Situation-Aware Access Control in Federated Data-as-a-Service for Maritime Search And Rescue

Samson Oni*, Zhiyuan Chen*, Adina Crainiceanu†, Karuna Joshi* and Don Needham†
* *University of Maryland Baltimore County, Baltimore, MD*
Email: {soni5, zhchen, kjoshi1}@umbc.edu
†*US Naval Academy, Annapolis, MD*
Email: {adina, needham}@usna.edu

***Abstract*—Maritime Search and Rescue missions involve complex operations in which multiple entities, playing different roles in dynamic situations, benefit from sharing mission-dependent data. We propose an approach to support situation-aware access control in a federated Data-as-a-Service architecture. We develop an ontology and rules to represent access control policies and a distributed reasoning framework to enforce these policies. We implement our proposed solution in a proof-of-concept system.**

***Keywords*-Maritime Search and Rescue, Ontology, Federated Systems, Data-as-a-Service, Semantic Web, Access Control**

I. Introduction

The United States Coast Guard carries out nearly 20,000 Maritime Search and Rescue (SAR) missions per year [1]. Maritime SAR missions often involve close collaboration between multiple entities such as private, military, or government agencies or vessels, and even foreign vessels.

Each entity has data that needs to be kept private, as well as data that needs to be shared to accomplish the SAR mission. Suitable solutions center around situation-aware access control [2], [3] since entities may join the mission at any time and data access decisions may depend on the situation.

Most existing work [2], [3] only considers context-awareness within a single organization, whereas maritime SAR operations typically require collaboration between multiple organizations.

We propose an approach to support situation-aware access control in a federated Data-as-a-Service architecture for maritime search and rescue missions. We have made the following contributions: 1) an ontology and a set of rules to help define access control policies for maritime search and rescue; 2) a distributed reasoning framework that enforces these rules and can be easily integrated with existing systems. Distributed reasoning is needed since some rules reason over data from multiple members. Existing work to support distributed reasoning [4] requires special systems. We propose a query rewriting approach that converts distributed reasoning to federated queries. This allows seamless integration of the access control policies into existing query interfaces such as SPARQL without implementing a specialized system. We implement our solution in a proof-of-concept system using the Apache Jena Fuseki SPARQL server.

II. System Architecture

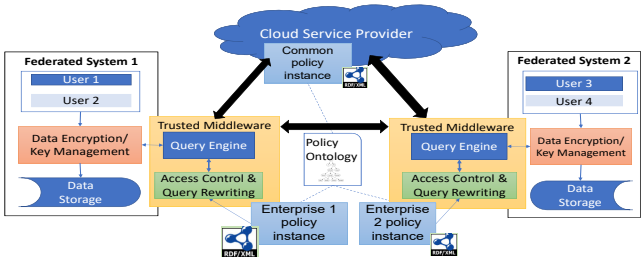

Figure 1. System Architecture

Figure 1 illustrates the overall architecture of our proposed system, which consists of federated members, their own storage infrastructure, and users. Data-as-a-Service is implemented in our architecture as a middleware layer. We use an existing web-based RDF storage and query interface (Apache Jena Fuseki) and use query rewriting to add situation-aware access control. We first develop an ontology to represent access control rules for each member. These rules can be stored at each member or, the rules common to all members can be stored in the cloud to allow every member to have access. Next, we go over the key components in our system.

A. Ontology and Rules

Situation-aware access control policies typically can be represented using the following components [2]:

- U: the set of users.
- R: the set of roles. R has a hierarchical structure.
- $UR \subseteq U \times R$: the assignment of users to roles.
- O: the set of data that can be shared.
- PU, PO: properties of data or users.
- SE: set of situation expressions typically on properties of users or objects (PU, PO).
- P: the set of permissions defined on O.
- $RP \subseteq R \times P$: the assignment of permissions to roles.
- $SEUR \subseteq 2^{SE} \times UR$, the set of situation-aware assignment of roles to users. 2^{SE} is power set of SE.

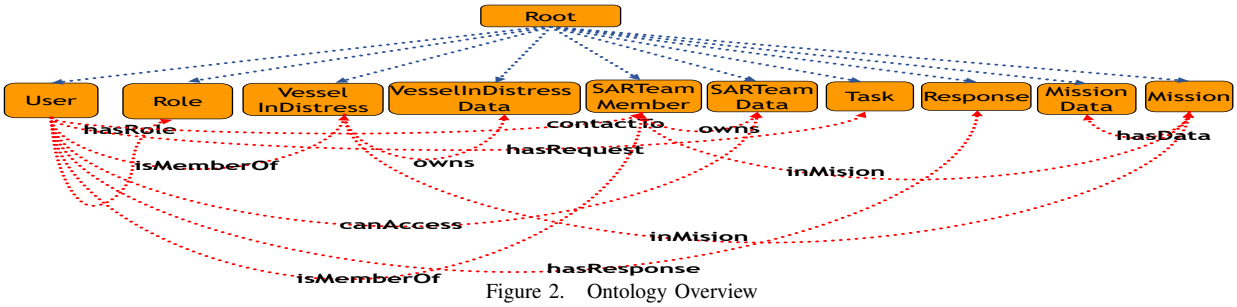

Figure 2. Ontology Overview

- $SERP \subseteq 2^{SE} \times RP$, the set of situation-aware role permission assignments.

We use the W3C Web Ontology Language (OWL) to define an ontology for maritime search and rescue. We examined current regulations [5] as well as existing ontologies for maritime vessels [6]. Figure 2 shows major classes and relationships (red arrows) between classes. The major classes include the vessel in distress and SAR team members, their data to be shared in the system, roles, users, allowed operations (read or write), rescue missions in which they are involved, tasks in the mission, etc.

The rules define the situation-aware assignment of permissions (RP and $SERP$) or roles ($SEUR$). Consider an example of situation-aware assignment of permissions. Suppose the shipmaster of a vessel in distress can get locations of nearby search and rescue ships.

Rule 1: User(?U) $\wedge$ hasRole(?U, DistressShipMaster) $\wedge$ isMemberOf(?U, ?I) $\wedge$ VesselInDistress(?I) $\wedge$ hasLocationPlace(?I, ?x1)$\wedge$ SARTeamMember(?S) $\wedge$ hasLocationPlace(?S, ?x2) $\wedge$ isWithinRangeOf(?x1, ?x2) $\rightarrow$ hasLocationAccess(?U, ?I)

Note that this rule depends on a dynamic situation that the vessel in distress and the other ship are close by (isWithinRangeOf(?x1, ?x2)).

B. Distributed Reasoning

We propose a query rewriting method that supports distributed reasoning, which is needed in federated Data-as-a-Service systems. E.g., suppose "John" is the shipmaster of a vessel in distress "Atlanta" and requests access to locations of rescue ships. This requires data from both the vessel in distress (to verify that John is the shipmaster) and the rescue ships. The following SPARQL query Q_1 will be sent to a rescue ship:

```
PREFIX sar:  <$http://sar.com/0.1/>
SELECT ?location
WHERE {
?S rdf:type sar:SARTeamMember .
?S sar:hasLocationPlace ?location . }
```

We assume that policy rules are not recursive. Algorithm 1 shows a query rewriting algorithm that enforces access control rules. Q is a SPARQL query, RS is the set of access control rules, u is the user issuing the query. We also assume there are m query endpoints. I is the set of predicates that do not appear in data (i.e., they need to be inferred). D_L is the set of predicates contained in local data. D_{Rj} is the set of predicates contained at the j-th remote endpoint and D_R be the union of all remote predicates.

Algorithm 1 : queryRewriting(Q, RS, u)

1: **repeat**
2: **for** $r \in RS$ **do**
3: **for** $p \in r$'s condition and $p \in I$ **do**
4: Find rules $r'_1, r'_2, \ldots, r'_k \in RS$ that has p in consequence
5: Replace p in r with disjunction of conditions of $r'_1, r'_2, \ldots, r'_k$
6: **end for**
7: **end for**
8: **until** No more rules are rewritten
9: Add a predicate p_c to Q to check whether user u has access to query result
10: **for** each rule $r_i \in RS$ that has p_c as consequence **do**
11: Generate a new query Q_i which is the same as Q but replace p_c with query patterns checking condition of r_i
12: For those query patterns checking predicates in D_{Rj}, generate a subquery Q_{ij} to query the j-th query endpoint
13: **end for**
14: Return union of all generated Q_i

The algorithm rewrites the rules such that any predicate in I in the condition part of a rule will be replaced with predicates in $D_L \cup D_R$. This is done similarly to backward chaining, i.e., repeatedly replacing such a predicate p with predicates in the condition part of a rule where p appears in the consequence part (lines 1 to 9). E.g., suppose there is another rule, Rule 2, which defines two locations as withinRangeOf if their Euclidean distance is within a threshold, then $isWithinRangeOf$ in Rule 1 will be replaced by the conditions of Rule 2.

In the second phase of the algorithm, the method rewrites the SPARQL query to check whether the user sending the query has access to the result. In the above example, we add query pattern "sar:John sar:hasLocationAccess ?S ." to the

query to check whether John has access to the location of ?S. This pattern matches the consequence of Rule 1 rewritten by the first phase. So it will be replaced by the conditions in the left hand side of Rule 1. After the first phase, all predicates in left hand side of a rule can be found in $D_R \cup D_U$. Predicates from remote members are sent as a subquery to the remote member's service endpoint. In the above example, query Q_1, the rewritten query, looks like:

```
PREFIX sar:  <http://sar.com/0.1/>
SELECT ?location
WHERE {
SERVICE <http://192.168.56.103:3030/sar
{sar:John rdf:type sar:User  ;
    sar:hasRole sar:DistressShipMaster ;
    sar:isMemberOf sar:Atlanta .
sar:Atlanta rdf:type sar:VesselInDistress ;
    sar:hasLocationPlace ?x1 .}
?S rdf:type sar:SARTeamMember .
?S  sar:hasLocationPlace ?location .
?x1  sar:isWithinRangeOf  ?location .}
```

Predicates have been added to the query to check whether John has access to the rescue ship's location. The subquery inside the SERVICE keyword will be sent to the vessel in distress and the remaining part will be executed locally.

III. Prototype Implementation and Evaluation

We have implemented a proof-of-concept system in which the ontology and rules are developed using Protege. We used Apache Jena Fuseki to store data as RDF triples and provide a web service interface to support SPARQL 1.1. queries over multiple endpoints. Each member of the federated system was implemented as an Apache Jena Fuseki server.

We have validated our proposed ontology with SAR domain experts from the U.S. Coast Guard and U.S. Navy. These experts confirmed that the domain assumptions, classes, properties, and relationships defined by our ontology support the extraction, aggregation, and sharing of SAR information.

We also conducted preliminary experiments to evaluate the performance of our proposed distributed reasoning framework. We ran a sample query asked by shipmaster of a vessel in distress to return medical equipment available on a SAR ship. We simulated both a centralized case when data and rules were stored in one place and a distributed case when data was stored in two different sites (one as SAR ship and the other as vessel in distress). We used our proposed method to rewrite the original query to check for permission. The rewritten query requires data from both sites. We simulated a network delay of 50 ms (typical in a 4G network) and 500 ms (typical in satellite communication) between the two sites. We also varied the number of conditions in the access control rule used in this query from 0 (i.e., no permission check) to 4.

Table I shows the execution time of various cases. The results show that the overhead of checking for permissions (the difference between 0 condition and non zero conditions) is quite small (no more than 20% when there are 4 conditions) and scales linearly with the number of conditions. The execution time in distributed settings is higher than the centralized setting due to network delay but the difference is still acceptable (around 45%-65% for 500 ms delay and around 17%-32% for 50 ms delay).

IV. Conclusion and Future Work

In this paper we propose an approach to support situation-aware access control for a federated Data-as-a-Service architecture for maritime search and rescue.

As future work, we will conduct more extensive experiments examining the performance of our solution. We also plan to develop distributed trust management, as data sharing in SAR depends on trust between members. One possible approach is to dynamically recompute a member's trust level based on its behavior (e.g., if a member's behavior deviates from most peers, its trust level will be reduced).

References

[1] United States Coast Guard Facts, "http://www.uscgboating.org/content/us-coast-guard-facts.php."

[2] S. S. Yau, Y. Yao, and V. Banga, "Situation-aware access control for service-oriented autonomous decentralized systems," in *Autonomous Decentralized Systems*. IEEE, 2005, pp. 17–24.

[3] D. Beimel and M. Peleg, "Using owl and swrl to represent and reason with situation-based access control policies," *Data & Knowledge Engineering*, vol. 70, no. 6, pp. 596–615, 2011.

[4] E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, A. ten Teije, and F. van Harmelen, "Marvin: Distributed reasoning over large-scale semantic web data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 4, pp. 305–316, 2009.

[5] International Maritime Organization (IMO), "International convention on maritime search and rescue," 1979.

[6] S. Brüggemann, K. Bereta, G. Xiao, and M. Koubarakis, "Ontology-based data access for maritime security," in *International Semantic Web Conference*. Springer, 2016, pp. 741–757.

No. Rule Conditions	Exe. Time (sec) Centralized	Exe. Time (sec) Distributed (50 ms delay)	Exe. Time (sec) Distributed (500 ms delay)
0	2.84	3.33	4.15
1	2.87	3.47	4.26
2	2.91	3.76	4.63
3	3.02	3.83	4.67
4	3.04	4.00	4.99

Table I
Execution time by varying number of rule conditions