RDF-X: A Language for Sanitizing RDF Graphs

Jyothsna Rachapalli, Vaibhav Khadilkar, Murat Kantarcioglu, Bhavani Thuraisingham
The University of Texas at Dallas
Richardson, Texas, USA
{jxr061100, vvk072000, muratk, bxt043000}@utdallas.edu

ABSTRACT

With the advent of Semantic Web and Resource Description Framework (RDF), the web is likely to witness an unprecedented wealth of knowledge, resulting from seamless integration of various data sources. Data integration is one of the key features of RDF, however, absence of secure means for managing sensitive RDF data may prevent sharing of critical data altogether or may cause serious damage. Towards this end we present a language for sanitizing RDF graphs, which comprises a set of sanitization operations that transform a graph by concealing the sensitive data. These operations are modeled into a new SPARQL query form known as `SANITIZE`, which can also be leveraged towards fine grained access control and building advanced anonymization features.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features; D.4.6 [**Operating Systems**]: Security and Protection—*Access Control*

Keywords

RDF; SPARQL; Access Control

1. INTRODUCTION

In the present day, data sharing has become a crucial aspect of many domains such as medical research or healthcare, national security, scientific experimental reproducibility, *etc.* Although data sharing is important, maintaining data privacy has become equally crucial. The need to release data yet conceal sensitive information within a dataset has led to the notion of data sanitization [1]. RDF data sanitization is the process of masking sensitive data in an RDF graph with a suitable replacement in order to mitigate the risk of data exposure. RDF sanitization can be useful under two usage scenarios. Firstly, when an RDF dataset needs to be outsourced or shared with a third party, in which case sanitization can be performed on the entire dataset before being shared. Secondly, in an access control like scenario where data is present in original/unmasked form but hidden from those who do not have access to it. In this case, sanitization is performed on the fly, on a per query basis on the subgraph of the RDF dataset that is being accessed by the user query. Further, notice that one of the crucial requirements to perform RDF graph sanitization is the ability to automatically generate the values with which to replace the sensitive data. Additionally, one may be required to perform automatic synchronization, *i.e.*, perform consistent replacement of a data item or an IRI not only in the sensitive triple but also in the rest of the graph. These critical features are however not present in any of the existing SPARQL operations/query forms such as `CONSTRUCT` or `UPDATE`. In this paper we present RDF-X, which is an extension of SPARQL that is specifically designed to perform RDF graph sanitization.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2577367.

2. RDF SANITIZATION OPERATIONS

We introduce a new SPARQL query form called `SANITIZE`, which comprises `SANITIZE` clause, `WHEREs` clause and an optional synchronization clause denoted with keyword `SYNC`. The `SANITIZE` clause specifies the sensitive graph that needs to be sanitized and the `WHEREs` clause specifies the sanitization operation being used and the pattern used to access the sensitive resource. We present four main types of sanitization operations namely Sanitize Node (SNode), Sanitize attributes of a node (Star), Sanitize Edge (SEdge) and Sanitize Path (SPath). The purpose of the SNode operation is to sanitize and protect a sensitive node/resource. This can be accomplished by masking or protecting the identifying attributes (object values) such as name, SSN, ID, *etc.*, of a sensitive resource (subject of a triple). An example SPARQL query Q1 to perform such data sanitization is shown below. Query Q1 sanitizes the object part of the input triple. When Q1 is run on an input graph G_q, only one triple is sanitized and the remaining graph stays the same, *i.e.*, the query transforms the ID value from "007-45678" to "xxx-xxxxx" as shown in Figure 1. If one needs to conceal ID values of all agents in G_q, one may use the variation as shown in Q2, where the sensitive data item is depicted by the graph pattern. Further, if one needs to conceal IRI's of all agents in G_q who worked on secret missions then one may use the variation of SNode shown in Q3. Here we try to hide a node or nodes identified in association with a sensitive resource (secret mission), which itself need not be hidden.

The Star operation is a more sophisticated version of the SNode operation. The intent of the Star operation is to mask a set of identifying attributes of a node or to mask a

node along with its identifying attributes. For *e.g.*, one may want to hide identifying attributes such as SSN, Name, Zip code, *etc.*, of persons of type Agent, as described by query Q4 and illustrated by Figure 2. Further, one may wish to perform synchronization, *i.e.*, not only hide Agent1 in given sensitive triples but also in remaining triples of the graph.

```
Q1: SANITIZE Gq WHEREs {SNode (Agent1 hasID "007-45678")}

Q2: SANITIZE Gq WHEREs {SNode (?s rdf:type Agent . ?s hasID ?o)}

Q3:SANITIZE Gq WHEREs {SNode (?s rdf:type SecMission . ?s ledBy ?o)}

Q4:SANITIZE Gq
    WHEREs {Star (Agent1 hasSSN hasName hasZip)} SYNC {Agent1}
```

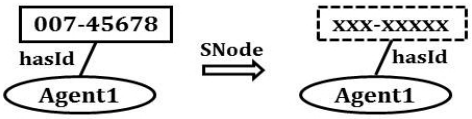

Figure 1: SNode

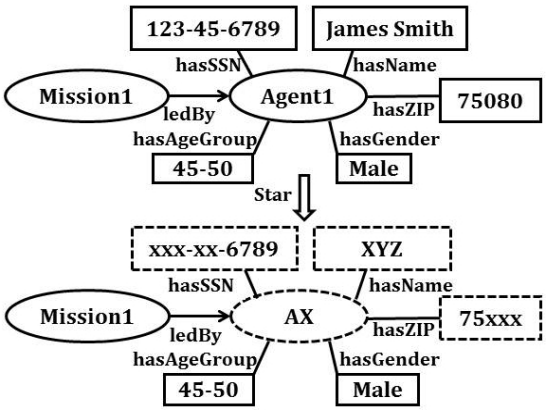

Figure 2: Star

The purpose of SEdge operation is to hide a relationship between two nodes or alternatively to perform edge contraction, *i.e.*, to protect an edge along with its two nodes (where one of the nodes can be a data value). The example query Q5 using SEdge is illustrated with Figure 3. In Q6, edge sanitization is performed on a class of individuals of type Agent. Further, synchronization is performed which consistently hides the sensitive Agent IRI's in the entire graph.

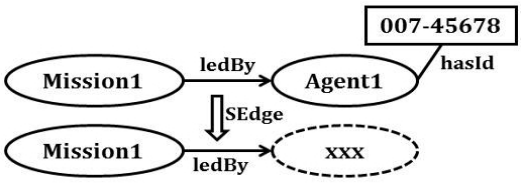

Figure 3: SEdge

```
Q5: SANITIZE Gq
    WHEREs {SEdge (Agent1 hasID 007-45678)} SYNC {Agent1}

Q6: SANITIZE Gq
    WHEREs {SEdge (?s rdf:type Agent . ?s hasID ?o)} SYNC {?s}

Q7: SANITIZE Gq WHEREs {SPath (R2 [used]+ ?o)} SYNC
```

The intent of the SPath operation is to protect a path containing multiple nodes connected by edges. Query Q7 using SPath, as illustrated in Figure 4, hides the derivation history of a report `R2`, which is expressed using the regular expression `(R2 [used]+ ?o)`. The set of triples or subgraph we get by evaluating the path pattern expression `(R2 [used]+ ?o)`, on G_q, comprises two triples: R2 used R1, R1 used Secret-Doc. Unlike SEdge, where we perform edge contraction, in SPath we simply replace each s, p and o value with it's corresponding sanitized value to preserve connectivity and correctness of the RDF graph. The operations SNode, Star, SEdge and SPath provide a range of techniques for sanitizing a variety of sensitive data items and together they form a sanitization language with rich expressivity. One of the major differences between our current work and a preliminary version of our work [2] is in terms of the language used to perform sanitization. The sanitization language presented in [2] was a modification of the SPARQL `SELECT` query form, whereas in this work we present a new query form called `SANITIZE`, which is specifically designed for RDF sanitization and resembles SPARQL `UPDATE` in terms of semantics. Additionally, introduction of Star operation and features such as synchronization, which is taken care of by the `SYNC` clause, make this language much more expressive.

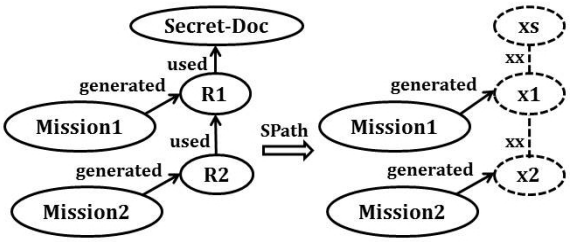

Figure 4: SPath

3. CONCLUSION

RDF-X provides a valuable layer of security for RDF stores with the ability to modify existing RDF data to remove identifiable, distinguishing or sensitive characteristics yet retain usability. This layer of security not only promotes integration of RDF datasets but also makes it practical. We presented a set of sanitization operations that are built as an extension to SPARQL, and as a result, any system with RDF data can build more complex security features.

4. ACKNOWLEDGMENTS

This work was partially supported by National Institutes of Health Grants 1R0-1LM009989 and 1R01HG006844, National Science Foundation (NSF) Grants Career-CNS-0845803, CNS-0964350, CNS-1016343, CNS-1111529, CNS-1228198 and Army Research Office Grant W911NF-12-1-0558.

5. REFERENCES

[1] M. Bishop, A. Singh, J. Cummins, B. Bhumiratana, S. Peisert, and D. Agarwal. Relationships and Data Sanitization: A Study in Scarlet. In *2010 Workshop on New Security Paradigms*. ACM, 2010.

[2] J. Rachapalli, V. Khadilkar, M. Kantarcioglu, and B. M. Thuraisingham. REDACT: A Framework for Sanitizing RDF Data. In *WWW (Companion Volume)*, pages 157–158, 2013.