On the Role of Primary and Secondary Assets in Adaptive Security: An Application in Smart Grids

Liliana Pasquale*, Mazeiar Salehie*, Raian Ali‡, Inah Omoronyia*, and Bashar Nuseibeh*†
* *Lero- Irish Software Engineering Research Centre, Limerick, Ireland*
{*Liliana.Pasquale, Mazeiar.Salehie, Inah.Omoronyia, Bashar.Nuseibeh*}*@lero.ie*
† *Department of computing, Open University, Milton Keynes, UK*
‡ *Bournemouth University, UK, rali@bournemouth.ac.uk*

***Abstract*—Adaptive security aims to protect valuable assets managed by a system, by applying a varying set of security controls. Engineering adaptive security is not an easy task. A set of effective security countermeasures should be identified. These countermeasures should not only be applied to (primary) assets that customers desire to protect, but also to other (secondary) assets that can be exploited by attackers to harm the primary assets. Another challenge arises when assets vary dynamically at runtime. To accommodate these variabilities, it is necessary to monitor changes in assets, and apply the most appropriate countermeasures at runtime. The paper provides three main contributions for engineering adaptive security. First, it proposes a modeling notation to represent primary and secondary assets, along with their variability. Second, it describes how to use the extended models in engineering security requirements and designing required monitoring functions. Third, the paper illustrates our approach through a set of adaptive security scenarios in the customer domain of a smart grid. We suggest that modeling secondary assets aids the deployment of countermeasures, and, in combination with a representation of assets variability, facilitates the design of monitoring functions.**

***Keywords*-Adaptive security; Smart grid; Assets; Adaptive software**

I. Introduction

Software systems are becoming increasingly dynamic and should adapt at runtime to cope with the variability of their requirements or the environment in which they operate. A critical part of adaptation concerns protecting critical assets of the system from variable threats. Changes in the protected assets and their corresponding security goals should also be accommodated. In this paper, we address proactive adaptive security [1], which tries to prevent attacks by adjusting security requirements and countermeasures. Engineering security requirements possesses a number of challenges. In many cases applying security countermeasures on the *primary assets* that have to be protected is not enough. An attacker may target other *secondary assets*, which are related to and can be used to harm their corresponding primary asset(s). For example, in smart grids, energy related information (secondary assets) may be exploited to cause appliances outage (primary assets). Secondary assets are important in engineering effective security requirements and countermeasures, since they enable us to identify additional software components and execution points where security countermeasures should be instrumented. For example, encrypting energy-related information may improve the availability of the associated appliances. Furthermore, assets in the protection boundary of the system can vary: the value of assets can change, new assets can be added or existing assets can be removed to/from the system. To accommodate these variabilities and protect assets continuously, it is necessary to monitor assets changes and adjust countermeasures appropriately at runtime. In our earlier work [1], we investigated the decision making mechanism necessary to reconfigure countermeasures at runtime. This paper focuses on how to design and apply the necessary monitoring functions for primary and secondary assets.

The paper provides three main contributions towards engineering adaptive security. First, it proposes a modeling notation to represent primary and secondary assets, along with their variability. To this end, we extend the goal and asset models proposed in our earlier work [1]. Second, we describe how to use the extended models in engineering security requirements and designing required monitoring functions. Monitoring functions aim to update the asset model at runtime. Third, we exemplify our approach through a set of adaptive security scenarios in the customer domain of smart grids. We suggest that due to criticality and variety of assets in the smart grid domain, it is suitable for investigating adaptive security.

The paper is organized as follows. Section 2 reviews security concerns in smart grids and discusses two adaptation scenarios. Section 3 describes the asset model and its link to security goals. Section 4 explains how the asset and goal models facilitate the deployment of security countermeasures and the design of asset monitoring functions. Section 5 reviews related work and, Section 6 concludes the paper.

II. Smart Grid as an Adaptive Security Domain

This section explains the main security concerns of smart grids, and provides two adaptive security scenarios.

A. Security in Smart Grids

A smart grid [2] is "a digitally enabled electrical grid that gathers, distributes, and acts on information about the behavior of all participants (suppliers and consumers) in order to improve the efficiency, importance, reliability, economics, and sustainability of electricity services." Among the different domains that compose the architecture of a smart grid we focus on the customer domain, which is the most appealing for adaptive security purposes. A simplified architecture of the customer domain is sketched in Figure 1[1]. Metering data report which appliances have been used, and when and how much energy they consumed. Metering data are transmitted to the internal EMS (Energy Management System), and to other external domains (e.g., supplier domain) through the HAN (Home Area Network) Gateway. The EMS transmits operational control data to the internal appliances. It also reads metering data to optimize the load on the appliances and the energy consumption, especially during the expensive high-peak periods.

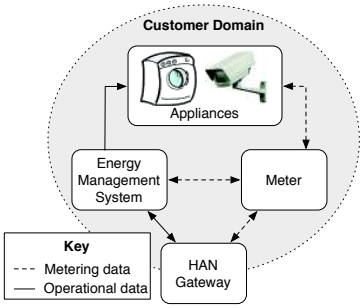

Figure 1. A simplified architecture of the customer domain.

The customer domain comprises several assets: energy, meter, EMS, metering data (including energy consumption), operational control data, audit trail, power-enabled appliances, and assets protected by these appliances. These assets can be primary or secondary. An asset is primary, when it is the ultimate object of value to the customer. However, sometimes a primary asset is not directly accessible to an attacker, who has to turn her/his attention to another secondary asset to achieve her/his primary objectives. For example, energy can be a primary asset since a potential attacker may gain economic advantages by stealing it. However, energy cannot always be directly stolen by an attacker, who may have to falsify metering data describing energy consumption, which are secondary assets in this case. Similarly, a building (with valuable objects inside) is a primary asset, since an attacker may wish to steal its goods. So, the attacker may try to deactivate a CCTV camera that controls the building, which is a secondary asset in this case. Secondary assets are linked to primary assets, since they may carry information about primary assets (e.g., metering data are primary assets for energy), or control them (e.g., a CCTV camera is a secondary asset for the building that is controlled).

[1]A complete architectural description of the customer domain can be found at http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7628.

B. Adaptive Security Scenarios

This section provides two adaptive security scenarios for a smart grid. These scenarios deal respectively with the variability of primary and secondary assets.

Scenario 1. Variability of primary assets may affect the value of related secondary assets and trigger changes in the security requirements and countermeasures. Changes in the asset value can be intrinsic (i.e., determined by the asset itself). For instance, the value of energy depends on its daily consumption, and changes in this value directly affects the value of other secondary assets related to energy (e.g., metering data and the meter itself). Consequently, in case the value of energy is low, performing authentication at the meter is not a critical requirement and a weaker authentication can be applied. Stronger authentication is chosen otherwise. The value of assets may also depend on other contextual factors. For example, the criticality of a building does not only depend on the goods that are in it, but also on whether tenants are in the building. This may also affect the criticality of related secondary assets and may modify the countermeasures that are applied. For example, the criticality of the CCTV camera may decrease when tenants are in the building. Consequently, in case the criticality of this asset is low, it is not necessary to forbid operations of remote disconnect, but it is just sufficient to authorize them. Primary assets can dynamically come into play, for example when a new appliance is acquired, and bring new secondary assets, security requirements and countermeasures. Existing assets can also be removed, for example, when an appliance is dismissed.

Scenario 2. The value of secondary assets on their own is not significant for security, but their criticality only depends on the criticality of their corresponding primary asset(s). Variability in the secondary assets may take place when they are added or removed to/from the system. For example, metering data can be stored in a supplementary device. In this case, a new secondary asset, associated with this additional storage device, must be linked to the metering data. As a consequence, new security requirements must be associated with this device, and suitable countermeasures (e.g., authentication and/or encryption) must be applied on it as well. Similarly, if a CCTV camera is replaced by an anti-theft sensor, an adaptation is triggered. First, all countermeasures that have been previously applied to the CCTV camera are removed. Then, a new set of countermeasures must be applied on the EMS (e.g., encryption of operational data) to protect its communication with the anti-theft sensor.

III. Asset and Goal Models

To represent all security concerns of a smart grid we leverage the goal and asset models, presented in earlier work [1]. Our goal model extends the KAOS model with a representation of vulnerabilities. The asset model represents the assets that need to be protected, their mutual relationships, and relates them to the corresponding security goals to be achieved. The asset model for the scenarios discussed in the previous section is shown in Figure 2. Each asset is associated with its secondary assets, if present. As explained before, a building is a primary asset and CCTV camera is considered as secondary asset. To turn off or disconnect appliances, an attacker may modify the operational data sent by the EMS. Hence, operational data is a secondary asset with respect to the CCTV camera and the heater. The EMS is a secondary asset with respect to the operational data, since an attacker may decide to gain the remote control of the EMS to violate the integrity of these data. Similarly, energy cannot be directly targeted by an attack, but its integrity[2] can be damaged by manipulating metering data or compromising the meter where these data are stored. Hence, metering data is a secondary asset for energy and meter is a secondary asset for metering data,

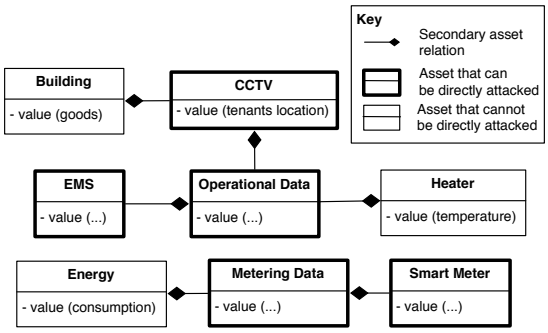

Figure 2. Asset model in the customer domain of a smart grid.

The goal model for the scenarios described in the previous section is shown in Figures 3 and 4. Figure 3 illustrates the main goals of smart grids, and decomposes them into functional and non-functional requirements (leaf goals). It also includes the vulnerabilities of the system, which may be brought by system operations, including security countermeasures, and domain assumptions. For example, a CCTV camera provides a set of standard functions (turn on, turn off, and replay), and allows users to remotely connect to it, by using login and logout operations. The operation *Turn Off* adds the vulnerability *V1*, since a malicious attacker may use it to deactivate video surveillance when the tenants are not in the building. To foster remote connection, the CCTV camera must be IP addressable. This domain assumption leads to the vulnerability *V2*, since a malicious attacker can remotely gain the control of the camera.

A smart grid must provide metering information. To this end, a meter should execute the following operations: show metering data, when required, read energy consumption log, update its current info, and store them internally. The operation *Store Info* introduces the vulnerability *V3*, since metering data may be read and manipulated by a malicious attacker. Remote connection is also supported by the meter, through login and logout operations. The operation *Login* introduces the vulnerability *V4*, since unauthorized attackers can remotely connect to it. Operational data must be also transmitted to the appliances to balance their load. To this end, the EMS must read energy usage, compute operational data and send them to the appliances. The operation *Send Op Data* is based on the domain assumption that data are transmitted through a wireless channel (*Wireless*). This assumption introduces the vulnerability *V5*, since messages sent over the wireless channel may be intercepted and modified by an attacker.

Figure 4 illustrates the security goals related to our scenarios, namely, availability and integrity, and associates each of them with the assets to be protected (in squared brackets). Security goals have a hierarchical structure and are decomposed into security requirements (leaf goals) and countermeasures. Each countermeasure is related to the vulnerabilities it tries to mitigate. For example, the goal *Availability[CCTV]* is decomposed into requirements *SR2* (authorize remote disconnect) and *SR3* (remote access to the CCTV camera must be authenticated). Requirement *SR2* can be achieved by using different countermeasures (e.g., forbid remote disconnect to all or grant it just to the admin user). These countermeasures mitigate the vulnerability *V1*. Requirement *SR3* can be achieved by applying proper authentication mechanisms (e.g., single factor or multi-factor) to remotely access to this appliance. These countermeasures mitigate the vulnerability *V2*. The priority of security goals also depends on the value of the primary assets that need to be protected. For example, the priority of the goal *Availability[CCTV]* depends on the value of the CCTV camera and any change in its priority may lead to the selection of a different set of countermeasures. In case the criticality of a CCTV camera is low, remote disconnect can be granted to a certain set of users. In case a CCTV camera is highly critical, remote disconnect can be denied for all users.

To support availability of appliances, it is also necessary to guarantee the integrity of their secondary assets (e.g., operational control data). In particular, each appliance must accept only operational control data coming from trusted sources (e.g., IP addresses) (*SR7*). These data must always report values (e.g. voltage) within a certain range (*SR8*), to avoid power outages. The countermeasures associated with *SR7* and *SR8* mitigate the vulnerability *V5*. To prevent energy theft, it is necessary to support the integrity of me-

[2]Note that in this case we refer to the integrity of energy consumption.

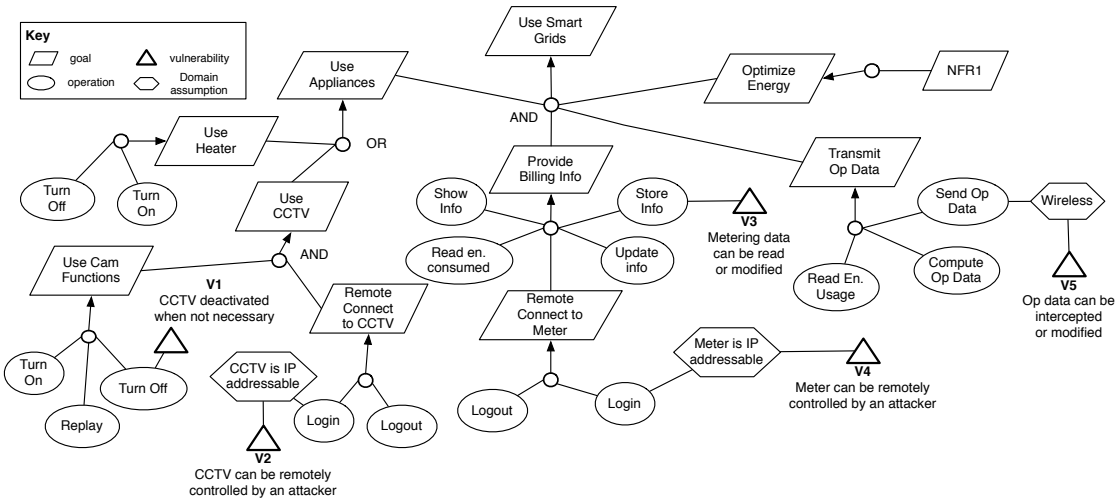

Figure 3. Goal model for the customer domain of smart grid.

tering data. For example, the amount of energy consumption must correspond to the energy consumed by the referenced appliance (*SR5*). The time of usage must correspond to the time when the referenced appliance was active (*SR6*). The countermeasures associated with *SR5* and *SR6* mitigate the vulnerability *V3*. To prevent energy theft, it is also necessary to foster integrity of the meter, which is a secondary asset for metering data, and, to this aim, the connection to the meter must be authenticated (*SR9*). A maximum number of consecutive invalid login attempts that a user can perform during a time period must be fixed (*SR10*). The countermeasures associated with these requirements mitigate the vulnerability *V4*. The link between assets and security goals along with the goal decomposition allows us to identify how and where countermeasures should be applied to satisfy their security goals. For example, the goal *Integrity[Energy]* depends on the integrity of *Metering Data*, which, in its turn, must be achieved by enforcing authentication at the *Meter* (*SR9*). In case a security goal or requirement is not associated with any asset, it automatically inherits the assets associated with its father. For example, *SR7* is directly associated with *Operational Data*.

IV. Engineering Security Requirements and Monitoring Functions

This section explains how secondary assets can be used to engineer security requirements and monitoring functions.

A. Engineering Security Requirements

To engineer security requirements it is necessary to develop their corresponding security countermeasures. These may avoid the execution of an operation, perform a set of actions before and/or after executing an operation, or modify the input and/or the output parameters of an operation. In this paper we do not automatically infer the implementation of the countermeasures, but we use secondary assets to identify the execution points where they have to be performed.

A possible way to implement security countermeasures is through AOP (Aspect Oriented Programming) [3], which allows us to inject existing security countermeasures (advices) at specific execution points (join points). Pointcuts can also be defined to group together a set of join points. We use AspectJ [4] to define the pointcuts used for our smart grid scenarios. For example, the following pointcut `counterm` identifies the execution points necessary to perform a countermeasure, when an operation (`op1`) is called on a target object (`obj`). We assume that *op1* is the operation that brings the vulnerability we want to mitigate.

```
pointcut counterm(Obj obj): target(obj)
    && call(* * op1(...));
```

In case an aspect denies the execution of an operation, the join point must be applied `around` the execution of that operation. This is the case, for example, of countermeasure *Grant to admin*, which may only allow the admin to perform an operation of remote disconnect on the CCTV camera. In this case, the following join point and advice is used.

```
around(Obj obj, int x): counterm(obj, x){
    ...
    if(admin) proceeds(x); }
```

The advice code will check the credential of the user, and, in case the user is the administrator, it `proceeds` with the code of the intercepted operation (i.e., `turn off`). Otherwise, the advice will return the control to the caller. In this case, the target object is the CCTV camera , which is the secondary asset associated with this countermeasure (see the parent goal *Availability[CCTV]* in Figure 4), and brings vulnerability *V1* that we are trying to mitigate.

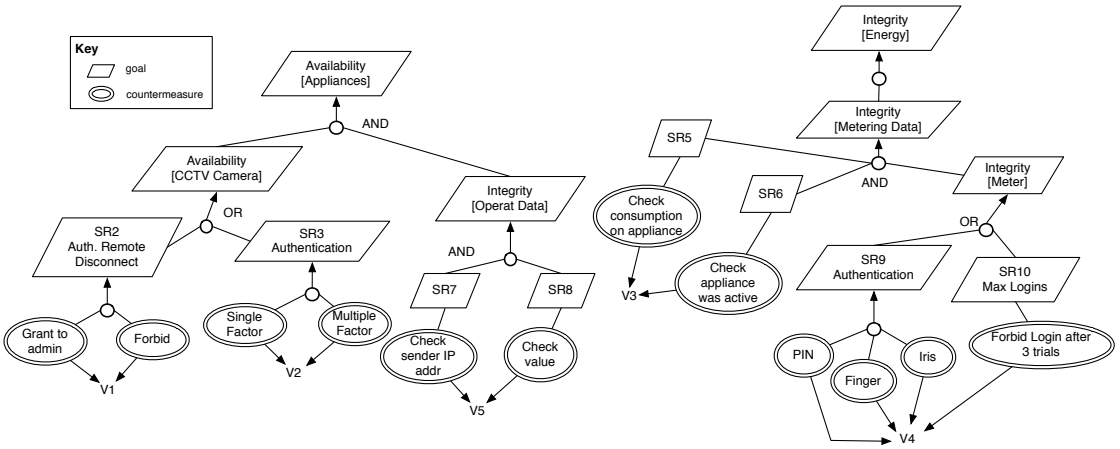

Figure 4. Security goals, requirements and countermeasures for the customer domain of smart grid.

In case an aspect performs some additional actions (`advice-code`) before or after the execution of the original operation, the following pointcuts/advices are respectively defined.

```
before/after(Obj obj, int x): counterm(obj, x){
  advice-code }
```

A `before` pointcut can be used for countermeasure *Check Appliance Was Active*. In this case, the operations in `advice-code` control whether an appliance was active at the time claimed by the metering data. The target object is the software component that manipulates metering data, which is the secondary asset associated with this countermeasure (see the parent goal *Integrity[Metering Data]*).

If an aspect modifies the input parameters of an operation, it must keep a reference to its input in its signature (`x`) and must perform additional actions (`before-advice-code(x)`) to modify parameter `x` and put into a new variable (`y`). The original operation is subsequently called by providing the new input (`proceed(y)`). The output parameter (`z`) of the original operation can be also further manipulated and returned by the advice.

```
int around(Obj obj, int x): counterm(m, x){
  int y =before-advice-code(x)
  int z = proceed(y);
  z = after-advice-code(z);
  return z; }
```

This template is adopted for example when we want to encrypt metering data before the store data operation at the meter, to mitigate vulnerability *V4*, and guarantee the integrity of energy. In this case, the operations in `before-advice-code` will encrypt the input data and will call the store data operation, without performing the operations in `after-advice-code`. The target object of this operation is the meter.

B. Engineering Monitoring Functions

Since we want to protect critical assets proactively, in this paper monitoring functions are not aimed to detect the occurrence of attacks. Instead, these update the values of primary assets depending on a set of contextual factors and propagate these values onto the corresponding secondary assets. Both contextual factors and secondary asset relationships are immediately derived from the asset model. Contextual factors can be *fixed* or *mutable*. Fixed factors do not change at runtime and, indeed, do not need to be monitored, although they can be manually modified. On the other hand, mutable factors can change dynamically and must be continuously tracked at runtime. Each factor has a value comprised 0 and 1, and impacts on the value of its corresponding asset with a certain weight. The value of an asset is computed by performing a weighted sum of the contributions of all its factors.

A model of the monitoring functions necessary for our example is provided in Figure 5. The criticality of the *Building* depends on factors *F1* and *F2*. *F1* is associated with the value of the goods that are in the building. It is a fixed factor, since, for simplicity, we assume that the goods in the building do not vary frequently. It can be eventually modified by the system administrator, for example, when a new device (e.g., computer) is installed in the building. *F2* is related to the location of the tenant and it is a mutable factor. *F1* and *F2* have respectively weights 0.7 and 0.3, since *F1* is more important to determine due to the criticality of the building. The value of *Energy* is associated with a mutable factor (*F3*), which is related to its daily consumption (*dc*).

The value of mutable factors depends on the satisfaction of some mutually exclusive constraints, which must be monitored with a certain frequency. Monitoring can be performed punctually (i.e., every time the data, on which the constraint depends, change), or with a certain frequency

(e.g., every second, minute, and so on). For example, factor *F2* (tenant's location) is associated with constraints *C1* and *C2*. *C1* verifies whether the tenant is in the building (e.g., the coordinates of her/his location are within those that delimit the building), while *C2* verifies the contrary. These constraints are checked every 10 minutes. In case *C1* is satisfied, the value assigned to $F1$ is 0.5, and is 1.0 otherwise. Factor *F3* (daily energy consumption) is associated with constraints *C3*, *C4*, and *C5*. They respectively verify whether $dc < 5kW$, $dc \in [5, 15)\ kW$, or $dc \geq 15kW$. These constraints are monitored daily and, in case one of them is satisfied, the value assigned to $F3$ is 0.25, 0.5 and 1.0, respectively. Secondary assets are valuable since they are the "means" to harm their associated primary assets. For this reason, secondary assets inherit the value of their corresponding primary asset(s) and must be protected as well. For example, a *CCTV* camera acquires the same value of the associated building.

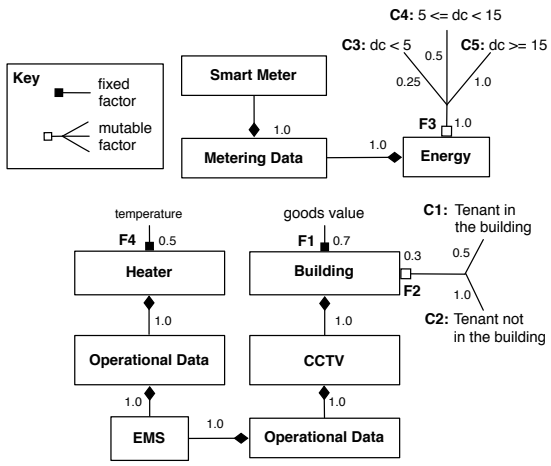

Figure 5. Monitoring functions of the smart grids example.

V. Related Work

We are not aware of any work on adaptive security in the smart grid domain. Most of the research efforts have investigated attack-oriented approaches to adaptive security (e.g., intrusion detection and tolerance). Debar et al. [5] define the goal of intrusion detection systems as "to discover breaches of security, attempted breaches, or open vulnerabilities that could lead to potential breaches". They also mention the possibility of applying countermeasures after attack detection. Julisch [6] uses root cause analysis to identify why an intrusion detection system raises alarms. The ultimate goal of this work is to reduce the number of alarms to be notified to users. Atighetchi et al. [7] target intrusion tolerance through strategies such as replicating key application components, attack containment and unpredictable change of configurations. Weise [8] listed several goals for adaptive security, including reducing remediation time, decreasing attack velocity and shrinking the attack surface. None of these works uses asset modeling and runtime monitoring to prevent attacks. Instead, existing solutions mainly address attack detection and reaction.

VI. Conclusions

This paper has proposed an approach to support adaptive security by explicitly modeling assets and linking them to their corresponding security goals. We also introduced the concept of secondary assets, which denotes assets that may be directly targeted by an attack to indirectly harm other primary assets. Secondary assets, along with an explicit representation of vulnerabilities, help engineering security requirements and the deployment of security countermeasures. Representing the variability of assets facilitates the design of asset monitoring functions. We used the smart grid as an application domain to demonstrate our approach.

We are planning to integrate our approach with the decision making mechanism we built in earlier work [1]. We also intend to improve monitoring functions with explicit support for detecting new vulnerabilities, for example, by monitoring domain assumptions. Finally, we will further investigate how to apply monitoring functions for reactive adaptation in order to detect occurrences of attacks.

VII. Acknowledgements

This work was supported by the EU 7th Framework Programme (FP7/2007-2013) grant 258109 (FastFix), and by Science Foundation Ireland grant 10/CE/I1855.

References

[1] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh, "Requirements-Driven Adaptive Security: Protecting Variable Assets At Runtime," Lero- Irish Software Eng. Research Centre, Tech. Rep. Lero-TR-2012-01, 2012.

[2] "Smart Grids European Technology Platform," http://www.smartgrids.eu/.

[3] G. Kiczales *et al.*, "Aspect-Oriented Programming," in *Proceedings of the 11th European Conference on Object-Oriented Programming*, 1997, pp. 220–242.

[4] "The AspectJ Project," http://www.eclipse.org/aspectj/.

[5] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Annals of Telecommunications*, vol. 55, no. 7, pp. 361–378, 2000.

[6] K. Julisch, "Clustering intrusion detection alarms to support root cause analysis," *ACM Transactions on Information and System Security*, vol. 6, no. 4, pp. 443–471, 2003.

[7] Atighetchi *et al.*, "Adaptive cyberdefense for survival and intrusion tolerance," *Internet Computing, IEEE*, vol. 8, no. 6, pp. 25–33, 2004.

[8] J. Weise, "Desiging an adaptive security architecture," 2008, wikis.sun.com/download/attachments/57526796/820-6825.pdf.