Reactive Security for Smart Grids Using Models@run.time-Based Simulation and Reasoning

Thomas Hartmann, Francois Fouquet, Jacques Klein, Gregory Nain, and Yves Le Traon

Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
{thomas.hartmann,francois.fouquet,jacques.klein,gregory.nain,
yves.letraon}@uni.lu
http://wwwen.uni.lu/snt

Abstract. Smart grids leverage modern information and communication technology to offer new perspectives to electricity consumers, producers, and distributors. However, these new possibilities also increase the complexity of the grid and make it more prone to failures. Moreover, new advanced features like remotely disconnecting meters create new vulnerabilities and make smart grids an attractive target for cyber attackers. We claim that, due to the nature of smart grids, unforeseen attacks and failures cannot be effectively countered relying solely on proactive security techniques. We believe that a reactive and corrective approach can offer a long-term solution and is able to both minimize the impact of attacks and to deal with unforeseen failures. In this paper we present a novel approach combining a Models@run.time-based simulation and reasoning engine with reactive security techniques to intelligently monitor and continuously adapt the smart grid to varying conditions in near real-time.

Keywords: Models@run.time, Reactive security, Reasoning engine, Smart grid, Model-driven engineering, Meta-modeling

1 Introduction

The vision of the smart grid promises to significantly increase the efficiency and reliability of the electricity grid and to seamlessly integrate micro generations and renewable energies. New services for electricity consumers, producers, and distributors will be created. One big step to turn this vision into reality is to use modern ICT to enable a two-way communication between customer devices and smart grid providers. On the one hand this facilitates advanced new features like remotely reading usage information from a meter or controlling devices through remote commands. On the other hand these new abilities make the smart grid more complex, making it inevitably more prone to failures, and more vulnerable to attacks. This introduces new challenges. Moreover, advanced features like

remotely disconnecting smart meters, makes the smart grid a valuable target for cyber attackers. Exploited vulnerabilities can result in the takeover of devices by an attacker, which can subsequently lead to serious crises as city blackouts. In particular, with a view to the rising cybercrime and given the importance of the electricity grid, it is essential to effectively protect it against attacks and failures.

Considering the complexity of smart grids and the fact that security techniques must dynamically evolve and improve over time to face future attacks and failures, we claim that proactive security techniques (like encryption, network-, and protocol security), although very useful, are not sufficient as a stand-alone approach. Instead, it must be anticipated that not all attacks and failures can be successfully prevented using proactive measures. We believe that, besides proactive security measures, a reactive and corrective security approach for smart grids is essential for at least two main reasons. First, it allows to deal with attacks and failures by monitoring and continuously adapting the smart grid to varying conditions like attacks, failures, and potential dangers —which together we refer to as *events*— in near real-time. Second, reactive security techniques allow to minimize the global impact of successful local attacks and failures. In this paper we present a novel approach combining a Models@run.time-based reasoning engine with reactive security techniques for smart grids. We mainly want to address security issues related to the stability and availability of the smart grid. By using an abstract model of state and behaviour of physical smart grid elements, a reasoning engine can simulate and explore potential actions on how to react to an event. For example, when an intrusion into a smart meter is detected, the reasoning engine could react by remotely deactivating the communication module of this smart meter to isolate it in order to avoid cascading failures (like reading potentially corrupted data from it). The models are used at runtime to monitor the smart grid with the intention of filling the gap between software models and the physical grid. Based on the Models@run.time paradigm the reasoning engine can simulate, explore, and evaluate different protection actions and their impacts in near real-time before the most appropriate ones (to secure and stabilize the grid) can be selected and applied to the real system.

The rest of this paper is organized as follows. Section 2 briefly introduces the background of this work: smart grids, Models@run.time, and reactive security. Section 3 details our Models@run.time-based simulation and reasoning engine and section 4 presents numbers from a real implementation of this approach. The related work is discussed in section 5. Finally, section 6 gives an outlook on future work before this paper concludes in section 7.

2 Background

2.1 Smart Grid

Today's electricity grid was designed for the demand of the 20th century where power generation was centralized and electricity was delivered from utilities to customers in a strictly one-way direction. This changes with the integration of

micro generations and renewable energies where electricity can be exchanged in both directions. Energy produced from private windmills, for example, can be sold to providers in times of high demand. Furthermore, electric vehicles could help to balance load by delaying their charge cycles or even transferring electricity back to the grid in peak times, as proposed in [21] and [10]. Modern ICT is applied to automate and control the electricity grid by enabling a two-way communication between customer devices and grid providers. This makes it possible to remotely read (consumption) data from meters and, what is more important, send commands to devices. This modernization of the electricity grid to meet the demands of the 21th century and especially it's distributed control ability is referred to as the *smart grid future* by Farhangi [15]. Bruno *et al.* [7] propose that a distributed control of smart grids can significantly improve its stability by locally smoothing the energy consumption. Among smart grid devices, smart meters are the cornerstones of the new infrastructure. While their initial task was mainly *automated meter reading* (AMR) [15], in future scenarios they tend to become highly interconnected and control other devices —like gas meters and micro generation devices— to build a so called *advanced metering infrastructure* (AMI) [15]. Electricity grids are typically controlled by SCADA (Supervisory Control and Data Acquisition) systems which control electricity production and delivery in real-time. These systems ensure the global stability of the grid by performing dynamic load balancing of electricity production, depending on customer consumption. SCADA systems have strong constraints concerning latency to ensure resilience of the grid in case of over-usage, as described by Aim *et al.* [1]. A challenge when designing smart grid infrastructures [7] is the coordination of SCADA systems and new communication networks across smart meters. SCADA systems typically focus on electricity production and delivery management, while smart meters and the smart grid network focuses on local consumption optimization and management.

2.2 Models@run.time

The smart grid aims to become a self-adaptive and self-healing system. Such systems usually need to analyze their surrounding environment and internal state in order to continuously adapt themselves to varying conditions. Therefore, building an appropriate abstraction model, which reflects the current context of the system is of key importance. Over the past few years, an emerging paradigm called *Models@run.time* [6], [27] proposes to use models both at design and runtime in order to support self-adaptive and intelligent systems. At design-time, following the Model-Driven Engineering (MDE) paradigm [22], models support the design and implementation of the system. The same (or similar) models are then embedded at runtime in order to support the reasoning processes of self-adaptive and intelligent systems. The idea behind this is that models offer a *simpler, safer and cheaper* [30] means to reason. In addition to the static structure of a system it is also possible to include the dynamic behavior in the model [29] to create a so-called executable model. The dynamic behavior of a system can be expressed using several paradigms such as stochastic queuing theory [36] or finite state

automata [9]. State machines [32] are a well known semantic to express behavior in terms of states and transitions, which are triggered in reaction to events.

2.3 Reactive Security

Reactive security follows the idea that it is nearly impossible to proactively prevent all kind of possible attacks and failures. Instead, it must be taken into account that techniques used for cybercrime will continuously evolve and — in some cases— outperform previously installed proactive security techniques. Whereas proactive security has to predict future attacks, which is very hard, reactive security has to minimize the effect of attacks, *e.g.* by learning from the past, which is in many cases easier [2]. Reactive security approaches aim to prevent attacks by intelligently monitoring and early reacting to changes [31], or to minimize the global effect of successful attacks. For example, a denial of service (DoS) or distributed denial of service (DDoS) attack on a smart grid concentrator can be countered reactively by dynamically putting the attackers on blacklists.

3 Models@run.time-Based Simulation and Reasoning

3.1 General Approach

We propose a reactive security approach for smart grids. As smart grids are becoming more and more complex and techniques used for cybercrime will continuously evolve, we believe that effective protection mechanisms for smart grids must be able to react dynamically to successfully counter attacks and failures. Thus smart grids need the ability to continuously adapt themselves in order to react to various events. Therefore, they need to analyze their surrounding environment and internal state. We suggest using an abstract model of state and behaviour of physical smart grid components. Based on the Models@run.time paradigm this model reflects the internal state of the smart grid and is continuously updated with state information of the physical smart grid components. It is a common approach for self-adaptive systems to regularly sample and store the context of the system in order to back the reasoning algorithms up with historical data. That the Models@run.time approach is suitable for large distributed and self-adaptive systems has, for example, been shown in [27, 28]. The model is used at runtime to simulate and explore different actions to react to events in real-time. The model, which represents the state of the smart grid at the time of an event, can be cloned to simulate several reactions on independent models. This happens in near real-time. Based on results of different simulations, appropriate corrective actions can be derived and either suggested for manual application or automatically applied to the real system. The basic concept of our Models@run.time-based reasoning engine is illustrated in figure 1.

The goal is twofold: first to prevent attacks and failures by intelligently monitoring and continuously adjusting the smart grid and second to minimize the

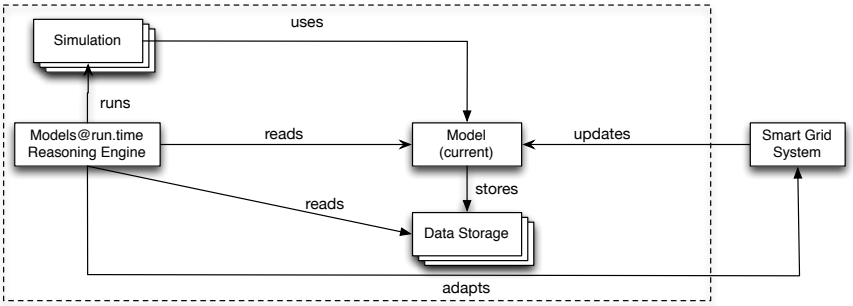

Fig. 1. Models@run.time-based reasoning engine

global impact of (successful) attacks and failures. The key thing to note is that our reasoning engine works reactive. It always searches for appropriate counter-measures to dynamically react to an event. Typical events, which we target to counter with our approach, are:

- **Intrusion detection**: It doesn't matter whether the attack is detected by our reasoning engine or by another tool (specific attacks might be best detected in specific layers by specific tools). The proposed reasoning engine is particularly suitable to detect attacks by identifying deviations from normal behaviour, usually called *anomaly-based detection* [4]. For instance, by continuously monitoring state information, state flapping (state oscillating between two configurations) of devices can be detected. By continuously monitoring network traffic and checking against the expected traffic, attacks like flooding an entity (smart meter, concentrator) with messages can be detected. Also, by monitoring sender and receiver of messages, suspicious messages can be identified. For example, a meter request to send consumption data to a device to which the corresponding meter is not logically registered to, is suspicious. Another example for a potentially suspicious behaviour would be if a large number of smart meters in one area receive the command to shut the electricity down (even if the command is send by a trusted entity). In addition to anomaly-based intrusion detection the proposed reasoning engine can be feed with data from other tools, *e.g. specification-based intrusion detection* systems [5], [4] or *signature-based intrusion detection* systems [4]. All this data can be aggregated and analyzed by the reasoning engine and used to derive appropriate counter-measures. For example, the communication module of affected smart grid devices, like smart meters, can be remotely deactivated to isolate it in order to avoid cascading failures. Another strategy would be to blacklist the device so that other devices no longer exchange messages with potentially corrupted devices.
- **Electrical load**: Based on the current load, combined with historical data (*e.g.* last 20 Monday evenings) the reasoning engine can predict how the load will likely develop and if a critical limit could be exceed. Besides creating

alarm messages, this information can be used to delay/encourage electric cars to charge. Similarly, the voltage level can be monitored and predicted to decide if local production units must be connected/disconnected.

- **Communication network traffic**: Based on the knowledge of the used protocols, the network technologies, and historical data the reasoning engine can simulate and evaluate the number of messages required for an action and thus predict the network load. This information can be used to delay actions (like sending consumption data) to keep the overall network load below a critical value.
- **DoS/DDoS**: DoS and DDoS attacks can be detected by the reasoning engine, *e.g.* by monitoring the network traffic and state information of attacked components. Potential attackers could be automatically added to a blacklist. Or, in case of an affected concentrator the reasoning engine could deactivate it and initiate that connected smart meters reconnect to other concentrators.
- **Frequency of disturbances**: In complex and distributed systems, like smart grids, it is normal that from time to time minor disturbances (like meters are temporarily not reachable) occur. By monitoring disturbances over time the reasoning engine can detect an unusual high frequency, which can indicate security problems.
- **State changes**: A frequent change of state (like repeated unsuccessful register intents) often indicates security issues of smart meters. Such problems can be detected by the reasoning engine and can for instance cause to deactivate the communication module of the concerning meters.

The counter-measures found by the reasoning engine can be either automatically applied to the real system or only proposed for a manual validation. It is conceivable that counter-measures first must be manually validated and then, based on this validation, the reasoning engine can automatically improve itself by learning from these decisions. For example, if a counter-measure for a certain event has been manually validated and confirmed for automatic execution, the reasoning engine can apply this solution in the future automatically. If a manual validation indicates that the proposed counter-measure is not appropriate, the reasoning engine can learn which counter-measure should be used instead (i.e. the counter-measure which is manually selected instead the one automatically proposed). Also, for reasons of safety, counter-measures with a very big impact on the grid may be only applied after a manual validation and confirmation.

One risk of our Models@run.time-based simulation and reasoning engine is that the model could not correctly reflect the state of the real system. This could for example be due to the fact that the model has not been updated since the last important state change of the smart grid system. This is known as *eventually consistent* [35]. In general, a model can always only reflect a partial view of a real system. This is a general problem that self-adaptive and intelligent systems face and has to be taken into account by the reasoning engine.

3.2 Smart Grid and Behaviour Model

Topology Model Our model for smart grids consists of different components: smart meters, repeaters, concentrators, SCADA systems, and a central control system and reflects how these components are connected. The basic structure of our model is inspired by the smart grid configuration currently deployed in Luxembourg [20], as we work in close collaboration with Creos Luxembourg S.A[1] on cyber security for smart grids. Figure 2 shows a simplified topology of the smart grid components described with our model. Each smart meter is connected to a concentrator, either directly or via one or several repeaters, and each concentrator in turn is connected to the central system. One or several SCADA systems are used to monitor and control the physical smart grid processes. The proposed

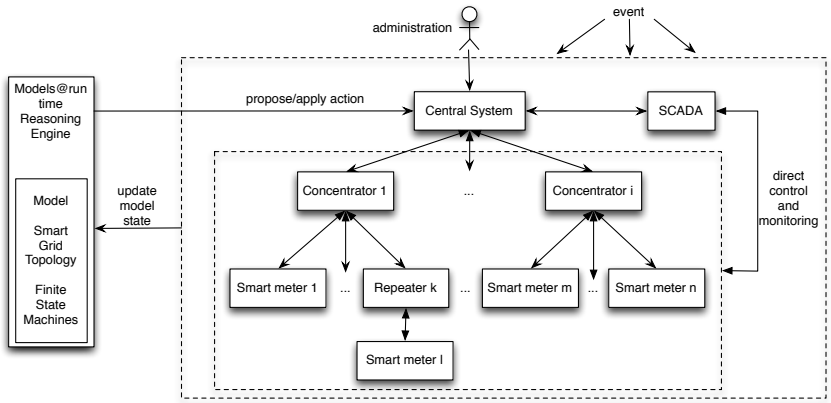

Fig. 2. Topology model

reasoning engine operates on top of a model representation of this structure.

FSM Model for Behaviour In order to model the behaviour of our structural smart grid components (smart meter, repeater, concentrator, central system, SCADA) we use Finite State Machines (FSM). The suitability of FSMs to model and simulate behaviour has been shown in [17, 32]. Each message sent to a component can be interpreted as an event for the corresponding FSM and can trigger a state change. Figure 3 shows a simplified representation how a typical smart grid process can be simulated using FSMs. For the sake of simplicity, all states which are not necessary for the example are omitted. It shows how a smart meter registers to a concentrator after starting up.

The initial state of each smart meter and concentrator is *inactive*. Lets now assume *concentrator 1* and *concentrator 2* are in state *active* and *smart meter 1*

[1] Creos Luxembourg S.A is the main grid operator in Luxembourg

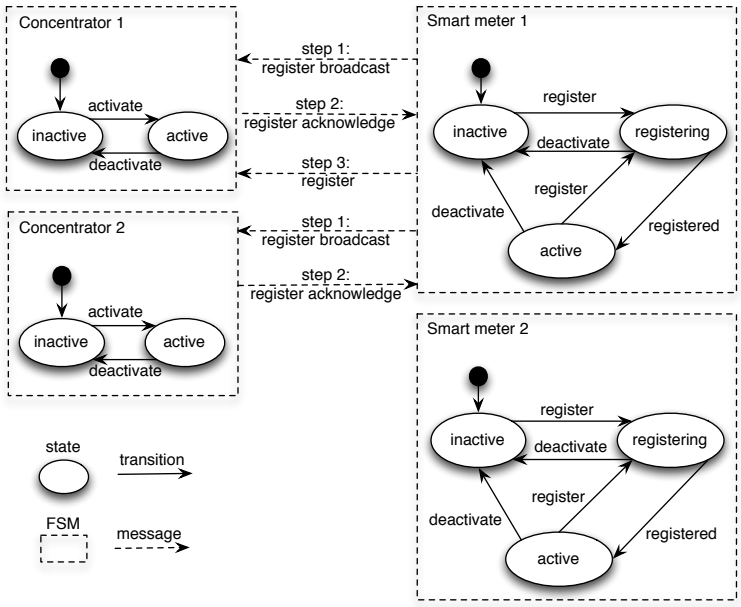

Fig. 3. FSM scenario for registering smart meters

and *smart meter 2* are in state *inactive*. When switching a smart meter on, it tries to register itself to a concentrator and enters the state *registering*. As illustrated in the figure, *smart meter 1* broadcasts its registering intent to all reachable concentrators (step 1). Upon receiving this message the reached concentrators send acknowledge or deny messages back to *smart meter 1* (step 2). Based on criteria such as signal strength and number of hops to the central system, *smart meter 1* decides to register to *concentrator 1*. It then sends a corresponding register confirmation back to *concentrator 1* (step 3). Depending on the real smart gird implementation (*e.g.* used protocols) this behaviour may vary. Again, the described protocol is inspired by the smart grid deployment in Luxembourg. By additionally taking the average size and payload of messages into account, even the impact of the network load can be simulated and evaluated.

This simple example illustrates how we simulate typical smart grid processes using FSMs to model the behavior of smart grid components. Another example is to simulate the effects of a deactivated communication link (or electricity link) of a concentrator; thus how many smart meters are affected, how long it will take until all of them are registered again, and so on.

3.3 Reasoning Engine Scope

Through observing and dynamically reacting our Models@run.time-based reasoning engine aims to support the smart grid to become self-adaptive and thus

self-healing. Intelligent and self-adaptive software systems need to analyze both their surrounding environments and their internal state in order to continuously adapt themselves to changing conditions. Therefore, building an appropriate model, to reflect the current context of such systems is of key importance. The model of our reasoning engine focuses on the basic state, structure, and behavior of physical smart grid components. This means that our model reflects if a smart meter is active or not, that a smart meter can receive commands, and that this may change the current state. It also covers knowledge of how these components are interconnected and thereby how certain actions cascade. For example the model contains the necessary information about which smart meters are affected if a concentrator fails. We do not intend to duplicate the complete runtime system, but intend to build an appropriate abstraction containing the relevant parts of the system. It aggregates and combines information collected from different layers. This means our model is not limited only to application oriented layers but can also take information of lower layers (*e.g.* data bases, network traffic) into account. However, it is not the goal of our model to reflect detailed physical processes like the control of electricity production and delivery management. Our smart grid model contains only knowledge which will be used by our reasoning engine to simulate and explore potential actions on how to react to attacks, failures, and potential problems (like local electrical overload) Moreover, we do not intend to replace any existing control systems, like SCADA systems, or security systems. Instead, our proposed reasoning engine complements such systems by aggregating their information to build an appropriate context model.

3.4 Searching Appropriate Counter-Measures

In order to derive appropriate counter-measures to face an event, our reasoning engine must be able to evaluate and compare different actions. Each action can potentially change the state of the model. The goal of the reasoning engine is to propose actions which lead to an *improvement* of the overall model state. First of all, this requires knowledge about what actions are applicable to face an event. This is domain knowledge and is in form of rules integrated in the knowledge base of our reasoning engine. Second, it requires evaluation functions for our model to compare different states.

Counter-Actions are the reactions of our reasoning engine in order to counter events. Examples for events and appropriate counter-actions are:

- Smart meter intrusion detection: deactivate communication module of the smart meter to isolate it and avoid additional damage.
- Disconnected smart meter (customers' electricity is off): send command to restart the electricity link.
- High/low local electric load level: delay/encourage charging of electric cars.
- High/low local voltage level: disconnect/connect local production units (where possible).

Evaluation Functions to evaluate a model we use a set of rules, which are part of the domain knowledge and added to the knowledge base of our reasoning engine. For instance, such a rule is that disconnected (electricity for a customer is down) smart meters are worse than connected ones or that the local electrical load should not exceed (or fall below) a certain value for a longer period of time. Further examples are: Smart meters which are regularly not reachable or have weak connections to their data concentrators lead to a worse state. This applies also for smart meters in error states and especially for smart meters where intrusions are detected. The combination of all rules allows to calculate an overall score for the model. This score in turn is used to compare model states. For example, a failure of a data concentrator leads to a decreasing model state. This failure affects all smart meters connected to this data concentrator since they can no longer communicate with the failed data concentrator, further decreasing the model state. One conceivable reaction could be to connect the smart meters to an alternative data concentrator. Each smart meter which is connected again to a data concentrator increases the model state. By comparing scores of models it can be evaluated if actions improve or downgrad a model state.

Selection the goal of the reasoning engine is to find (and select) appropriate counter-reactions to face an event. The procedure is as follows: from a set of possible counter-reactions (knowledge base), the reasoning engine simulates the (independent) application of different actions using the model and evaluates which actions are the most appropriate ones (leads to the best model state). It is important to notice that the simulation and selection of the counter-reactions happen in near real-time. As a first approach we implemented a greedy [33] algorithm. But other algorithms which are not limited to search local optimums but also consider steps before and after the current, might be far more useful and are subject to study in future work.

3.5 Scalability

Since smart grids can consist of a huge number of components the scalability of our approach is very important. Operations to navigate or manipulate our model at runtime must be very efficient in terms of time and space. Therefore, we are working on a Models@run.time framework, called Kevoree Modeling Framework [16], which is specifically designed for this purpose. It is also conceivable to split the smart grid model into sub-models and distribute the reasoning over multiple nodes. An appropriate strategy is to split the model accordingly to the topology of the grid, such as deploying one instance of the reasoning engine on each data concentrator. Each of these local reasoning engines can then monitor one part or region of the smart grid. The information of the local reasoning engines can be combined on a global level.

4 Case Study Luxembourg

4.1 Scenario

We have implemented a concrete smart gird model based on the approach discussed in section 3. The smart grid test deployment in Luxembourg, which is currently deployed by our industrial partner Creos S.A., is the template for our abstract model. The topology of our model primarily consists of: Smart meters, data concentrators, physical cables, consumption and production data, GPS location data of devices, logical communication connections, and routing tables. Overall, the concrete model of our case study includes around 250 nodes (smart meters, data concentrators), 30 physical cables, and 25.000 consumption data sets per day. We currently cross compile our model as well as our simulation and reasoning engine for the Java Virtual Machine [19] and for JavaScript [11]. For our scenario the model, simulation, and reasoning engine are small and efficient enough to be executed entirely in a web browser running on a standard laptop (MacBook Pro i5 2.4 Ghz, 16 GB RAM).

4.2 Example: Malicious Shutdown Commands

In this example we implemented a detection and protection reaction for potentially malicious shutdown commands. The reasoning engine monitors the state of entities (smart meters, repeaters, concentrators, central system) and detects if a striking number of entities (more than 10% in a region) are remotely shutdown within a certain time range. The sender of the malicious shutdown commands is added to a blacklist by the reasoning engine to avoid that additional entities in this region will be affected. We implemented a greedy algorithm to detect entities, which are shutdown, and to automatically start them again. A corresponding model evaluation function rates an entity, which is shutdown worse than an entity which is started and the counter-measure is to restart and connect the concerned entity again. If an entity cannot be connected again (*e.g.* a smart meter can only be connected if a repeater or concentrator is available) the algorithm proceeds and tries the next entity and so forth. The algorithm stops if either all entities are started and connected again, or if none of the remaining entities, which are shutdown can be started and connected. The algorithm is executed in the range of milliseconds to a few seconds (in the worst case that all meters and concentrators are shutdown). This is what we consider near real-time. Since it is conceivable that a large number of entities are intentionally shutdown, *e.g.* for maintenance, it is possible to deactivate this detection (or only the reaction) for shutdown commands in the reasoning engine. This concrete example demonstrates the feasibility of near real-time reactive security at the range of a city.

4.3 Example: Electric Load Prediction

Based on our discussed approach and model we have implemented an electric load prediction. The idea is to predict if the electric load value in a region —

in our case study around 50 smart meters— will likely exceed a critical value. If that is the case, the maximum allowed consumption for the corresponding meters can temporally be reduced by our reasoning engine to avoid an electric overload. Since an electrical grid usually can maintain an overload for a few seconds or minutes [8] the reasoning engine has to react within this time range. The reasoning process consists of an electrical load prediction for a specific point of the grid (one smart meter). Both the current electric load and past values (the consumption history of one month) at this meter as well as from the surrounding meters are taken into consideration. This prediction is continuously performed on a few dozen grid points and a linear regression of the average electric load values of the meters (over a certain period of time) is computed. The complete reasoning process for our case study is computed within a time range of a few seconds. Again, the example demonstrates the feasibility of near real-time reactions of our reasoning engine.

5 Related Work

Cyber security is a major concern of smart grids. Therefore, a lot of work is paying attention to this topic. An analysis of security threats and challenges in smart grids can be found in [26], [23], [13]. This work indicates the importance of smart grid security and privacy and shows significant weaknesses and attack points of smart grids. Many other authors like [4], [34], [12] focus their studies on smart meter security: intrusion detection systems, redundant meter reading, and privacy. Others, like Zhao et al. [37] focus their work on cryptography and a secure authenticated key exchange for smart grids. The above-mentioned work discusses important proactive security measures to improve security and privacy in smart grids. Unlike this work, our approach focuses on reactive security techniques, which we believe can complement proactive security measures to improve security in smart grids. In particular we intend to improve the self-healing aspect of smart grids. An interesting approach based on game-theoretic models for reactive security in general (not connected to smart grids in specific) is presented in [2]. Learning based game-theoretic techniques could be interesting for our reasoning engine to find appropriate counter-measures to face events and can be explored in future work. Godfrey et al. [18] suggest to use simulation techniques for an analysis of complex smart grid control schemes. This work focuses mainly on the exact simulation (incl. latency) of control messages. Kundur et al. [24] also use simulation techniques to study the potential severity of physical impacts of cyber attacks. A combination of hardware and software for a detailed simulation of a smart grid is presented in [25]. Their so-called *SmartGridLab* aims to provide researchers with a platform to conveniently and efficiently compare different smart grid designs. Just as the above-mentioned work, our approach suggests to use simulation techniques. In contrary to this work, we do not intend to simulate a complete smart grid one-to-one. Instead, we aim to dynamically counter attacks, failures, and potential dangers by simulating and evaluating different protection reactions in near real-time. Therefore,

we use a model abstraction of a smart grid at runtime to be able to perform different simulations in real-time and finally to decide how to react. Baumeister [3] presents an exhaustive literature review specifically on smart grid cyber security. A more general survey on smart grid technologies, which also includes a review of smart grid cyber security literature, can be found in [14]. To the best of our knowledge there is no related work combining Models@run.time techniques and a reasoning engine to a reactive security approach for smart grids.

6 Future Work

In future work we will explore more complex algorithms, like genetic [33] or game-theoretic [2] ones, to find appropriate counter-measures to face events. Especially algorithms, which are not limited to search local optimums but also consider steps before and after the current, will be subject to study. Another approach we would like to explore is to use techniques and methods from artificial intelligence in order to learn from previous situations and thus automatically improve our reasoning engine. Furthermore, several functions to evaluate the state of our model will be investigated. We will implement and simulate more complex and realistic use cases to continuously evaluate and improve our approach.

7 Conclusion

Ensuring a satisfactory level of security for smart grids is critical and challenging. We introduced a reactive security approach to face this challenge by both 1) reasoning at high level to take the right decision and 2) reacting in near real-time. Unlike many other works, which mainly focus on proactive security techniques, our approach is completely reactive. Given the complexity of smart grids, we believe that a reactive security approach is essential to either entirely prevent, or at least to minimize the global impact of (successful) attacks and failures. The novelty of our approach is the combination of a Models@run.time-based reasoning engine with reactive security techniques to react in near real-time. Using a lightweight model representation of the physical smart grid elements, our approach allows to simulate and evaluate different counter-measures in real-time in order to dynamically protect the smart grid with the most appropriate ones. We presented an abstract model of the physical smart grid elements and used FSMs to model the behaviour of the elements. We believe that using Models@run.time together with a reasoning engine can introduce a new approach of reactive security for smart grids and can help to develop the electricity grid of today into a more secure and adaptive smart grid of tomorrow that can verify and supervise itself.

Acknowledgments

The research leading to this publication is supported by the National Research Fund Luxembourg (grant 6816126) and Creos Luxembourg S.A. under the SnT-Creos partnership program.

References

1. S.M. Amin and B.F. Wollenberg. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.
2. A. Barth, B. I. P. Rubinstein, M. Sundararajan, J. C. Mitchell, D. X. Song, and P. L. Bartlett. A learning-based approach to reactive security. *CoRR*, abs/0912.1155, 2009.
3. Todd Baumeister. Literature Review on Smart Grid Cyber Security. Technical Report CSDL-10-10, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, December 2010.
4. R. Berthier, W.H. Sanders, and H. Khurana. Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In *SmartGridComm*, pages 350–355, 2010.
5. Robin Berthier and William H. Sanders. Specification-based intrusion detection for advanced metering infrastructures. In Leon Alkalai, Timothy Tsai, and Tomohiro Yoneda, editors, *PRDC*, pages 184–193. IEEE Computer Society, 2011.
6. G. Blair, N. Bencomo, and R.B. France. Models@ run.time. *Computer*, 42(10):22–27, 2009.
7. S. Bruno, S. Lamonaca, M.L. Scala, G. Rotondo, and U. Stecchi. Load control through smart-metering on distribution networks. In *PowerTech, 2009 IEEE Bucharest*, pages 1–8, 2009.
8. J. C. Cepeda, D.O. Ramirez, and D.G. Colome. Probabilistic-based overload estimation for real-time smart grid vulnerability assessment. In *Transmission and Distribution: Latin America Conference and Exposition (T D-LA), 2012 Sixth IEEE/PES*, pages 1–8, Sept 2012.
9. Axel Cleeremans, David Servan-Schreiber, and James L McClelland. Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381, 1989.
10. S. Deilami, A.S. Masoum, P.S. Moses, and M. A S Masoum. Real-time coordination of plug-in electric vehicle charging in smart grids to minimize power losses and improve voltage profile. *Smart Grid, IEEE Transactions on*, 2(3):456–467, 2011.
11. ECMA International. *Standard ECMA-262 - ECMAScript Language Specification*. 5.1 edition, June 2011.
12. C. Efthymiou and G. Kalogridis. Smart grid privacy via anonymization of smart metering data. In *SmartGridComm*, pages 238–243, 2010.
13. G. N. Ericsson. Cyber Security and Power System Communication–Essential Parts of a Smart Grid Infrastructure. *IEEE Transactions on Power Delivery*, 25(3):1501–1507, July 2010.
14. Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart grid—the new and improved power grid: A survey. *Communications Surveys Tutorials, IEEE*, 14(4):944–980, 2012.
15. H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28, 2010.

16. F. Fouquet, G. Nain, B. Morin, E. Daubert, O. Barais, N. Plouzeau, and J. Jzquel. An eclipse modelling framework alternative to meet the models@runtime requirements. In R. B. France, J. Kazmeier, R. Breu, and C. Atkinson, editors, *MoDELS*, volume 7590 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2012.
17. M. Fowler. *Domain Specific Languages*. Addison-Wesley Prof., 1st edition, 2010.
18. T. Godfrey, S. Mullen, R.C. Dugan, C. Rodine, D.W. Griffith, and N. Golmie. Modeling smart grid applications with co-simulation. In *SmartGridComm*, pages 291–296, 2010.
19. James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley. *The Java Language Specification*. California, USA, java se 7 edition, February 2012.
20. Robert Graglia. Smart grid luxembourg, September 2013.
21. Christophe Guille and George Gross. A conceptual framework for the vehicle-to-grid (V2G) implementation. *Energy Policy*, 37(11):4379–4390, 2009.
22. Stuart Kent. Model driven engineering. In *IFM*, 2002.
23. Himanshu Khurana, Mark Hadley, Ning Lu, and Deborah A. Frincke. Smart-grid security issues. *IEEE Security & Privacy*, 8(1):81–85, 2010.
24. D. Kundur, Xianyong Feng, Shan Liu, T. Zourntos, and K.L. Butler-Purry. Towards a framework for cyber attack impact analysis of the electric smart grid. In *SmartGridComm*, pages 244–249, 2010.
25. Gang Lu, D. De, and Wen-Zhan Song. Smartgridlab: A laboratory-based smart grid testbed. In *SmartGridComm*, pages 143–148, 2010.
26. Patrick McDaniel and Stephen McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security and Privacy*, 7(3):75–77, May 2009.
27. B. Morin, O. Barais, J. Jezequel, F. Fleurey, and A. Solberg. Models@ run.time to support dynamic adaptation. *Computer*, 42(10):44–51, 2009.
28. Brice Morin, Olivier Barais, Gregory Nain, and Jean-Marc Jezequel. Taming dynamically adaptive systems using models and aspects. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 122–132, Washington, DC, USA, 2009. IEEE Computer Society.
29. Pierre-Alain Muller, Franck Fleurey, and Jean-Marc Jézéquel. Weaving executability into object-oriented meta-languages. In *Model Driven Engineering Languages and Systems*, pages 264–278. Springer, 2005.
30. Jeff Rothenberg, Lawrence E. Widman, Kenneth A. Loparo, and Norman R. Nielsen. The nature of modeling. In *Artificial Intelligence, Simulation and Modeling*, 1989.
31. Brent R. Rowe and Michael P. Gallaher. Private sector cyber security investment: An empirical analysis. In *WEIS*, 2006.
32. Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.
33. Haleh Vafaie and Ibrahim F. Imam. I.: feature selection methods: Genetic algorithms vs greedy-like search. In *In: Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, 1994.
34. D.P. Varodayan and G.X. Gao. Redundant metering for integrity with information-theoretic confidentiality. In *SmartGridComm*, pages 345–349, 2010.
35. Werner Vogels. Eventually consistent. *Queue*, 6(6):14–19, October 2008.
36. R. W. Wolf. Stochastic modeling and the theory of queues. *Printice Hall*, 1989.
37. Fangming Zhao, Yoshikazu Hanatani, Yuichi Komano, Ben Smyth, Satoshi Ito, and Toru Kambayashi. Secure authenticated key exchange with revocation for smart grid. *Innovative Smart Grid Technologies, IEEE PES*, 0:1–8, 2012.