RT-SPDM: Real-time Security, Privacy & Dependability Management of Heterogeneous Systems

Konstantinos Fysarakis[1], George Hatzivasilis[1], Ioannis Askoxylakis[2], and Charalampos Manifavas[3]

[1]Dept. of Electronic & Computer Engineering, Technical University of Crete, Greece
{kfysarakis, gchatzivasilis}@isc.tuc.gr
[2]FORTH
asko@ics.forth.gr
[3]Dept. of Informatics Engineering, Technological Educational Institute of Crete, Greece
harryman@ie.teicrete.gr

Abstract. The need to manage embedded systems, brought forward by the wider adoption of pervasive computing, is particularly vital in the context of secure and safety-critical applications. This work presents RT-SPDM, a framework for the real-time management of devices populating ambient environments. The proposed framework utilizes a formally validated approach to reason the composability of heterogeneous embedded systems, evaluate their current security, privacy and dependability levels based on pre-defined metrics, and manage them in real-time. An implementation of Event Calculus is used in the Jess rule engine in order to model the ambient environment context and the rule-based management procedure. The reasoning process is modeled as an agent's behavior and applied on an epistemic multi-agent reasoner for ambient intelligence applications. Agents monitor distinct embedded systems and are deployed as OSGi bundles to enhance the real-time management of embedded devices. A Service Oriented Architecture is adopted, through the use of the Devices Profile for Web Services standard, in order to provide seamless interaction between the framework's entities, which exchange well-formed information, determined by the OASIS CAP standard. Proof-of-concept implementations of all entities are developed, also investigating user-friendly GUIs for both the front-end and back-end of the framework. A preliminary performance evaluation on typical embedded devices confirms the viability of the proposed approach.

Keywords: SOAs, DPWS, event calculus, formal methods, security validation, metrics composition, JADE, Jess, OSGi, policy-based access control

1 Introduction

Advances in computing and communication technologies have enabled a new reality where interconnected computing systems, in various forms, permeate our environments, aiming to enhance all aspects of our everyday lives. These significant changes did not leave the industrial and enterprise environments unaffected, with ubiquitous

adfa, p. 1, 2015.

computing acting as an enabler for new business opportunities and services, but also providing more sophisticated tools for monitoring and managing the existing business functions and infrastructures.

However, the above also introduce significant challenges in terms of the security and privacy of the data processed, stored and communicated by these systems [1]. The complexity and heterogeneity of the underlying infrastructures, systems and networks, along with the limited resources of typical embedded devices, only exacerbate said issues. While existing networking and security mechanisms are updated and adapted to handle the vast population of these resource-constrained devices, higher level, machine to machine interactions, are a requirement in order to effectively monitor and manage the infrastructure, allowing the use of its full potential.

This paper presents RT-SPDM, a framework for supervising the wide range of ubiquitous computing systems, heterogeneous in nature, that are typically found in enterprise and industrial environments (e.g. embedded platforms, devices featuring sensors & actuators, personal computers, smart phones etc.). RT-SPDM leverages the benefits of Service Oriented Architectures (SOAs) to allow real-time monitoring the Security, Privacy and Dependability (SPD) of local and remote systems and their corresponding subsystems, using a formally validated reasoning process to monitor the changes in their SPD states.

This work is organized as follows: Section 2 sets the background and highlights related research efforts, Section 3 details the framework's components and architecture, Section 4 presents the approach followed to produce a proof-of-concept implementation and a preliminary evaluation of its performance, and, finally, Section 5 features the concluding remarks.

2 Background

In [2] we have presented a preliminary version of a composition verification and security validation reasoning system, where event-driven model-based methods have been proposed to describe the behavior of a dynamic system in a formal manner.

The work presented here extends this reasoning system by adopting the multi-metric formation and normalization process [3] and the composition evaluation formulas of the medieval castle approach [4]. We propose the use of standardized mechanisms and protocols, both for context-aware modelling as well as real-time control and management of the devices. The resulting system implements a formal methodology in system composition verification and SPD validation, supporting a metric-driven reasoning process, via SOA-based interfaces that enable seamless access to the various devices and their functional elements.

2.1 Service-oriented Architectures in Embedded Systems

The heterogeneity of ubiquitous devices has compelled researchers and developers alike to focus on mechanisms that guarantee interoperability, providing seamless access to the various devices and their functional elements. Service Oriented Architectures

(SOAs) evolved from this need to have interoperable, cross-platform, cross-domain and network-agnostic access to devices and their services. This approach has already been successful in business environments, as SOAs allow stakeholders to focus on the services themselves, rather than the underlying hardware and network technologies.

As Service-oriented Architectures (SOAs) are widely used, there is now an effort to apply this technology on embedded systems as well. To enable these services in such systems, several standards and platforms have been proposed. The Devices Profile for Web Services (DPWS) [5] OASIS standard targets resource-constrained embedded devices and enables secure web service messaging, discovery, description, and eventing.

The Open Service Gateway initiative (OSGi) [6] is a standard module system and service platform in Java and implements a complete and dynamic component model. Components are modeled as bundles for deployment, which can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot.

In the proposed framework, DPWS is integrated into OSGi by implementing operators which are controlled by the relevant system components via an OSGi interface, as will be detailed in later chapters. Embedded devices specify their type and the provided services in DPWS. Each system component implements a component operator bundle that handles its underlying DPWS devices and their services.

2.2 Knowledge Representation and Reasoning

Knowledge Representation and Reasoning (KR&R) is the research field of Artificial Intelligence (AI) that deals with the representation of information about the world in a manner that it is understandable by a machine and can be utilized in order to solve complex tasks. KR&R is strongly related with semantic technologies which encode knowledge to express ontologies, and logic languages which include reasoning rules for processing that knowledge.

Event Calculus (EC) [7] is a logic language for representing and reasoning about actions and their effects as time progresses. An implementation of EC in Java and the rule engine Jess (Jess-EC) is utilized [8]. The whole reasoning model is formed as an agent's reasoning behavior and is implemented on the Java Agent Development framework (JADE) [9], implementing an epistemic multi-agent reasoner. The agents utilize the FIPA standardized Agent Communication Language (ACL) [10] to exchange information; a conflict resolution mechanism between agents' local knowledge is also supported.

The Common Alerting Protocol (CAP) [11], an XML-based data format OASIS standard for exchanging public warnings and emergencies, is used to model semantic information that is exchanged between the entities. The CAP alerts are transformed into Jess-EC events and trigger the rule-based reasoning of CompoSecReasoner and AmbISPDM. The abovementioned two operations are then combined to achieve a holistic metric-driven SPD management.

2.3 Metrics and Methodologies for Quantifying Security

Metrics for evaluating aspects like security and performance are becoming an integral feature in system development. Metrics provide a quantitative assessment of a system's compliance with the application's requirements. This also enables comparisons between different system settings based on objective factors, facilitating the selection of the ones that are more appropriate with respect to the design and/or specific deployment criteria. Moreover, it enables metric-driven management procedures for real-time systems.

In [3] a multi-metric approach is proposed for forming and normalizing different types of metrics in order to compose them and estimate the overall security level of a system. Each metric consists of a triple vector <Security, Privacy, Dependability>, referred to as SPD, where each parameter takes a value from [0-100]. Moreover, each metric has a value-type (e.g. time in ms or security in bits) and takes values from a set (e.g. time from [0.1-5] ms or encryption key length of [128,192,256] bits). Every value in the set is normalized and mapped to an equivalent SPD. All the provided metrics are then composed under a specific formal composition strategy to produce the current overall SPD level of the system.

The medieval castle approach [4] forms the basis of the method adopted to compose the metrics and quantify the security level of the system. Using this approach, a system is considered as a castle and the security mechanisms as the castle's doors. An intruder tries to break through these doors and reach the treasure rooms inside the castle, i.e. the system's assets that must be protected. The difficulty in passing through a door is represented by a relevant metric and the minimum effort to reach a treasure room is the final security level of the system.

3 The Proposed Framework

3.1 Overview

The proposed framework is a multi-agent system, implemented in the JADE platform which, via the OSGi middleware, is able to monitor and manage DPWS devices. The AI reasoning process (CompoSecReasoner and AmbISPDM) is an event-based model checker that is based on EC. The context theory is aware of the security, privacy and dependability aspects of the underlying embedded system (component types and composition, implemented technologies and their different configurations, SPD metrics etc.).

In order to effectively study and model embedded systems and their security characteristics, their architecture is segregated into four layers. These layers, from bottom-up, are: **Node** (represents all the embedded devices themselves), **Network** (consists of nodes connected in networks), **Middleware** (the management-software of the networks) and **Overlay** (formed by RT-SPDM agents, who control distinct sub-systems and exchange high-level security- privacy- and dependability-related information).

The proposed framework adopts the above layered approach. RT-SPDM agents, implemented using the JADE platform, are part of the overlay, while the necessary DPWS and OSGi mechanisms operate at the middleware.

The framework's entities are deployed using DPWS, allowing each device to expose the services and operations needed to transmit its current status (to allow real-time monitoring) but also to receive commands (for management purposes). DPWS can be used to expose not just the proposed framework's monitoring/management operations, but also those related to their functional elements (e.g. the various sensors and actuators), allowing system owners to leverage the SOA benefits across their whole infrastructure. By exploiting these benefits, the proposed framework facilitates the communication of critical information regarding the secure & privacy-aware and dependable operation of the devices.

This information is then aggregated to a control node, where it is composed using EC-based rules, in a formally validated manner. The system's agents base their reasoning process on Jess-EC, which can perform, among others, automated epistemic, temporal and casual reasoning for dynamic domains with real time events, preferences and priorities; features that are necessary in the context of this work. The various SPD states and properties of each device are modelled as fluents, and the system changes as events of EC, all implemented using the abovementioned extended Jess-EC. Rules that trigger the reasoning process of the agent are also supported, producing a metric-driven management of the infrastructure according to its SPD level. The updates on the current state as well as the changes that need to be enforced by the agent(s) are all realized via appropriate DPWS-based mechanisms present on all devices. These changes can be triggered by human interaction (e.g. from system operators) or by a predefined set of rules. Furthermore, in the case of large organizations or when monitoring facilities in segregated premises, a multi-agent system can be deployed, where each agent monitors the SPD levels of a subset of the infrastructure and communicates with the rest of the agents and/or a central agent to provide enterprise-wide monitoring & management.

The overall state, once calculated, is presented in a form which is usable and simple to understand by the system's operators, enabling real-time assessment of the security and dependability posture of the infrastructure and allowing for a timely response to changes in the system state (either via human interaction or in an automated manner). Thus, the proposed framework simplifies threat assessment and risk management. Moreover, it enables the interfacing with more sophisticated, intelligence-driven, management mechanisms, potentially fully automating the management of the ecosystem (e.g. for automatic incident response).

3.2 Core Components

RT-SPDM consists of two components: CompoSecReasoner and AmbISPDM. The former uses the reasoning system presented in [2] in order to implement an extended composition validation and security verification, while the latter models the management theory of the ambient environment and manages the system in real time.

CompoSecReasoner.

In the proposed methodology, technologies and protocols are modeled as attributes. A security analysis is performed for every developed attribute and a relevant SPD is defined for each security level that is provided. Then, the evaluated metrics and properties of the system are determined, including how they may be affected by the underlying attributes at runtime. This information is encoded in the evaluation functions of the relevant metrics and properties.

In order to model metrics and other measurable system parameters in a uniform manner, the previously-cited multi-metric approach and normalization processes are adopted. Each measurable value is modelled as an SPD vector (i.e. <Security, Privacy, Dependability>) and evaluated as in the original approach (i.e. in [3]). These parameters are used to reason about the security, privacy and dependability of a system and whether it can fulfill its specific application requirements. The SPD values are also integrated in the AI reasoning process to accomplish metric-driven real-time management.

For the system composition, verification and security validation, the system is considered as a set of components of four layers (node, network, middleware, overlay) and the components of each layer are composed of sub-components of the adjacent lower layer.

Each component: has a set of sources – data that are processed; a set of attributes – technologies and protocols; performs some operation – a series of attributes; achieves specific levels of SPD – based on metrics and its sub-components; and satisfies a set of properties under some conditions. The components, sources, attributes, operations, metrics and properties attain specific SPD values based on the core evaluation process of the formal methodology. Attributes possess a static SPD for each configuration setting, e.g. encryption with 128-bit keys will have a lower "S" factor in its (S,P,D) triplet compared to encryption with 256-bit keys, assuming a device that can operate with both key lengths. An operation's SPD is derived by evaluation formulas based on the medieval castle approach discussed above. The SPD of a source is the SPD of the last operation that processes it.

The metrics and the properties are layer specific and evaluate their SPD at runtime based on evaluation functions. A component's SPD is the summation of the component's metrics and the minimum SPD of the underlying sub-components (i.e. the weaker inner link). The component's SPD is constrained by the SPD of the higher layer component that contains it (i.e. the weaker outer link). Components are composed to higher layer components by composition operations. In order to perform such an operation, a component must be able to execute the operation's attributes (functional requirements) and achieve specific SPD for relative metrics and properties (non-functional requirements that are determined by the operation). When composition is successful, the composition verification and security validation process are revisited. A similar strategy is followed for decomposition operations.

AmbISPDM.

The core of the reasoning process is an event-based model checker which extends EC and is implemented in the Jess rule engine. CompoSecReasoner reasons about the system composition and SPD validation while AmbISPDM models the security and safety related management strategy and the system's administration through real-time technologies. The whole reasoning process is transformed into a JADE agent's reasoning behavior and implemented as a multi-agent epistemic reasoner.

The agents are then encapsulated into OSGi bundles and are deployed on Knopflerfish [12], an open-source implementation of OSGi. Each agent controls a distinct system via the OSGi middleware, while the various agents communicate with each other using the ACL language, with messages containing CAP data exchanged via JADE. The embedded devices specify their type and provided services using DPWS, which is also used to facilitate the exchange of CAP messages between managed devices and agents. Each underlying system component (node, network and middleware) that communicates directly with the agent creates a component-operator that controls the component's services. The agent and the component operators exchange well-formed information determined by the CAP scheme. The DPWS-related components are developed using the WS4D-JMEDS API [13] and are also integrated into OSGi bundles.

3.3 Security Mechanisms

The protection of messages exchanged between RT-SPDM entities is of critical concern. The framework can protect the communication between the managed embedded devices and their respective RT-SPDM agents through the use of the WS-Security standard [14] typically used alongside DPWS. Said standard can provide end-to-end security, non-repudiation, alternative transport bindings, and reverse proxy/common security token in the application layer.

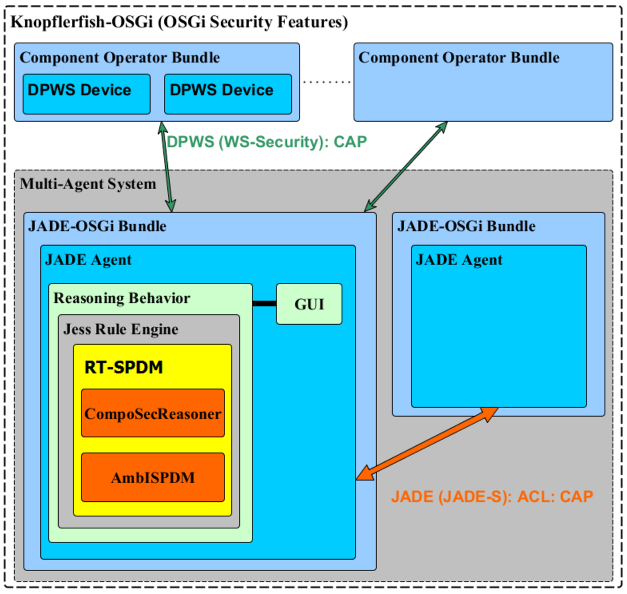


Fig. 1. Software Layers of the Proposed Framework

With regard to the protection of the communications between agents, the JADE-S add-on can be used to safeguard the ACL messages exchanged, by providing user authentication, agent actions authorization against agent permissions, as well as message signature and encryption. The OSGi security features can additionally be used to provide inner-platform security on both agents and component operator bundles (i.e. managed devices) by limiting bundle functionality (e.g. which bundles can be started/stopped, when, by whom etc.) to pre-defined capabilities. **Fig. 1**, illustrates the software layers of RT-SPDM.

4 Proof-of-Concept Implementation

As a proof-of-concept example, the proposed framework was developed and used to emulate monitoring and management of an ambient environment, e.g. a smart building, a smart production floor or a critical infrastructure [15].

The agents evaluate the current SPD levels of their underlying sub-systems and the system as a whole (CompoSecReasoner). They also monitor their respective domains, managing them based on SPD levels and the reasoning process (AmbISPDM). The agents can change the configurations of a system in order to increase the security level (e.g. increase the size of the encryption keys) when an attack is detected and then return to the previous state (to conserve resources) when the attack is over.

Moreover, a Policy-Based Access Control (PBAC) mechanism [16] is integrated for managing access to the infrastructure's resources based on the active policy set; said mechanism exploits and extends the DPWS functionality already present on the framework's devices, to realize the necessary mechanisms (i.e. Policy Enforcement Points on the embedded devices and Policy Decision and Information Points on the control nodes). It can be further exploited to allow automatically enforce high-level security requirements (e.g. stemming from risk analysis) [17].

Using these mechanisms, RT-SPDM is able to trigger changes to both the security mechanisms employed to protect the PBAC's messaging but also to the current active policy set. So, for example, in a safety-related incident, the agents can help the personnel in the case of a fire alarm (e.g. enabling them to unlock doors that they were previously not allowed to access), sacrificing security in doing so (thus reporting a lower "S" factor for the current SPD state). Moreover, it could allow system operators to monitor the exact location of personnel in the building (with obvious benefits to their timely rescue), but moving to an SPD state with lower "P", as privacy has been lost by enabling this direct monitoring. Finally, as some of the devices may be damaged in the incident, redundant hardware will have to take up their role (e.g. a backup camera taking over from the main one), which will be depicted as a drop in dependability (i.e. the "D" in the system's SPD state). When the fire is extinguished, the system returns to the normal SPD state.

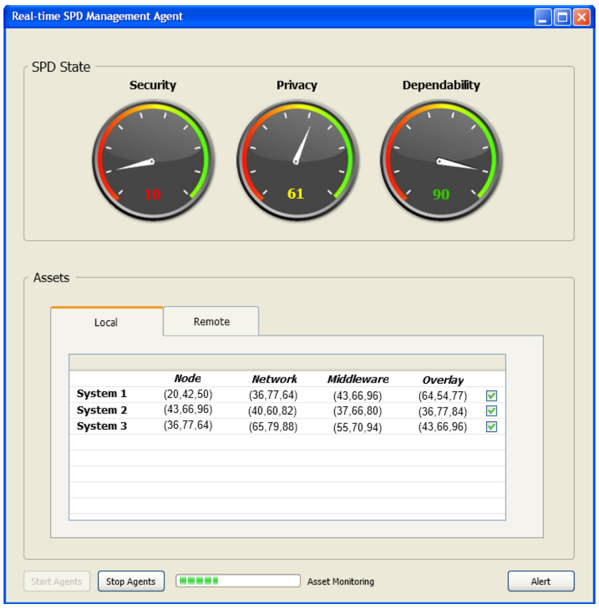
4.1 User Interface

While the technical aspects detailed in the previous chapters are important, a decisive factor to the practical success of any framework of this nature will be its usability. The accessibility both in terms of the usability of its user interface, as well as the success of visualizing the system's current state in a user-friendly and intuitive manner are important, as well as ensuring that it imposes minimal requirements from the end-user in terms of training, configuration and maintenance [18].

The inner workings of the system are irrelevant to the operator, who may or may not be technically proficient. Moreover, its output should be accessible to higher management, who, in the context of businesses, are usually people with limited time and accustomed to handling interfaces with dashboards, gauges and other high-level representations [19]. The preference to dashboard layouts is not restricted to higher management, but has been shown to be preferable to users in general [20], while recent research indicates there are significant benefits to this approach, helping address various business challenges (e.g. to provide a global view on resource occupation, aiming to enhance decision-making for both human and non-human resource management at run-time [21]). Thus, this is the approach adopted for visualizing the Agents' outputs and the corresponding user interface.

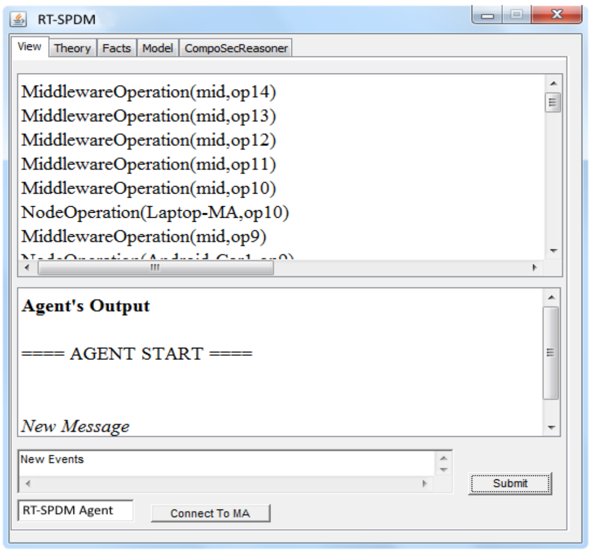
The SPD levels derived from RT-SPDM are plain numbers, e.g. (56,75,89), since they are the results of the composition process detailed in the previous sections. Though these numbers are appropriate to be transmitted and computed by machines (i.e. 'machine to machine' interfaces), when it comes to presenting them to human operators in user interfaces, there is the need for making their representation more intuitive. That would make the SPD level much easier to understand in order to allow operators to be aware of the situation and possibly operate manual countermeasures wherever appropriate. The appropriate presentation of information is highly dependent on the specific domain and application, however a basic template could be applied to most domains. To this end, and in order to further facilitate the comprehension of the SPD state for a human operator, the use of a graphical scale and appropriate colors should be used (e.g. red for all values below 40, yellow for values from 40 to 70 and green for values from 70 to 100). An example of this approach, chosen for the proof-of-concept implementation of RT-SPDM, appears in **Fig. 2**.

Other than the main GUI detailed above which presents the aggregated SPD state of the monitored infrastructure, each RT-SPDM agent also features its own backend GUI. This is aimed to advanced users/system operators, providing low level information of the SPD status changes reported by each subsystem. The events appearing as they happen on the screen, but these changes are also logged in the corresponding files for offline auditing. A screenshot of this user interface appears in **Fig. 3**.



	Node	*Network*	*Middleware*	*Overlay*	
System 1	(20,42,50)	(36,77,64)	(43,66,96)	(64,54,77)	✓
System 2	(43,66,96)	(40,60,82)	(37,66,80)	(36,77,84)	✓
System 3	(36,77,64)	(65,79,88)	(55,70,94)	(43,66,96)	✓

Fig. 2. The proof-of-concept RT-SPDM master agent GUI



```
MiddlewareOperation(mid,op14)
MiddlewareOperation(mid,op13)
MiddlewareOperation(mid,op12)
MiddlewareOperation(mid,op11)
MiddlewareOperation(mid,op10)
NodeOperation(Laptop-MA,op10)
MiddlewareOperation(mid,op9)
```

Agent's Output

```
==== AGENT START ====
```

New Message

Fig. 3. The backend GUI of RT-SPDM's agents

4.2 Performance Evaluation

The RT-SPDM proof-of-concept modules were deployed on platforms that are expected to be found in typical ambient environments. The end devices consisted of three

Beaglebone [22] embedded hardware platforms (720MHz ARM Cortex-A8 processor, 256MB of RAM). The sub-system agents and the master agent run on a desktop PC (Core i5 CPU, 8GB RAM), as is expected to be the case in actual deployments (e.g. a command & control backend system). The devices were interconnected via wired Ethernet, as they are assumed to be part of a smart building's infrastructure and the risk of relying on wireless connections to control critical functions (e.g. cyber-physical systems controlling actuators) is high. A preliminary performance evaluation was conducted to investigate the feasibility of the proposed mechanisms. Issuing 200 requests to the monitored devices, while triggering changes to their SPD state in-between, yielded promising performance, as illustrated in **Fig. 4**. The spikes noted occur when SPD changes take place (thus the agents are informed and the system SPD state is altered).

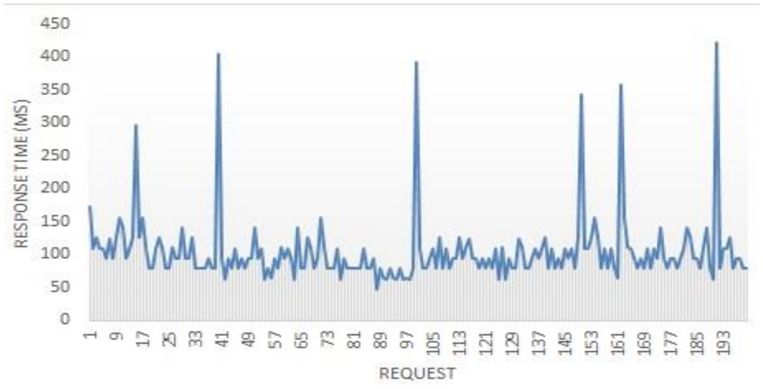


Fig. 4. Response time of the system as the SPD changes occur in the fire-alarm scenario.

5 Conclusions

This work presented RT-SPDM, a framework for the real-time management of the Security, Privacy and Dependability of ambient environments. The multi-agent system is implemented in the JADE platform and deployed on OSGi middleware for controlling embedded devices, leveraging the benefits of SOA to provide a platform-agnostic solution with seamless integration across heterogeneous systems. The core of the reasoning process is an event-based model checker which extends the Event Calculus, while Security-related theory is modelled and implemented as a formal framework for system composition verification, security validation and metric-driven management of SPD states. The proposed solution is applied to a smart-buildings scenario, where it configures the underlying systems at runtime to counter attacks and perform AI reactive plans to retain the safety of the employees in cases of emergency. A performance evaluation of the proof-of-concept implementation was conducted on typical embedded devices and the results were promising with regard to the feasibility of the proposed framework. An extended performance evaluation will be presented in future work.

Acknowledgments. Part of this work has been supported by the Greek General Secretariat for Research and Technology (GSRT), under the ARTEMIS JU research program nSHIELD (new embedded Systems arcHItecturE for multi-Layer Dependable solutions) project. Call: ARTEMIS-2010-1, Grant Agreement No.:269317.

References

1. Mayes, K. and Markantonakis, K.: Information Security Best Practices. Secure Smart Embedded Devices, Platforms and Applications, Springer, part II, pp. 119-144 (2013).
2. Hatzivasilis, G., Papaefstathiou, I., Manifavas, C., Papadakis, N.: A Reasoning System for Composition Verification and Security Validation. 6th NTMS 2014, pp. 1–4 (2014).
3. Eguia, I., Del Ser, J.: A Meta-heuristically Optimized Fuzzy Approach towards Multi-metric Security Risk Assessment in Heterogeneous System of Systems. PECCS. pp. 231–236 (2014).
4. Walter, M., Trinitis, C.: Quantifying the security of composed systems. Parallel Processing and Applied Mathematics. pp. 1026–1033. Springer (2006).
5. Devices profile for web services, version 1.1, http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf
6. Open Services Gateway Initiative (OSGi), http://www.osgi.org/
7. Mueller, E.T.: Commonsense Reasoning. (2006).
8. Patkos, T., Plexousakis, D.: Epistemic Reasoning for Ambient Intelligence. ICAART (1). pp. 242–248 (2011).
9. Java Agent DEvelopment (JADE) Framework, http://jade.tilab.com/
10. FIPA, A.C.L.: FIPA ACL message structure specification. Found. Intell. Phys. Agents, http//www. fipa. org/specs/fipa00061/SC00061G.html (2002)
11. OASIS: Common Alerting Protocol Version 1.2, http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf
12. Knopflerfish - Open Source OSGi service platform, http://www.knopflerfish.org/
13. WS4D-JMEDS DPWS Stack, http://sourceforge.net/projects/ws4d-javame/
14. Lawrence, K., Kaler, C., Nadalin, A., Monzilo, R., Hallam-Baker, P.: Web Services Security: SOAP Message Security 1.1, https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
15. Petroulakis, N.E., Askoxylakis I.G., Tryfonas T.: Life-logging in smart environments: challenges and security threats. IEEE ICC 2012, pp. 5680-5684 (2012).
16. Fysarakis, K., Papaefstathiou, I., Manifavas, C., Rantos, K., Sultatos, O.: Policy-based access control for DPWS-enabled ubiquitous devices. 2014 IEEE ETFA. pp. 1–8 (2014).
17. Tsoumas, V., Tryfonas, T.: From risk analysis to effective security management: towards an automated approach. Inf. Manag. Comput. Secur. 12, 91–101 (2004).
18. Hollan, J.D., Hutchins, E.L.: Designing User Friendly Augmented Work Environments. Designing User Friendly Augmented Work Environments. pp. 237–259 (2010).
19. Beuschel, W.: Encyclopedia of Decision Making and Decision Support Technologies. IGI Global (2008).
20. Read, A., Tarrell, A., Fruhling, A.: Exploring user preference for the dashboard menu design. 42nd Annual Hawaii International Conference on System Sciences, HICSS (2009).
21. Linden, I.: Proposals for the integration of interactive dashboards in business process monitoring to support resources allocation decisions. J. Decis. Syst. 23, 318–332 (2014).
22. BeagleBone System Reference Manual, RevA3_1.0, http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE_SRM.pdf