Towards Self-protecting Smart Metering: Investigating Requirements for the MAPE loop

Mazeiar Salehie*, Liliana Pasquale*, Inah Omoronyia* and Bashar Nuseibeh*†
* *Lero- Irish Software Engineering Research Centre, Limerick, Ireland*
Email: {Mazeiar.Salehie, Liliana.Pasquale, Inah.Omoronyia}@lero.ie
† *Department of computing, Open University, Milton Keynes, UK*
Email: Bashar.Nuseibeh@lero.ie

***Abstract*—Smart grids are increasingly proliferating all over the world to leverage electricity infrastructures with information technology. Smart metering, particularly Advanced Metering Infrastructure (AMI), is an enabling technology for realizing smart grids by collecting and processing energy consumption logs and managing energy for customers and utility companies. Security is one of the main concerns of smart metering, and potential threats and attacks to this technology have been discussed since the early initiatives. Considering the unbounded and changing nature of security problems, especially in complex and critical cyber-physical systems, smart metering security concerns can not be always addressed at design time. Autonomic self-protection promises to address runtime security concerns in proactive and reactive ways. In this paper, we focus on the customer domain of smart metering, and investigate potential reactive self-protection scenarios by concentrating on requirements. To this aim, we analyze a sample set of published security requirements from the AMI-SEC forum [1] to derive self-protection requirements. Then we discuss how these requirements can be linked to the MAPE loop [2] in the autonomic computing architecture.**

***Keywords*-Self-protecting; Smart metering; Security requirements; Autonomic computing; Smart grid;**

I. INTRODUCTION

Smart gird technology aims at leveraging the power grid with information technology to improve efficiency, flexibility and reliability of managing energy resources and power-enabled devices. Advanced Metering Infrastructure (AMI) is a way to realize the smart grid. AMI offers a rich structure for collecting and processing metering information for customers and utility companies. However, AMI flexibility and its rich set of features could introduce vulnerabilities to power grid and energy management. In the last five years numerous articles have been published on AMI and smart grid security, and how new changes in the power gird could be threatful.

Generally, security cannot be guaranteed, and its bounded view at design-time would not probably protect all the valuable assets in the boundary of systems in all situations. Security requirements may fail, cannot be satisfied anymore, or domain assumptions may not be valid anymore. In these cases the system may need changes in security countermeasures and even requirements. Autonomic computing proposed self-protection as one of the major properties of autonomicity to address these runtime security concerns [2]. However, existing smart grid and metering infrastructures and technologies have not considered emerging security problems at runtime and self-protection. Our main motivation is to investigate self-protection in smart metering and explore this area as a new domain for software adaptation.

This motivation leads us to define the problem of self-protection in smart metering from the Requirements Engineering perspective. Requirements engineering for autonomic and self-adaptive software systems is not well established and is still challenging, as discussed by Cheng et al. [3], and Salehie and Tahvildari [4]. This paper does not aim to propose a process for requirements engineering of self-protecting systems. Instead, we intend to explore runtime self-protection needs in smart metering and to investigate possible requirements.

Requirements in this problem domain, which we call them *self-protection requirements (SPR)*, should be elicited from stakeholders and/or derived from the security requirements of managed elements in smart metering systems. However, requirements and architecture specifications are intertwined, as discussed by Nuseibeh [5]. Therefore, we should consider the target architecture in elaborating and refining SPRs. In this paper, we adopt the autonomic computing reference architecture [6] including the MAPE loop.

Identifying SPRs for the entire AMI needs an extensive analysis of security requirements in customer, generator and distributer domains. For this paper, we narrow the scope of SPRs to the customer domain of smart metering. This allows us to investigate potential self-protection scenarios without dealing with the complexity of the entire metering infrastructure. Our contributions in this paper are: i) analyzing security requirements published by AMI-SEC [1] to investigate whether self-protection in smart metering is essential, ii) defining a sample set of self-protection requirements based on AMI-SEC document, and iii) linking these self-protection requirements to the MAPE loop in the autonomic reference architecture.

The rest of this paper is organized as follows. Section 2 gives a brief review of smart metering and self-protecting state of the art. Section 3 provides a short review of smart metering and its significant entities, especially in the customer domain. Section 4 analyzes security issues in smart metering. Section 5 discusses how self-protection requirements can specify the autonomic system behavior, and Section 6 derives a sample set of such requirements in smart metering by considering the MAPE loop. Section 7 draws conclusions and points out potential future directions for this work.

II. Related work

Up to our knowledge, there is no work on self-protecting smart metering. First, we briefly reviews the self-protecting systems in practice, and how they define self-protection. Then, we look at the state of the art in requirements engineering for autonomic and adaptive software. Although, there is no research effort specifically on self-protection requirements, reported experiences are quite useful for this paper.

A. Self-protecting systems

Self-protecting solutions are often referred to artificial immune, intrusion detection and similar systems in the literature (e.g., see [7] [8] [9]). The IBM autonomic computing architectural blueprint defined self-protecting as [6]: "To anticipate, detect, identify, and protect against threats." What stakeholders expect from such a self-protecting system has been defined by these four capabilities, although the last one is general. It is not clear that after identifying threats what will happen, and the definition does not refine the "protect" capability.

The vision of autonomic computing elaborates further this definition [2]: "Autonomic systems will be self-protecting in two senses. They will defend the system as a whole against large-scale, correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures. They will also anticipate problems based on early reports from sensors and take steps to avoid or mitigate them." In this definition, problems seem refer to threats, and a self-protecting system is supposed to anticipate them similar to the first definition. But attacks are specifically addressed and an interesting point is that unintentional failures are also considered as potential threats if they propagate through the system. Chess et al. [10] focused on security aspects of autonomic computing, from an architectural point of view. They discuss that the autonomic manager should monitor and control the managed element, and control the access to its input and output channels. Requirements were not explicitly discussed in this work, but Chess et al. highlighted two general properties of detection and restoration.

Debar et al. [11] defined the goal of intrusion detection systems in the MAFTIA project as "to discover breaches of security, attempted breaches, or open vulnerabilities that could lead to potential breaches". They mentioned the possibility of applying countermeasures after detection as well. Several efficiency properties such as performance, fault tolerance, accuracy and completeness were also mentioned for these systems.

Julisch from IBM Zurich research lab [12] used root cause analysis to identify why an intrusion detection system raises alarm. The ultimate goal of that work was to reduce the number of alarms to be notified to users. Atighetchi et al. [13] from BBN technologies targeted intrusion tolerance through strategies such as replicating key application components, attack containment and unpredictable change of configurations.

In another related work, Joel Weise, from the adaptive security architecture project in Sun Microsystems, listed several goals for adaptive security, including reduce remediation time, reduce threat amplification, decrease attack velocity and shrink the attack surface [14]. Automatic repairing of discovered vulnerabilities is also addressed by some industrial research. For example, ShieldGen [15] is a tool provided by Microsoft research to generate patches in order to prevent zero-day attacks.

None of these work clearly and explicitly defined the problem of self-protecting by requirements specifications. But they explained features and capabilities expected from a self-protecting system. These capabilities can be mapped to the monitoring, analyzing, planning and executing functions in the MAPE loop. The aforementioned capabilities particularly related to analyzing and planning. Detecting, diagnosis, and anticipating (prognosis) are linked to analyzing, and tolerating, repairing, recovering and similar capabilities are related to planning.

B. Autonomic and adaptation requirements

Two high-level aspects of requirements need to be discussed in engineering autonomic and adaptive software, and in this context self-protecting software: i) functional and non-functional adaptation requirements, and ii) the role of requirements at runtime. The former should be linked to the adopted architecture, which in this paper is the autonomic architecture, and most often includes a control loop. But for requirements at runtime, three roles are discussed in the RE community: i) monitoring requirements, ii) evaluating requirements as targets for adaptation, and iii) evolving requirements.

Fickas and Feather [16] highlighted the importance of requirements monitoring at runtime for evolution. They posed two questions to figure out how we can know when a system needs to be evolved, and how we can use the acquired knowledge to orchestrate the evolution. Souza et al. [17] noted that "adaptation takes place when requirements are found to be partially/un-fulfilled." They introduced awareness requirements as a way to deal with adaptation requirements and their fulfillments at runtime, which mostly

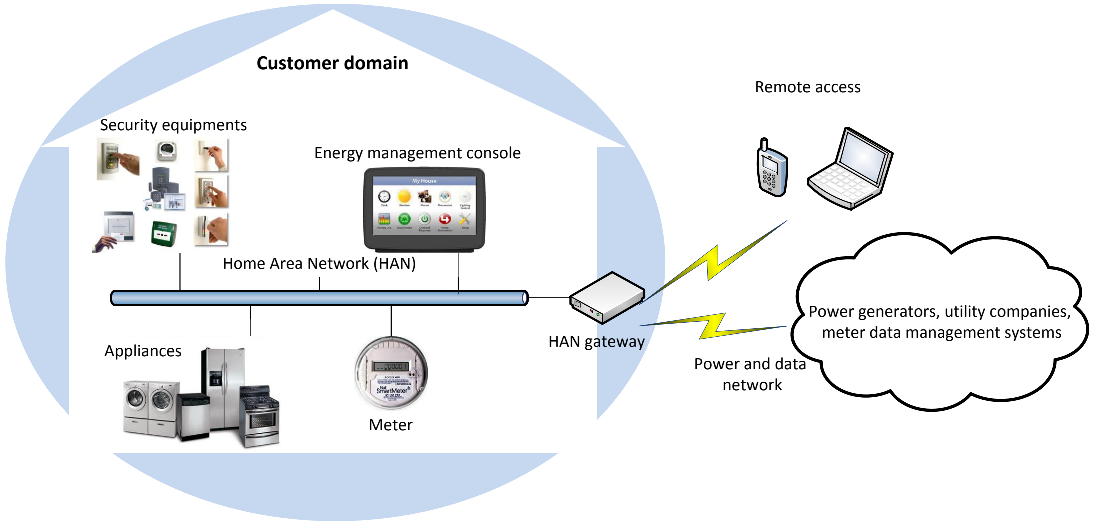

Figure 1. A general schema of smart metering system by focusing on the customer domain.

address monitoring and analyzing functions in the MAPE loop. Awareness requirements refer to success, failure, performance and other properties of software requirements. Mylopoulos defined adaptivity requirements as [18] "awareness requirements that can also talk about changes of status for another requirement." In this view operationalization of adaptation requirements is reactive rather than proactive (proactive behavior in this view is addressed by requirements at the development time).

Baresi et al. [19] proposed FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) based on fuzzy goals to deal with fulfilment of requirements at runtime. Adaptation goals define countermeasures for dissatisfaction of system goals. These countermeasures are responsible for changes in the goal model to prevent violation of a goal, enforce a particular goal, or move to substitute one. Salehie and Tahvildari [20] proposed GAAM (Goal-Attribute-Action Model) as a runtime model for representing system and adaptation goals. Autonomic manager uses GAAM to monitor goals and select appropriate actions in each situation.

The existing research body partially addresses what we need to deal with autonomic requirements. Representing and managing these requirements need to provide new processes, languages, verification methods, and traceability from requirements to architecture and source code. Particularly, defining and managing self-protection requirements have not been investigated. Compared to other self-* properties, uncertainty level is higher, and monitoring self-protection requirements is more difficult.

III. Smart Metering in AMI

Figure 1 depicts the high level structure of a smart metering system, by concentrating on the customer domain. In this domain, *meter* records energy consumption data and other logs of the system (including any changes in the configuration). *Management console* provides the user interface and management services to configure and monitor the system behavior. *Electrical/electronic equipment* is power consumer including regular appliances and critical equipment for safety and security (e.g., anti-theft system). Equipment can be smart appliances that have local energy management units. *Home Area Network (HAN)* is the internal data network between meter, management console and equipment. *HAN gateway* is the interface between HAN and the external world. There may be also some local power generation sources in the customer domain, such as solar panels. Customers and maintenance crews can access HAN remotely to monitor or control devices in the customer domain.

Major benefits of AMI include [21]: two-way communication between consumers and providers, easier consumption management, better system resilience, and easier metering data management and billing. Generally, smart grid is an evolving domain and some experts believe that existing AMI solutions still do not address smart grid objectives. For example, Turner and Taft[1] argued that AMI focuses on customer premises, while smart grid includes many other assets and data in generation, transmission and distribution. In the context of this paper, only the customer domain is discussed and AMI addresses this domain adequately for our objectives.

Security concerns of the smart grid have been discussed since the very beginning, and security often has been mentioned as the top issue to be taken into account in realizing smart metering. Smart grid is believed to be much vulnerable

[1]D. Turner, J. Taft, "Smart Grids and AMI: Understanding the Big Picture", www.energypulse.net, September 2008

than existing power grids, mainly due to an extended attack surface [22], which is associated to augmenting information technology to the power grid. Therefore, security in AMI is extremely significant for stakeholders, and AMI-SEC task force provided security requirements to elaborate what are expected from a smart grid employing AMI [1]. Several other organizations have also provided detailed guidelines for smart grid security, e.g., NIST published a three-volume guideline for security and privacy concerns and requirements [23]. In this paper we consider the AMI-SEC set of security requirements, which roughly organized based on security goals.

IV. Analyzing security in the customer domain of smart metering

In this section, we analyze security requirements and architecture artifacts published by AMI-SEC. We briefly review valuable assets, threats, security requirements and vulnerabilities, and give our view of these artifacts.

A. Valuable assets and corresponding security threats

In the customer domain of smart metering system, considering Figure 1, three categories of assets are recognizable:

- *Energy* is the main critical asset in the customer domain that needs to be protected.
- *Information* in the metering system, including metering audit trail, billing data, pricing data, and personal information.
- *Equipment and applications for metering and energy management* in the customer premise also need to be protected, including the meter, management console, networking devices, and the applications running on equipment.
- *Electrical and electronic appliances* can be the critical assets that need to be protected. Equipment may be security controls that protect other valuable assets. For example a digital access control mechanism that controls valuable devices, documents or VIPs. In this case, energy disconnection or interruption could harm the protected assets.

Notable threat agents for the entire AMI system can be [24]: customer, insider, outsider, and prior insider. However, human errors, equipment failures and natural disasters may also lead to security threats as well [25]. In a typical customer domain, threat agents would be customer (e.g., house or organization owner), outsiders (e.g., a neighbor), or prior insider (e.g., previous tenants). Because AMI employs information technology and the metering is accessible remotely from the network (and often from web), most of the common threats in Internet would be likely. However, energy and its related assets may motivate a host of other threats for financial gain, fraud and sabotage [21].

Almost all of the existing self-protecting systems have targeted cyber security and mainly information security. However, security in smart metering is about cyber-physical assets, and this is what makes self-protection different from other practices. Especially energy as an asset brings new challenges to security and self-protection, since threatening this asset could harm other assets in the system. Considering asset types, the following threats in a customer domain are notable:

- Threats regarding energy as the primary asset: The most significant attack in this category is energy theft, by a customer or an external agent (e.g., a neighbor). This attack can be realized by methods such as meter tampering, meter bypassing, and log modification. Also power shut down (i.e., denial of service) is under this category.
- Threats regarding energy as the secondary asset: In this category two cases may happen. First, power-enabled security controls may be targeted. For example, access control systems, surveillance cameras, and network security devices. The power outage or interferences would knock off these functions and would ease access to protected assets. Second, depending on facilities the smart metering system provides, power-enabled appliances might be manipulated in a way to be harmful to people or assets in a property. For example, safety incidents (fire or electrocution) may happen by manipulating electrical appliances (i.e., usurpation threats).
- Threats regarding energy-related information assets: There may be other assets not in the above categories prone to security attacks. For instance, audit trail may be modified to conceal the energy theft, but this asset could be also misused in other ways. Such assets could be misused towards confidentiality and privacy breaches (e.g., revealing location and types of used devices), vandalism, fake utility retail, and harm to reputation (e.g., manipulating the metering data to frame someone).

For the scenarios in this paper, we focus on threats to energy assets and related information.

B. AMI-SEC security requirements

AMI-SEC published the AMI system security requirements covering three categories [24]: primary services (addressing major goals such as availability), secondary services (including cryptographic and resource management services) and assurance (including accountability and access control). We base the scenarios and self-protection requirements on the primary services section of this document.

Table I shows several security requirements from AMI-SEC document [1]. Authorization requirements are denoted by AZR, availability requirements by AVR, and accountability requirements by ACR.

AZR1 limits the number of concurrent users, perhaps to ease access control and to avoid flooding attacks. However, assume a situation when the system cannot afford even the

Table I
A SAMPLE SET OF SECURITY REQUIREMENTS FROM AMI-SEC DOCUMENT [1]

Category	Security requirement
Authorization	AZR1: The security function shall limit the number of concurrent sessions for any user to [assignment: organization-defined number of sessions] on the system.
Authorization	AZR2: The security function shall enforce the most restrictive set of rights, privileges or accesses needed by users or workstations (or processes acting on behalf of users) for the performance of specified tasks.
Availability	AVR1: The security function shall ensure that each access to all shareable resources shall be mediated on the basis of the subjects assigned priority.
Accountability	ACR1: The organization shall allocate sufficient audit record storage capacity and configure auditing to reduce the likelihood of exceeding storage capacity.
Accountability	ACR2: The security function shall [actions to be taken in case of possible audit storage failure] if the audit trail exceeds [pre-defined limit].

required number of sessions for some users. This could be the case of a denial of service attack by a customer in order to reject accessing to the metering log from the utility company. AZR2 limits the access to the system services due to performance concerns of some specified tasks, and AVR1 guarantees access to the system resources based on the user priority. These two requirements can adjust access to data and energy management services, but they can also impact on each other as we explain later in Section VI. ACR1 and ACR2 refer to the audit record storage capacity and when it is about to be filled. The audit trail is a valuable record in the metering system, because it includes the consumption log and changes in the security controls. An energy theft attack most likely clears its footprint in this record.

C. Implicit self-protection capabilities in AMI

The AMI-SEC document [1] implicitly or partially noted some self-protection capabilities in smart metering. For example, the document includes a set of eight requirements for anomaly detection services, and among these the following requirement is related to the context of this paper:

"The organization shall implement control system incident handling capabilities for security incidents that includes preparation, detection and analysis, containment, eradication, and recovery."

This is a general self-protection requirement, and the AMI security requirements analysts admitted that unexpected incidents should be detected and responded. But this requirement needs to be elaborated and refined further, particularly regarding the target architecture. There are also some other security requirements scattered in other categories that implicitly address adaptation and self-protection. For example, under accounting category there is the following requirement:

"The security function shall take [assignment: list of actions] upon detection of a potential security violation."

These actions are related to recovering, repairing, preventing and tolerating security violations, which can be runtime actions for self-protection. Among these self-protection capabilities, detection has been addressed more than the others. For example Berthier et al. [26] focused on intrusion detection, and interestingly their approach is specification-based to identify deviation from desired behaviors (close to runtime requirements monitoring). They argued that although specification-based intrusion detection in AMI is able to find unknown attacks, it is more expensive and provide little information for diagnosis. Robinson and Stuber [24] also noted that system controls are required for detective, deterrent, preventive and corrective behaviors in AMI without elaborating them further. Berthier et al. [26] also mentioned three main classes of requirements– including prevention, detection and mitigation/resilience– for secure AMI without discussing them as runtime self-protection capabilities.

V. DERIVING AND DECOMPOSING SELF-PROTECTION REQUIREMENTS FOR THE MAPE LOOP

Self-protection requirements can be directly elicited by the stakeholders and/or can be derived from managed element artifacts, such as SLAs and particularly security requirements. In the former case, customers and utility company owners/administrators express what they expect from self-protection, for instance to deal with failures of security requirements. In the latter, security requirements of managed elements are analyzed to derive self-protection requirements. In this view self-protection requirements refer to other requirements, in a sense of meta-requirements. This is similar to what Souza et al.[17] defined as awareness requirements for general software adaptation, although those requirements consider only monitoring. In this paper, we focus on deriving self-protection requirements from already published security artifacts.

We denote self-protection requirements by SPR, which can be defined based on failures, conflicts and other conditions for the security requirements in metering system:

- *Failures and frequent failures of security requirements*
- *Relationships and conflicts between security requirements*

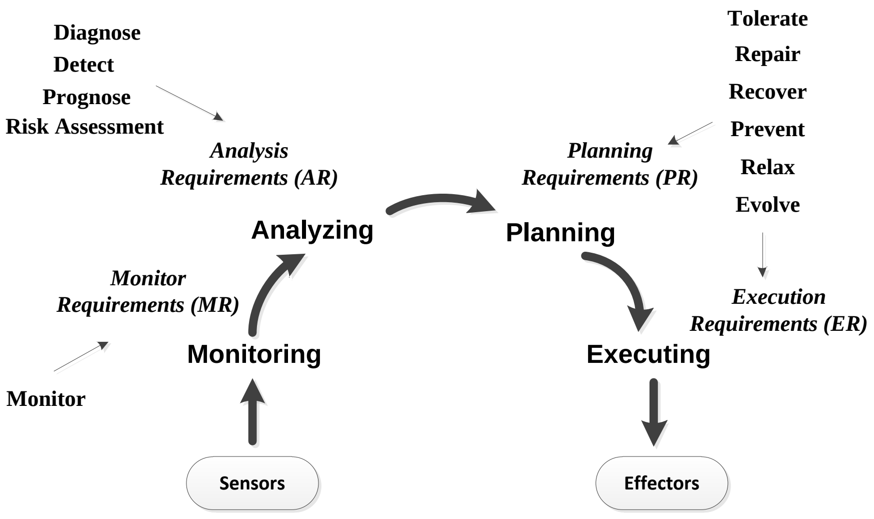

Figure 2. Decomposing self-protection requirements (SPR) for the MAPE loop

- *Relationships and conflicts between security and other quality requirements (e.g., performance and usability)*
- *Failures and frequent failures of self-protection requirements*

As noted before, due to the links between requirements and architecture specifications, we cannot elaborate self-protection requirements without considering the target architecture. By adopting the autonomic reference architecture [6], SPRs are decomposed to four categories of requirements corresponding to the MAPE loop, as illustrated in Figure 2.

First, *MRs (Monitor Requirements)* need to be defined to specify what can be and should be monitored, including security requirements and probably other SPRs. Second, *ARs (Analysis Requirements)* are defined to determine when adaptation is required to be performed. This includes detecting failures of security requirements and possibly SPRs, probably diagnosis and prognosis of a detected issue, and also risk assessment when there is even no requirement failure or attack. Third, *PRs (Planning Requirements)* specify strategies or tactics for decision making in self-protection. This may include tolerating security breaches (e.g., intrusion), repairing a problem, recovering from an attack, preventing an attack (or preventing its progress), or relaxing/evolving security requirements. The last but not the least, *ERs (Execution Requirements)* specify what should be considered in applying adaptation actions? What if adaptation actions cannot be applied and something goes wrong?

Two other significant entities in Figure 2 are sensors and effectors. The autonomic manager should be aware of what monitors and effectors are provided by managed elements, and what are constraints to use them at runtime. For example, the overload of sensors and conflicts between effectors should be specified. Self-protection requirements, particularly, MRs and ERs need to take into account characteristics and limitations of sensors and effectors.

As discussed by Cheng et al. [3], in adaptation requirements flexibility and uncertainty need to be considered by changing the vocabulary we use to express requirements. In traditional requirements normally “shall” and “must” enforce a behavior on the system, while in an autonomic system “ought to”, “might” and “may” are more appropriate to postpone the behavior selection to runtime. When the autonomic manager has more information about the situation, it can decide what to do based on existing alternatives, provided at design time or discovered during runtime.

VI. Defining self-protection requirements in the customer domain

In this section, we derive several sample self-protecting requirements from the AMI-SEC document. We investigate selected requirements in Table I for this purpose. First SPRs are derived and then they will be refined regarding the MAPE loop.

A. Sample self-protection scenarios

1) First scenario (Authorization and availability): In some situations the number of concurrent sessions may need adjustment to give access to higher priority users (e.g., maintenance agents). This means that AZR1 may not fail, but because the lack of resources high priority users cannot access the system. As noted before, this could be a DoS attack. In this scenario, the autonomic manager should handle AZR1 by considering AVR1. The following self-protection requirement is defined to address this scenario:

SPR1: *If a user (or subject) cannot access resources and AZR1 does not fail, the autonomic manager ought to enforce AVR1 for higher priority users.*

2) Second scenario (Authorization and performance): Assume the specified tasks in the AZR2 requirement in Table I do not have acceptable performance. This may be because of different reasons including component failures, but AZR2 aims to limit the access in order to release resources for high priority tasks. If autonomic manager observes that this cannot happen, in other words AZR2 fails, access control policies could be changed temporarily to tolerate the poor performance situation. Obviously, the autonomic manager cannot block all possible accesses to the metering system. For this purpose, a self-protecting requirement is defined as following :

SPR2: *The autonomic manager might modify access control rules when performance of [specified tasks in AZR2] cannot be satisfied by AZR2, considering other performance requirements.*

This requirement by itself is not enough to specify the autonomic manager behavior. As we show later in Section VI-B, this SPR1 is decomposed into requirements for the MAPE loop. Again, here we use “might”, because changing access control rule might not be the most appropriate option for that situation, perhaps considering performance requirements.

3) Third scenario (Accountability): For the third scenario, consider requirements ACR1 and ACR2 in Table I. In this scenario two threats may exist. First, something similar to a buffer-overflow attack may occur by adding a malicious action that deletes the records in case of an overflow. Note that the action may log an audit trail transfer or back up, but it actually deletes the trail. In this case, the autonomic manager should monitor actions taken because of ACR2 to detect any possible security breach. A self-protection requirement can be defined for this case:

SPR3: *The autonomic manager shall supervise the actions performed to satisfy ACR2 to prevent the loss of audit trail data.*

In SPR3, we used “shall” to oblige the autonomic manager. Although, this is a strict behavior, preventing malicious actions on the audit trail can be done in different ways.

Second, the audit trail overflow may happen frequently, which may cause problems for system availability. The autonomic manager needs to monitor ACR2 and detect its frequent failures. Also, ACR1 might not be satisfied and proper actions may need to be taken accordingly. The following self-protection requirement is defined for this purpose:

SPR4: *The autonomic manager shall take a proper action when ACR1 fails possibly because of frequent audit trail overflows (ACR2 causes actions frequently).*

B. Decomposing metering self-protection requirements for the MAPE loop

The next step is mapping SPRs to the MAPE loop by defining MRs, ARs, PRs and ERs.

1) First scenario: In this scenario, a monitor requirement needs to keep track of access failures and the number of concurrent sessions for different users:

MR1: *The autonomic manager shall monitor any access failure to system and the number of concurrent sessions for each user using [provided sensors].*

Also an analysis requirement is needed for checking failure or success of AVR1:

AR1: *If a user cannot access a resource, while other users with lower priority are currently accessing that resource, AVR1 fails.*

Another analysis requirement can be defined to figure out the failure cause. In this case, the planning function can respond effectively (e.g., providing more resources if DoS attack is not the reason). A possible planning requirement would be:

PR1: *The autonomic manager ought to manage lower priority users (or processes) to allow access to the high priority user based on AVR1.*

ER in this scenario is defined as following:

ER1: *The autonomic manager ought to satisfy PR1 using [specific effectors in the metering system] when AVR1 fails.*

The effectors are provided by the metering software (i.e., managed element). For example, PR1 may be satisfied by disconnecting at least one of the sessions of the lower priority users (or processes). Here the effector could be provided by the energy management console or the meter. In this case, AZR1 may be violated, because the number of allowed sessions might not be provided for users with a lower level of priority. This situation can be handled by considering priority of these two security requirements.

But what if this scenario happens frequently? The autonomic manager might change AZR1 to lower the limit of concurrent sessions for lower priority users. For example the following planning requirement might handle this case:

PR2: *if AVR1 fails frequently for high priority users, the system might change the number of concurrent sessions for lower priority users in AZR1.*

But the system needs an analysis requirement to know a frequent failure:

AR2: *if AVR1 fails [a specific number] of times in a [specific time period] for users with priority more than [specified level], AVR1 fails frequently.*

However, PR2 and AR2 can be defined in another way by referring to PR1. In this case, if PR1 is applied frequently, then AZR1 should be changed. This is an example in which self-protection requirements can refer to each other. In this situation, ER2 is defined similarly to ER1, based on specific effectors provided to satisfy PR2.

2) Second scenario: For the planning function, we may define a planning requirement like this:

PR3: *If AZR2 fails, and performance of [specified tasks in AZR2] is unacceptable, the system ought to block low-priority users for [a certain amount of time].*

Of course we need to monitor AZR2 and detect its dissatisfaction. MR1 and MR2 define two such monitor requirements

MR2: *The system shall monitor the performance of [specified tasks in AZR2].*

MR3: *The system shall monitor accesses and actions from users and all the processes using [provided sensors].*

AR1 is defined for detecting the AZR2 failure:

AR3: *If in [a certain period of time] the system cannot provide the specified level of performance for [specified tasks in AZR2] due to user access, AZR2 fails.*

As noted before, the performance problem may not because of user access, and the analysis requirement should figure out whether this is the root cause of the problem. Note that in this case the security requirement AZR2 addresses performance directly, but this is not necessarily the case, and the autonomic manager may consider other requirements of the managed element as well.

ER3 can be defined as:

ER3: *The autonomic manager may apply actions taken by PR3 and hold the condition unless an emergency happens (may change priorities).*

The emergency may be defined for security or safety-critical cases. For example, consider when fire is detected in a house and customer, which might be low priority user, wants to access the system.

3) Third scenario: For this scenario, a monitor requirement needs to observe actions taken by ACR2:

MR4: *The autonomic manager shall monitor actions that are taken by ACR2 on the audit trail using [provided sensors].*

An analysis requirement can be defined as following to deal with the first case in the third scenario:

AR4: *If the audit trail is flushed by an action specified in ACR2 without reporting to the assigned utility company, the action is malicious.*

A response can be defined by a planning requirement as:

PR4: *If a malicious action flushes the audit trail, the action must be blocked (or removed).*

ER4 is similar to ER3 with different effectors.

For the second case of the third scenario, in which ACR2 causes performing actions frequently, this analysis requirement can be defined:

AR5: *If ACR2 causes performing actions on audit trail [a specific number] of times in a [specific time period], actions could be suspicious or ACR1 cannot be satisfied.*

The autonomic manager may eventually notify the customer and utility company that ACR1 cannot be satisfied and the storage capacity should be increased or the period of data transmission or backup should be decreased. A planning requirement may change ACR1 to increase the audit record storage capacity.

PR5: *If the audit trail exceeds from its limit frequently (ACR2 is fired frequently), the system ought to increase the record storage capacity.*

ER5 is set to assure PR5 is satisfied, but in reality there may be no additional resources to added automatically by system. In this case ER5 notifies the customer and the utility company lack of enough storage capacity.

VII. Conclusions and future work

This paper explored smart metering as an application domain for autonomic self-protection. Cyber-physical security issues, and protecting energy and its related information in a complex smart metering system may not be easy to address at design time. By considering the customer domain in AMI, we investigated a few possible security scenarios that need runtime management. We defined the corresponding self-protection requirement for each scenario. These requirements were elaborated later for the autonomic reference architecture [6] and especially the MAPE loop. This mapping improves requirements traceability, which in turn may be useful for maintainability and verification. A notable point is that these requirements can refer to each other. For example, a planning or execution requirement can be addressed by monitoring requirements to deal with their failures.

Autonomic/adaptation requirements have some differences from traditional requirements. Flexibility, variability and uncertainty should be reflected in autonomic requirements, yet there is no language for expressing these requirements. In this paper, based on guidelines from Cheng et al. [3], we expressed self-protection requirements and their refinements in the MAPE loop. This approach can reflect autonomic aspects better than traditional requirements, although verification and assurance would be more difficult (which is expected for an autonomic system).

To extend this work on self-protection requirements, several future directions can be taken. Self-protection requirements can be defined based on the hierarchy of security goals. In this way, the metering system goal model is extended with self-protection goals to monitor and analyze satisfaction state of security goals. Another point is that self-protection requirements may be used only for developing the autonomic manager. But if the autonomic manager includes AMI security requirements as runtime objects, self-protection requirements can be represented at runtime as well. These requirements could be similar to self-protection requirements in this paper or live adaptation goals.

The scenarios we discussed in this paper are reactive, since they happen when requirements fail. But, a self-protecting system could be proactive to change security controls even when no security requirements have failed. For instance, the system can be risk-adaptive to increase the protection level in case risk increases. When the threat level in a region increases or valuable assets are added to customer premises, the system may need to change the security control. We noted that these cases can be addressed using runtime models, although we did not cover them in this paper.

In this paper, we did not discuss security issues of the autonomic manger. Self-protection requirements may be defined to protect an autonomic manager and its MAPE functions and policies as critical assets. We also did not address coordination between self-protection and other autonomic properties. This aspect is also important in engineering an autonomic manager in practice. Using requirements and models at runtime could facilitate handling the multi-objective nature of autonomic management.

ACKNOWLEDGMENT

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

REFERENCES

[1] B. Brown *et al.*, "AMI system security requirements," 2008, AMI-SEC Task Force.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[3] B. H. Cheng, R. Lemos, H. Giese *et al.*, "Software engineering for self-adaptive systems: A research roadmap," pp. 1–26, 2009.

[4] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. on Autonomous and Autonomic Systems*, vol. 4, no. 2, pp. 1–42, May 2009.

[5] B. Nuseibeh, "Weaving together requirements and architectures," *Computer*, vol. 34, no. 3, pp. 115–119, 2001.

[6] "An architectural blueprint for autonomic computing," IBM white paper, 2006, http://www.mendeley.com/research/an-architectural-blueprint-for-autonomic-computing/.

[7] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.

[8] P. Kabiri and A. Ghorbani, "Research on intrusion detection and response: A survey," *International Journal of Network Security*, vol. 1, no. 2, pp. 84–102, 2005.

[9] D. Dasgupta, Z. Ji, and F. Gonzalez, "Artificial immune system (ais) research in the last five years," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 1. IEEE, 2003, pp. 123–130.

[10] D. M. Chess, C. Palmer, and S. R. White, "Security in an autonomic computing environment," *IBM System Journal*, vol. 42, no. 1, pp. 107–118, 2003.

[11] 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.

[12] 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.

[13] M. Atighetchi, P. Pal, F. Webber, R. Schantz, C. Jones, and J. Loyall, "Adaptive cyberdefense for survival and intrusion tolerance," *Internet Computing, IEEE*, vol. 8, no. 6, pp. 25–33, 2004.

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

[15] W. Cui, M. Peinado, H. Wang, and M. Locasto, "Shieldgen: Automatic data patch generation for unknown vulnerabilities with informed probing," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 252–266.

[16] S. Fickas and M. S. Feather, “Requirements monitoring in dynamic environments,” in *Proc. of the Int. Symp. on Requirements Eng.*, ser. RE ’95, 1995, pp. 140–147.

[17] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, “Awareness requirements for adaptive systems,” in *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems*, ser. SEAMS ’11, 2011, pp. 60–69.

[18] J. Mylopoulos, “Awareness and adaptivity requirements,” 2010, http://www.inf.usi.ch/seams/files/mylopoulosSlides.pdf.

[19] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy goals for requirements-driven adaptation,” in *Proc. of Int. Requirements Eng. Conference (RE)*. IEEE, 2010, pp. 125–134.

[20] M. Salehie and L. Tahvildari, “Towards a goal-driven approach to action selection in self-adaptive software,” *Software: Practice and Experience*, p. TBA, 2011, http://onlinelibrary.wiley.com/doi/10.1002/spe.1066/pdf.

[21] W. Sikora, M. Carpenter, and J. Wright, “Smart grid and AMI security concerns,” 2009, http://inguardians.com/pubs/Smart_Grid_AMI_Security_Concerns-20090723.pdf.

[22] S. McLaughlin, D. Podkuiko, and P. McDaniel, “Energy theft in the advanced metering infrastructure,” *Critical Information Infrastructures Security*, pp. 176–187, 2010.

[23] NIST, “Guidelines for smart grid cyber security,” 2010, http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7628.

[24] R. Robinson and M. Stuber, “Risk analysis for advanced metering,” 2008, itron engineering white paper.

[25] K. Stouffer, J. Falco, and K. Scarfone, “Guide to industrial control systems (ics) security (nist800-82),” http://csrc.nist.gov/publications/drafts/800-82/draft_sp800-82-fpd.pdf.

[26] R. Berthier, W. Sanders, and H. Khurana, “Intrusion detection for advanced metering infrastructures: Requirements and architectural directions,” in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 350–355.