Future Internet **2015**, *7*, 225-256; doi:10.3390/fi7030225

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

EDAS: An Evaluation Prototype for Autonomic Event-Driven Adaptive Security in the Internet of Things

Waqas Aman * and Einar Snekkenes

Norwegian Information Security Laboratory (NISLab), Gjøvik University College, 2815 Gjøvik, Norway; E-Mail: einar.snekkenes@hig.no

* Author to whom correspondence should be addressed; E-Mail: waqas.aman@hig.no; Tel.: +47-611-35392.

Academic Editor: Steven Furnell

Received: 11 March 2015 / Accepted: 23 June 2015 / Published: 8 July 2015

Abstract: In Internet of Things (IoT), the main driving technologies are considered to be tiny sensory objects. These objects cannot host traditional preventive and detective technologies to provide protection against the increasing threat sophistication. Furthermore, these solutions are limited to analyzing particular contextual information, for instance network information or files, and do not provide holistic context for risk analysis and response. Analyzing a part of a situation may lead to false alarms and later to unnecessary and incorrect configurations. To overcome these concerns, we proposed an event-driven adaptive security (EDAS) model for IoT. EDAS aims to observe security events (changes) generated by various things in the monitored IoT environment, investigates any intentional or unintentional risks associated with the events and adapts to it autonomously. It correlates different events in time and space to reduce any false alarms and provides a mechanism to predict attacks before they are realized. Risks are responded to autonomically by utilizing a runtime adaptation ontology. The mitigation action is chosen after assessing essential information, such as the risk faced, user preferences, device capabilities and service requirements. Thus, it selects an optimal mitigation action in a particular adverse situation. The objective of this paper is to investigate EDAS feasibility and its aptitude as a real-world prototype in a remote patient monitoring context. It details how EDAS can be a practical choice for IoT-eHealth in terms of the security, design and implementation features it offers as compared to traditional security controls. We have explained the prototype's major components and have highlighted the key technical challenges.

Keywords: Internet of Things; adaptive security; eHealth; event-driven architecture; risk management; event correlation

1. Introduction

A recent report by EMC and the International Data Cooperation (IDC) details that by the year 2020, 30 billion of the connected devices in the world, constituting IoT, will be small and smart wireless devices characterized by their autonomous behavior [1]. They further reason that IoT will create new business models, capture real-time information in critical infrastructure, extend services offered by traditional modes and will provide a global visibility platform. Analyzing this report and other similar research, one can conclude that IoT can bring phenomenal extensions to the ICT-based services offered in the business, as well as in public domains.

The potentials offered by IoT can become more efficacious and reliable if other challenges it faces, such as networking, standardization, security and privacy issues, are carefully sorted out [2]. From a security perspective, a recent study made by OWASP and HP® [3,4] details a number of serious vulnerabilities that IoT has still to address. The report highlights that 60% of the things web interfaces are prone to web-related attacks, such as cross-site scripting (XSS) attacks; 90% of the things collect at least one piece of personal information; 70% of the devices communicate via unencrypted channels; and 70% of the devices are susceptible to account enumeration attacks. These are some severe concerns particularly for IoT-enabled health services, where the type of information communicated is mostly personal.

Current security solutions, like firewalls, intrusion detection systems (IDS), *etc.*, or even small anti-virus programs are not feasible for this sensory, tiny, low-resourced thing-driven network. Even if we somehow tailored a miniature version of them, they still would not achieve much. Because, as standalone mechanisms, they can only assess a particular set of vectors when a host is under a possible compromise situation. A security breach usually consists of multiple and associated attack vectors, means and targets. Analyzing only a part of them independently of their association in a context may yield false positives and negatives [5]. Analyzing risks irrespective of the context may further results in inappropriate mitigation decisions. Consider a scenario where a physician, currently on holiday, using her smartphone is given authorization by a role-based access control (RBAC) system to access patient personal information from an unusual place on a weekend. From an RBAC point of view, this activity seems to be legitimate, and the system should grant access. However, if the entire context, *i.e.*, the unusual place, current status and access time, is analyzed, one can conclude that there is a risk involved if access is granted, *i.e.*, the smartphone might have been compromised. Hence, current preventive and detective security solutions are either not feasible or do not have the intelligence to assess the situation in a holistic context.

In social services enabled by IoT, for instance eHealth concerning remote patient monitoring, mitigation decisions based on inaccurate risk information may disrupt service availability and can be life threatening. Examining the stated facts, statistics and resource-constrained nature of things in IoT, we proposed an event-driven adaptive security architecture (EDAS) [6] in an eHealth scenario where remote patients are continuously monitored. EDAS is an autonomous risk-based adaptive security architecture that monitors and analyzes security events generated by things for potential threats and responds to the corresponding risks by adapting an optimal mitigation action. The mitigation response is selected in a way such that user preferences, service and security requirements are all assessed before a decision is made. For autonomous adaptation, we proposed a runtime security ontology containing the necessary knowledge for optimized adaptation. For event monitoring, collection, analysis and correlation, we listed a variety of techniques that can be utilized. The threat and risk analysis routines are suggested to be performed at a resourceful remote machine to avoid heavy computations on a resource-constrained thing. The things only use their out-of-the-box event framework to generate and communicate events. Events are collected and analyzed to investigate any potential threat that they pose and are correlated in a context to reduce any false positive or negatives.

This paper describes an IoT-eHealth prototype showing how EDAS can be implemented as a real-world artifact. The primary objective is to investigate the feasibility of EDAS as an event-driven security architecture and whether its adaptation control loop adds value to the security of IoT. From the prototyping activity, evaluation and comparison detailed in this paper, we conclude that EDAS can be a more practical choice for IoT security as compared to traditional security controls. It provides a holistic security solution, complies with the resource-constrained nature of things in IoT, provides extensibility, allows existing traditional systems to be monitored and substantially increases the overall throughput of electronic security operations. We detail the prototype's major components individually and how they are utilized collectively to ensure adaptive security. Furthermore, this paper highlights the key technical challenges and discusses how they can be approached. For demonstration purposes, we have adopted OSSIM [7] as an event monitoring and analysis tool. However, we suggest that any appropriate statistical, probabilistic, rule-based or other methods, tools or techniques can be utilized for event monitoring and correlation in the architecture, as long as it has complex event processing (CEP) [8] capabilities. We have already categorized these techniques in [6].

The rest of the paper is organized as follows. In Section 2, an overview of the proposed architecture will be briefly revised. Section 3 explains the major components of the prototype and describes their technical design and features. A case study demonstrating the prototype will be presented in Section 4. The prototype feasibility will be argued and detailed in Section 5 to evaluate the architecture aptitude. Related work is discussed in Section 6. In Section 7, key challenges concerning EDAS development, possible solutions and further work will be discussed. Finally, the paper will be concluded in Section 8.

2. Proposed Architecture

EDAS [6] is an autonomous risk-based event-driven adaptive security architecture for IoT. It monitors security changes, *i.e.*, thing-generated events, in the IoT environment, analyzes the associated threat(s) and adapts appropriate and optimized security configurations against the risk faced. At the abstract level, EDAS complies with the IBM autonomic control loop that consists of the sensing, analyzing, planning and execution components [9]. It consists of two major components: The EDAS platform and the event sources, as shown in Figure 1.

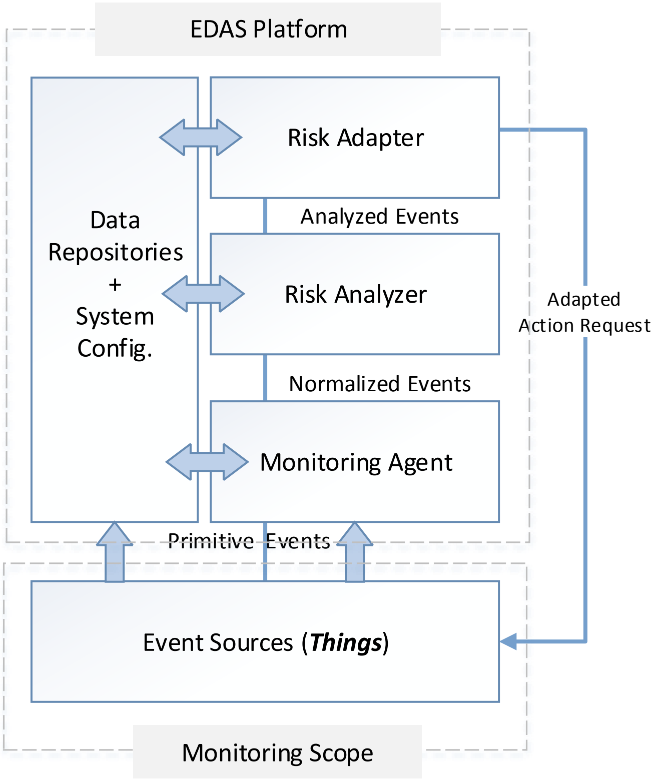


Figure 1. Abstract diagram of event-driven adaptive security architecture (EDAS) architecture.

The event sources reflect the monitored environment consisting of all of the critical things in IoT, *i.e.*, the applications, devices, objects, *etc.*, that are crucial for service delivery. Thus, the scope of the monitored things defines the risk management scope for adaptive security. In a typical IoT-eHealth scenario, as shown in Figure 2, event sources correspond to all of the applications, sensors, smart devices and actuators, both in the patient and hospital domains, that are essential for the reliable, secure and efficacious operations of remote patient monitoring services.

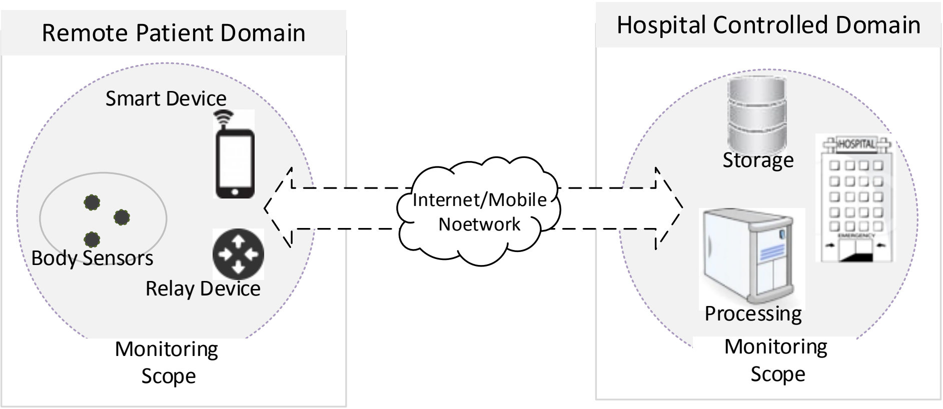


Figure 2. IoT-eHealth environment.

In the proposed architecture context, the event source is any of the mentioned things that can react to any change or event it experiences within itself or in the environment in which it operates. Reaction refers to triggering appropriate actions and processes, generating or logging information that detail the actions taken and communicating these changes and information across the network. This elaboration makes our event source (thing) description more aligned with that of the Cluster of European Research Projects on the IoT (CERP-IoT). According to CERP [10], "things are active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to the real/physical world events and influencing it by running processes that trigger actions and create services with or without direct human intervention".

The EDAS platform consists of a set of methods and tools necessary to continuously monitor and analyze these thing-generated primitive events in a context-aware manner to investigate any potential security threats and associated risks. As events arrive from different sources in different formats, they are filtered and normalized by the *Monitoring Agent* to remove any redundant events and to shape them into a universal format for risk analysis and adaptation. The *Risk Analyzer* investigates potential threats and risks associated with these events using a correlation engine, such that false alarms are avoided. This is ensured by correlating events in space and time in a context [11]. An unacceptable risk and its corresponding details are referred to the *Risk Adapter* where an adaptation engine utilizes a runtime security ontology to select a mitigation action from a pool of possible actions to reduce the risk impact. The selected action is sent to the local adaptor process embedded in the thing (event source) where the new security settings received are applied. The effects of the adapted changes are also recorded, monitored and analyzed again. Hence, security monitoring, analysis and adaptation is done in a continuous autonomic control loop fashion. Necessary information regarding event correlation, risk quantification, system configurations, adaptation requirements, normalization scripts and event databases is accessed and updated along this process via the repository component.

3. EDAS Prototype Specifications

We have developed an evaluation prototype to investigate the feasibility of the EDAS architecture as a real-world implementation. The test settings were designed to reflect a working IoT-eHealth environment where a remote patient with wearable body sensors is monitored from a hospital site. The test environment consists of the following hardware and software components.

At the remote patient domain:

- Libelium open source eHealth Sensor Shield V.2.0
- Arduino Uno R2: A 16-MHz 32 K micro-controller for the eHealth Shield
- RN-XV 171 IEEE 802.11 b/g-compatible Wifi module
- XBee Communication Shield for communicating Arduino serial data over Wifi
- Samsung Galaxy SIII-Mini as a relay device

At the hospital-controlled domain:

- MySQL 5.6.12 as a storage for primitive medical and security events
- Apache 2.4.4 to display real-time patient vital signs enabled with HighCharts API

- Sixty four-bit OSSIM V.4.6.1 as the EDAS monitoring and analysis platform
- A Jena-SPARQL-enabled JAR as an adaptation engine for accessing, modifying and storing the RDF /XML adaptation ontology

3.1. Technical Setup

Encrypted patient body temperature values are collected using a wearable temperature sensor, which are communicated to the hospital site via a smartphone as a relay device. In the patient domain, an app on the smartphone receives, parses and directs temperature readings and any security event generated by the sensor to their respective event databases. The temperature values are extracted by the hospital health application, where they are decrypted and displayed as a continuous real-time graph for medical diagnosis. The security events are pulled by the EDAS platform to investigate and respond to any potential threats. The EDAS platform is a standalone system contained in the hospital-controlled domain. A context diagram of the environment is shown in Figure 3.

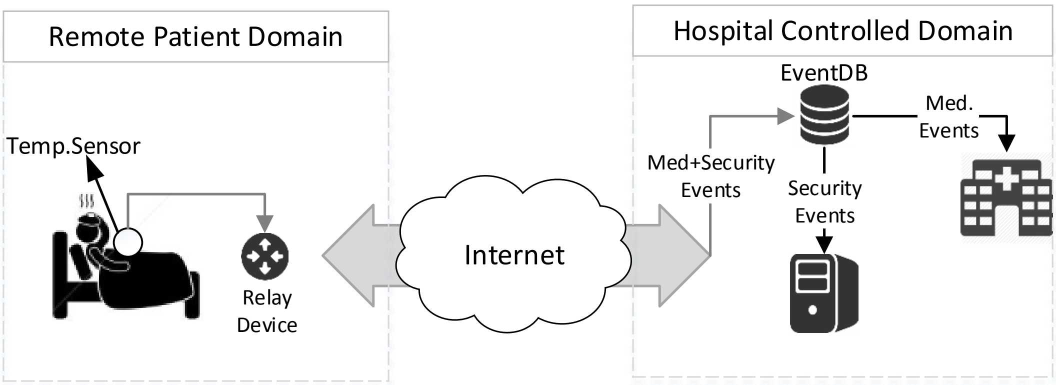


Figure 3. Prototype environment.

3.2. Architectural Overview

From a structural point of view, EDAS is designed as a component-based architecture (CBA) [12], where the design is fragmented into functional components necessary for achieving adaptive security. The components interact with each other using provided (output) and required (input) interfaces whenever an event is generated. Hence, from a communication perspective, EDAS utilizes an event-driven architecture (EDA) [8] in which events generated by a monitored component (thing) trigger concerned components. Events are considered as security changes and are monitored and correlated using a complex event processing (CEP) method. A CBA and EDA design make EDAS a reusable, replaceable, extensible, interoperable and independent architecture.

3.3. Event Sources

An event source, a thing in the EDAS context, can be visualized in functional layers, as shown in Figure 4. The application layer contains the actual application(s), temperature sensing in our case, along with the necessary security components, tools and protocols. The processing layer must have at least two components for EDAS to work efficiently, an event framework and a local adaptor. The *Event Framework* typically comes as an out-of-the-box service with most applications and includes an event handler and a logging utility.

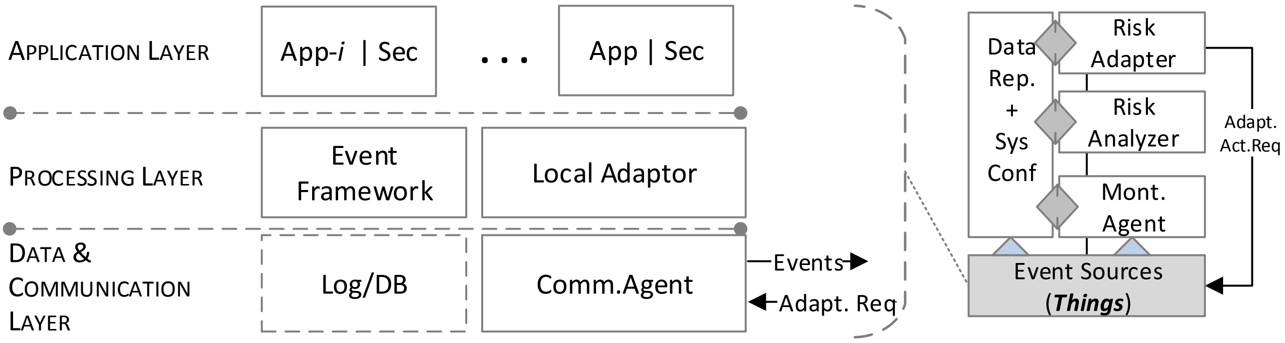
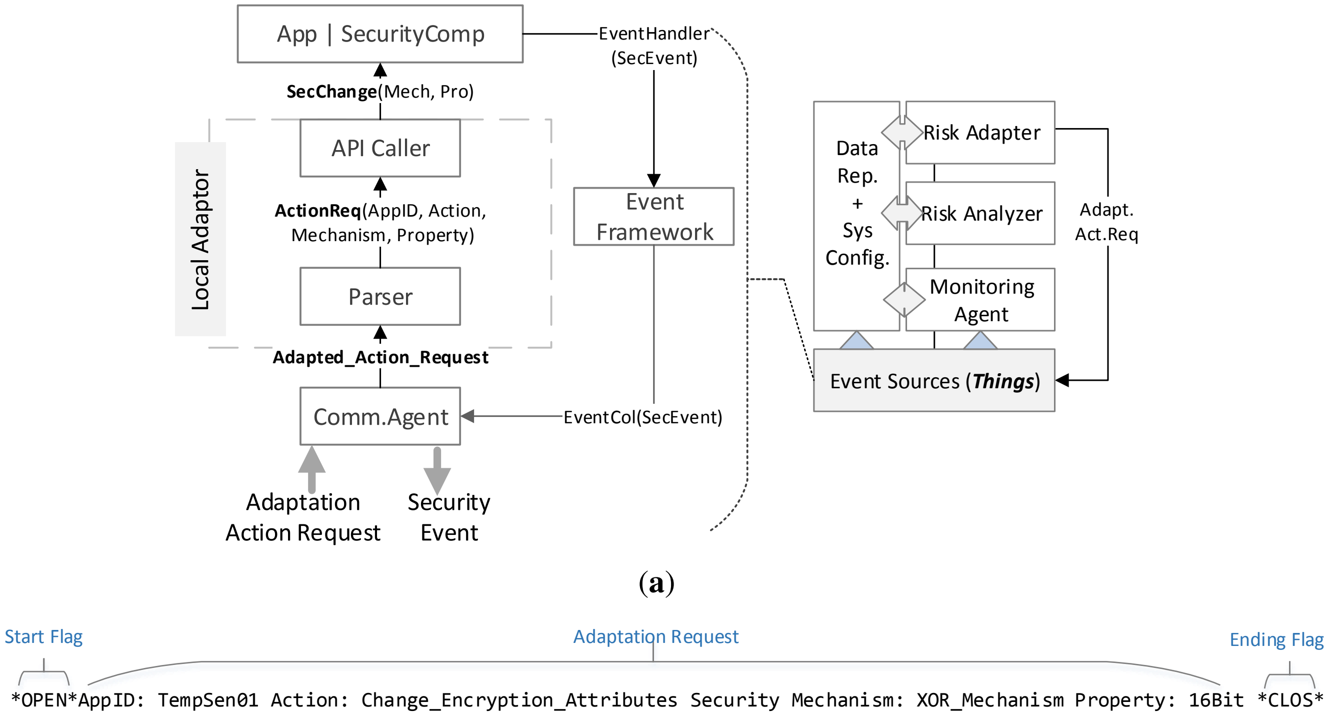


Figure 4. Event source abstraction.

Microsoft defines an event as a message generated by an object to signal the occurrence of an action, which qualifies a user interaction or any system change, *etc.* [13]. Events are thing-specific (applications or devices) information with unique attributes describing a particular change. They can be status information notifying about the battery, CPU or memory levels; a thing internal change, such as a computation error caused within a body sensor; a particular type of user interaction, such as updating a bank account number, a password change or inserting incorrect information; or an external stimuli, e.g., a location change update notified by a GPS sensor. Events are received by a handler, for instance the Java Event Handler [14], which listens to a particular event raised and invokes further methods necessary for handling. One typical method is to log that event into a local or remote file or DB present on the data and communication (DC) layer.

The *Local Adaptor* parses the adaptation request and calls the particular application and security API to adopt the new settings locally. The adaptation request received is a string detailing the new security parameters to be adapted as a result of deemed risk and an *AppID* identifying the application that raised the event, which will adopt the new settings. The local adaptation process is shown in Figure 5a, and an example adaptation request is shown in Figure 5b. Most of the low-end devices, such as body sensors, are not equipped with the local event logging facility because of the limited storage capacity and sometimes do not have remote logging capabilities. The EDAS *Communication Agent* hooks onto the output stream of the event framework via the *EventCol* interface, collects the events as they are raised locally and sends them to the EDAS platform via an event DB using HTTP request. Besides event communication, the agent also serves incoming connections. Thus, complying with the memory constraints, this design does not require any local storage for the events. Events from sources like a smartphone or those having a local logging facility, such as a file or DB, can be collected through the *LogCol* interface from the facility. Event source components and the interfaces between them can be seen in Figure 6.



(**a**)

```
*OPEN*AppID: TempSen01 Action: Change_Encryption_Attributes Security Mechanism: XOR_Mechanism Property: 16Bit *CLOS*
```

(**b**)

Figure 5. Local adaptation at the thing level. (**a**) Local adaptation process; (**b**) example adaptation request.

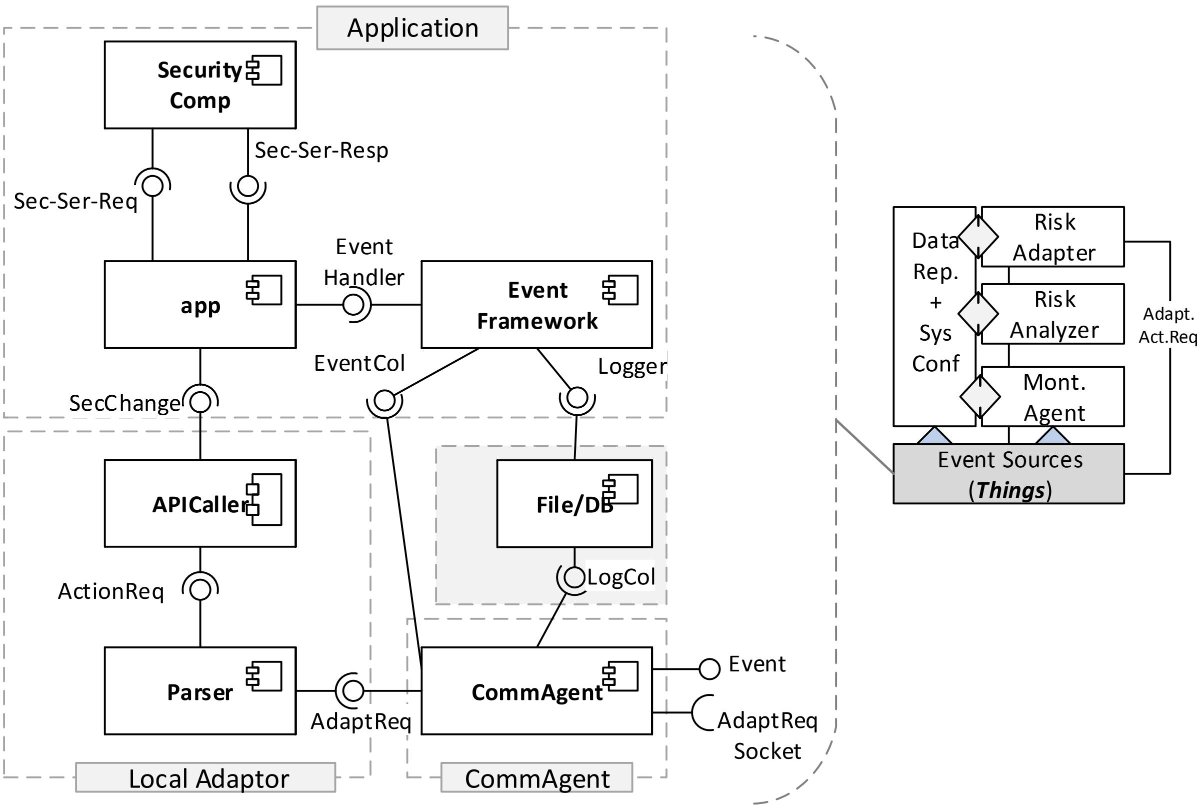
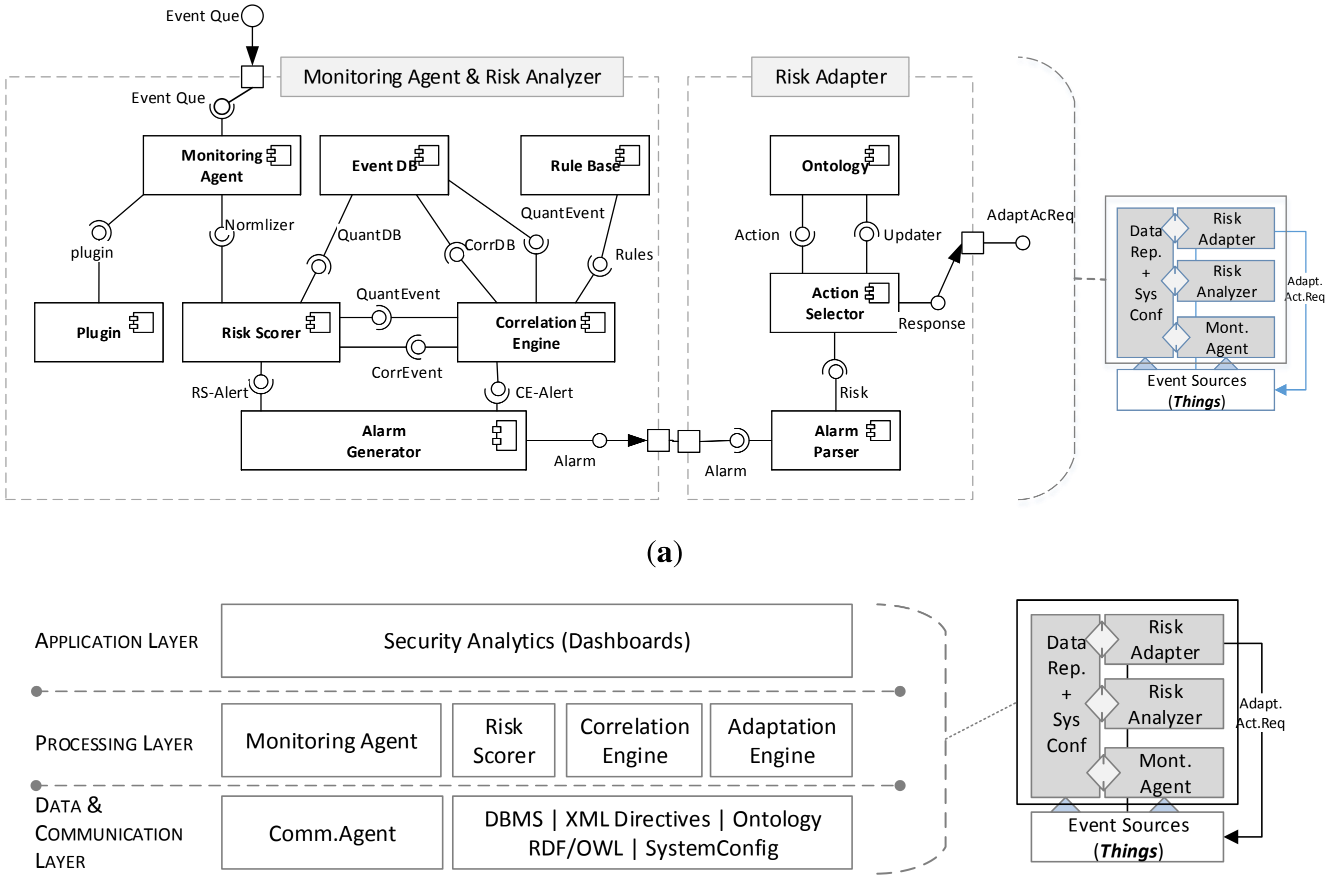


Figure 6. Event source component diagram.

3.4. The EDAS Platform

The objective of the EDAS platform is to monitor, filter, normalize and analyze primitive events coming from the monitored things in the IoT-environment. Furthermore, it decides a risk mitigation strategy per the risk faced, user and service preferences and thing capabilities. OSSIM is used as a monitoring and correlation platform in the prototype. The essential components and interfaces involved in this process are shown in its component diagram, Figure 7a, whereas a layered visualization of the platform is shown in Figure 7b. The components are explained as follows.



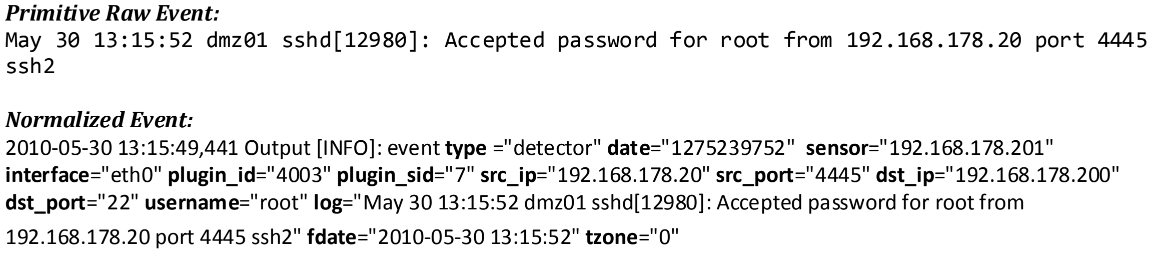
(**a**)

(**b**)

Figure 7. The EDAS platform. (**a**) EDAS platform component diagram; (**b**) The EDAS platform layered architecture.

3.4.1. Monitoring Agent

The *Monitoring Agent* reads, filters and normalizes events communicated by various sources. For each event source, there is a unique component called a plugin that performs the said operations. A plugin reads events from an event queue, a log file or DB, filters them using a white-listing technique and transform them into a standard format for analysis. Potential security events matching a regular expression (RegEx) or an SQL query are forwarded for normalization, considered as critical security events, while the unmatched ones are discarded. During normalization, essential event attributes are extracted as variables that are later used in the risk analysis process. Each event source and event type are assigned a unique plugin ID and plugin security ID (SID), respectively. These identifiers uniquely identify specific sources and event types in the IoT environment and are utilized in later processes. An example of an event before and after normalization with essential attributes for analysis and adaptation is shown in Figure 8.



Primitive Raw Event:

```
May 30 13:15:52 dmz01 sshd[12980]: Accepted password for root from 192.168.178.20 port 4445
ssh2
```

Normalized Event:

```
2010-05-30 13:15:49,441 Output [INFO]: event type ="detector" date="1275239752"  sensor="192.168.178.201"
interface="eth0" plugin_id="4003" plugin_sid="7" src_ip="192.168.178.20" src_port="4445" dst_ip="192.168.178.200"
dst_port="22" username="root" log="May 30 13:15:52 dmz01 sshd[12980]: Accepted password for root from
192.168.178.20 port 4445 ssh2" fdate="2010-05-30 13:15:52" tzone="0"
```

Figure 8. Example primitive and normalized events.

Events can be distinguished based on the information they contain. Based on the type of events and their content, the monitoring agent's (the plugin) filtration criteria decide which of them should be considered for risk analysis. For instance, medical events concerning vital body signs are not to be classified as security events and may be filtered out unless they are involved in a related biometric authentication system. Furthermore, among other essential attributes, events are generated with a key property generally called level, for instance, see Microsoft event properties [15]. The level indicates an event's importance or its severity level. Although the level definition is thing specific, it can provide essential information to distinguish how critical a particular event is and how it should be assessed in the risk analysis process. In the prototype, OSSIM transformed this importance level into event priority, discussed in the next section.

3.4.2. Risk Scorer and Correlator

During normalization, the plugin defines which SID has to be assigned a particular event. An SID definition includes its ID, priority, reliability and a description. These fields are registered for a particular event type in a MySQL DB present at the data layer. Priority and reliability values together with the asset (event source or thing) value are used to quantify the risk associated with a particular event [16]. The *Risk Scorer* performs this quantification. In OSSIM, asset and priority values reflect the importance of the event source and the event respectively. A higher value implies a high event importance. For instance, a higher value can be given to an error or warning level event than to an information level event. Similarly, in a remote patient monitoring system, a critical asset, such as a wearable sensor, is given a higher asset value as compared to a smart device that the patient uses for other medical purposes. The reliability of an event (SID) specifies its probability as an actual attack. It is an attack probability level asserting the chances that a particular SID may yield to a real compromise and is used to deal with false alarms.

As we have suggested earlier that any CEP-supported analysis method can be utilized in EDAS, we cannot recommend any particular risk formulation equation because of the different risk perceptions. Below, we include the OSSIM risk equation [16] as an example and for the purpose of explaining the prototype. OSSIM calculates the risk of an event as follows:

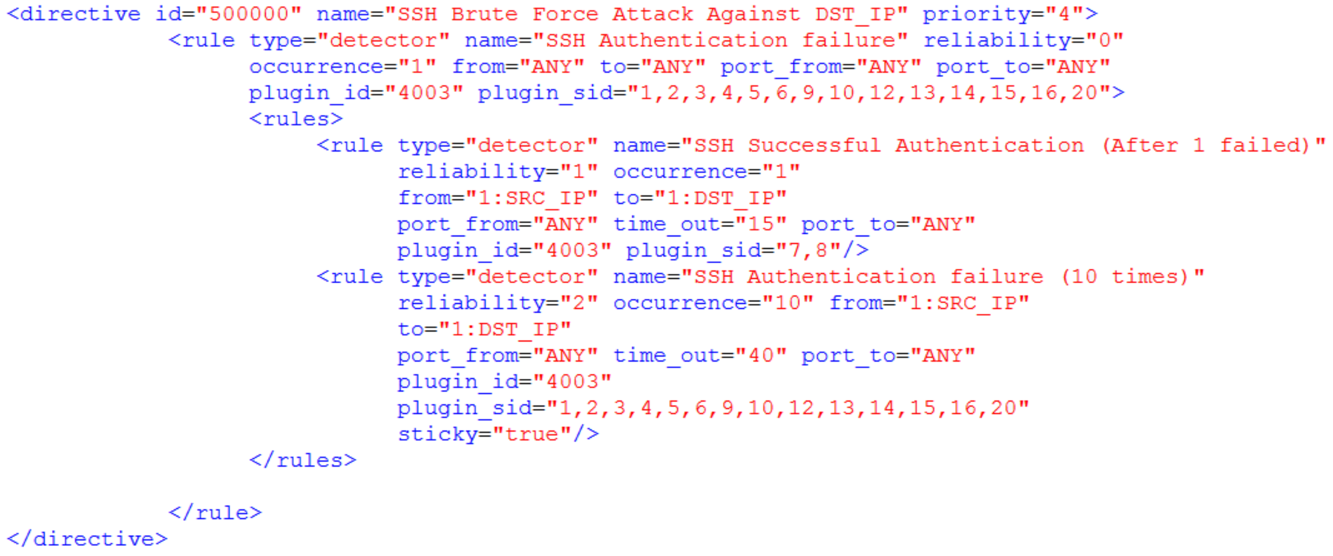
$$Risk(Event) = (Asset \times Priority \times Reliability/25) \quad (1)$$

where Asset and Priority can take a value from [0, 5] and Reliability from [0, 10]. The division by 25 is made to restrict risk to 10 different risk levels.

Risk quantification is based on the event's primitive information. In some cases, several events over a period may contribute towards an incident. In such circumstances, if a risk is calculated on a single independent event, it may lead to false positives and negatives, which may further yield to the selection and implementation of inappropriate adaptation strategies. To avoid such situations, the *Correlation Engine* correlates different potential events over a period in a definitive context and decides whether there is a risk involved or not. It modifies the event's reliability as per a faced situation (context) when multiple potential events are detected in a specified interval. In other words, as more probable events are detected in the same correlation context, its reliability is increased. Thus, the correlation makes the overall threat reliability more accurate as more events occur in the same context.

In OSSIM, a context defined for event correlation is a sequence of different events observed in a particular time frame. It is stored as an XML directive in a file and is activated when a particular SID is detected [16]. An XML correlation directive contains a rule set. The first rule is called the triggered rule, as it activates the potential threat context to be analyzed. Each rule specifies an event occurrence and defines a new reliability for the associated threat context. Thus, a risk is analyzed in a context-aware manner, where events are correlated in time and space.

An example directive is shown in Figure 9. This example directive captures repeated SSH log-in attempts and the corresponding contextual events (SIDs) generated as a result of a failed attempt. It can be seen that the reliability of the threat context is analyzed with each rule. Correlation is performed in a particular time frame captured in the `time_out` variable. With each rule, it is made clearer whether the events correspond to authorized attempts where one can forget or mistype log-in credentials or to a compromise, such as brute force. Hence, with each rule (event occurrence), the threat context reliability is analyzed again, and risk alarms are raised accordingly. Event correlation makes analysis more accurate and reduces the possibility of false alarms.



```
<directive id="500000" name="SSH Brute Force Attack Against DST_IP" priority="4">
            <rule type="detector" name="SSH Authentication failure" reliability="0"
                  occurrence="1" from="ANY" to="ANY" port_from="ANY" port_to="ANY"
                  plugin_id="4003" plugin_sid="1,2,3,4,5,6,9,10,12,13,14,15,16,20">
                  <rules>
                       <rule type="detector" name="SSH Successful Authentication (After 1 failed)"
                             reliability="1" occurrence="1"
                             from="1:SRC_IP" to="1:DST_IP"
                             port_from="ANY" time_out="15" port_to="ANY"
                             plugin_id="4003" plugin_sid="7,8"/>
                       <rule type="detector" name="SSH Authentication failure (10 times)"
                             reliability="2" occurrence="10" from="1:SRC_IP"
                             to="1:DST_IP"
                             port_from="ANY" time_out="40" port_to="ANY"
                             plugin_id="4003"
                             plugin_sid="1,2,3,4,5,6,9,10,12,13,14,15,16,20"
                             sticky="true"/>
                  </rules>

            </rule>
</directive>
```

Figure 9. Example OSSIM correlation directive.

3.4.3. Adaptation Engine

The *Adaptation Engine* decides a mitigation action for a particular user and thing in a given risk context. It takes the risk information from the *Risk Analyzer* and calls a runtime RDF/XML ontology. The proposed security adaptation ontology, shown in Figure 10, contains the security metrics, device capabilities and user preferences necessary to decide an optimized adaptation action from a pool of available actions. By optimized action, we refer to a response that is selected after assessing user, service and device requirements in a particular risk context. Entities and associated relationships in the ontology along with examples are detailed in our previous work [6]. The ontology entities are grouped into three contexts; user, thing and security. Each context captures the respective knowledge, as well as current runtime security settings necessary for deciding an adapted action. While selecting an action, the engine utilizes an Apache Jena-SPARQL API-enabled [17] script to query and update the stored ontology. Updating is performed to ensure that the ontology as a runtime knowledge platform is aware of all of the current configurations that may be required in possible succeeding adaptation decisions.

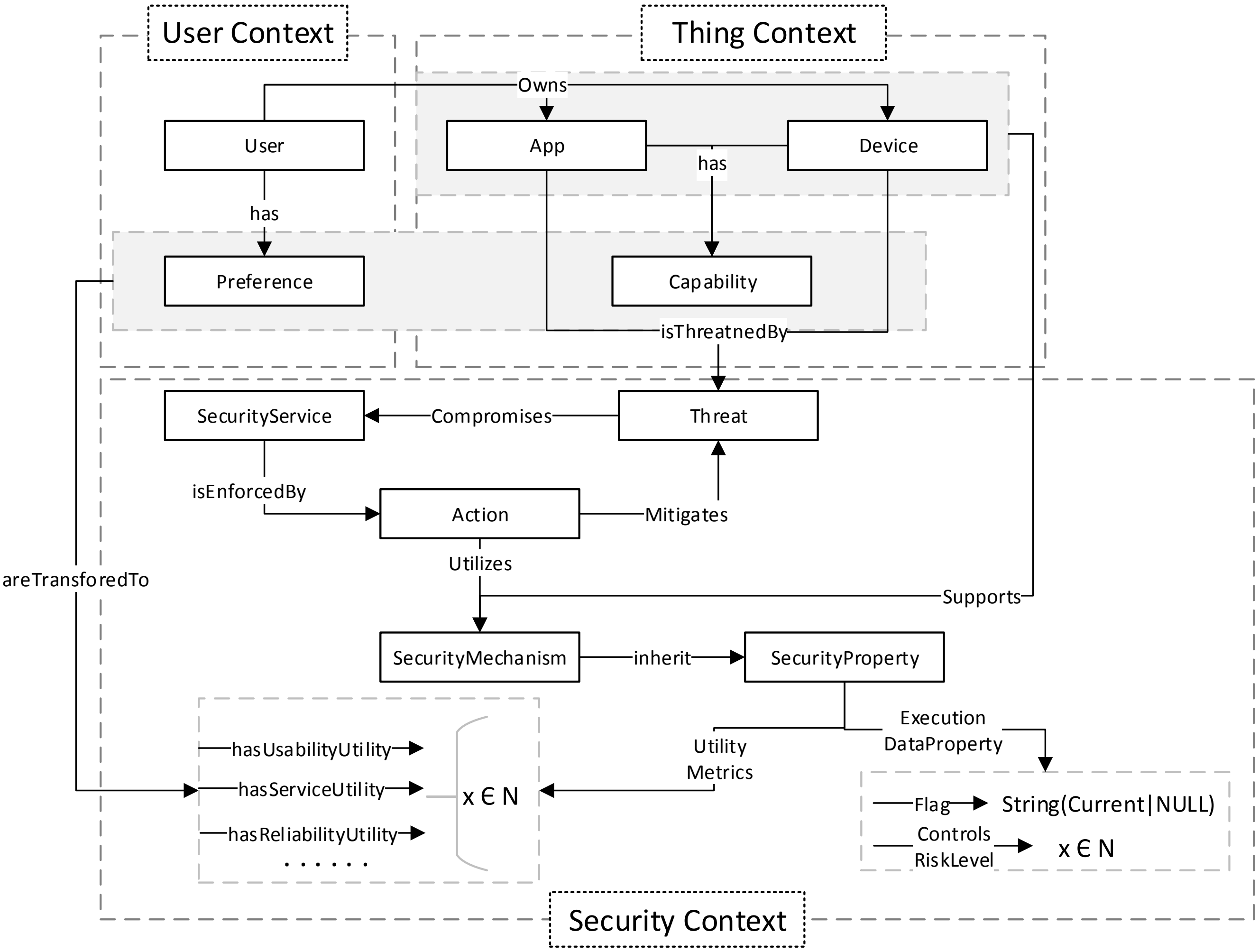


Figure 10. Security adaptation ontology.

The adaptation engine is activated when it receives a risk alarm from the risk analyzer. The alarm is a token detailing the risk components, *i.e.*, the Threat-ID, Risk-Level, and Device-ID. The adaptation action is selected in a stepwise procedure as shown in Table 1. The table describes all of the signatures pertaining to the ontology elements, *i.e.*, subjects, predicates and objects, which are accessed at each step of the procedure. The ontology is developed in an RDF/XML format using the Protege tool [18]. Different types of ontology elements used in the table are described as follows: a `Class` refers to a concept of interest in the ontology. `Class Objects` are the members of a class. `Object Property` is the relationship between one or more members of one class with one or more members of another class, and a `Data Property` refers to a particular attribute of a class object. Example objects and a description of classes used in the adaptation process are given in Table 2. For a detailed description of all of the concepts and relationships in the proposed ontology, refer to our previous work [6].

Table 1. The adaptation action decision process.

Step No.	Type: Subject	Type: Predicate	Type: Object	Type: Return
Step 1	**Description:** *identifying a particular threat faced in the ontology*			
	`Class:` threats	`Data Property:` hasThreatID	`String:` Threat-ID, e.g., "DOS5001"	`Class Object:` a threat object
Step 2	**Description:** *listing possible actions that address the threat identified in Step 1*			
	`Class:` actions	`Object Property:` mitigates	`Class Object:` threat	`Class Object:` action objects
Step 3	**Description:** *identifying security mechanisms utilized by actions identified in Step 2*			
	`Class Object:` action objects	`Object Property:` utilizes	`Class:` security mechanism	`Class Object:` security mechanism objects
Step 4	**Description:** *identifying the device facing the threat*			
	`Class:` devices	`Data Property:` hasID	`String:` Device-ID e.g., "192.168.1.3"	`Class Object:` A Device object.
Step 5	**Description:** *extracting only device-supported security mechanisms from those identified in Step 3*			
	`Class Object:` a device object	`Object Property:` supports	`Class Object:` security mechanism objects	`Class Object:` security mechanism objects
Step 6	**Description:** *listing properties that are utilized by mechanism identified in Step 5*			
	`Class Object:` security mechanism objects	`Object Property:` inherit	`Class:` security property	`Class Object:` security property objects
Step 7	**Description:** *selecting properties addressing a particular risk level from the properties identified in Step 6*			
	`Class Object:` security property objects	`Data Property:` controlsRiskLevel	Integer: risk-Level e.g., 1, 2, 3...	`Class Object:` security property objects
Step 8	**Description:** *extracting utility metrics values for individual property identified in Step 6*			
	`Class Object:` security property objects	`Data Property:` hasUsabilityUtility, hasConfUtility, ...	`Integer:` Utility-value e.g., 1,2,3...	`Integer:` utility-value

Table 2. Classes description in the adaptation process.

Class	Description	Example class objects
Device	Monitored devices and their attributes	Sensor, smartphone, desktop
Threat	Threats known in the environment	DoS, brute force, malware infection
Actions	Mitigation responses to be adapted to defend against a threat	Enforce a CAPTCHA , change cipher, change routing algorithm
Security mechanism	Methods, tools, algorithms used by an action	DES, AES, XOR, CAPTCHA, password length
Security property	Adjustable attributes of a security mechanism	Eight-char password, image CAPTCHA, audio CAPTCHA, 128-bit key

As the final step (Step 9), a security property object having the maximum weighted utility among those identified in Step 7 is selected as the most optimal property. The corresponding mechanism and action are extracted, and an adaptation request string is constructed. The request is sent to the event source as an optimal response to the faced threat and to be adopted locally. Metrics, such as hasUsabilityUtility, hasReliabilityUtility and hasConfUtility, *etc.*, are attributes associated with each security property object. They reflect the property utility in terms of user and service requirements and device capabilities. These utility metrics are derived from the mentioned classes in the ontology.

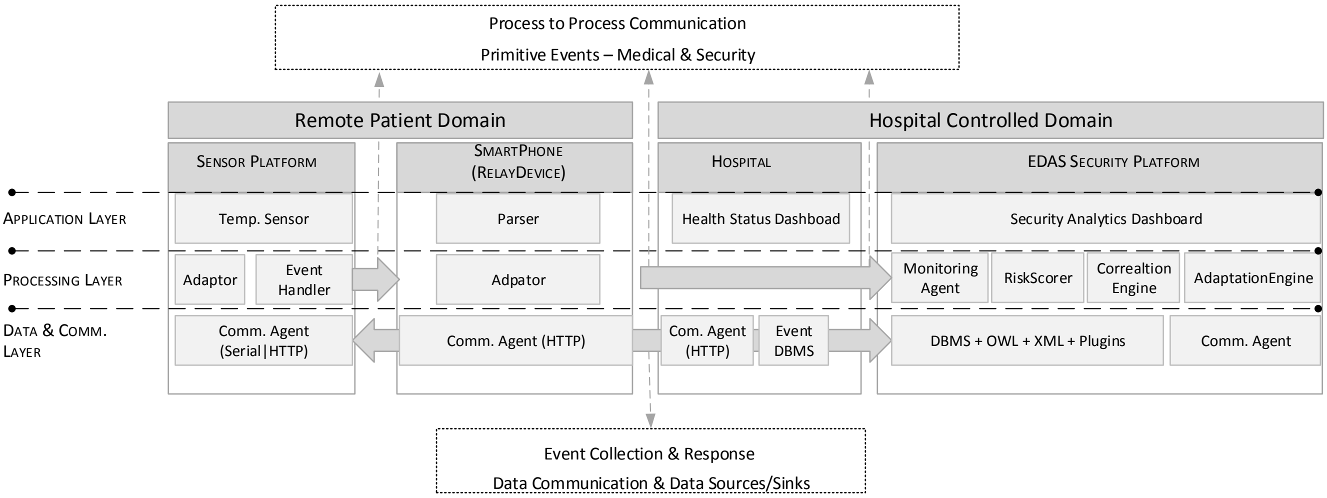


Figure 11. Prototype architecture categorized into functional layers.

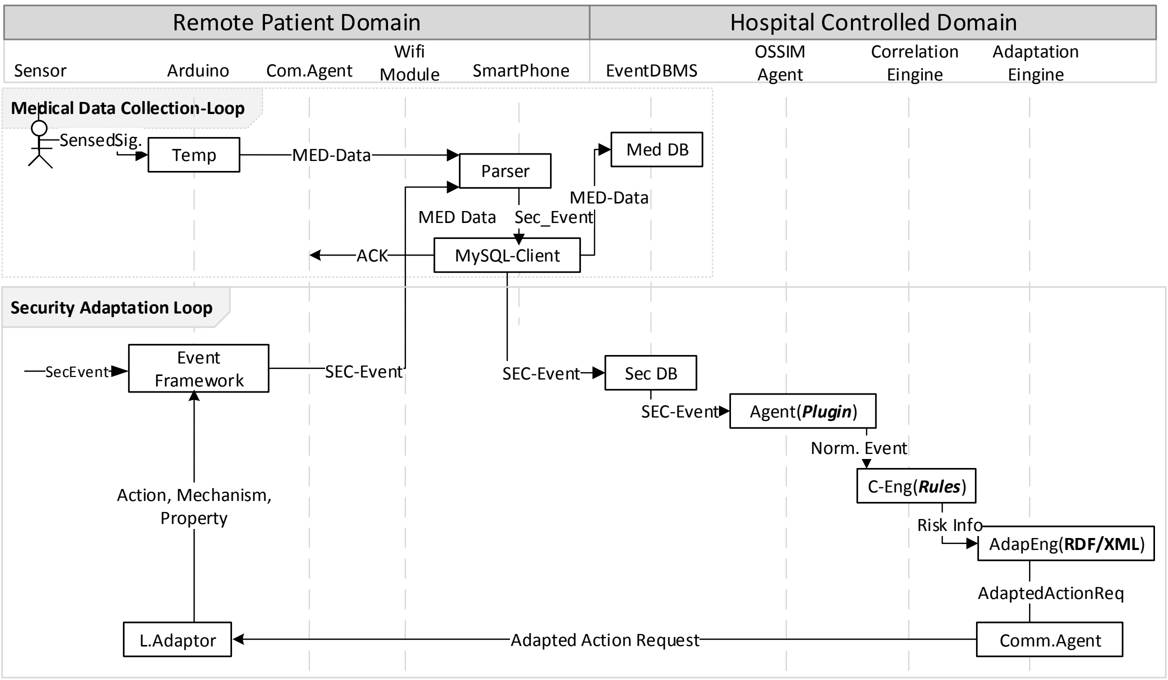


Figure 12. EDAS message sequence chart.

The prototype test environment as a whole can be categorized into functional layers as shown in Figure 11. The relay device, a smart device, is used primarily for parsing medical data and security events arriving at the same HTTP connection. It uses a MySQL client to send these events to the respective DBs contained in the event DBMS for further investigation. Utilizing a smart device, such as a smartphone, as a relay also makes the entire system more usable. Patients will be able to monitor their health status locally with the help of an app installed without querying the health journals at the hospital site. Furthermore, it can make the patient monitoring more usable in mobility scenarios, therefore increasing the overall eHealth service utility. Figure 12 shows an abstract level message transfer between the major processes that we developed and designed in our prototype. It can be seen that data are collected using two loops, medical data collection and security adaptation, both executed in parallel.

4. Case Study

A test case study concerning user confidentiality and service availability is developed to demonstrate EDAS as a real-world prototype. The case study characterizes a tradeoff between confidentiality and availability. The case study is based on a general phenomenon that encrypting messages consumes more energy if longer key lengths are utilized and *vice versa*. Pre-shared keys and respective indexes are used in the case study. A higher index corresponds to a key with increased length. The state transition diagram depicting the case study security adaptation is shown in Figure 13. *Stable State 1* is assumed to be the initial state. Different cipher key lengths for encrypting medical data are adapted when the *LowBattery* or *ChargingUp* events are generated by the temperature sensor.

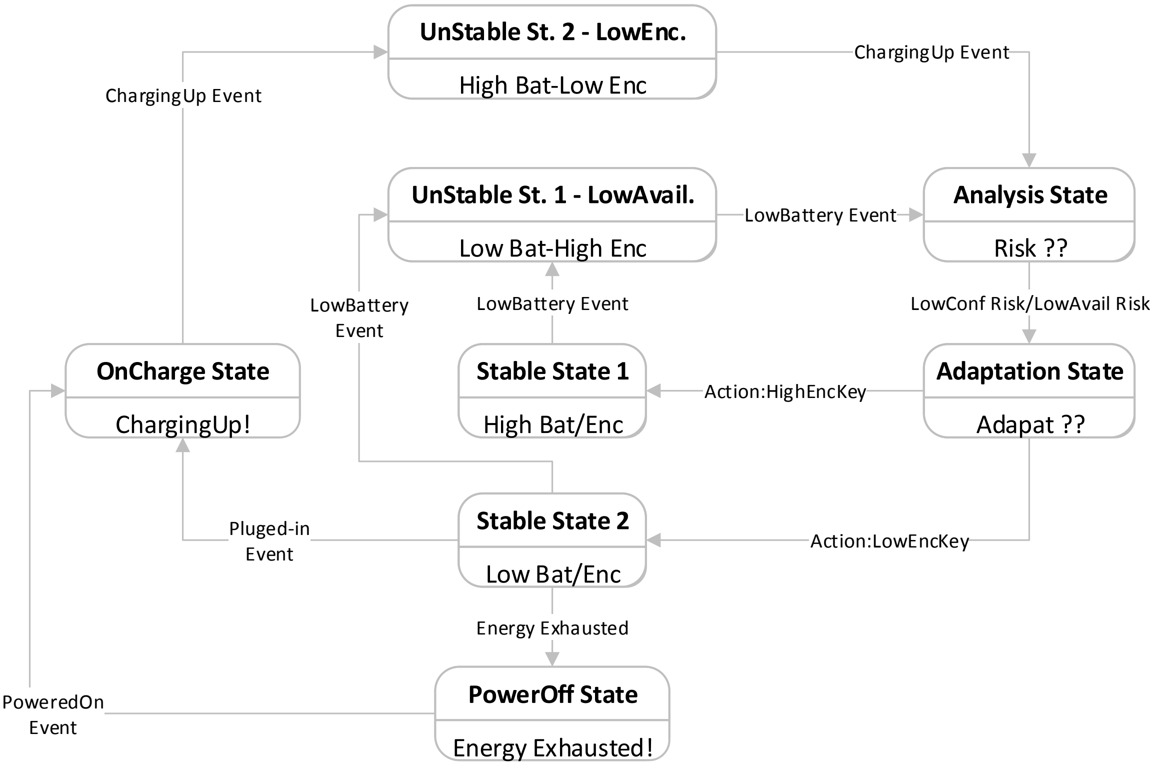
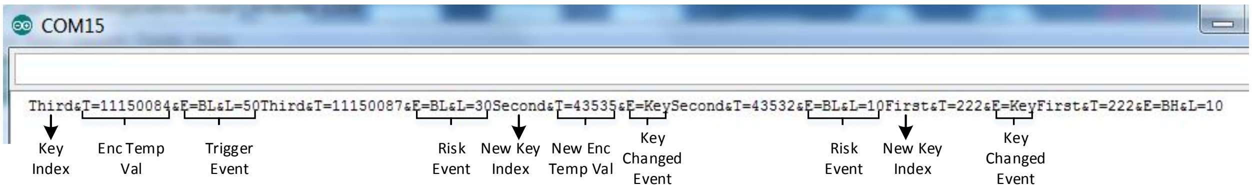


Figure 13. Adapting security to low availability/confidentiality risks.

The case study consists of two scenarios to reflect a confidentiality-availability tradeoff situation. In the first scenario, EDAS decides to ensure service availability as opposed to keeping a high confidentiality level when the sensor battery level drops below a certain threshold. Therefore, encryption keys with decreased lengths are adapted to meet the primary requirement, availability, of a continuous patient monitoring system. In the second scenario, confidentiality is preferred over availability. Confidentiality is regained and key lengths are increased as the battery is recharged to a particular threshold, indicating that the sensor is steadily available to meet a particular service level. Key lengths are gradually increased and decreased as per the observation of threshold battery level events. This adaptation process is performed continuously until the sensor changes a state.

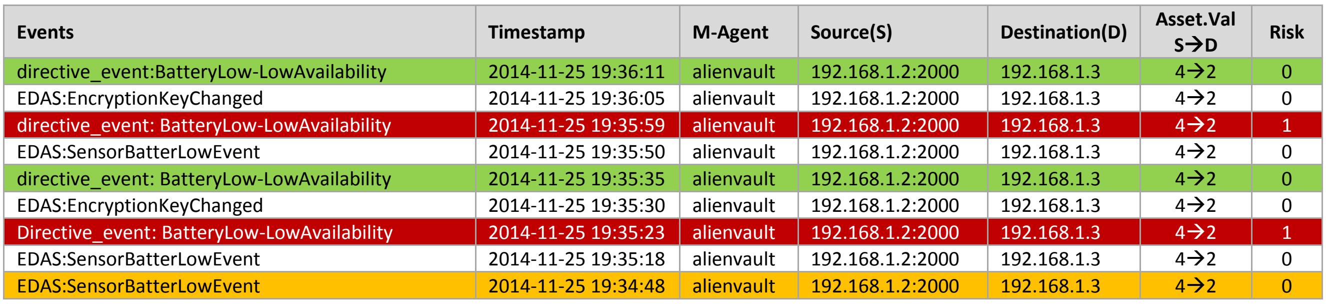
- Scenario 1 (Low Availability Risk: High Encryption with Low Battery)

Low availability (context or directive) risk is raised when there is a *LowBattery* event with the level being less than *X*%, and the encryption is still done with an increased key length. The corresponding risk is reduced when a *KeyChanged* event is generated by the event source (temperature sensor) after a reduced encryption key length is adapted. The corresponding screenshots are displayed in Figures 14 and 15.



```
Third&T=11150084&E=BL&L=50Third&T=11150087&E=BL&L=30Second&T=43535&E=KeySecond&T=43532&E=BL&L=10First&T=222&E=KeyFirst&T=222&E=BH&L=10
```

Figure 14. Scenario 1: sensor screen: decreased key lengths are adapted when battery level drops.

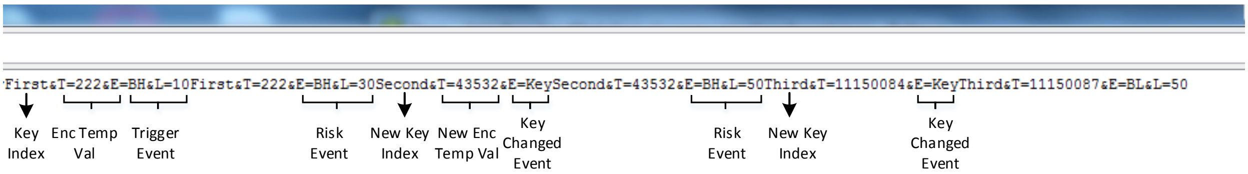


Events	Timestamp	M-Agent	Source(S)	Destination(D)	Asset.Val S→D	Risk
directive_event:BatteryLow-LowAvailability	2014-11-25 19:36:11	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:EncryptionKeyChanged	2014-11-25 19:36:05	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
directive_event: BatteryLow-LowAvailability	2014-11-25 19:35:59	alienvault	192.168.1.2:2000	192.168.1.3	4→2	1
EDAS:SensorBatterLowEvent	2014-11-25 19:35:50	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
directive_event: BatteryLow-LowAvailability	2014-11-25 19:35:35	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:EncryptionKeyChanged	2014-11-25 19:35:30	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
Directive_event: BatteryLow-LowAvailability	2014-11-25 19:35:23	alienvault	192.168.1.2:2000	192.168.1.3	4→2	1
EDAS:SensorBatterLowEvent	2014-11-25 19:35:18	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:SensorBatterLowEvent	2014-11-25 19:34:48	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0

Figure 15. Scenario 1: EDAS platform dashboard screen (modified): the *LowAvailability* alarm is raised (as risk = 1) whenever a *BatteryLow* event is detected and is reduced when a *KeyChanged* event is observed after adaptation. Color legend: yellow, trigger event; red, alarm (unacceptable risk); green, alarm (acceptable risk); white, event detected.

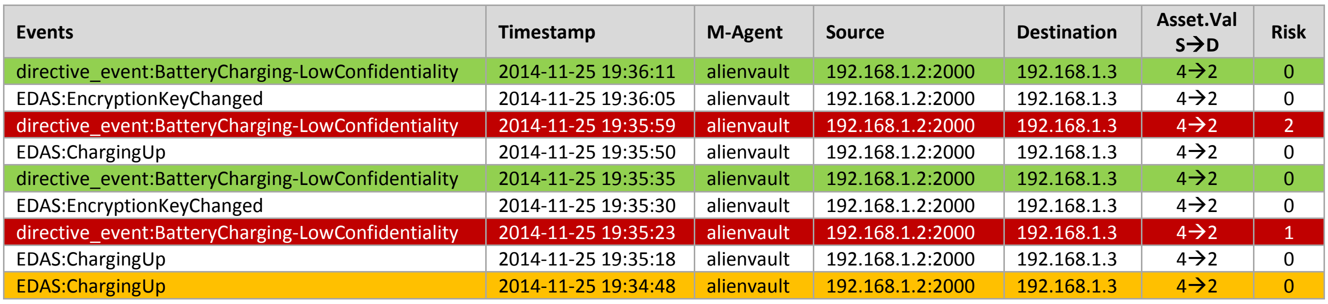
- Scenario 2 (Low Confidentiality Risk: Low Encryption with High Battery)

The low confidentiality alarm is raised when the *BatteryChargingUp* event is detected with a level greater than *Y*%, and the encryption is still done with a reduced key length. The corresponding risk is reduced when an increased key length is adapted and a *KeyChanged* event is discovered after increased key lengths are selected. See the corresponding screenshots in Figures 16 and 17.



```
First&T=222&E=BH&L=10First&T=222&E=BH&L=30Second&T=43532&E=KeySecond&T=43532&E=BH&L=50Third&T=11150084&E=KeyThird&T=11150087&E=BL&L=50
```

Figure 16. Scenario 2: sensor screen: encryption adapts to increased key lengths when the battery is recharged to a threshold level.



Events	Timestamp	M-Agent	Source	Destination	Asset.Val S→D	Risk
directive_event:BatteryCharging-LowConfidentiality	2014-11-25 19:36:11	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:EncryptionKeyChanged	2014-11-25 19:36:05	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
directive_event:BatteryCharging-LowConfidentiality	2014-11-25 19:35:59	alienvault	192.168.1.2:2000	192.168.1.3	4→2	2
EDAS:ChargingUp	2014-11-25 19:35:50	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
directive_event:BatteryCharging-LowConfidentiality	2014-11-25 19:35:35	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:EncryptionKeyChanged	2014-11-25 19:35:30	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
directive_event:BatteryCharging-LowConfidentiality	2014-11-25 19:35:23	alienvault	192.168.1.2:2000	192.168.1.3	4→2	1
EDAS:ChargingUp	2014-11-25 19:35:18	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0
EDAS:ChargingUp	2014-11-25 19:34:48	alienvault	192.168.1.2:2000	192.168.1.3	4→2	0

Figure 17. Scenario 2: EDAS platform dashboard screen (modified): the *LowConfidentiality* alarm is raised (as risk = 1, 2) whenever a *BatteryChargingUp* event is detected and is reduced when a *Key Changed* event is detected after adaptation. Color legend: yellow, trigger event; red, alarm (unacceptable risk); green, alarm (acceptable risk); white, event detected.

5. Feasibility and Evaluation

This section evaluates EDAS as an event-driven security concept and system architecture and will detail lessons learned from the prototyping activity. We will discuss how EDAS can be the right tool for ensuring real-time risk management in dynamic environments, such as IoT, and how it complements existing ICT infrastructure. Moreover, EDAS is compared with architectures corresponding to traditional security controls to investigate its feasibility as a viable solution for IoT.

5.1. Dynamic Real-Time Autonomous Risk Management

The primary objective of EDAS is to ensure continuous and dynamic risk management capabilities in the IoT. Changes (events) in the monitored environment are collected and analyzed as soon as they are observed. IoT is thought to be a self-adaptive and self-organized network, and things are deemed to be autonomous [1]. Self- and autonomous adaptation capabilities are necessary in IoT [19,20]. These properties ensure dynamic adaptation to avoid potential management delays caused by human intervention. EDAS overcomes this issue by placing the user preferences, service requirements and analyst knowledge in the correlation directives and adaptation ontology before the system start-up. Thus, it empowers the system to monitor, analyze and adapt to the risks faced autonomously and dynamically in an optimum manner.

5.2. Context Awareness

The term context has been defined differently by various authors. However, a more general definition is given by Abowd *et al.* [21] as any information that can be used to characterize and recognize the situation of an object, person or place. Context provides vital information regarding the who, what, where, when and why of a situation [22]. In a computing environment, this context or information is usually offered by the system- or application-generated events. It can be seen in Figure 8 that the information in the primitive events precisely provides essential attributes necessary to qualify both definitions stated. Hence, EDAS captures the fundamental unit of a system change, *i.e.*, the event, to set a clear and distinct ground for context-aware risk analysis.

Furthermore, in real-world scenarios, a compromise is usually a combination of different attack vectors, modifications, tools and targets, which may trigger a series of different events originating from different sources as the compromise progresses. EDAS addresses such situations through event correlation. Using a correlation method, such as a rule-based OSSIM correlation directive, an analyst could define a particular compromise context defined as a rule set. The context captures all of the potential events that signify a potential compromise and can accurately qualify whether a risk is involved or not. Correlating events from different sources provides a broad view of understanding the context and holistic information. Thus, it reduces any false alarms that may be caused by analyzing events independently. Nonetheless, if events are seen as a series of steps towards a particular compromise situation (context), then by exploiting the precautionary principle [23,24], EDAS can predict these steps and can respond to corresponding threats (events) before they are realized as actual attacks.

5.3. Preferences and Capability-Based Adaptation

Adapting security changes autonomously by only considering the impact of the risks faced is a security risk itself. In such cases, changing security parameters may negatively affect service attributes, such as throughput or latency, and thing resources. Re-configurations based only on the security context of a threat without assessing its impact on the service requirements can cause unnecessary adaptation that may cause serious problems [25]. Such strategies can cause service disruption in environments like IoT or WSN, where the main driving technologies are battery-powered and resource-limited devices. Therefore, other factors, such as the device and application capabilities, as well as service requirements, need to be considered while an adaptation strategy is decided, as performed in EDAS.

Nonetheless, in a user-centric service, such as IoT-eHealth, the user, *i.e.*, the patient, as well as the medical staff, preferences should also be assessed while new security settings are adapted. The EDAS runtime adaptation ontology stores these capabilities, requirements and preferences before system start-up and transforms them into metrics with respective utilities against a particular *SecurityProperty*. This enables the adaptation engine to choose a mitigation action from the available action pool, such that its weighted utility has a maximum value for a given user using a particular service in a specific risk situation.

5.4. Development and Deployment

Besides some component engineering, the EDAS development involves a technology integration process. It utilizes the thing event framework included in almost all mature applications and devices as a logging, troubleshooting and debugging facility. However, local adopters, which are merely a string parser and API caller, must be developed and should have read and execute permissions in order to execute the call to a specific security component against a specified application. Local adaptation can also be performed via the application where the adaptation request received is passed as arguments to override the method that ensures security. As for the variety of technology used in the IoT, a platform-independent design, e.g., Java, can be opted to develop and integrate uniform local adopters across the monitored environment.

As most devices in IoT have low computational power and sometimes perform only specific sensing or actuation routines, the analysis burden has been taken away from the thing level to a resourceful machine (EDAS platform). Event sources can be monitored from anywhere. If the event source is addressable in the environment and if it can generate and communicate events, no matter where it is located (remote or on-site), it can be integrated with EDAS. This implies that traditional systems, such as firewalls, databases, file/application servers and other critical information systems, with built-in well-defined event frameworks, can also be integrated in and monitored by EDAS, as they have key roles in the overall service delivery.

Component-based architectures (CBA) and event-driven architectures (EDA) have the ability that their operational components can be distributed over the network as they are loosely coupled [8,12]. EDAS can be used as a standalone platform to monitor, analyze and adapt to the risk faced. However, it inherits the CBA and EDA concepts, and thus, its major operations can be distributed over multiple locations in different hierarchical settings. For instance, in an IoT-eHealth perspective, a city hospital *X* can analyze the threats related to the environment it operates in by a local EDAS risk analyzer. Same settings can be established for a city hospital *Y*. However, a principal adaptation engine in location *Z* governed by a central security policy can be deployed to decide a mitigation response based on *X's* and *Y's* risk information to reduce the risk level in either of the domains. Such distributed settings enable security analysts to isolate and focus on a set of threats concerning a particular location context and may add more to the precision of the risk analysis process.

5.5. Architectural Comparison

This section provides a comparison of EDAS with traditional security architectures and corresponding controls. They are grouped into the following categories based on their architecture, however irrespective of the particular prevention and detection methods they utilize:

- Host-based: controls that manage security locally on an end-user machine, e.g., an anti-virus, host IDS, firewalls, *etc.*
- Endpoint: controls that protect end-user machines, but are managed by a central entity, e.g., Endpoints in Microsoft System Endpoint Protection.

- Agent-based: controls that protect and manage the monitored environment based on the information gathered by specialized agents; for example, an agent-based IDS.
- Centralized: controls that provide security as a central stand-alone entity; for instance, an enterprise firewall or a network-based IDS.
- Distributed: a security architecture that utilizes various prevention or detection systems distributed over a network to facilitate advanced security analysis; for example, a distributed firewall or IDS.

The objective is to highlight whether the concept of EDAS as an event-driven security architecture and adaptive security solution adds value to an IoT-based service, such as eHealth. There is comprehensive work concerning system and software architecture evaluation, e.g., [26,27]. However, we present a simple comparison based on the architectural concept and on the prototyping exercise we performed to reflect on the extent to which these candidates qualify or support a given architecture quality attribute in the IoT-eHealth context.

One can find an extensive list of these attributes in the literature. However, we have selected a list of architecture quality attributes from [28–31], which covers most of these attributes. Furthermore, we have used selected attributes from these sources as some of them, though having little differences, can be defined interchangeably. For instance, attributes like modifiability, evolvability, adaptability, configurability, reusability and customizability can be accumulated for a change in a system for which a single and, more common, word, maintainability, can be used, which we have adopted. Some attributes are intentionally dropped as they are more focused on software architecture as opposed to system architecture, e.g., portability in [28]. Hence, the attributes used here are considered and defined in a system perspective and not in a software context. Furthermore, we have added a few functional attributes, such as monitoring scope and threat detection accuracy, to reflect on a candidate aptitude as a security solution. These attributes are described in Table 3. Table 4 depicts the comparison where (++) implies that an attribute is positively qualified or supported, a (+) indicates partial qualification or support of the attribute indicating that there is some design dependency involved, and (-) indicates an absence of the attribute.

Table 3. System architecture quality attributes.

	S.No	Attributes	Ref.	Description
Runtime Execution	1	Interoperability	[29–31]	The ability of a system to be utilized in diverse environments
	2	Reliability	[28–31]	The ability of a system to continue intended operations over time
	3	Usability	[29,31]	The measure of how well the user requirements are met for using the system (in terms of the user's security requirements)
	4	Latency	[28–31]	The reaction time of a system to an event/incident/threat
	5	Throughput	[28–31]	The number of events/threats/incidents responded to in a given time interval

Table 3. *Cont.*

	S. No	Attributes	Ref.	Description
Security	6	Security	[29–31]	The capability of a system to stand against a potential threat concerning the C-I-A services
	7	Monitoring Scope		The various types of contextual information/assets the system can monitor and analyze
	8	Adaptability	[31]	The ability of a system to systematically and autonomously regulate its behavior and re-configure its settings (here Security Adaptation only)
	9	Threat Detection Accuracy		The capability of a system to accurately detect threats to avoid false positives/negatives
Design	10	Simplicity	[28]	A measure reflecting how functionalities are separated from one another to keep things clear and easy to understand, isolate and develop
	11	Extensibility	[28,31]	The ease with which a system functionality can be extended by adding more components to it
	12	Maintainability	[28–31]	The ability of a system to be easily modified when requirements are changed
Support	13	Supportability	[29,31]	The ability of a system to provide helpful information to resolve errors, trace user activity and related issues
	14	Testability	[29,31]	A measure depicting how well a test criteria can be created, executed and evaluated against the system

Table 4. EDAS *vs.* traditional security controls.

	Attribute	EDAS	Host	Endpoint	Agent-Based	Centralized	Distributed
Execution	Interoperability	++	-	-	-	++	++
	Reliability	++	-	-	-	-	++
	Usability	++	+	+	-	-	-
	Latency	+	++	++	+	+	+
	Throughput	++	+	+	+	+	+
Security	Security	++	+	+	+	+	++
	Monitoring Scope	++	+	+	+	+	+
	Adaptability	++	-	-	-	-	-
	Threat Detection Accuracy	++	+	+	+	+	++
Design	Simplicity	+	++	++	+	+	+
	Extensibility	++	-	-	+	+	+
	Maintainability	+	++	+	+	+	+
Support	Supportability	++	++	++	++	++	++
	Testability	+	++	++	+	+	+

5.5.1. Interoperability

A CBA design makes a component more independent and reusable [12]. EDAS is based on the CBA style, which makes its components interoperable. This is also true for distributed systems. Moreover, it is based on the event processing concept, which is a typical facility in almost all applications.

Hence, irrespective of the type and nature of an application, device or network, it is an environment or platform-independent architecture that can be used in any context where the monitored objects can generate and communicate events. On the other hand, host, endpoints and agent-based solutions are designed for particular platforms performing specialized tasks. They may share data with external systems, but cannot be operated on different platforms.

5.5.2. Reliability

The CBA and EDA designs ensure loose coupling between the system components, which enables component distribution and redundancy to support reliability, separation of functionalities and avoids single point failures [8,12]. These properties as desirable design attributes can also be found in distributed systems. Other architectures, being single entities, may expose and threaten the whole enterprise architecture or asset if compromised [32].

5.5.3. Usability

Traditional security architectures and corresponding controls are designed to protect resources, such as a server, files or subnets. They are driven by a resource-specific policy irrespective of the user requirements. EDAS offers a user- and service-centric security solution. All requirements pertaining to the user, service, as well as critical resources are considered and assessed in individual adverse contexts before any decisions are made. End-user solutions may accommodate user preferences to some extent, but overall, the emphasis is the resource.

5.5.4. Latency and Throughput

Architectures designed for end users, *i.e.*, host and endpoint architectures, perform analysis locally where the events of interest occur and, thus, have low latency. The other listed architectures, including EDAS, process events away from the point of occurrence and are subjected to delays caused by the network and communication. However, the EDAS autonomous adaptation property decreases any response management delays caused by a human in the loop. Thus, it results in maximum throughput as compared to the rest of the architectures.

5.5.5. Security

We have already established in Section 1 how traditional controls are not feasible for resource-constrained things and how they lack analysis of a context holistically, though they do provide security to a certain level. We have also detailed how EDAS addresses these issues by transferring computations required at a thing level to a resourceful machine (EDAS platform) and correlating different types of events in time to provide grounds for accurate analysis that reduces potential false alarms. Its cross event correlation feature can analyze a spectrum of threats, such as power exhaustion, confidentiality, intrusions, *etc.* Traditional controls can only defend against a defined set of threats. These can be the network, the web, local files or OS-related risks. Hence, their security aptitude is very restricted, and due to this lack of scope and context, they usually result in false alarms [5,11].

5.5.6. Monitoring Scope

EDAS can monitor any event source as long as it is accessible and can communicate the events that it generates. This makes the monitoring scope of EDAS much broader as compared to traditional mechanisms that only monitor a specific concern.

5.5.7. Adaptability

To overcome potential management delays and to meet the dynamic nature of IoT, EDAS offers autonomic security adaptation. Adaptation is a key desirable attribute in IoT environments [19,20]. Traditional architectures and solutions lack security adaptation and approach it in a manual manner where responses to threats are managed by a human in the loop. Of course, not all actions or configurations, for instance plugging in a wire or charging a smartphone to ensure availability, can be automated. However, all electronic operations can be automated, provided there are no physical engagements required. The objective of automation is to minimize the administrator or analyst interfacing with the system to increase throughput. In such circumstances, risk analysts may focus more on designing new criteria and rules for threat analysis, which can be added to the EDAS platform as security updates. Automation may also reduce the cost of the overall administration, as less effort will be required due to minimal manual configurations. The feasibility and degree to which an adapted action or activity can be automated is use case-, scenario- and risk context-specific and is beyond the scope of this study.

5.5.8. Threat Detection Accuracy

Please refer to Section 5.5.5 Security of the comparison discussing event correlation.

5.5.9. Simplicity

As a CBA, EDAS separates concerns into functional components. At the component level, separating concerns make it easier for developers to isolate, understand, verify and develop functionalities with fewer complexities [28]. However, at the system level and size of the monitored environment, its complexity is increased as more components are added. This notion also affects agent-based, centralized and distributed architectures. End-user architectures are comparatively simple to design, develop and implement due to the fewer number of components involved.

5.5.10. Extensibility and Maintainability

EDAS, as well as agent-based, centralized and distributed architectures are relatively hard to maintain and may demand increased cost due to the increased number of components involved. End-user solutions have a limited scope and fewer components. Thus, they take less effort and cost to maintain them. However, this maintenance advantage comes with a limitation of extensibility. Since end-user architectures are designed to meet specific and limited objectives, they cannot be extended to accommodate other systems. Agent-based systems can also be extended at the cost of an extra component, *i.e.*, the agent. Centralized and distributed architectures may also provide room for extension. However, the limited context they protect limits their extension scope. While EDAS can be extended to accommodate any system or thing that has a potential event framework.

5.5.11. Supportability

Almost all traditional security systems are equipped with mature event frameworks and logging mechanisms that can be used to resolve errors, failure and trace user activity. Since EDAS utilizes the same utility, it can potentially provide the same level of support.

5.5.12. Testability

While it is easy to create a test criterion against an individual component in a modular design, such as in a CBA, it is relatively complex to validate the entire system [29]. The increased number of components in EDAS, agent-based, centralized and distributed architectures makes it also hard to test and validate them. Interactions may be required in components distributed across network locations, which may make testing more complex [31]. On the other hand, end-user systems can easily be tested, as they are considerably less complex.

6. Related Work

Since IoT comprises diverse technologies with varying capacities, there is a need to design appropriate architectures and mechanisms, such that information sharing and communication can be made more efficient. Credible work has been concluded in this regard, while others are still under research. Below, we discuss a few of them.

The OpenIoT project [33] details an open source architecture that aims to connect Internet-enabled things with cloud computing, thus enabling ICT companies to offer sensor-based solutions. The architecture of [34] consists of three layers, namely application, virtualization and physical plane. The cloud computing capabilities are placed in the virtualization plane. Sensor middlewares are placed in the physical plane, which collects, filters and normalizes sensors data and communicates them to the cloud. The application plane contains utilities that enable users to control and monitor the sensor. These utilities also control requests made from connected services.

A similar model is also produced in the IoT-A project [35]. However, the artifact, IoT Architectural Reference Model (ARM) [36], provides a more abstract and domain-specific description as opposed to detailing the technicalities. It consists of three major components. An IoT Reference Model stipulates the high abstraction level definitions, as well as information and communication models. These definitions and abstract models are driven by the business vision, scenarios and stakeholder requirements and serve as the second major component. Its IoT Reference Architecture provides a reference for developing IoT compliant architectures and mainly details various views, such as functional, deployment, operational and perspectives, such as security, resilience, performance and interoperability, derived from the scenarios and requirements.

Antii *et al.* [37] proposed a self-adaptive architecture for smart spaces, which utilizes an information security measurement ontology (ISMO) to carry out the adaptation process. The model is inspired from IBM's MAPE-K control loop [38]. Though the authors provide a detailed view of the architecture, they did not explicate how user requirements and things' hardware capacities should be addressed while new security strategies are adapted to a given situation.

Other adaptive security solutions and studies, for instance risk adaptable access control (RAdAC) [39], context-sensitive adaptive authentication [40], the RSA Adaptive Authentication platform [41], security event information management solutions, such as AlienVault Unified Security Management [42] and HP ArcSight [43], *etc.*, either emphasize a single security service, e.g., authentication or confidentiality, or lack automated adaptation as an essential risk management component.

EDAS is focused on events corresponding to any security-related service or activity, *i.e.*, intrusion, confidentiality, energy, mobility, *etc.* Conceptually, EDAS can be related to the IoT-A ARM model, as it is driven by requirements and scenarios designed for IoT-enabled eHealth in remote patient monitoring settings. It is based on IoT-eHealth essentials that we have identified previously as functional, security and risk management requirements in [44]. Though designed primarily for IoT-eHealth, we suggest that EDAS, as an event-driven architecture, can be utilized for any IoT-enabled services where things can generate and communicate events. However, this proposition further needs to be investigated.

7. Discussion and Further Work

In this section, we discuss issues related to how the architecture itself can be made secure, its dependency on event frameworks, concerns related to event communication and how EDAS can provide a holistic approach towards the increasing risk sophistication. These are a few top level challenges, observations and possible further work that will be discussed in this section.

7.1. Securing the Architecture

Since EDAS aims to evaluate the security of a critical infrastructure by capturing and communicating security information (events), its adaptation loop also needs to be protected. The protection becomes more serious when its components are deployed in distributed settings. It must be ensured that the events, carrying security information, are communicated via well-protected channels and protocols and remain genuine and protected during the communication. Furthermore, access to the platform needs to be protected and well managed, and mechanisms should be provided to ensure its availability. To meet these requirements, we suggest the following:

(i) The EDAS platform should also monitor itself. This implies that there is a security monitoring and adaptation loop within the platform itself. Thus, EDAS should be considered as a critical event source in the architecture and needs to be monitored as other monitored objects in the scope.

(ii) Security tools, such as firewalls, intrusion detection, access controls and availability monitoring applications, should be installed on the platform to provide the required security services to it. The events generated by these applications can also be integrated into the architecture to make the analysis process more accurate.

(iii) Event communication protocols should provide security mechanisms for confidentiality and integrity. Furthermore, it should be energy efficient to comply with the resources of low-end devices, such as sensors.

EDAS is primarily dependent on an event framework that can generate, handle and log meaningful events. In EDAS, we assume that event framework has already been programmed into the thing. This facility is typically available as an out-of-the-box solution for various purposes, including debugging, error tracking, security logging and other troubleshooting operations. In IoT-eHealth, it is a must have component for patient safety to ensure that the body sensors are working reliably.

7.2. Event Communication

While most applications (things) log events, they do not have mechanisms to communicate them to remote destinations. For this purposes, various protocols can be utilized that can read from the event log file or hook onto the event framework output stream and communicate them remotely in a timely manner. Some of these are logging specific, for instance SYSLOG [45] and SNMP, while general protocols, like HTTP, REST architecture [28] or MQTT [46], can also be utilized. However, the question to investigate here is: which of these will be more reliable and efficient to be used as communication agents in low-end devices as sensors in IoT? As already mentioned, these events contain critical information that should be protected during communication. Therefore, the event communication protocol needs to ensure that channel is well protected. Furthermore, to ensure energy efficiency and fast delivery factors, like the protocol's packet size, request-response time and other QoS attributes needs to be evaluated before it is adopted. For instance, just to reflect on this issue, Stephen in an experiment [47] shows that MQTT saves about 30% of battery as compared to HTTPS when 1024 messages each of one byte are sent over a period of 1 h. In the same experiment, he also shows that MQTT can send about 94-times more messages than HTTPS over a 3 G network and 72-times more messages over a WiFi connection. An experiment like this implies that one must consider the communication protocol in an environment like IoT where resource-limited devices are the primary driving force.

In EDAS, we assume that the security analysts who design the threat correlation contexts have a keen understanding of the different types of events generated by a multitude of things in the monitored scope. It seems to be a complex task; however, most mature products (things) are provided with well-documented reference material of which they can take advantage.

7.3. The Security Adaptation Ontology

The security adaptation ontology is pre-configured and populated with the necessary knowledge before the system start-up. At this stage of development, we consider pre-determined ontology data based on the expert knowledge. Therefore, ontology modification at this stage refers to updating pointers in the existing knowledge contained as RDF data markups to reflect the current security posture of the monitored environment. Automated knowledge management, e.g., adding new threat members to the *Threat* entity in the ontology, can be done either by utilizing a machine learning technique, which can learn from the situations being dealt with, by employing update service resources or by other related methods. Such management methods are not considered in the study scope.

Furthermore, we have used only abstract level contextual requirements, such as patient's preferred privacy, availability, usability levels and a few service and device physical capabilities for each *Property* used in the ontology. However, these requirements are vital metrics for adaptation and must be carefully understood and designed. Further research is required to analyze and formulate such metrics and measurements that can reflect user preferences, security and services requirements and thing capabilities, as well as how they influence each other during the adaptation process. Potential work to approach in this regards can be security metric and quality of protection (QoP) modeling techniques, such as [48–50].

7.4. Dealing with Advanced Threats

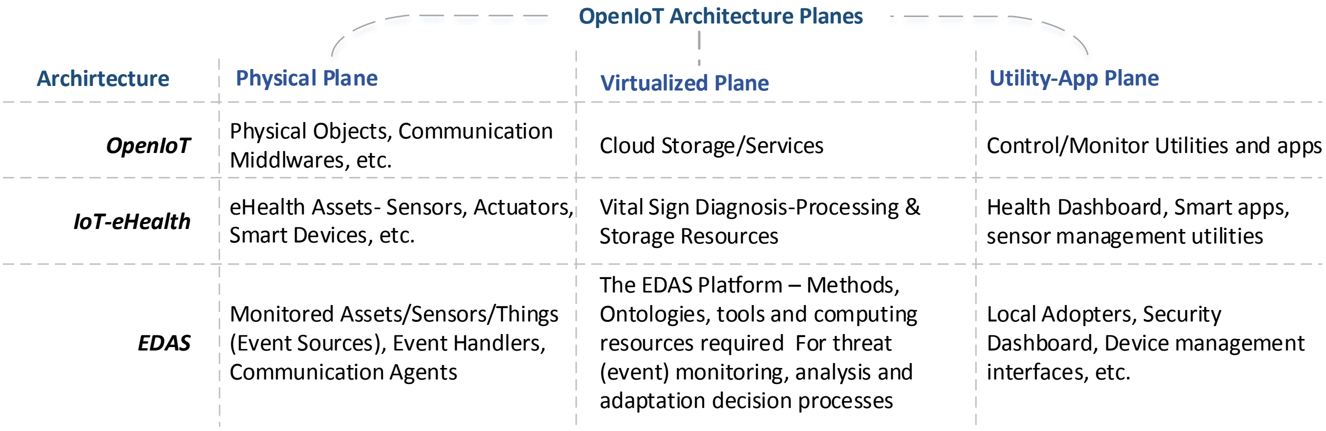
Traditionally, mainly the inbound communications are considered the most concerning for which only preventive controls are employed. However, they may not be sufficient: first, because they may not be suitable for the increasing threat sophistication and, secondly, they mainly focus on the inbound communication [51]. For instance, traditional intrusion detection or prevention systems are based on the concept of intrusion and, in general, are used to detect and analyze inbound communications only. Thus, they lack protection against the insider threat. Furthermore, depending on their architecture, they either analyze network packets or host information. Hence, the scope and context they analyze is limited. Advanced threats, also known as advanced persistent threats (APT) [52], are considered to be highly sophisticated, possibly exploiting zero-days and can be very challenging for a complex and diverse network like the IoT. Therefore, relying on a single analysis technique or method may be insufficient to address the threat landscape faced. We should have appropriate multiple detection capabilities as a second line of defense, which should consider both the outsider and insider threat.

An in-/out-bound activity may trigger a number of events. If we focus only on the activity type, the chances are that we may omit critical events necessary for rigorous analysis. EDAS ensures event-driven analysis where events from multiple activities, irrespective of their type, can be investigated during event correlation. Furthermore, we suggested that any CEP-compatible methods and tools can be utilized, including the traditional solutions, as they may provide various security alerts (events) as an input. Hence, combining different analysis techniques and tools, such as rule-based, profiling and resource reputation measures, statistical methods, behavior analysis, *etc.*, to collectively analyze a situation may appropriately address advanced threats. Lastly, if the events handled in EDAS are adequately logged and managed, they can be utilized as key learning resources for enhancing the underlying analysis techniques.

7.5. Utilization in IoT System Architectures

Several system models and architectures have been proposed for IoT. We have discussed a few of them in Section 6, such as the Open IoT [34] and IoT-A [36] architectures. One may also find domain-specific system architectures in the literature related to grids, vehicular systems, eHealth, *etc.* These architectures detail technological, system or application perspectives of IoT. They categorize physical objects, applications and services to distinguish and model consumer and service operations. They may describe the means and methods that can be used to communicate between the possible interfaces and how the architecture can be employed as a cloud service or in a particular business or public domain, *etc.* EDAS proposes a security perspective and architecture detailing how a particular service, *i.e.*, security adaptation, can be modeled, such that it can be adopted in any of the mentioned or related IoT system architectures.

As an example, Figure 18 instantiates how EDAS can be utilized in the OpenIoT architecture [34] in the context of IoT-eHealth. The first row represents the OpenIoT architecture itself and the distribution of different things, services, applications and utilities it employs at each plane it describes. The second row shows the IoT-eHealth and its corresponding elements as a possible application archetype of the OpenIoT architecture. The last row specifies the possible implementation of EDAS and its components in the OpenIoT. The physical plane will encompass the event sources, *i.e.*, the monitored assets, such as sensors, smart devices and their event frameworks. The virtualized plane will consist of the methods and tools necessary for risk analysis and response, *i.e.*, the EDAS platform. It can either be implemented in the cloud or as a dedicated server inside the hospital-controlled environment. Utilities, such as local adopters, security dashboards and other admin tasks, can be implemented in the utility-app plane. The figure dictates that if adaptive security is desired in an OpenIoT-related architecture, EDAS and its components can be readily implemented in it as illustrated. Furthermore, as previously mentioned, EDAS offers independent and extensible components that can be distributed and employed as per the requirements of a given IoT system architecture, such as the OpenIoT.



OpenIoT Architecture Planes

Archirtecture	Physical Plane	Virtualized Plane	Utility-App Plane
OpenIoT	Physical Objects, Communication Middlwares, etc.	Cloud Storage/Services	Control/Monitor Utilities and apps
IoT-eHealth	eHealth Assets- Sensors, Actuators, Smart Devices, etc.	Vital Sign Diagnosis-Processing & Storage Resources	Health Dashboard, Smart apps, sensor management utilities
EDAS	Monitored Assets/Sensors/Things (Event Sources), Event Handlers, Communication Agents	The EDAS Platform – Methods, Ontologies, tools and computing resources required For threat (event) monitoring, analysis and adaptation decision processes	Local Adopters, Security Dashboard, Device management interfaces, etc.

Figure 18. EDAS utilization in OpenIoT architecture.

8. Conclusions

We have explained and evaluated the feasibility of an autonomous event-driven adaptive security prototype based on our proposed architecture. We have expressed how our proposed concept of event-driven security and adaptation ontology ensures context-aware adaptation and complies with device capabilities, as well as user, QoS and security preferences. In the evaluation, there is potential evidence that EDAS, with its event-driven concept and adaptation control loop, is satisfying the requirements needed for managing risk in a dynamic environment, such as the IoT. It provides a real-time risk management platform and ensures autonomous and context-aware adaptive security. We have suggested that traditional security controls are not feasible for IoT in terms of resources and security. However, they can be still used in the traditional settings and can provide input to EDAS for security analysis, as they have a mature set of event frameworks. Hence, EDAS also motivates the merger of traditional enterprise architecture with the thing world: for instance, merging traditional eHealth systems with remote patient monitoring with wearable sensors to make the health system more reachable, accessible and efficient. Critical development and implementation issues, such as secure and reliable event communication, protecting the architecture and metrics required for adaptation, are highlighted, which can be further explored to make EDAS a more reliable solution for IoT-enabled services.

Acknowledgment

This work presented is sponsored by the Adaptive Security in Smart IoT in eHealth (ASSET, 2012–2015) project funded by the Research Council of Norway under Grant Agreement No. 213131/O70. We wish to thank our project colleagues and the anonymous reviewers for their valuable comments and suggestions.

Author Contributions

Waqas Aman and Einar Snekkenes designed the concept and structure of the article. Waqas Aman developed the prototype and performed the scenarios implementation. Waqas Aman and Einar Snekkenes wrote and reviewed the article.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. Available online: http://www.emc.com/leadership/digital-universe/2014iview/internet-of-things.htm (accessed on 21 December 2014).
2. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
3. Smith, C.; Miessler, D. The Internet of Things Research Study. Available online: http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf (accessed on 24 February 2015).
4. OWASP Internet of Things Top 10 Project. Available online: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Top_Ten_Project (accessed on 24 February 2015).
5. Shackleford, D. *Real-Time Adaptive Security*; Technical Report, SANS; 2008. Available online: http://www.sans.org/reading-room/whitepapers/analyst/real-time-adaptive-security-34740 (accessed on 21 December 2014).
6. Aman, W.; Snekkenes, E. Event Driven Adaptive Security in Internet of Things. In Proceedings of the Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2014), Rome, Italy, 24–28 August 2014; pp. 7–15.
7. AlienVault OSSIM: The Open Source SIEM. Available online: http://www.alienvault.com/open-threat-exchange/projects (accessed on 21 December 2014).
8. Michelson, B.M. *Event-Driven Architecture Overview*; Patricia Seybold Group: Bridgewater, MA, USA, 2006.

9. Ganek, A.G.; Corbi, T.A. The dawning of the autonomic computing era. *IBM Syst. J.* **2003**, *42*, 5–18.
10. Sundmaeker, H.; Guillemin, P.; Friess, P.; Woelfflé, S. *Vision and Challenges for Realising the Internet of Things*; European Commission Information Society and Media DG: Brussels, Belgium, 2010.
11. MacDonald, N. *The Future of Information Security Is Context Aware and Adaptive*; Gartner RAS Core Research Note G00200385; Gartner: Stamford, CT, USA, 2010.
12. Szyperski, C. *Component Software: Beyond Object-Oriented Programming*, 2nd ed.; Addison-Wesley Longman Publishing Co., Inc.: New York, NY, USA, 2002.
13. Microsoft. Handling and Raising Events. Microsoft Developer Network. Available online: https://msdn.microsoft.com/en-us/library/edzehd2t(v=vs.110).aspx (accessed on 21 December 2014).
14. Java Class Event Handler. Available online: https://docs.oracle.com/javase/7/docs/api/java/beans/EventHandler.html (accessed on 21 December 2014).
15. Microsoft. Event Properties. Available online: https://technet.microsoft.com/en-us/library/cc765981.aspx (accessed on 24 April 2015).
16. Karg, D. OSSIM Correlation Engine Explained, 2004. Available online: https://www.alienvault.com/docs/correlation_engine_explained_worm_example.pdf (accessed on 21 December 2014).
17. ARQ—A SPARQL Processor for Jena. Available online: http://jena.apache.org/documentation/query/ (accessed on 21 December 2014).
18. Protege. A Free, Open-Source Ontology Editor and Framework for Building Intelligent Systems. Available online: http://protege.stanford.edu/ (accessed on 5 March 2015).
19. Ma, H.D. Internet of things: Objectives and scientific challenges. *J. Comput. Sci. Technol.* **2011**, *26*, 919–924.
20. Caceres, R.; Friday, A. Ubicomp systems at 20: Progress, opportunities, and challenges. *IEEE Pervasive Comput.* **2011**, *11*, 14–21.
21. Abowd, G.D.; Dey, A.K.; Brown, P.J.; Davies, N.; Smith, M.; Steggles, P. Towards a better understanding of context and context-awareness. In *Handheld and Ubiquitous Computing*; Springer-Verlag: London, UK, 1999; pp. 304–307.
22. Abowd, G.D.; Mynatt, E.D. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput. Hum. Interact.* **2000**, *7*, 29–58.
23. Stewart, A. On risk: perception and direction. *Comput. Secur.* **2004**, *23*, 362–370.
24. Pieters, W. Security and privacy in the clouds: A bird's eye view. In *Computers, Privacy and Data Protection: An Element of Choice*; Springer: Dordrecht, The Netherlands, 2011; pp. 445–457.
25. Metzger, A.; Chi, C.H.; Engel, Y.; Marconi, A. Research challenges on online service quality prediction for proactive adaptation. In Proceedings of the IEEE 2012 Workshop on European Software Services and Systems Research-Results and Challenges (S-Cube), Zurich, Switzerland, 5 June 2012; pp. 51–57.
26. Kazman, R.; Klein, M.; Clements, P. *ATAM: Method for Architecture Evaluation*; Technical Report, DTIC Document; Carnegie Mellon University Software Engineering Institute: Pittsburgh, PA, USA, August 2000.

27. Kazman, R.; Bass, L.; Webb, M.; Abowd, G. SAAM: A method for analyzing the properties of software architectures. In Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, 16–21 May 1994; IEEE Computer Society Press: Washington, DC, USA, 1994; pp. 81–90.
28. Fielding, R.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University of California, Irvine, CA, USA, 2000.
29. Microsoft. *Chapter 16- Quality Attributes. Microsoft Application Architecture Guide*, 2nd ed.; Microsoft Corporation: Redmond, WA, USA, 2009.
30. Barbacci, M.; Klein, M.H.; Longstaff, T.A.; Weinstock, C.B. *Quality Attributes*; Technical Report, DTIC Document; Carnegie Mellon University Software Engineering Institute: Pittsburgh, PA, USA, December 1995.
31. O'Brien, L.; Merson, P.; Bass, L. Quality Attributes for Service-Oriented Architectures. In Proceedings of the International Workshop on Systems Development in SOA Environments (SDSOA 2007), Minneapolis, MN, USA, 20–26 May 2007 ; IEEE Computer Society: Washington, DC, USA, 2007; pp. 3–9.
32. Fulp, E. Parallel Firewall Designs for High-Speed Networks. In Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006), Barcelona, Spain, 23–29 April 2006; pp. 1–4.
33. Open Source cloud solution for the Internet of Things. OpenIoT Project. Available online: http://www.openiot.eu/ (accessed on 03 July 2015).
34. Dimitropoulos, P.; Soldatos, J.; Kefalakis, N.; Bengtsson, J.E.; Giuliano, A. *D 2.3 OpenIoT Detailed Architecture and Proof-of-Concept Specifications*; Technical Report; 2013. Available online: http://www.openiot.eu/sites/all/themes/corporateclean/Files/OpenIoT_D23.pdf (accessed on 24 April 2015).
35. Internet of Things Architecture (IoT-A) Project. An EU FP7 Project. Available online: http://www.iot-a.eu (accessed on 03 July 2015).
36. *Project Deliverable D1.2-Initial Architectural Reference Model for IoT*; Technical Report, IoT-A Project; 2011. Available online: http://www.iot-a.eu/public/public-documents/documents-1/1/1/d1.2/at_download/file (accessed on 24 April 2015).
37. Evesti, A.; Suomalainen, J.; Ovaska, E. Architecture and knowledge-driven self-adaptive security in smart space. *Computers* **2013**, *2*, 34–66.
38. Kephart, J.O.; Chess, D.M. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50.
39. McGraw, R.W. Risk Adaptable Access Control, 2009. Available online: http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Paper0001.pdf (accessed on 24 April 2015).
40. Hulsebosch, R.; Bargh, M.S.; Lenzini, G.; Ebben, P.; Iacob, S.M. Context sensitive adaptive authentication. In *Smart Sensing and Context*; Springer Berlin: Heidelberg, Germany, 2007; pp. 93–109.
41. RSA. RSA Adaptive Authentication. A Comprehensive Authentication and Fraud Detection Platform, 2012. Available online: http://www.emc.com/collateral/data-sheet/h11429-rsa-adaptive-authentication-ds.pdf (accessed on 24 April 2015).

42. AlienVault Unified Security Management. Available online: http://www.alienvault.com/products (accessed on 24 April 2015).
43. Security Information & Event Management (SIEM) Solutions ArcSight. Available online: http://www8.hp.com/us/en/software-solutions/siem-security-information-event-management/ (accessed on 24 April 2015).
44. Aman, W.; Snekkenes, E. An Empirical Research on InfoSec Risk Management in IoT-based eHealth. In Proceedings of the Third International Conference on Mobile Services, Resources, and Users (MOBILITY 2013), Lisbon, Portugal, 17–22 November 2013; pp. 99–107.
45. System Logger: Syslog Linux Man Page. Available online: http://linux.die.net/man/3/syslog (accessed on 31 May 2014).
46. MQ Telemetry Transport, MQTT. Available online: http://mqtt.org/ (accessed on 21 December 2014).
47. Nicholas, S. Power Profiling: HTTPS Long Polling *vs.* MQTT with SSL, on Android, 2012. Available online: http://stephendnicholas.com/archives/1217 (accessed on 21 December 2014).
48. Savola, R.M.; Heinonen, P. Security-measurability-enhancing mechanisms for a distributed adaptive security monitoring system. In Proceedings of the IEEE 2010 Fourth International Conference on Emerging Security Information Systems and Technologies (SECURWARE 2010), Venice, Italy, 18–25 July 2010; pp. 25–34.
49. Ksiezopolski, B.; Zurek, T.; Mokkas, M. Quality of Protection Evaluation of Security Mechanisms. *Sci. World J.* **2014**, doi:10.1155/2014/725279.
50. Rusinek, D.; Ksiezopolski, B.; Wierzbicki, A. Security Trade-Off and Energy Efficiency Analysis in Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2015**, *501*, 943475.
51. Cole, E. Advanced Persistent Threat (APT) and Insider Threat, 2012. Available online: http://goo.gl/ZdOWvO (accessed on 24 April 2015).
52. Pingree, L.; MacDonald, N.; Firstbrook, P. Best Practices for Mitigating Advance Persistant Threats, 2013. Available online: http://goo.gl/StNQEz (accessed on 24 April 2015).