2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems

Ontology-based Security Adaptation at Run-time

Antti Evesti and Eila Ovaska
VTT Technical Research Centre of Finland
Finland
e-mail: antti.evesti@vtt.fi, eila.ovaska@vtt.fi

***Abstract*—This paper describes how software is able to autonomously adapt its security mechanisms based on knowledge from security ontology. Security adaptation is required because a software's environment changes during run-time. Thus, all security requirements cannot be defined beforehand. To achieve security adaptation, we have combined a security ontology that defines security mechanisms, security objectives, and high level security measurements. The run-time security adaptation utilises this security ontology to adapt security mechanisms or their parameters to fulfil security requirements for each environment and usage situation. The novelty of this approach comes from the utilisation of ontologies and security measurements, which makes adaptation flexible. We validate our security adaptation with a case study in a smart space environment. The case study proves that security adaptation is able to work autonomously without other user actions.**

Keywords-component; Security ontology, dynamic adaptation, quality management

I. INTRODUCTION

Nowadays, it is not enough to decide the security mechanisms used only at the time of a software's design. Present requirements demand that the software product is able to select the most suitable security mechanisms at run-time. This is so because many applications are executed in a constantly changing environment and it is therefore not possible to define all of these changes beforehand. In addition, all security mechanisms are not applicable in all possible environments, i.e. security mechanism is not supported or mechanism requires too much calculation capacity – causing a need to adapt application at run-time. Mobile devices used in smart environments face this kind of situations in particular. Although, the environment changes a user wants to preserve the particular security level. Therefore, run-time security adaptation ensures that the user is able to concentrate to application usage – instead of configuring its security features constantly. Moreover, an application with adaptation capabilities is able to offer increased performance because security mechanisms are used only when needed – not all the time.

Few approaches exist where changing security requirements and attack situations are the focal point. In [1] Security Adaptation Management (SAM) is presented. The purpose of SAM is to allow more performance and a less secure state to the system when it is not under an attack. When the number of attacks increases the SAM adapts the system to use different implementations to achieve a more secure state. This work utilises the term adaptation spaces for defining possible security breaches and related countermeasures. SAM concentrates on buffer overflow cases and defends against them. Alia et al. present an adaptation model for handling trade-offs between QoS (Quality of Service) and security concerns in [2]. The adaptation model consists of a component framework model, the context in a general sense, adaptability dimensions, user's preferences, and a utility function for describing an adaptation strategy. The utility function selects which application variant will be used in each situation – the selection is based on the user's preferences and context information. Lamprecht et al. presents a Secure Socket Layer (SSL) based run-time security adaptation in [3, 4] – called an adaptive SSL. The adaptive SSL selects an appropriate security mechanism for SSL session, i.e., performs a renegotiation, when the environment changes. The main interest of the authors is security adaptation from the resource consumption and performance viewpoint. However, a threat level is also mentioned as a possible trigger for the adaptation. Myllärniemi et al. present in [5] a security variability approach called KumbangSec – which is intended to work at an architectural level. With this kind of approach the purpose is to derive a different product for different security requirements. Therefore, the security mechanism is bound at design-time and changes are not possible afterwards. Hence, this kind of approach is insufficient in those cases when the security requirements of an application or security threats change during run-time.

In this paper we present our work towards applications which are able to adapt their security mechanisms at run-time. Security is a complex area, containing several bindings to other quality attributes, like reliability and performance. In addition, some security objectives are contradicting, like non-repudiation and privacy. These complexities are reason to utilise ontologies to describe security. Our adaptation approach is based on a security ontology with measurements, and context information representing changes in the environment and usage of application. The novelty of our approach comes from the utilisation of ontologies and security measurements, which ensures flexible and autonomous adaptation also in resource-restricted systems – like mobile devices used in smart spaces. In addition, the approach takes an asset's value into account that reflects directly to the security requirements of an application. The ontology is used to describe security mechanisms, their properties and supported security objectives, whereas security measurements are triggers for the security adaptation. In this work, we adopt security measurements

978-0-7695-4232-4/10 $26.00 © 2010 IEEE
DOI 10.1109/SASO.2010.11

from risk management approaches. Furthermore, utilisation of context information makes it possible to recognise different security requirements for the same application depending on its usage. This is essential when the same application is used for different purposes. Moreover, the approach is applicable for different threats.

After the introduction, background information is given. Thereafter in section 3 parts of the used security ontology and security measurements are presented. Section 4 presents our adaptation approach. Section 5 gives a case example and section 6 contains discussion and future work ideas. Finally, conclusions close the paper.

II. BACKGROUND

ISO/IEC defines security in [6] as follows: The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them. Furthermore, in some sources security is thought to be a composition of confidentiality, integrity and availability [7, 8]. Common criteria [8] list security failures for these attributes as unauthorised disclosure, modification, and loss of use, respectively. [8] defines an asset as an entity that someone presumably places value upon. This means that almost anything can be an asset. However, in this work we define an asset as data stored in a device or a message in a communication channel – as depicted in Fig. 1.

Risk management is the process of identifying risk, assessing risk, and taking steps to reduce risk to an acceptable level [9]. Risk management utilises threat likelihood and the impact caused by a threat exercise to calculate a risk level. The NIST (National Institute of Standards and Technology) integrates risk management to the SDLC (Software Development Life Cycle), i.e. 1) initiation, 2) development or acquisition, 3) implementation, 4) operation or maintenance, and 5) disposal in [9]. Naturally, phase number four – operation or maintenance – can be thought of as a phase where run-time security adaptation is applicable. The NIST 800-30 [9] suggests performing risk management activities in phase four periodically or whenever major changes are made to the system or its environment. However, in smart spaces these changes in the execution environment happen constantly, which we thought of as context changes.

There are several definitions for context, for instance from Chen and Kortz [10]: Context is a set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user. In this work we view context from two perspectives. Firstly, the environment context means broadly an application's execution environment, i.e. location, devices and people around, execution platform and so on. Secondly, the usage context means the purpose of an application's usage, i.e. entertainment, work and so on.

Fig. 2 presents our vision of run-time security management published in [11]. In this initial vision, service discovery, adaptation, measuring, and reasoning phases form a sequential process with a feedback loop. Adaptation is performed based on available services, their security properties, and the desired security. Next, the achieved security is monitored by means of measuring. The achieved security value acts as an input for the reasoning activity. If the desired security is not achieved, the reasoning phase calls the adaptation and an alternative security mechanism is selected. The context affects the desired and achieved security levels as well as available security mechanisms and resources. Now in this work, we concentrate on the adaptation phase and its prerequisites.

Adaptation is the ability of software to adapt its functionality according to the environment and usage context, refined from [12]. In this work, functionality means those security mechanisms intended to achieve a particular level of security.

In [13] Zhou defines ontology and its possible use as follows: Ontology is a shared knowledge standard or knowledge model defining primitive concepts, relations, rules and their instances which comprise topic knowledge. It can be used for capturing, structuring and enlarging explicit and tacit topic knowledge across people, organizations and computer and software systems. There are several security ontologies available for different purposes, listed and compared, for example in [14]. Our earlier work also compares a few security ontologies from the viewpoint of run-time applicability [11]. The existing security ontologies are intended to be used in design-time – like ontologies in [15, 16] – or alternatively, for service discovery and matchmaking purposes – like those ontologies in [17, 18]. Added to these, Herzog et al. presented a comprehensive ontology of information security in [19].

Based on Savola et al. [20] security metrics are used for decision support, especially in risk management for mitigating, cancelling or neglecting security risks. Therefore, metrics that might be useful for different purposes will be associated with risk analysis. Security metrics and measurements can be used for decision support, especially in assessment and prediction [20]. In this work, security measurements are used to detect the difference between the required and achieved security level of an application.

Figure 1. Assets in our work

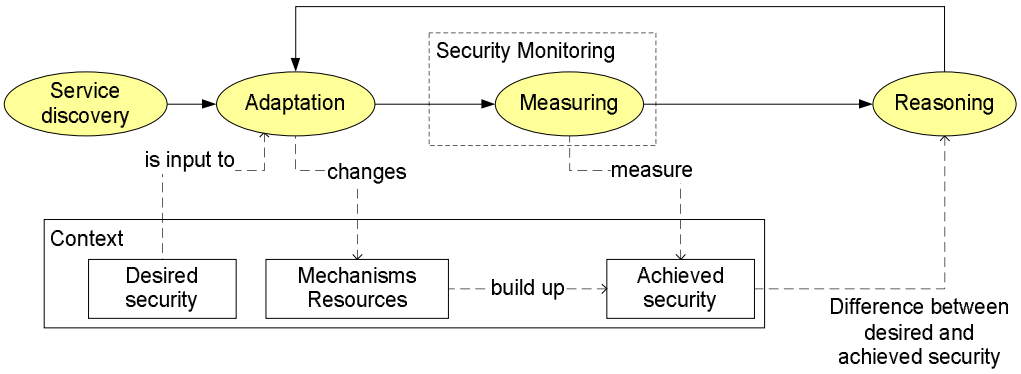

Figure 2. Vision of run-time security management

III. Security Ontology and Measurements

Security adaptation at run-time requires that devices and applications in these devices are able to communicate and exchange security-related information. To make this communication possible security ontologies are needed. Support for run-time security adaptation requires that the security ontology contains security mechanisms, objectives, security measurements, and connections between these. From these concepts, the connections are essential for adaptation, since they make it possible to know which objectives, i.e. high level requirements, can be achieved with a particular mechanism. The security ontology utilised in this work adopts security objectives and mechanisms from [18, 19]. However, measurement part conform the ontology from Savolainen et al. [16]. TABLE I. presents an initial part of a security ontology combined for this work. The first column lists classes, the second column lists the properties of each class and their ranges, and the third column gives examples of possible instances. The ontology definition is work in progress but the current version presents the most important concepts from the run-time adaptation viewpoint. Ontologies in [18, 19] contain dozens of objectives and security mechanisms – all of these are important but copying them to this paper is not reasonable at this point.

Some classes of the security ontology are quite straightforward – listing mechanisms etc. Whereas, some other classes are more complex, like classes related to security measurements. For this security ontology, we defined risk level measurements for confidentiality and integrity based on the proposal in [21] presented in TABLE I. which conforms to the idea of a risk-level matrix from [9]. The vertical axis denotes a threat's likelihood and the horizontal axis means the severity of consequences. We posit that the severity of consequences is an analogue for an asset's value defined in the security ontology – in NIST 800-30 this is defined as impact [9].

For defining likelihoods of threats we have to know what these threats are, and thus we decided to use the following high level threats for confidentiality and integrity respectively: "unauthorised disclosure of information" and "unauthorised modification of information" defined in [8]. Based on these threat definitions and the table above we can define separated risk measurements for confidentiality and integrity as follows:

$$RiskForConfidentiality = Tl * Av \quad (1)$$

$$RiskForIntegrity = Tl * Av \quad (2)$$

Where *Tl* means threat's likelihood and *Av* is an asset's value. Hence, values for these risk levels are calculated in a similar way as TABLE I. shows but it is notable that the values are in the interval [0, 100] not only as values presented in the table. However, we map these numerical values to the risk levels – similarly as made in [21], i.e. low (0-10), moderate (11-39), and high (40-100).

TABLE I. Part of Security Ontology

Class	Properties {Range}	Possible instances
SecurityAsset	hasValue {[0, 100]}	Message Data
SecurityObjective	achievableWith {securityConcept} fulfilmentMeasured-With {SecurityMeasurement}	Confidentiality Integrity Availability Authentication
SecurityConcept	support {securityObjective} utilises {credential}	AES and DES support Confidentiality MD5 supports Integrity
Credential		Fingerprint, Password
SecurityMeasurement	hasValue measuredObjective {securityObjective}	RiskForConfidentiality RiskForIntegrity
Device	hasSecurityConcept {securityConcept} hasCurrentlyUsed SecurityConcept {SecurityConcept} hasMeasuredSecurity {SecurityMeasurement}	MobileDevice with entertainment application

TABLE II. RISK LEVELS ADOPTED FROM [18]

Threat Likelihood	Severity of consequences			
	Insignificant (10)	Marginal (40)	Critical (70)	Catastrophic (100)
Incredible (0.1) Level 1	10 *0.1=1	40*0.1=4	70*0.1=7	100*0.1=10
Improbable (0.2) Level 2	10 *0.2=2	40*0.2=8	70*0.2=14	100*0.2=20
Remote (0.4) Level 3	10 *0.4=4	40*0.4=16	70*0.4=28	100*0.4=40
Occasional (.0.6) Level 4	10 *0.6=6	40*0.6=24	70*0.6=42	100*0.6=60
Probable (0.8) Level 5	10 *0.8=8	40*0.8=32	70*0.8=56	100*0.8=80
Frequent (1.0) Level 6	10 *1.0=10	40*1=40	70*1=70	100*1=100

As said in section 2, an asset can be almost anything but in this work an asset is defined as a message in a communication channel or data stored in a device. However, an asset's value (*Av*) for its owner depends on the content and usage of the asset. Hence, we do not try to define these asset values in this work – instead this information will be collected from user preferences. For calculating threat likelihood *Tl* we propose the following equation:

$$\begin{cases} Tl = 0, \text{ when } Da \geq Ia \\ Tl = Ia - Da, \text{ when } Da < Ia \end{cases} \quad (3)$$

Ia means actions that increase the threat likelihood and *Da* actions that decrease the likelihood. At the moment, we do not try to give numeric values for *Da* and *Ia* variables – instead variables get values describing how many levels they affect to the threat likelihood. As TABLE I. shows *Tl* gets values from the interval [0, 1], and these values are mapped to six levels from incredible (level 1) to frequent (level 6). Naturally, when *Da* is bigger or equals with *Ia* then *Tl* gets value 0, which maps to the incredible level. In this time, we make an assumption that *Ia* and *Da* are independent from each other, e.g. change in a *Da* value does not affect to an *Ia* value. In order to achieve automatic adaptation, calculation of this formula has to be made automatically during the execution based on information stored in the ontology and context information collected from the environment. It is important to bear in mind that we can give exact definitions for *Tl* values 0 and 1 as Hunstad et al. did in their work [22] for probabilistic security values, but precise values between these end points are more complicated as they mention. Hence, we know that threat likelihood 0 means that there is no possibility for threat's realisation, and likelihood 1 means that the threat will certainly be realised.

Utilisation of security mechanisms affects the variable *Da* (Decreasing actions). Therefore, the security ontology has to contain a connection from mechanisms to *Da* variable in the measurement class. For instance, measurement *RiskForConfidentiality* is affected by encryption algorithms like AES (Advanced Encryption Standard) and DES (Data Encryption Standard). Thus, the device software has to select the appropriate encryption mechanism and its parameters to decrease the *RiskForConfidentiality* measurement's value. TABLE III. contains security mechanism assessment, i.e. how many levels mechanisms reduce threat likelihood. Humstad et al. [22] note that the rating of security functions is a difficult task. Thus, it is impossible to give exact values how much a particular security mechanism decreases the particular threat. However, we defined initial *Da* values for security mechanisms in the security ontology in order to test how well the security ontology is able to support run-time security adaptation. The current usage of security mechanisms can give suggestive values for *Da* value selection, for example Anderson mentions in [23] that NSA (National Security Agency) approved AES with 128 bit keys for secret purposes and AES with 256 bit keys for top secret purposes. However, values in TABLE III. change when time goes on because vulnerabilities will be found and attackers learn new ways to attack. In addition, use cases in different contexts will offer valuable feedback for assessing these values. Thus, values in the ontology have to be updated when new knowledge appears.

On the contrary, environment issues affect the variable *Ia* (Increasing actions). Basically these can be anything that happens in the application's environment, i.e. knowing the context is essential to elicit this information. Added to this, the following information can be also used when estimating the increasing actions variable (*Ia*): asset's value because it may be a motive for an attacker, information from vulnerability databases, and log information from intrusion detection systems (IDS). In any case, it is better to overestimate this variable. 0gives examples of events that increase threat likelihood *Tl*.

As a conclusion the security ontology describes security objectives, mechanisms supporting these objectives, and risk based measurements for measuring fulfilment of these objectives. Therefore, the security ontology contains essential information for application adaptation – presented in the next section.

TABLE III. INITIAL SECURITY MECHANISM ASSESSMENT

Security mechanism / parameters for confidentiality	Initial *Da* value
AES 256 bit	Reduce 5 Levels
AES 192 bit	Reduce 4 levels
AES 128 bit	Reduce 3 levels
DES	Reduce 1 level
Security mechanism for integrity	**Initial Da value**
SHA-2	Reduce 4 levels
SHA-1	Reduce 3 levels
MD5	Reduce 2 levels
MD4	Reduce 1 level

TABLE IV. INCREASING ACTIONS

Event in the environment	Initial Ia value
Untrusted agent in the environment	Increase 2 levels
Abnormal behaviour of agent	Increase 3 levels

IV. APPLICATION ADAPTATION

The adaptation takes place at the start-up and run-time phases alike. Clearly, the start-up phase adaptation is performed when an application is started. The run-time phase adaptation takes place during application execution – when a major change occurs in the environment or usage of an application. In practice, both adaptation cases are similar from the adaptation viewpoint because similar input information collection and mechanism selection activities are required in both cases. However, the adaptation during the run-time phase requires more sophisticated mechanisms from an implementation perspective, e.g. re-establishing network connections, when compared to the adaptation at start-up phase. In our adaptation approach, the above described security ontology and security measurements are used in both start-up and run-time adaptation phases. The approach is initially intended for smart spaces but the purpose is that it can also be utilised in other environments. The block diagram of the application's adaptation is presented in Fig. 3 – containing both start-up and run-time phase adaptations. The diagram shows the actions required to perform before or during the action of adaptation. In the figure, rectangles mean information and rounded rectangles depict actions. Grey boxes represent information which is applicable only during the run-time phase. Furthermore, in the start-up phase we use the term collecting input information instead of the monitoring term, used at the run-time phase.

A. Start-up Phase Adaptation

The first action is elicitation of security requirements and the required levels for these requirements. The context of application affects these, especially the context related to the usage of application. As an example, entertainment usage and surveillance usage of an application elicit different security requirements and related levels. For entertainment usage, only availability matters and its required level is quite low. This is contrary to surveillance usage which requires availability and integrity at a critical level – adopting terms from TABLE I. Thus, an application gets a list of security requirements and levels. After that, security mechanisms supporting these requirements can be retrieved from the security ontology. Retrieving supporting mechanisms from the security ontology is possible because ontology contains connection from objectives to mechanisms, i.e. *achievableWith* property presented in TABLE I. For instance, if confidentiality and integrity is required retrieving ontology produces a list of supporting mechanisms – like AES, DES, and MD5.

Next, a control analysis action is performed. This is required because the security ontology is intended to be general, and thus may contain mechanisms not implemented in an application or alternatively not applicable in the current environment. The outcome from the control analysis action is a set of applicable security mechanisms. The final action in the collecting input information activity is measuring. In this action, the forthcoming security of the application is measured, meaning the security level that will be achieved without any security mechanisms or alternatively with default mechanisms defined in user preferences when the application starts.

As a conclusion, we have the following information available for the adaptation: security requirements and levels, applicable security mechanisms, and measured security levels. The adaptation action is executed if the measured security fails to reach the requirements set. Based on input information, the adaptation action selects the security mechanisms to be used and their parameters.

B. Run-time Phase Adaptation

The run-time adaptation phase is for the most part similar to the above described start-up phase adaptation. However, some differences exist. Firstly, the adaptation action is triggered based on monitoring results. Thus, the adaptation is performed when required – not only periodically at a moment defined earlier. Secondly, the manner in which adaptation takes place differs when the application is already running, for instance re-establishing network connection might be needed. The monitoring activity monitors earlier mentioned context changes, i.e. changes in usage context and environment context.

The change in the usage of an application may cause change in security requirements or their levels. Consequently, the currently used security mechanisms do not fulfil these new security requirements – recognised by means of measuring. The security ontology defines connection from security objectives to security measurements, i.e. *fulfilmentMeasuredWith* property presented in TABLE I. Thus, ontology offers knowledge needed to measure requirements fulfilment.

Another possibility for changes comes from the environment. These changes affect how well the required securities are met – not the requirements themselves. For instance, a new device appears that poses a threat to the application's operation, or alternatively, the environment changes in a way that prevents the usage of the earlier selected security mechanism. Both of these context changes can create a need to adapt security mechanisms or the parameters of the application.

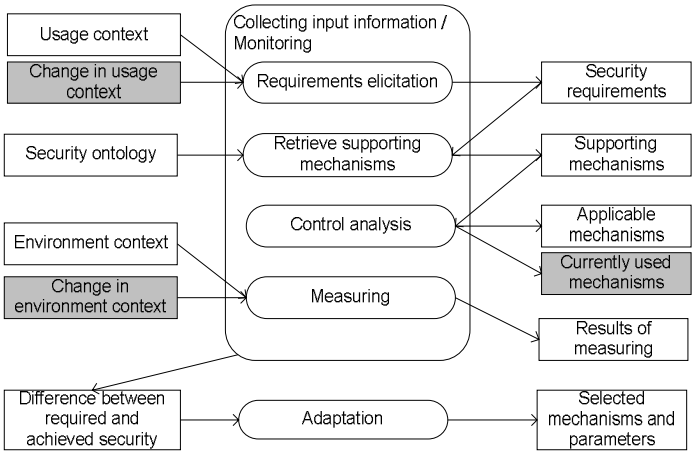

Figure 3. Adaptation block diagram

V. CASE EXAMPLE

We tested our run-time adaptation with a greenhouse demonstration [24]. The demonstrator does not cover the whole adaptation presented in the previous section, i.e. automatic requirements elicitation is not yet implemented – which is required when usage context changes. However, start-up and run-time adaptation is built with the above described ontology and measurements, and changes in the environment context. Therefore, demonstrator measures requirements fulfilment and adapts application accordingly. The smart greenhouse demonstrator contains a miniature greenhouse with several sensors and actuators. In addition, the demonstrator contains two stakeholder groups, i.e. a gardener and the customers of the greenhouse. The run-time adaptation is performed in the gardener application, running in a mobile device. Hence, gardener's device uses security mechanisms only when needed, and thus, is able to save resources.

The demonstrator is built upon the Smart-M3 platform [25] – intended to establish smart spaces where devices can use information and its semantics. The main concepts in the Smart-M3 are SIB (Semantic Information Broker) and agents. SIB mediates information between agents which produce and consume information. In other words, agents do not communicate directly with each other. In the demonstrator, SIB is running in a Linux laptop offering a wireless TCP connection for agents.

A. Start-up Phase Adaptation

Firstly, a gardener arrives at the greenhouse, which is his place of work. The greenhouse smart space authenticates the gardener and assigns him worker privileges. Thus, the gardener is able to look at sensor values from the greenhouse and to change actuator states. The gardener performs these actions with his mobile device, Nokia N810 Internet tablet, containing the gardener's application implemented by Python – user interface is presented in Fig. 4. The first security adaptation takes place immediately when the gardener receives his worker privileges and joins the greenhouse smart space, i.e. security adaptation at start-up phase. The adaptation goes as presented in Fig. 3 – thus the required securities and levels are first decided. The application is intended for working purposes. Therefore, both confidentiality and integrity are critical security requirements and related risk levels have to be in the low level. For these requirements, the security ontology returns the following list of mechanisms for the gardener's application: AES or DES for confidentiality and MD5 for integrity. However, control analysis reveals that the gardener's application does not support DES encryption, and thus the applicable mechanisms are AES and MD5.

The final thing before adaptation is security measuring – based on measurements presented in section 3. This moment, the greenhouse smart space contains only trusted agents, i.e. agents for sensors and actuators. Thus, threat likelihood variable *Tl* – related to confidentiality – in equation 3 remains in level 1 (incredible) because environment does not contain any increasing actions (*Ia*). However, the consequences of possible security breaches might be remarkable harmful to the gardener or the greenhouse, and thus, the severity of consequences (Asset Value *Av*) is in a critical level, when compared to TABLE I. With these values we get following *RiskForConfidentiality* measurement, as TABLE I. shows:

*Incredible (0.1) * Critical (70) = Low level (7)*

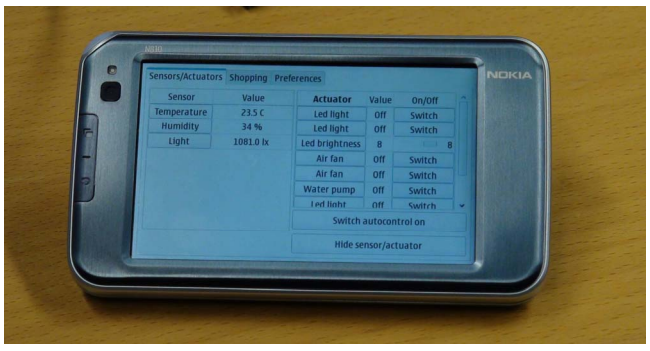

Figure 4. Gardener application in Nokia N810

Similarly a value for *RiskForIntegrity* measurement can be calculated by utilising the same asset value, i.e. critical. However, the threat likelihood (*Tl*) value for integrity risks differs from confidentiality because there has been a possibility that some untrusted third party device has manipulated sensor values before the gardener arrived. In 0it is defined that untrusted agent in the environment increases *Ia* value with 2 levels. Thus, *Tl* increases from the first level to the third level, i.e. to remote level. Hence, utilising equation 2 *RiskForIntegrity* measurement gets value 28 as follows.

*Remote (0.4) * Critical (70) = Moderate level (28)*

Based on these calculations, the gardener's application makes a decision to use hash function (MD5) in communication to ensure integrity, but without encryption because the *RiskForConfidentiality* was below 10, i.e. in the low level. Utilisation of MD5 hash function decreases *Tl* value to level 1 and ensures that *RiskForIntegrity* value also decrease to the low level, which was required. The upper sequence diagram in Fig. 5 shows these start-up adaptation actions.

B. Run-time Phase Adaptation

In the second phase, the retail area of the greenhouse opens and customers arrive to the greenhouse, causing a need for run-time adaptation. The monitoring action recognises that the environment context changes due to these customers and their mobile devices joining to the greenhouse smart space. The required security does not change because the context change comes from environment. However, the arriving customers cause the value of the *RiskForConfidentiality* measurement to increase. This is because customers' mobile devices are regarded as untrusted agents, i.e., an increasing action (*Ia*) in equation 3. *Tl* value increases from the first level to the third level and asset value *Av* remains at the critical level – as described earlier. Using equation 1 *RiskForConfidentiality* measurement reveals that risk level is raised to the moderate level. Hence, the required confidentiality level is no longer reached – causing a need to run-time phase adaptation.

Next, retrieval of supporting mechanisms and control analysis actions are performed. It is also possible to use the security mechanisms list retrieved earlier, in the start-up phase, but it may contain mechanisms already cracked. In this phase, the control analysis action returns two separated lists, i.e. applicable mechanisms (AES) and currently used mechanisms (MD5). Based on this, AES encryption with a key length of 128 bits is selected for ensuring confidentiality. TABLE III. showed that this mechanism decreases *Tl* value three levels, which is enough to reaching risk level low. The lower sequence diagram in Fig. 5 shows the run-time phase adaptation.

The case example showed that it is possible to adapt an application both at the start-up and run-time phases. The security monitoring is able to detect changes in the current security level by means of measuring. Thus, adaptation is performed when required, which ensures selection of the optimal security mechanism.

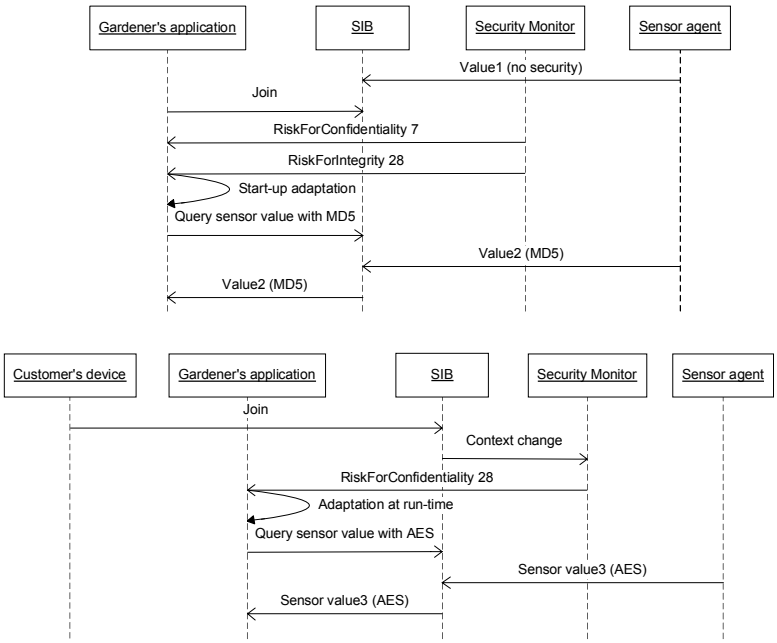

Figure 5. Sequence diagrams for start-up phase and run-time pahse adaptation.

VI. Discussion and Future Work

It is possible to find some essential differences when comparing our approach to the related work mentioned in the introduction section. There are, however, also a few similarities. The Security Adaptation Management (SAM) [1] adapts the used implementation if the current one does not fulfil the required security. However, the adaptation is intended to take place in attack and buffer overflow situations. In some cases, recognising an attack is difficult, or alternatively, when an attack is recognised it might be too late to adapt the software. Our approach recognises security related risks that might cause security breaches, which makes it possible to operate proactively. SAM uses adaptation spaces for defining security breaches and their countermeasures. In our work, this information is stored in the security ontology by using security requirements and mechanisms. The adaptation model presented in [2] concentrates strongly on trade-offs. The presented model makes it possible to select an application variant that best satisfies the user preferences in the current context. In this adaptation model, the user gives the required securities and their importance levels in user preferences. By contrast, our approach elicits these requirements from context information. The adaptive SSL presented by Lamprecht et al. [3, 4] adapts security mechanisms of the SSL session. Performance is a trigger for the adaptation. Authors mention a possibility to use a threat level as a trigger for adaptation but this area is not covered yet. Naturally, the adaptive SSL concentrates to adapt security mechanisms of communication. Thus, it is not able to adapt security mechanisms used to protect assets in a user device. Finally, when compared to design time security variability approaches like [5] it can be noted that the run-time phase adaptation offers more flexibility, especially in those cases where all security related requirements cannot be defined at design-time. As a conclusion, all of these approaches are intended to select the most suitable security mechanism to fulfil security requirements in different situations. Thus, the most significant difference comes from the techniques used to trigger these adaptations.

Risk based security measurements ensure that our approach is able to work in different threat situations. In addition, it is easy to add new security measurements to the ontology. Moreover, current measurements take an asset's value into account. As an example – from our case study domain – data from a temperature sensor is not valuable for a customer but might be very important for the gardener who is making decisions based on this sensor data. Thus, an asset's value is thought to be an important factor in adaptation. The adaptation utilises context information to observe changes in environment and usage of the application. At this moment, changes in the environment context are monitored. However, in the future we will also monitor the usage context of an application that is used to elicit security requirements. The security ontology used in this work is under construction. The ontology is combined from three separate sources containing essential parts for the adaptation, such parts as measurements, mechanisms and supported security objectives. Nevertheless, additions and refinements are still needed, especially related to measurements, context issues, and connections between different concepts. Future studies will concentrate on security measuring and context issues and how different contexts affect the required and achieved securities.

Additional research is also needed in security measurements and their calibration techniques. Measurements related to security mechanisms also facilitate evaluation of how good a particular mechanism really is – because in this work we were obliged to use estimated values in calculations. Moreover, it is important to make performance tests, in order to see how much memory and CPU consumption our approach demands. However, the performed case example does not reveal any major overheads. Also, negotiation techniques between devices should be taken into account because security mechanisms used in a communication requires that both sides support a selected technique. In our case example gardener's application and SIB supported same security mechanisms but sometimes more sophisticated negotiation techniques will be needed. Applicable solutions for the negotiations can be found for instance from existing service matchmaking approaches.

The presented case example is based on the first laboratory experience. The case example is intended to give an overview of the applicability of the adaptation approach and facilitate the future field tests. Currently, we are implementing the wider demonstrator. The second demonstrator contains also requirements decision part based on the usage context of the application. In addition, the second demonstrator utilises more sophisticated security measures as a trigger for an adaptation action. Therefore, the results from the second demonstrator offer valuable input for the evaluations related to feasibility of adaptation decisions and performance overhead.

VII. Conclusions

In this work we presented our run-time security adaptation approach that utilises security ontology and measurements. The high level risk measurements for confidentiality and integrity are presented. Adaptation takes place either at an application's start-up or during the run-time phase. In addition, adaptation is performed at the mechanism and parameter levels, which is essential for many security mechanisms. The designed security ontology supports the adaptation approach and offers a possibility for future changes and extensions. The case example showed that software is able to adapt its security mechanisms during run-time even in a mobile device.

Acknowledgment

This work has been carried out in the ARTEMIS JU SOFIA project funded by Tekes, VTT, and the European Commission.

REFERENCES

[1] H. Hinton, C. Cowan, L. Delcambre and S. Bowers. "SAM: Security Adaptation Manager," *Proceedings of 15th Annual Computer Security Applications Conference 1999 (ACSAC '99),* pp. 361-370, 1999.

[2] M. Alia and M. Lacoste. "A QoS and security adaptation model for autonomic pervasive systems," *32nd Annual IEEE International Computer Software and Applications Conference, COMPSAC 2008,* pp. 943-948, 2008.

[3] C. J. Lamprecht and A. P. A. van Moorsel. "Runtime Security Adaptation Using Adaptive SSL," *Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium,* pp. 305-312, 2008.

[4] C. J. Lamprecht and A. P. A. van Moorsel. "Adaptive SSL: Design, Implementation and Overhead Analysis," *First International Conference on Self-Adaptive and Self-Organizing Systems, 2007. SASO '07.,* pp. 289-294, 2007.

[5] V. Myllärniemi, M. Raatikainen and T. Männistö. "KumbangSec: An approach for modelling functional and security variability in software architectures," *First International Workshop on Variability Modelling of Software-Intensive Systems,* pp. 61-70, 2007.

[6] ISO/IEC 9126-1:2001. *Software Engineering - Product Quality - Part 1: Quality Model.* 2001,

[7] A. Avižienis, J. -. Laprie, B. Randell and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing,* vol. 1, pp. 11-33, 2004.

[8] ISO/IEC 15408-1:2009, *Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and General Model.* International Organization of Standardization, 2009,

[9] G. Stoneburner, A. Goguen and A. Feringa. "Risk management guide for information technology systems," *Special Publication 800-30,* 2002.

[10] G. Chen and D. Kotz. "A Survey of Context-Aware Mobile Computing Research," *Technical Report TR2000-381*2000.

[11] A. Evesti, E. Ovaska and R. Savola, "From security modelling to run-time security monitoring," *European Workshop on Security in Model Driven Architecture (SECMDA),* pp. 33-41, 23 - 26 Jun. 2009. 2009.

[12] M. Matinlassi and E. Niemelä. "The impact of maintainability on component-based software systems," *29th Euromicro Conference,* pp. 25-32, 2003.

[13] J. Zhou. "Knowledge Dichotomy and Semantic Knowledge Management," *Industrial Applications of Semantic Web,* pp. 305-316, 2005.

[14] C. Blanco, J. Lasheras, R. Valencia-García, E. Fernández-Medina, A. Toval and M. Piattini. "A systematic review and comparison of security ontologies," *3rd International Conference on Availability, Security, and Reliability (ARES 2008),* pp. 813-820, 2008.

[15] B. Tsoumas and D. Gritzalis. "Towards an Ontology-based Security Management," *20th Advanced Information Networking and Applications 2006 (AINA 2006),* pp. 985-992, 2006.

[16] P. Savolainen, E. Niemelä and R. Savola. "A taxonomy of information security for service centric systems," *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007),* pp. 5-12, 2007.

[17] G. Denker, L. Kagal and T. Finin. "Security in the Semantic Web using OWL," *Information Security Technical Report 10(1),* pp. 51-58. 2005.

[18] A. Kim, J. Luo and M. Kang. "Security Ontology for annotating resources," *LNCS,* vol. 3761, pp. 1483-1499, 2005.

[19] A. Herzog, N. Shahmehri and C. Duma. "An ontology of information security," *Techniques and Applications for Advanced Information Privacy and Security: Emerging Organizational, Ethical, and Human Issues* pp. 278-301. 2009.

[20] R. M. Savola and H. Abie. "On-Line and off-line security measurement framework for mobile ad hoc networks," *Journal of Networks, 4(7),* pp. 565-579, 2009.

[21] D. S. Herrmann. *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI.* 2007,

[22] A. Hunstad, J. Hallberg and R. Andersson. "Measuring IT security - A method based on Common Criteria's security functional requirements," *5th Annual IEEE System, Man and Cybernetics Information Assurance Workshop (SMC),* pp. 226-233, 2004.

[23] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* (2nd ed.) 2008,

[24] A. Evesti, M. Eteläperä, J. Kiljander, J. Kuusijärvi, A. Purhonen and S. Stenudd, "Semantic Information Interoperability in Smart Spaces," *8th International Conference on Mobile and Ubiquitous Multimedia (MUM'09),* 22 - 25 Nov. 2009.

[25] Smart-M3, http://sourceforge.net/projects/smart-m3/