A Security Policy Infrastructure for Tactical Service Oriented Architectures

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. Tactical networks are affected by multiple constraints related to the limited node characteristics and the availability of resources. These constraints within the highly dynamic tactical environment, impose significant limitations to the functionalities and efficiency of current generic security policy frameworks. Earlier studies have provided a risk analysis of tactical service oriented architectures (SOA), and a set of fine-grained protection goals in correspondence to the aforementioned constraints. Furthermore, web ontology language has been identified as a suitable mediator towards the requirements and opportunities imposed by tactical SOA. Thus, in this article we present a security policy framework dedicated to tactical networks, as it has been developed within the project TACTICS.

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

1 Introduction

Tactical networks are of Ad-Hoc nature, subjected to a variety of constraints related both to the limited operational characteristics of the deployed nodes and the scarcity of network resources. Such constraints impede the attainment of requisite protection goals, by rendering current generic solutions unsuitable, due to limited adaptability over the network dynamics. For that purpose, within the project TACTICS (TACTICal Service oriented architecture), suitable security solutions have been developed, tailored to the characteristics of tactical service oriented architectures. Within this scope our study aims to identify and support fine-grained protection goals over the initial over provisioned operational stages, but mainly through the anticipated degraded and disrupted mission execution phases.

Earlier studies [1],[2] presented a detailed risk analysis of tactical SOA, investigating the impact of the aforementioned constraints across the three stages of tactical operations (Preparation-Execution-Debrief). Furthermore, suitable security requirements and protection goals have been identified, referring to the security of communication procedures, transitive information, data at rest and service choreography related processes. Finally, the feasible benefits of exploiting the unique characteristics of service

oriented architectures have been identified, aiming to utilise them for the enhancement of the implemented security mechanisms.

The results of these studies have been consequently utilised for the extraction of functional requirements in respect to the developed security policy mechanisms [3],[4]. These requirements include constraints related to scalability, real time dynamic adaptability, cross layer implementation and distributed deployment. A parallel evaluation between the identified functional policy requirements and the constraints imposed by the nature of tactical SOA, was undertaken for the examination of suitable security policy frameworks. This examination included commonly used mechanisms, such as WS - Security, SAML[5], XACML[6] and Ponder[7], as well as recent semantic (REI[8], KAOS[9], ROWLBAC[10], Kolter et al.[11], Trivellato et al.[12]) and trust management frameworks (cassandra[13], Tulip[14], RT[15], Peer-Trust[16]). This analysis promoted the use of web ontology language (OWL) as the most suitable solution in respect to the requirements of tactical SOA. Thus, the same study presented a tactical policy framework and our initial results regarding its conceptualisation.

In this paper we present a detailed analysis of this security policy framework dedicated to tactical SOA, as it has been designed within TACTICS. Section 2 introduces the developed tactical service infrastructure, focusing on the security related services, their interactions and functionalities. Section 3 presents the core policy model in accordance to the decision process, along with the required steps for the policy formalization. Finally, section 4 includes a simplified example of the prototype implementation developed for validation and demonstration purposes.

2 Tactical Service Infrastructure-TSI

Four distinct instances of tactical nodes have been assumed within TACTICS, each of whom supports the delivery of a defined associated functionality set, through standard interfaces. The studied tactical node types are:

- TSI Node-Dismounted: Carried by individual soldiers.
- TSI Node-Mobile: Integrated in single vehicles.
- TSI Node-HQ: Integrated in semi-permanent headquarters.
- TSI Node-Custom: Unmanned operational node.

The internal TSI components along with a subset of the defined core functionalities are presented at figure 1, while the security related services are highlighted(Yellow). The middle-ware has been divided into two vertical stacks, as it was presented in detail by Thorsten et al. [17] namely:

1. **Processing Pipeline:** It comprise of the following sub-components:
   - *Service Mediator:* Supports functionalities related to session management, message exchange and message adaptation. The defined functionalities include but are not limited to locate remote services, create proxy services, support various message exchange patterns and adjust message priority.
   - *Message Handler:* Supports functionalities related to message forwarding and message transport. The defined functionalities include but are not limited to message format translation, next hop identification, message monitoring and message storage management.

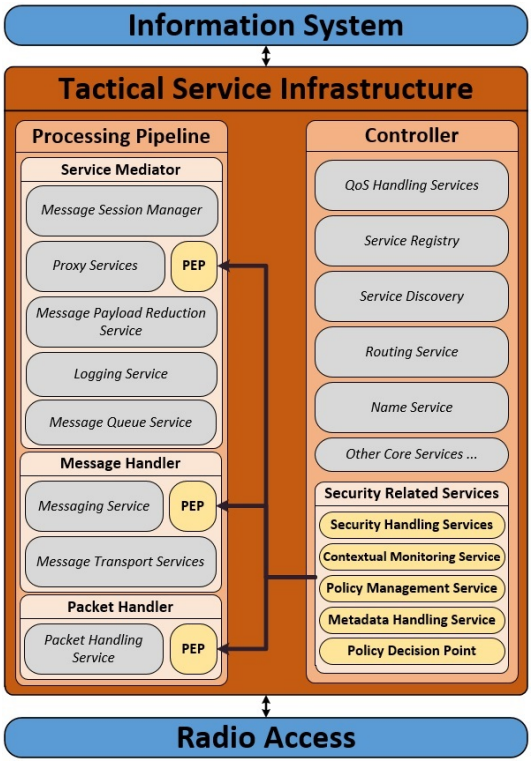

Fig. 1. Defined internal components of TSI nodes.

- *Packet Handler:* Supports functionalities related to packet forwarding and packet scheduling. The defined functionalities include but are not limited to reliability handling, packet queue handling and packet release to radio.

2. **Controller:** It includes core services responsible for the supervision of the aforementioned services, deployed across the processing pipeline layers. The defined functionalities include but are not limited to trigger resource reservation, update service endpoints, select routing protocol and enforce encryption mechanisms.

The aforementioned security services along with the interactions supported by the defined interfaces are presented at figure 2. As described earlier in detail [18], the functionalities of these elements can be summarised as:

- **Security Handling Service-(SH):** A service that monitors network parameters and actors behaviour or requests, where actors can be users, nodes and services. Accordingly it identifies the requirement for a specific action, initiating a corresponding action request. Additionally, SH stores precomputed policy decisions, either from the mission preparation stage or by earlier requests during mission execution, for optimization of resource utilization.

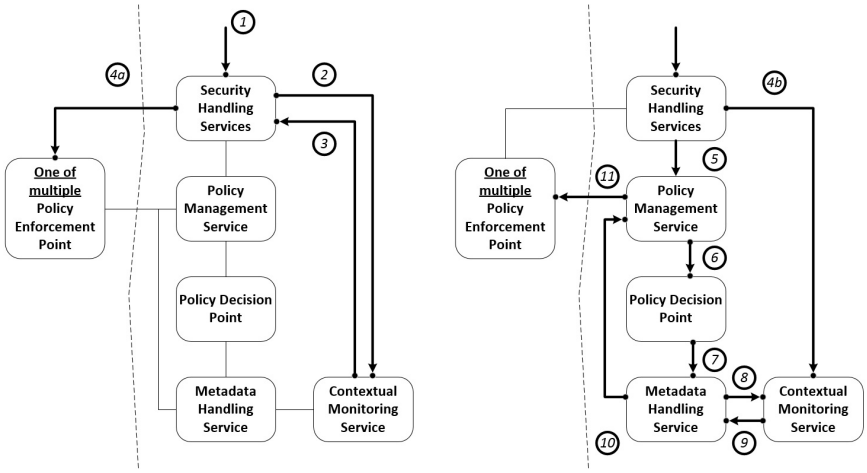

Fig. 2. Interaction of security services within the TSI.

- **Policy Management Service-(PM):** A service that is responsible for the successful resolution of the action request in accordance to the current network parameters and its subsequent transfer for enforcement.
- **Policy Decision Point-(PDP):** It contains the policy rules mapped to the available action requests, in the form of prioritised description logic queries.
- **Metadata Handling Service-(MH):** An ontological knowledge-base that incorporates static and dynamic attributes required for reasoning over the aforementioned policy rules. Reasoning occurs at the MH in accordance to a static copy of the ontological structure at the time of the action request in order to maintain policy consistency.
- **Contextual Monitoring Service-(CM):** A service that monitors timely values of the dynamic attributes utilised across the policy rules, while it computes statistical and aggregated values populating MH upon request.
- **Policy Enforcement Point-(PEP):** A service responsible for the enforcement of the generated or precomputed policy decisions, by use of the locally implemented mechanisms.

While in respect to the functionalities of the implemented interfaces:

- **1:** SH receives a trigger for the initiation of an action request. The trigger can be either external (e.g. Access request by a user, service invocation request by a service, message prioritization request by Quality of Service (Qos) mechanisms) or internal by monitoring the values of the dynamic attributes stored at CM (e.g. node trust levels, node location updates, service choreography statistics).
- **2:** SH requests from CM the current values of the attributes related to the given action request. These values are compared with a predefined range for which the precomputed policy decisions are valid.
- **3:** CM replies with the timely values of the requested dynamic attributes.

– **4a:** If the received attribute values correspond to the predefined ranges, the precomputed policy decision is transferred to the corresponding PEP for enforcement. In this scenario the procedure is successfully terminated at this stage.
– **4b:** If the received attribute values are outside the predefined ranges, SH sends a request to CM for a static copy of the monitored parameters with a unique identifier.
– **5:** SH sends an action solution request to the PM including the unique identifier.
– **6:** PM sends the same bundle (Action Solution Request, Unique Identifier) to the PDP, which retrieves the stored set of prioritised rules corresponding to the given action request.
– **7:** PDP populates the bundle with the first priority rule (Action Solution Request, Unique Identifier, 1st Priority Rule) and transfers it to the MH.
– **8:** MH requests the values of the monitored parameters corresponding to the received Unique Identifier.
– **9:** MH receives the aforementioned values and populates a locally stored copy of the ontological knowledge-base. At this stage, reasoning occurs using this copy and the received 1st Priority Rule.
– **10:** The identified instances are transferred to PM. (Note: If no instances have been identified, steps 6 to 10 are repeated using the complementary prioritised rules)
– **11:** The policy decision is transferred to the PEP for enforcement.

3 Formal Policy Modelling

3.1 Core Policy Model

The formal policy model has been constructed by mapping the aforementioned architectural elements to the required functionalities, as presented at figure 3. The decision process within the formal policy model is:

$$\begin{aligned} Individual\_Domain \cap Individual\_Capability = \{&Individual\_Action(k), \\ &Individual\_Action(k+1), ..., Individual\_Action(k+i)\} \end{aligned} \tag{1}$$

Where:

$$\begin{aligned} Individual\_Action(k) \widehat{=} \{&Individual\_Rule[k(z)], Individual\_Rule[k(z+1)], ..., \\ &Individual\_Rule[k(z+j)]\} \end{aligned} \tag{2}$$

And:

$$Observable\_Objects \xrightarrow{Individual\_Rulek(z)} Governing\_Mechanisms_{Individual\_Action(k)} \tag{3}$$

While the elements constituting the formal policy model have been defined as:

– **Domains:** The tactical policy domains have been identified in accordance to the protection requirements as Planning, Protection, Detection, Diligence and Response. These generic core domains can be extended or refined in order to support fine-grained definition of policy governance.
  – **Individual Domain:** A singular Domain corresponding to the evaluated action.

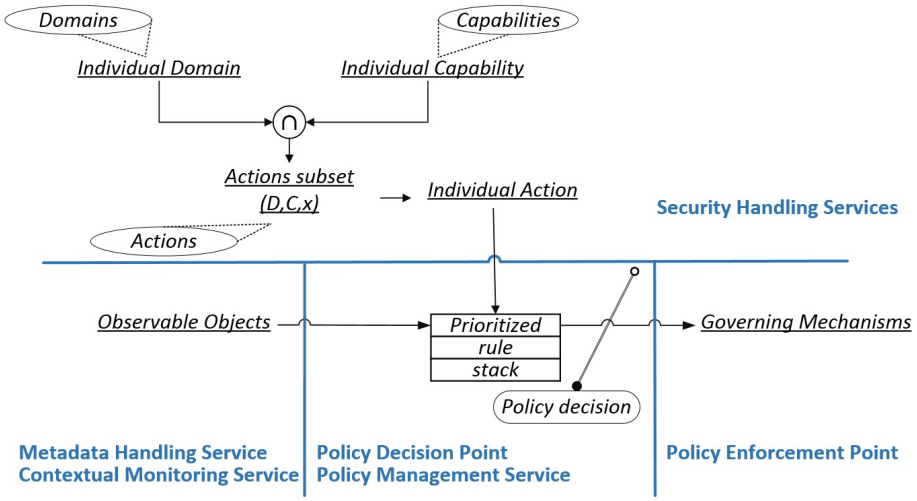

Fig. 3. Visualisation of the decision process within the formal policy model.

– **Capabilities:** TACTICS defined a distinct set of capabilities as part of the developed Tactical Reference Architecture (TRA), in accordance to contemporary operational requirements and the existing NATO Capability View (NAF-NCV-2/7 [19]). The extended list of defined capabilities includes Effects Management, Fire Support, Combat Service Support and Shared Situational Awareness.
  – **Individual Capability:** A singular Capability corresponding to the evaluated action.
– **Actions:** Actions are defined as the intersection of Domains and Capabilities, in the sense of enforcing the Domain requirements upon the operational Capabilities. Thus, defining fine grained policy sub-trees such as Planning/Effect Management, Protection/Shared Situational Awareness or Response/Intrusion Detection.
  – **Actions subset:** A subset of available,suitable and prioritised responses in respect to the defined Actions, by the activation and tailored management of the available Governing Mechanisms. In that sense the Action ”Protection/Message Transmission”, may correspond to an Action subset that includes various cryptographic and credential management services
  – **Individual Action:** A singular policy response across the examined Action subset.
  Note: The definition of these elements allow the Security Handling Services to identify and initiate fine-grained policy decisions, as mapped in a prioritised order to the monitored Observable Objects and actor behaviour or requests.
– **Observable Objects:** Monitored network parameters of static and dynamic nature, as predefined during the mission preparation stage. Observable Objects refer to Service, Information, Network, Radio, Node and Subject attributes, formulating a complete description of the tactical SOA ecosystem upon which policy reasoning is achieved.
  Note: The Metadata Handling Service maintains a static local knowledge accord-

ing to the values of Observable Objects in an ontological knowledge base, while the Contextual Monitoring Service is responsible for the monitoring of dynamic Observable Objects and the calculation of their timely, statistical and aggregated values.

- **Prioritized rule stack:** A set of predefined and prioritized rules dedicated to the governance of each Individual Action. Every rule is constructed as a description logic query for instance identification, with increased granularity as a function of Observable Objects.
  Note: The definition of multiple rules for the governance of each Individual Action allows the on-line adaptation of policy decisions to the dynamic network conditions, in contrast to singular implementations. The communication between the Policy Decision Point and Policy Management Service facilitates the selection of the most suitable governing rule at the decision time, according to predefined prioritizations
- **Governing Mechanisms:** Services deployed within the Policy Decision Point capable of enforcing the policy decision in respect to an examined Individual Action.
  Note: The deployed Governing Mechanisms can be generic or mission specific, related to a variety of security requirements such as authentication, authorisation, cryptography, session management, access control, integrity control, error handling/logging, validation and public key infrastructure.

3.2 Policy Formalization

The formalisation of the core policy model elements within the security TSI services, is based on suitable description logic fragments and executed in six consecutive steps. These steps are in direct mapping to the decision process, as presented in equations 1,2 and 3. Various detailed resources exist in respect to knowledge representation with description logic[20]. Thus, the purpose of this subsection is not to provide an exhaustive reference to this topic, but an insight to the elements crucial for the formalization of the developed security policy model:

- **Equation 1**
  – *Step 1-Definition of Domains:*
  Individual Domains are initially formalised as empty disjoint ontology classes, using terminological box concept definitions. These classes are consequently populated with the defined Actions, formalising extensional knowledge in the form of simple membership assertions, as:

$$hasDomain(AccessDenial, Response) \tag{4}$$

  A closed world assumption must be enforced in order to accommodate the functionality of the Security Handling Services in respect to Action identification. This is achieved in ontology editors by the definition of restricted equivalences for each domain class using a functional data property (e.g. hasDomain). As an example in OWL functional syntax, this is defined as:

  Declaration(Class(:Domains))

Declaration(Class(:Response))
SubClassOf(:Response :Domains)
EquivalentClasses(:Response DataHasValue(:hasDomain "Response"))
Declaration(DataProperty(:hasDomain))
FunctionalDataProperty(:hasDomain)
DataPropertyRange(:hasDomain DataOneOf("Defined Domains"))
Declaration(NamedIndividual(:AccessDenial))
DataPropertyAssertion(:hasDomain :AccessDenial "Response"xsd:string)

– *Step 2-Definition of Capabilities:*
Capabilities are formalised and populated similarly to Domains, as:

Declaration(Class(:Capabilities))
Declaration(Class(:MessageAuthenticityAssurance))
SubClassOf(:MessageAuthenticityAssurance :Capabilities)
EquivalentClasses(:MessageAuthenticityAssurance DataHasValue(:hasCapability "MessageAuthenticityAssurance"))
Declaration(DataProperty(:hasCapability))
FunctionalDataProperty(:hasCapability)
DataPropertyRange(:hasCapability DataOneOf("Defined Capabilities"))
Declaration(NamedIndividual(:DigitalSignatureValidation))
DataPropertyAssertion(:hasCapability :DigitalSignatureValidation "MessageAuthenticityAssurance"xsd:string)

– *Step 3-Definition of Actions and Grouping into Actions subsets:*
Actions are formalised as individuals with the use of unary predicates and categorised into Action subsets with the use of existential quantifications and value restrictions. This is achieved in ontology editors with the definition of data properties of suitable granularity. As mentioned earlier, the Security Handling Service initiates an Action based policy request in accordance to external or internal triggers. An external trigger is directed to a singular Action (e.g. Domain:Protection/ Capability:ServiceAccessControl/ Action:AccessMessagingService), but an internal trigger is based on the dynamic values of predefined Observable Objects leading to the identification and evaluation of multiple actions defined as an Action subset. Thus the Actions forming each Action subset must be prioritised in order to accommodate this functionality, allowing the identification and enforcement of the most suitable policy decision in accordance to the existing resources. Description logic allows the fine-grained definition of Actions. In the previous simplified example, the Action definition is represented in OWL functional syntax as:

Declaration(DataProperty(:hasActionSetID))
Declaration(DataProperty(:hasActionSetPriority))
Declaration(DataProperty(:hasCapability))
Declaration(DataProperty(:hasDomain))
Declaration(DataProperty(:hasGoverningMechanism))

Declaration(DataProperty(:hasRuleSetID))
Declaration(NamedIndividual(:AccessMessagingService))
FunctionalDataProperty(:hasActionSetID)
DataPropertyRange(:hasActionSetID xsd:integer)
FunctionalDataProperty(:hasActionSetPriority)
DataPropertyRange(:hasActionSetPriority xsd:integer)
FunctionalDataProperty(:hasCapability)
DataPropertyRange(:hasCapability DataOneOf("Defined Capabilities"))
FunctionalDataProperty(:hasDomain)
DataPropertyRange(:hasDomain DataOneOf("Defined Domains"))
DataPropertyRange(:hasGoverningMechanism xsd:string)
FunctionalDataProperty(:hasRuleSetID)
DataPropertyRange(:hasRuleSetID xsd:integer)
DataPropertyAssertion(:hasActionSetID :AccessMessagingService
"9632654"xsd:integer)
DataPropertyAssertion(:hasActionSetPriority :AccessMessagingService
"1" xsd:integer)
DataPropertyAssertion(:hasCapability :AccessMessagingService
"ServiceAccessControl" xsd:string)
DataPropertyAssertion(:hasDomain :AccessMessagingService
"Protection" xsd:string)
DataPropertyAssertion(:hasGoverningMechanism :AccessMessagingService
"AuthServ23" xsd:string)
DataPropertyAssertion(:hasRuleSetID :AccessMessagingService
"86514665" xsd:integer)

It must be noted that in terms of ease of implementation and deployment, the same procedure can be used for the definition of Action clusters according to invocation and statistical patterns. Utilising constrained class equivalences and exceptions, Actions of separate Action subsets can be efficiently grouped and mapped into common policy rules, significantly minimising resource consumption under heavily constrained scenarios.

– **Equation 2**
  – *Step 4-Definition of Prioritised rule stack per Action:*

The notable expressive power of description logic fragments originates from the extended set of available constructors, including but not limited to elements of first order logic (e.g intersection, union, complement, universal/ existential restriction) and role oriented (e.g. role union/ chains/ transitivity/ hierarchy). The full extend of available constructors can be exploited at this step for the definition of detailed rules of increased granularity, incorporating both unary and binary predicates in accordance to the security requirements.

Thus, a prioritized rule stack of increasing complexity is defined per Action, facilitating the adaptation of the security policy to dynamic network conditions. The least-priority/least-complexity rule for each Action is defined as a default escape policy expression (i.e. deny-override, permit-override, deny-by-default, permit-by-default) depending on the type of the Action, for use in highly congested tactical

environments and node isolation scenarios. Concurrently, the rules of highest priority can designedly incorporate sets of unary and binary predicates, referring to discrete adaptations of the security policy to the real time network conditions for the given Action.

- **Equation 3**
  – *Step 5-Extraction of Observable Objects and knowledge base construction:*
  Observable Objects correspond to the aforementioned unary and binary predicates referring to service, information, network, radio, node and subject attributes as incorporated within the policy rules. Observable Objects can be defined in ontology editors as object and data properties, enforcing suitable schema constructs (e.g. subPropertyOf, range), relations to other properties (e.g. inverseOf), logical characteristics (e.g. transitive, symmetric) and global cardinality restrictions (e.g. InverseFunctionalProperty, FunctionalProperty). Depending on the granularity requirements of the defined policy rules aggregated and statistical Observable Objects can also be constructed and incorporated, allowing their utilisation across rules of distinct priority levels.
  – *Step 6-Mapping of Individual Actions to Governing Mechanisms:*
  This step is initiated during Step-3 by the definition of suitable DataPropertyAssertions, and finalised by a constrained mapping between actions and suitable Governing Mechanisms for their enforcement. This is achieved by the definition of simple membership assertions, similar to those presented in previous steps.

4 Prototype Implementation

TACTICS has defined sixty requirements with ”MUST” priority, forty with ”SHOULD” and seven with ”COULD”, thirty-four of which are security dedicated as briefly discussed earlier [2][1]. An overall prototype implementation has been realised according to sections 2 and 3, in order to validate the satisfaction of these requirements under the distinct tactical constraints. This implementation was targeted to four common tactical operation types (1-Reconnaissance Surveillance and Target Acquisition, 2-MEDical EVACuation, 3-Convoy mission, 4-Intervention Patrol), separated into a multitude of corresponding episodes (e.g. Sensor data acquisition, Blue force tracking, Mobility management, Improvised Explosive Device detection and report, Ordering and Tasking). Here we present the security policy formalization, in respect to the interface functionalities as presented at sections 2 and 3, for one of the investigated episodes.

4.1 Transitive service invocation

The presented example is part of the transitive service invocation scenarios of the convoy mission use case. Nodes N1 and N2 are mounted on vehicles that belong to a tactical convoy, with N1 being the command vehicle and N3 a hand-held device (TSI Node Dismounted) allocated to a member of N2 personnel. The scenes of the episode are:

1. N1 requires an image from the Area of Operation(AoO) of N2
2. N1 Identifies available services*

3. N1 Identifies local service provider*
4. N1 Transmits corresponding request to N2
5. N2 Transmits corresponding request to N3
6. N3 Evaluates service access request*
7. N3 Invokes service
8. N3 Identifies image compression requirement*
9. N3 Identifies local service provider*
10. N3 Transmits uncompressed image to N2
11. N2 Evaluates service access request * (According to image attributes and N3 credentials)
12. N2 Invokes service
13. N2 Transmits compressed image to N1

The overall execution of a transitive service invocation corresponds to a variety of Actions including interactions between the Information System, TSI, and Radio Access, with load both on the northbound/ southbound interfaces and core service invocations within and across the involved tactical nodes. For clarity these functionalities have been distributed across multiple use cases, while those corresponding to this scenario are marked as "*". Although multiple security policy decisions are involved within a transitive service invocation, this scenario is one of those dedicated to investigating specific aspects of the service choreography functionalities. Thus, actions related to message transmission and queuing, bandwidth allocation or service substitution refer to the invocation of a variety of TSI core services[17], which are not within the scope of this scenario.

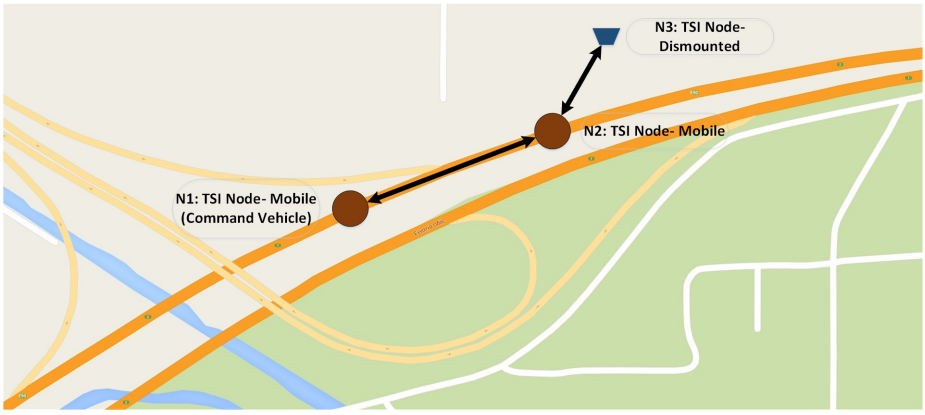

Fig. 4. Visualisation of Transitive service invocation scenario.

The policy formalisation in OWL functional syntax for the presented steps 1-6, can be extracted for this episode as:

- *Step 1-Definition of Domain:*
  Only the Protection Domain is required within the given scenario, defined as presented as section 3.2.
- *Step 2-Definition of Capabilities:*
  The given scenario refers to the Service_Choreography and Situational_Awareness capabilities, defined as presented as section 3.2.
- *Step 3-Definition of Actions and Grouping into Actions subsets:*
  The presented functionalities correspond to four of the Actions within the Action subsets defined by the Protection/Service_Choreography and Protection/Situational_Awareness intersections, namely:
  1. Service_ServiceAvailabilityIdentification
  2. Node_LocalServiceProviderIdentification
  3. Service_ServiceAccessRequestVerification
  4. Information_ImageAttributeIdentification

  which are defined as presented as section 3.2.
- *Step 4-Definition of Prioritised rule stack per action:*
  As described earlier, making use of the extended expressive power of description logic allows the construction of complex security policy rules, validating unary and binary predicates as needed by the specific Action. Using as a simplified example the Node_LocalServiceProviderIdentification Action the Prioritised rule stack in Manchester syntax can have the form:
  1. 1st priority rule:
     Node_SupportsService value ”TacticsImaging”
     Node_hasUser some AllSubjects
     (User_hasTrustLevel value ”High”) and
     ((Node_hasTrustLevel value ”High”) or (Node_hasTrustLevel value ”Medium”))
     Node_hasAoO value ”AoO12341”
     (User_hasRank value ”COL”) or (User_hasRank value ”CPT”)
     Node_hasMissionType value ”Convoy”
     (Node_hasOperationalGroup value ”G2”) and (Node_hasType value ”TSI_ND”)
     Node_hasSupportRadioITUDesignation value ”UHF”
     Node_hasSupportProtocol value ”TLS/SSH”
  2. 2nd priority rule:
     Node_SupportsService value ”TacticsImaging”
     Node_hasUser some AllSubjects
     (User_hasTrustLevel value ”High”) and
     ((Node_hasTrustLevel value ”High”) or (Node_hasTrustLevel value ”Medium”))
     Node_hasAoO value ”AoO12341”
     Node_hasOperationalGroup value ”G2”
     Node_hasSupportRadioITUDesignation value ”UHF”
     Node_hasSupportProtocol value ”TLS/SSH”
  3. 3rd priority rule:
     Node_SupportsService value ”TacticsImaging”
     Node_hasUser some AllSubjects
     (User_hasTrustLevel value ”High”) and ((Node_hasTrustLevel value ”High”)
     or (Node_hasTrustLevel value ”Medium”))

Node_hasAoO value ”AoO12341”
Node_hasSupportProtocol value ”TLS/SSH”
4. 4th priority rule:
Node_SupportsService value ”TacticsImaging”

– *Step 5-Extraction of Observable Objects and knowledge base construction:*
Using the previous rule set as an example the Observable Objects can be extracted as:
1. Data properties (Unary predicates)
User_hasTrustLevel, Node_hasTrustLevel, Node_hasAoO, User_hasRank, Node_hasMissionType, Node_hasOperationalGroup, Node_hasType, Node_hasSupportProtocol
2. Object properties (Binary predicates)
Node_SupportsService, Node_hasUser, Node_hasSupportRadioITUDesignation
The overall extracted Observable Objects incorporated within the security policy knowledge-base are defined as presented as section 3.2 and described earlier [1].

– *Step 6-Mapping of individual Actions to Governing Mechanisms:*
This step depends on the locally implemented services across the nodes deployed for a given tactical operation. Thus, as an example in the given scenario, the Service_ServiceAvailabilityIdentification Action would have as first priority Governing Mechanism the distributed service registry, while the security policy knowledge-base could also serve as a secondary Governing Mechanism for redundancy purposes.

5 Conclusions

In this article we have presented a security policy framework dedicated to tactical SOA, aiming to satisfy the established protection requirements under the constraints of tactical environments. The developed architecture has been presented, focusing on the functionalities of core services and an insight of the defined interfaces. Furthermore, the formal policy model was presented along with the required policy formalisation steps. The prototype implementation has provided a validation of the requirement for an easily deployed, lightweight, cross-layer and dynamically adaptable security infrastructure. Thus, our future plans include the further evaluation with the use of the developed use cases and the preparation of the field-demonstration along with the overall TACTICS architecture.

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. 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.
2. 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.
3. 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)*.
4. 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*, 2015.
5. OASIS, "OASIS Security Services (SAML) TC."
6. C. D. P. K. Ramli, H. R. Nielson, and F. Nielson, "The Logic of XACML," *Science of Computer Programming*, vol. 83, pp. 80–105, Apr. 2014.
7. N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," in *Policies for Distributed Systems and Networks* (M. Sloman, E. Lupu, and J. Lobo, eds.), vol. 1995 of *Lecture Notes in Computer Science*, pp. 18–38, Springer Berlin Heidelberg, 2001.
8. L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, K. Sycara, and G. Denker, "Authorization and privacy for semantic Web services," *Intelligent Systems, IEEE*, vol. 19, pp. 50–56, Jul 2004.
9. A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken, "KAoS Policy Management for Semantic Web Services," *Intelligent Systems, IEEE*, vol. 19, pp. 32–41, July 2004.
10. T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. H. Winsborough, and B. Thuraisingham, "ROWLBAC - Representing Role Based Access Control in OWL," in *Proceedings of the 13th Symposium on Access control Models and Technologies*, (Estes Park, Colorado, USA), ACM Press, June 2008.
11. J. Kolter, R. Schillinger, and G. Pernul, "Building a Distributed Semantic-aware Security Architecture," in *New Approaches for Security, Privacy and Trust in Complex Environments* (H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, eds.), vol. 232 of *IFIP International Federation for Information Processing*, pp. 397–408, Springer US, 2007.
12. D. Trivellato, N. Zannone, M. Glaundrup, J. Skowronek, and P. S. Etalle, "A semantic security framework for systems of systems," *International journal of cooperative information systems*, vol. 22, pp. 1–35, April 2013.
13. M. Becker and P. Sewell, "Cassandra: distributed access control policies with tunable expressiveness," in *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings*, pp. 159–168, June 2004.
14. M. Czenko, J. Doumen, and S. Etalle, "Trust Management in P2P Systems Using Standard TuLiP," in *Trust Management II* (Y. Karabulut, J. Mitchell, P. Herrmann, and C. Jensen, eds.), vol. 263 of *IFIP The International Federation for Information Processing*, pp. 1–16, Springer US, 2008.

15. N. Li, J. Mitchell, and W. Winsborough, "Design of a role-based trust-management framework," in *2002 IEEE Symposium on Security and Privacy, 2002. Proceedings.* , pp. 114–130, 2002.
16. W. Nejdl, D. Olmedilla, and M. Winslett, "PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web," in *Secure Data Management* (W. Jonker and M. Petkovi, eds.), vol. 3178 of *Lecture Notes in Computer Science*, pp. 118–132, Springer Berlin Heidelberg, 2004.
17. 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.
18. V. Gkioulos, A. Flizikowski, A. Stachowicz, D. Nogalski, K. Gleba, and J. Sliwa, "Interoperability of Security and Quality of Service Policies Over Tactical SOA," *Submitted for review at: Military Communication conference-MILCOM*, 2016.
19. NATO, "Nato c3 classification taxonomy." `https://www.act.nato.int/article-8a`, 2012 March.
20. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003.