Security Infrastructure for Service Oriented Architectures at the Tactical Edge

Vasileios Gkioulos[1] and Stephen D. Wolthusen[1,2]

[1] Norwegian Information Security Laboratory, Norwegian University of Science and Technology, Norway
{vasileios.gkioulos,stephen.wolthusen}@ntnu.no
[2] School of Mathematics and Information Security, Royal Holloway, University of London, United Kingdom

Abstract. The requirement for enabling network centric warfare through the accommodation of network-enabled capabilities, promoted the use of service oriented architectures (SOA) within military networks. The initial response of the academic and industrial communities was to utilize standard enterprise SOA. The developed solutions were well adjusted to the strategic domain, where node and network constraints were minimal. Yet, experience gained from the battlefields of the last decade, has proven that the tactical domain imposes a set of unique constraints, that render such solutions inefficient for the tactical edge.
The project TACTICS, supported by the European Defense Agency, focuses on the study and development of a SOA dedicated to tactical networks. In this paper we present the designed security service architecture, as developed in accordance to the requirements identified in our earlier studies. Each service is presented as an architectural element within the TACTICS TSI (Tactical Service Infrastructure), aiming to highlight the distinct functionalities of the security infrastructure towards the efficient enforcement of security controls at the tactical edge.

Keywords: Ad-Hoc · Security · Service Oriented Architectures · Tactical networks. · Tactical Service Infrastructures

1 Introduction

The introduction of SOA across the strategic domain of military networks has been promoted by the increasing requirement for the integration of Network Enabled Capabilities (NEC), within the developed C2 (Command and Control) and C4I (Command, Control, Communications, Computers and Intelligence) systems. Extending this paradigm to the tactical domain, is expected to allow the widespread incorporation of Network Centric Warfare (NCW), by improving situational awareness and increasing network flexibility, adaptability and responsiveness at the tactical edge.

However, standard enterprise SOA have been proven across the AoO (Areas of Operations) of recent conflicts to be unsuitable for tactical networks, due to their rapidly evolving nature and constrained resources. The project TACTICS [1] is oriented towards the theoretical and experimental analysis of contemporary tactical networks, in respect to the feasibility and required adaptations for the deployment of SOA. Consequently, and in accordance to these studies, a Tactical Service Infrastructure (TSI) has

been defined and experimentally demonstrated. The TSI architecture [2] was developed according to the NATO Architecture Framework 3.1, including twenty discrete architectural perspectives.

Focusing on the security aspects of such an architecture, our study was initiated by analysing system specific constraints and requirements, arising due to terminal and network characteristics across the three mission stages (Preparation, Execution, Debrief). This allowed the identification of fine-grained security requirements and protection goals, maintaining the necessary distinction between the communication [3] and service domains [4].

Accordingly, these requirements have been translated into corresponding functional characteristics, for a security policy framework and service infrastructure, that would be suitable for the investigated environment. Furthermore, an extended state of the art review, revealed the weaknesses of existing mechanisms but also suitable adaptations that would satisfy the identified requirements under the imposed constraints [5]. These initial studies, allowed us to analyse, define and develop a suitable security policy framework [6], along with the corresponding distribution [7], reconciliation [8] and QoS (Quality of Service) interoperability [9] mechanisms.

In this article we present the core security service infrastructure, as developed within the TACTICS TSI in accordance to the aforementioned studies. These components are suitably adjusted towards satisfying the identified requirements, by facilitating the operation of the developed security policy framework and supporting mechanisms. The remainder of this paper is structured as follows: Section 2 presents the functionalities and interactions of each developed service, as an architectural element towards the extraction of valid policy decisions. Subsequently, section 3 includes a discussion over the operational complexity of the security service infrastructure, in accordance to early results from the ongoing field and laboratory experiments/demonstrations.

2 TACTICS Security Architecture

The developed security architecture, consists of two distinct groups of services, namely core and functional. The functional services are responsible for the enforcement of the requisite protection goals, by instantiating the distinct mechanisms (e.g. encryption algorithms, access control, intrusion detection), while the core security services are responsible for the governance of these mechanisms, in accordance to predefined security policies. In this section we present these components (Figure 1), aiming to highlight the processes involved in the extraction of suitable policy decisions (Figures 2 and 3). It must be noted that the security policy framework developed within TACTICS for the accommodation of the requirements imposed by contemporary tactical SOA, has been presented earlier in detail [6] and is outside the scope of this article.

The main functionalities of each service as presented in figures 1, 2 and 3 can be summarised as:

- **Security Handling service**
  1. Initiate the internal policy decision extraction process.
  2. Store and identify the applicability of precomputed policy decisions.
- **Policy Management service**

1. Control the policy decision extraction process.
2. Prioritize pending policy decision requests.

– **Policy Decision Point service**
1. Securely store the prioritized rule stacks that have been defined for each available policy decision request.

– **Metadata Handling service**
1. Accommodate the defined ontological knowledge base (Including both the Terminological-box and Assertional-box) and the selected inference engines.
2. Extraction of policy decisions.

– **Contextual Monitoring service**
1. Monitoring and collection of dynamic attributes.
2. Generation of statistical and aggregated data.
3. Triggering of event driven policy decisions to the Security Handling service.
4. Update of Metadata Handling service A-box to current values.

– **Policy Enforcement Point service**
1. Translation and enforcement of extracted policy decisions.

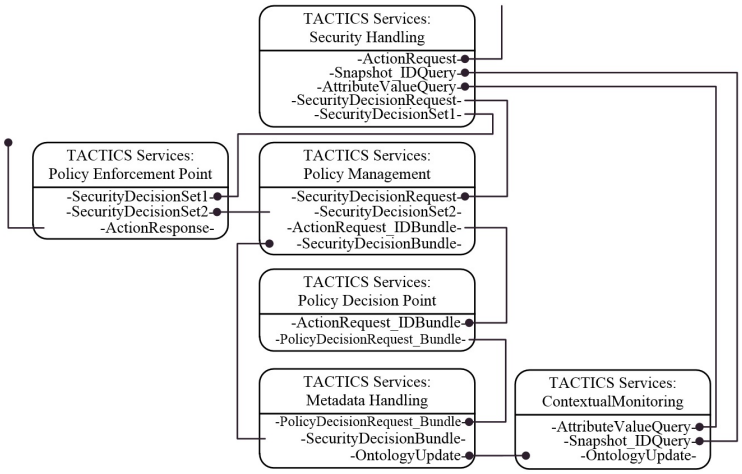

Fig. 1. Interfaces of the developed core security services.

2.1 Security Handling Service

Description: The Security Handling (SH) service operates as the internal to the security architecture action (policy decision) requester. The service can be invoked either externally (By a predefined set of core and functional services, for which the required interfaces have been established for the invocation of corresponding policy decisions) or internally (By the Contextually Monitoring (CM) service, for event driven policy decisions). The service accommodates precomputed policy decisions for the reduction of

Fig. 2. Sequence diagram for valid precomputed policy decisions.

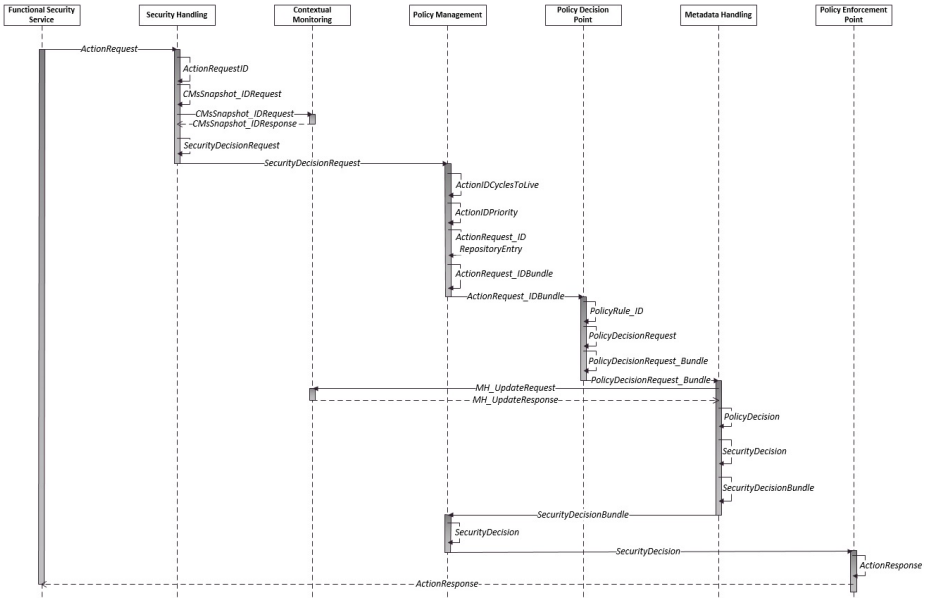

Fig. 3. Sequence diagram for the on-line extraction of policy decision.

the computational overhead imposed by the security architecture. These are established at the mission preparation stage, according to a statistical analysis of previous invocation logs and the use of computational intelligence methodologies. Thus, precomputed

policy decisions can be established for a constraint range of the required semantics, and after local evaluation, be directly applied without the invocation of the complete security service stack (Figure 2). When such precomputed policy decisions are not available or applicable, the SHs must compose and forward a security decision request to the subsequent security services, providing all the required information for the adaptation, prioritization and successful extraction of valid policy decisions.

Invocation: Invocation Originator $\Longrightarrow$ Invocation Form

1. Set of core and functional services (RequestorID) $\Longrightarrow$ ActionRequest.
2. Contextual Monitoring (CM) service $\Longrightarrow$ Pre-established ActionRequest according to attribute threshold alert.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$

1-$\rightharpoonup$ Receive ActionRequest
2-$*$ Generate ActionRequest_ID
3-$*$ Identify existence of precomputed PolicyDecision. According to ActionRequest_ID
a-$*$ IF(TRUE)
i-$*$ Identify required attributes
ii-$*$ Generate AttributeValuesRequest
iii-$\leftharpoondown$ Send AttributeValuesRequest to CMs for timely values
iv-$\rightharpoonup$ Receive AttributeValuesResponse from CMs
v-$*$ Evaluate attributes of precomputed PolicyDecision
1-$*$ IF(TRUE)
a-$*$ Generate SecurityDecision
b-$\leftharpoondown$ Send SecurityDecision to PEPs for enforcement
2-$*$ IF(Not TRUE)
a-$*$ Generate CMsSnapshot_IDRequest
b-$\leftharpoondown$ Send CMsSnapshot_IDRequest to CMs
c-$\rightharpoonup$ Receive CMsSnapshot_IDResponse from CMs
d-$*$ Generate SecurityDecisionRequest
e-$\leftharpoondown$ Send SecurityDecisionRequest to PMs
b-$*$ IF(Not TRUE)
i-$*$ Generate CMsSnapshot_IDRequest
ii-$\leftharpoondown$ Send CMsSnapshot_IDRequest to CMs
iii-$\rightharpoonup$ Receive CMsSnapshot_IDResponse from CMs
iv-$*$ Generate SecurityDecisionRequest
v-$\leftharpoondown$ Send SecurityDecisionRequest to PMs

2.2 Policy Management Service

Description: The Policy Management (PM) service operates as the controller of the security services that are involved in the policy decision extraction process. The PMs can be explicitly invoked by the Security Handling (SH) service, or execute functionality according to the input received from the Metadata Handling (MH) service. The invocation from the SHs includes all the required information for the management of a viable policy decision extraction within a security decision request. In addition to the action request related elements, an aggregated metric of the available local node resources (for

the prioritization of policy decision requests) and a refresh alert based on predefined constraints (for the update of the MHs to the current state of dynamic semantics) are also included. Upon receipt of a security decision request, the corresponding cycles to live and priority are identified according to the received available resources metric. Consequently, a corresponding entry is generated and included within a repository with all the pending security decision requests. The repository entries are prioritized and a bundle is created, including those requests that can currently be served. The cycles to live of those requests is reduced and the bundle is forwarded to the Policy Decision Point (PDP) service. It must be noted that the cycles to live metric is critical, since it affects the maximum complexity of the policy rule that will be used for the resolution of the security decision request.

Invocation: Invocation Originator $\Longrightarrow$ Invocation Form

1. Security Handling (SH) service $\Longrightarrow$ SecurityDecisionRequest.
2. Metadata Handling (MH) service $\Longrightarrow$ SecurityDecisionBundle.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$

- For invocation type 1:
  1-$\rightharpoonup$ Receive SecurityDecisionRequest from SHs
  2-$*$ Extract RefreshAlert from SecurityDecisionRequest
  3-$*$ Extract ResourceAvailability from SecurityDecisionRequest
  4-$*$ Identify ActionIDCyclesToLive (For the received ActionID)
  5-$*$ Identify ActionIDPriority (For the received ActionID)
  6-$*$ Generate ActionRequest_IDRepositoryEntry
  7-$*$ Update ActionRequest_IDRepository (Enter new entry)
  8-$*$ Prioritize ActionRequest_IDRepository (According to 3, 4, 5)
  9-$*$ Generate ActionRequest_IDBundle
  10-$*$ Update (Reduce by one) ActionIDCyclesToLive in ActionRequest_IDRepository (For those included in the ActionRequest_IDBundle)
  11-$\leftharpoondown$ Send ActionRequest_IDBundle to PDPs
  12-$*$ Update ActionRequest_IDRepository (Remove entries with ActionIDCyclesToLive equal to zero)
- For invocation type 2:
  1-$\rightharpoonup$ Receive SecurityDecisionRequest from MHs
  2-$*$ Extract PolicyDecision /s
  3-$*$ Extract ActionRequest_ID /s
  a-$*$ IF(PolicyDecision TRUE)
  i-$*$ Generate SecurityDecision /s
  ii-$\leftharpoondown$ Send SecurityDecision/s to PEPs for enforcement
  iii-$*$ Delete ActionRequest_IDRepositoryEntry /s
  b-$*$ IF(PolicyDecision NotTRUE) OR IF(ActionRequest_IDRepository NotEMPTY)
  i-$*$ Prioritize ActionRequest_IDRepository
  ii-$*$ Generate ActionRequest_IDBundle
  iii-$*$ Update (Reduce by one) ActionIDCyclesToLive in ActionRequest_IDRepository (For those included in the ActionRequest_IDBundle)
  iv-$\leftharpoondown$ Send ActionRequest_IDBundle to PDPs

v-∗ Update ActionRequest_IDRepository (Remove entries with ActionIDCyclesToLive equal to zero)

2.3 Policy Decision Point Service

Description: The Policy Decision Point (PDP) service operates as a repository of the predefined policy rules. Each ActionID is mapped at the mission preparation stage to a set of ActorIDs, SubjectIDs and RequestorIDs in accordance to a corresponding set of semantics (Referring to Services, Information, Networks, Radios, Nodes and Subjects). These mappings constitute the predefined policy rules. Thus, a set of prioritized policy rules of increasing granularity are defined for any given range of the allowed ActionRequest_IDs. Furthermore, an escape rule of least priority is defined for each ActionRequest_ID range, in order to allow the enforcement of security policy decisions under heavily constrained local-node and radio resources. The received ActionIDCyclesToLive indicator defines which of the prioritized rules should be utilized for the given policy reasoning cycle. Accordingly, upon receipt of an ActionRequest_IDBundle, the individual ActionRequest_IDs are separated and bound to the corresponding policy rules (PolicyRule_ID). This is achieved by the individual evaluation of their ActionIDCyclesToLive and rule identification across their predefined rule-sets. The generated PolicyDecisionRequest_Bundle contains these ActionRequest_ID/PolicyRule_ID pairs and the received RefreshAlert indicator.
Invocation: Invocation Originator $\Longrightarrow$ Invocation Form
1. Policy Management (PM) service $\Longrightarrow$ SecurityDecisionRequest.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$
1-$\rightharpoonup$ Receive ActionRequest_IDBundle from PMs
2-∗ Extract RefreshAlert from ActionRequest_IDBundle
3-∗ Extract individual ActionRequest_ID / ActionIDCyclesToLive pairs
4-∗ Identify PolicyRule_ID according to ActionIDCyclesToLive
5-∗ Generate PolicyDecisionRequest /s
6-∗ Generate PolicyDecisionRequest_Bundle
7-$\leftharpoondown$ Send PolicyDecisionRequest_Bundle to MHs

2.4 Metadata Handling Service

Description: A variety of semantic web frameworks can be used for the implementation of the Metadata Handling (MH) service, such as CubicWeb, RDF4J(Sesame), Mulgara, Open Semantic Framework and Jena. The MHs receives a bundle of policy decision requests and updates the local ontology, if required so by the received RefreshAlert. The value of the RefreshAlert originates from the Contextual Monitoring (CM) service (From CMsSnapshot_IDResponse), which bound to an ActionRequest initiation, is used to update the local ontology through the MH_UpdateRequest/ Response process. After this update, the exact functionality order depends on the selected semantic web framework. Yet, the required functionalities are: Structure ontological construct $>$ Invoke reasoner $>$ Query local ontology (According to the received PolicyRule_ID) for policy decision. The result of this process is then matched with the corresponding

ActionRequest_ID and transferred back to the Policy Management (PM) service.
Invocation: Invocation Originator $\Longrightarrow$ Invocation Form
1. Policy Decision Point (PDP) service $\Longrightarrow$ PolicyDecisionRequest_Bundle.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$
1-$\rightharpoonup$ Receive PolicyDecisionRequest_Bundle from PDPs
2-$*$ Extract RefreshAlert
a-$*$ IF RefreshAlert TRUE
i-$\leftharpoondown$ i. Send MH_UpdateRequest to Contextual Monitoring service
ii-$\rightharpoonup$ Receive MH_UpdateResponse from CMs
iii-$*$ Update local ontology
3-$*$ Extract PolicyDecisionRequest/ s from PolicyDecisionRequest_Bundle
4-$*$ Create reasoner
5-$*$ Insert ontological terminology and assertions
a-$*$ For(all PolicyDecisionRequest/ s)
i-$*$ Extract PolicyRule_ID from PolicyDecisionRequest
ii-$*$ Query local ontology according to PolicyRule_ID
iii-$*$ Extract PolicyDecision
iv-$*$ Generate SecurityDecision
6-$*$ Generate SecurityDecisionBundle
7-$\leftharpoondown$ Send SecurityDecisionBundle to Policy Management (PM) service

2.5 Contextual Monitoring Service

Description: The Contextual Monitoring (CM) service is not strictly bound to the security architecture, since it serves multiple other actors and services including the quality of service (QoS) architecture. The functionalities of CMs relate to the maintenance of local awareness over the context under which the tactical nodes operate, including local and remote dynamic information, related to services, information, networks, radios, nodes and subjects. These information are collected locally through other services and by exploiting cross layer functionalities. Furthermore, entries in the CMs can be updated globally utilizing policy administration processes. It must be noted that CMs can also generate aggregated and statistical data for use within policy rules of limited priority. This allows the definition of simplified policy rules of limited computational complexity, for use under constrained network or local resources. For the two invocation cases initiated by the Security Handling (SH) service, the CMs only returns timely values of the corresponding attributes. Yet, for the invocation initiated by the Metadata Handling (MH) service, the exact implementation of this process is system specific and can vary significantly in terms of the syntax, context or both, regarding the information transferred through the MH_UpdateResponse. In this sense, the generated MH_UpdateResponse may refer to a complete and updated policy copy or only the timely values of the dynamic data and object properties (In which case their incorporation occurs at the MHs, during inserting the ontological terminology and assertions Line 5 of MHs: functionalities).
Invocation: Invocation Originator $\Longrightarrow$ Invocation Form
1. Security Handling (SH) service $\Longrightarrow$ AttributeValuesRequest.

2. Security Handling (SH) service $\Longrightarrow$ CMsSnapshot_IDRequest.
3. Metadata Handling (MH) service $\Longrightarrow$ MH_UpdateRequest.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$

- For invocation type 1:
  1-$\rightharpoonup$ Receive AttributeValuesRequest from SHs
  2-$*$ Extract requested attributes
  3-$*$ Extract attribute values
  4-$*$ Generate AttributeValuesResponse
  5-$\leftharpoondown$ Send AttributeValuesResponse to SHs
- For invocation type 2:
  1-$\rightharpoonup$ Receive CMsSnapshot_IDRequest from SHs
  2-$*$ Extract timely value of 'ResourceAvailability' semantic
  3-$*$ Extract timely value of 'RefreshAlert' semantic
  4-$*$ Generate CMsSnapshot_IDResponse
  5-$\leftharpoondown$ Send CMsSnapshot_IDResponse to SHs
- For invocation type 3:
  1-$\rightharpoonup$ Receive MH_UpdateRequest from MHs
  2-$*$ Generate MH_UpdateResponse
  3-$\leftharpoondown$ Send MH_UpdateResponse to MHs

2.6 Policy Enforcement Point Service

Description: The Policy Enforcement Point (PEP) service operates as the output of the core security policy architecture towards the rest of the security or TSI services deployed in the processing pipeline. The role of the PEPs is to identify the service that provides the functionalities required for the enforcement of the policy decision, translate it to a suitable format for enforcement, and communicate it to the initial RequestorID.

Invocation: Invocation Originator $\Longrightarrow$ Invocation Form

1. Security Handling (SH) service $\Longrightarrow$ SecurityDecision.
2. Policy Management (PM) service $\Longrightarrow$ SecurityDecision.

Functionalities: Internal=*, Input=$\rightharpoonup$, Output=$\leftharpoondown$

1-$\rightharpoonup$ Receive SecurityDecision from SHs or PMs
2-$*$ Extract RequestorID from ActionRequestID
3-$*$ Generate ActionResponse
4-$\leftharpoondown$ Send ActionResponse to RequestorID

2.7 Functional Security Services

Additionally to the aforementioned core security architecture components, a variety of functional services can be incorporated in a modular manner through the TSI processing pipeline. These services refer to the enforcement of all the predefined protection goals (e.g. cryptography, management of digital certificates, access control, authentication, credential management, integrity protection, information labelling and filtering, security token management, provenance assurance).

In addition to some non security related services (e.g. packet queue, service registry, message session management), these functional security services are expected to

invoke the extraction of policy decisions. Therefore, these services are assigned a RequestorID, and incorporate the appropriate interfaces towards the Security Handling service and from the Policy Enforcement Point service (Denoted earlier as the singular ActionRequest and ActionResponse interfaces). These services can be defined following standardized processes. Yet, the developed architecture allows the incorporation of national and tailored solutions, satisfying the requirement for modularity towards the security enforcement mechanisms.

3 Test case based validation

As presented earlier, the designed TSI is targeted to the tactical domain. Thus the test cases used for the validation of the designed architecture were developed in accordance to common tactical operations, the experience gained from recent battlefields and the analysis of future requirements. The used tactical operations (e.g. Convoy, Reconnaissance Surveillance and Target Acquisition (RSTA), intervention patrol, medical Evacuation (MEDEVAC), cordon and search, area denial) have been separated to specific use cases (e.g. Blue force tracking, Common Operational Picture (COP) distribution, injection of high mobility nodes, improvised explosive device (IED) detection and report, interoperability with police forces) and detailed episodes (Addressed request/reply, multihop service invocation, service discovery, transitive service delivery, node isolation).

The communication between the defined core security services is achieved using SOAP (Simple Object Access Protocol) messages, allowing the remote procedure call across the services. It is apparent that the service functionalities as presented earlier, correspond mainly to simple message modifications or substitutions. In this case a dedicated process receives a SOAP message (request) that contains all the required parameters, and transforms it into an invocation of the corresponding method. The resulting SOAP message (response) contains the required parameters for the continuation of the policy decision extraction process. Following this model, as presented in figure 3, an ActionRequest (according to its components) is mapped to a SecurityDecisionRequest by the Security Handling service. Consequently, the SecurityDecisionRequest (according to its components) is mapped to an ActionRequest_IDRepositoryEntry by the Policy Management service, while the process continues until the extraction of a valid ActionResponse towards the corresponding functional security service.

According to the results of our experiments, it is important to note that the complexity and dynamic adaptability of the developed mechanism is situated at the structure of the ontological knowledge base, the governing policy rules, the fine grained definition of action requests and the detailed incorporation of the available semantics, as described earlier [6, 5, 7, 8, 3]. Contrary to that, the functionalities of the presented core security services are kept at a low complexity level aiming for clear separation of duties within the policy decision extraction process. Thus, the identification of the appropriate SecurityDecisionRequest by the Security Handling service is a low complexity matching/-querying process, despite of the fine-grained definition of security actions as a conjunction of the security domains (e.g protection, detection, diligence, response) and network capabilities (e.g. NCV-NATO Capability View).

The executed validation experiments highlighted the functionalities of the Metadata Handling service, and more precisely the reasoning phase (See: Metadata Handling Service /Functionalities/5.a.i to iv), as the process with most significant impact in terms of computational complexity within the policy decision extraction process. Aiming to counteract this obstacle and maintain the support of the required network functionalities, under a constrained operational status or across low capacity nodes, a variety of countermeasures have been deployed within the security policy mechanism, which are visible in the functionalities of the presented services.

- The Security Handling service can incorporate precomputed policy decisions, when this has been deemed necessary at the mission preparation stage.
- The Policy Management service utilises resource availability metrics at the prioritization of the ActionRequest_IDRepositoryEntries.
- The Policy Decision Point service connects each ActionRequest_ID to a PolicyRule_ID in accordance to resource availability metrics. Thus, for each reasoning cycle the complexity of the utilised policy rule depends on the locally available computational capacity. Additionally, as presented earlier, a default policy escape rule must be defined (Across the prioritized dedicated rule stack) for each possible ActionRequest_ID for use under highly congested scenarios.
- The Metadata Handling service can incorporate supplementary reasoners (OWL, OWL Mini, OWL Micro) and instances of the local ontological knowledge base, for use under highly congested scenarios.
- Finally, a dedicated policy distribution mechanism has been developed [7], for the purpose of allowing the core security service architecture presented in this article, to be operable across the various platforms deployed within a tactical network.

4 Conclusions

Our research within the security aspects of TACTICS is tripartite. The first completed aspect was to analyse the requirements, validate, and recommend suitable controls and mechanisms for their attainment (e.g. Recommendation of suitable solutions for the enforcement of the identified protection goals through the functional services). Consequently, the development of a suitable security policy framework, able to support and govern the functionality of the aforementioned mechanisms was required, and has been developed as presented earlier. The last major contribution towards a tactical SOA, has been presented in this article and relates to the design of a core security service architecture, able to instantiate the functionalities of the other two elements. The developed architecture provides configuration flexibility in a modular manner, while satisfying the defined requirements dynamically under varying network conditions. Additional SOA benefits include the information flow and performance improvement, maintaining the capacity to integrate existing or tailored assets, with reduced development and management cost. In our future work we intent to utilize our existing experimental results with the experience gained from the recent demonstration of the overall TACTICS TSI, towards the fine-grained adaptation of the developed mechanisms to the realistic conditions of contemporary areas of operations.

Acknowledgments

The results described in this work were obtained as part of the European Defence Agency project TACTICS (Tactical Service Oriented Architecture). The TACTICS project is jointly undertaken by Patria (FI), Thales Communications & Security (FR), Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie FKIE (DE), Thales Deutschland (DE), Leonardo (IT), Thales Italia (IT), Norwegian University of Science and Technology (NO), ITTI (PL), Military Communication Institute (PL), and their partners, supported by the respective national Ministries of Defence under EDA Contract No. B 0980.

References

1. A. Aloisio, M. Autili, A. D'Angelo, A. Viidanoja, J. Leguay, T. Ginzler, T. Lampe, L. Spagnolo, S. D. Wolthusen, A. Flizikowski, and J. Sliwa, "TACTICS: tactical service oriented architecture," *CoRR*, vol. abs/1504.07578, 2015.
2. T. A. Lampe, C. Prasse, A. Diefenbach, T. Ginzler, J. Sliwa, and S. McLaughlin, "TACTICS TSI Architecture," *International Conference on Military Communications and Information Systems ICMCIS*, 2016.
3. V. Gkioulos and S. D. Wolthusen, "Securing Tactical Service Oriented Architectures," *2nd International Conference on Security of Smart cities, Industrial Control System and Communications (SSIC)*, 2016.
4. V. Gkioulos and S. D. Wolthusen, "A Risk Analysis Approach Over Network Centric Warfare and Tactical Service Oriented Architectures," *Submitted for review at: 7th International Conference on Mathematical Methods, Models and Architectures for Computer Networks Security*, 2017.
5. V. Gkioulos and S. D. Wolthusen, "Enabling Dynamic Security Policy Evaluation for Service-Oriented Architectures in Tactical Networks," *Norwegian Information Security Conference 2015 (NISK-2015)*.
6. V. Gkioulos and S. D. Wolthusen, "A Security Policy Infrastructure for Tactical Service Oriented Architectures," *2nd Workshop on the Security of Industrial Control Systems and of Cyber-Physical Systems (CyberICPS 2016), in conjunction with ESORICS 2016*.
7. V. Gkioulos and S. D. Wolthusen, "Constraint Analysis for Security Policy Partitioning Over Tactical Service Oriented Architectures," *Advances in Networking Systems Architectures, Security, and Applications - of Springer's Advances in Intelligent Systems and Computing*, 2016.
8. V. Gkioulos and S. D. Wolthusen, "Reconciliation of Ontologically Defined Security Policies for Tactical Service Oriented Architectures," *International Conference on Future Network Systems and Security-FNSS*, 2016.
9. V. Gkioulos, A. Flizikowski, A. Stachowicz, D. Nogalski, K. Gleba, and J. Sliwa, "Interoperability of Security and Quality of Service Policies Over Tactical SOA," *IEEE Symposium on Computational Intelligence for Security and Defense Applications (IEEE CISDA 2016) - IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2016)*, 2016.