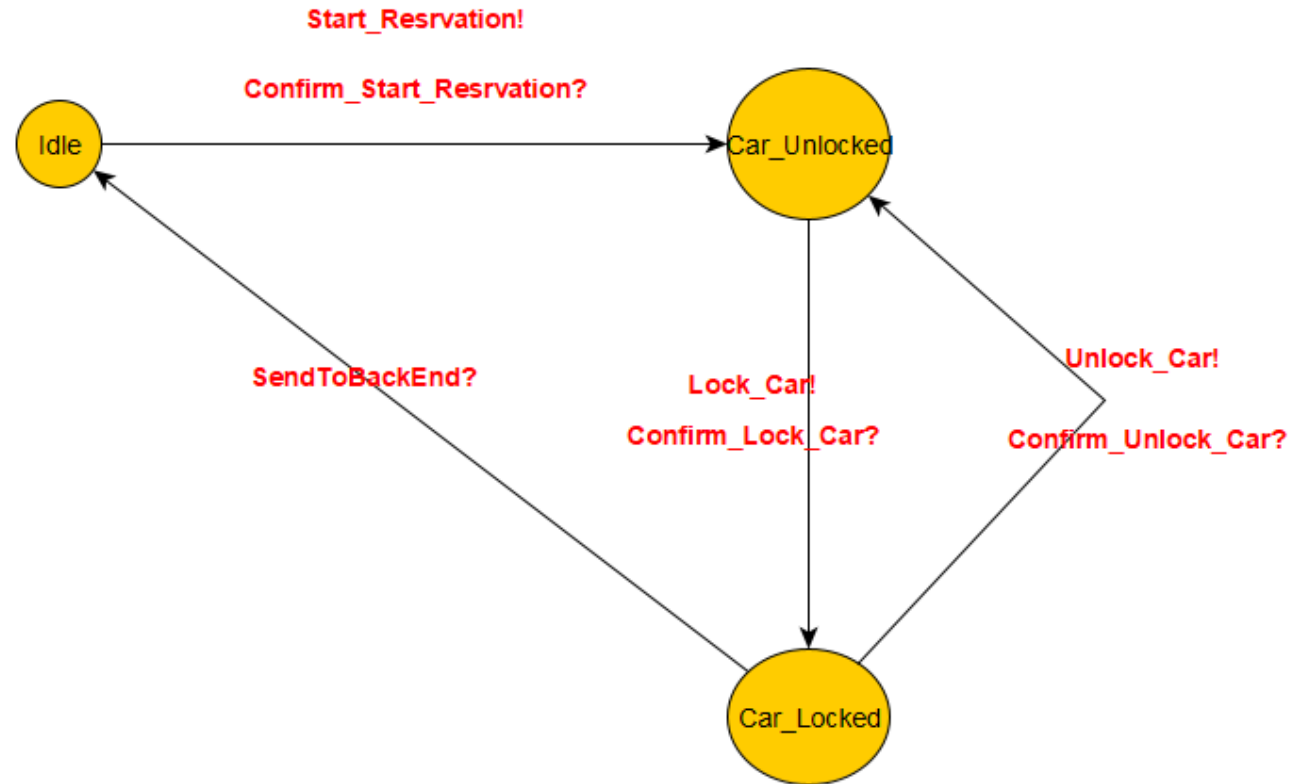
- Behavior:
- The system is idle at first
- The booking is added
- The booking is started
- The car in locked
(intermediate stop)
- The session is continued
- The session is ended
- Car locked



Back Office Model



RFID Model



Risk Analysis

- Risk analysis is the process of identifying and analyzing potential issues that could negatively impact key business initiatives or projects.
- Identify states and behaviors of the system that can be targets for the attacker
- Example:
 - States carLocked and carUnlocked
- Some of the targets are present in:
 - The car door lock
 - The booking state

Translate goals into properties

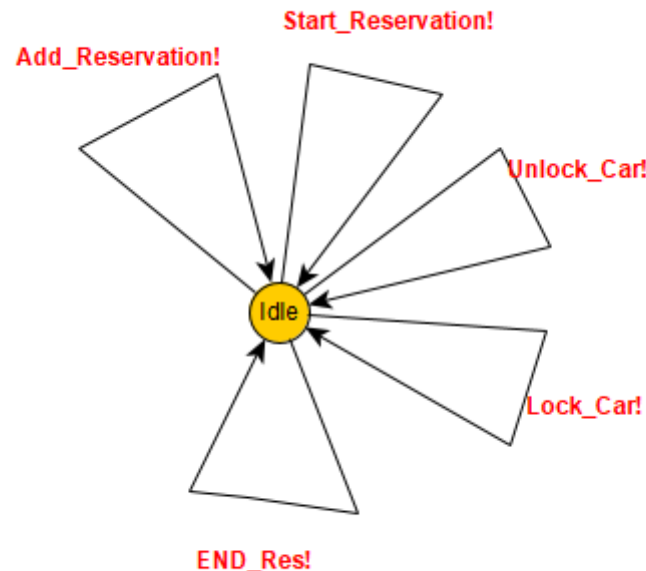
- The attacks or the goals of the attackers are translated into UPPAAL properties.
- There are two ways to define properties :
 - From the system's point of view - to verify that the system functions well
 - From the attacker's point of view – to verify that the attack was successful
- We used the system point of view
 - Make sure the system is resilient to the attacks

Translate goals into properties

- The properties are checked in the UPPAAL simulator
 - If the verification is successful → The attack failed
 - If the verification fails → The system is faulty or the attack was successful
 - A counter example is given

How to capture the attacker

- An automaton is added to the model to represent an attacker
- The attacker can use all capabilities at anytime.
- It can interfere in a scenario and cause it to block or to act in an unexpected way.



Results

- In some cases, the attacker is able to keep the vehicle open or closed and blocks the system
 - Service Availability
 - Once the car is unlocked, at some point in the future, the system goes back to idle and the car can be booked again.
 - If the car is locked, it will eventually be unlocked

Conclusion

- Reading the source code and the scenarios
- Agreeing on a representation
- Risk analysis feedback
 - We were able to detect some risks on the system
 - Car still open after the session ends
 - Car is locked and can not be unlocked
- UPPAAL was used only for feasibility analysis and prototyping
- Further work:
 - Another language or tool is to be provided due to UPPAAL limitations

References

- [1] Dean C. Wardell et al. “A Method for Revealing and Addressing Security Vulnerabilities in Cyber-physical Systems by Modeling Malicious Agent Interactions with Formal Verification”. In: *Procedia Computer Science* 95 (2016). Complex Adaptive Systems Los Angeles, CA November 2-4, 2016, pp. 24–31. issn: 1877-0509. doi: <https://doi.org/10.1016/j.procs.2016.09.289>. url: <http://www.sciencedirect.com/science/article/pii/S1877050916324619>
- [2] J. C. Mitchell and J. Bau. “Security Modeling and Analysis”. In: *IEEE Security & Privacy* 9.03 (May 2011), pp. 18–25. issn: 1558-4046. doi: [10.1109/MSP.2011.2](https://doi.org/10.1109/MSP.2011.2).
- [3] W. Yu et al. “Modeling and Verification of Online Shopping Business Processes by Considering Malicious Behavior Patterns”. In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2016), pp. 647–662