# Powering Filtration Process of Cyber Security Ecosystem Using Knowledge Graph

Claude Asamoah[1], Lixin Tao[2*], Keke Gai[3], Ning Jiang[4]

*Abstract*—**Cyber Security breaches and attacks are on the ascendancy as corporations, governments, universities, and private individuals are conducting their business and personal transactions on the web. This increasing participating on the web necessitates that robust and efficient cyber security systems need to be put in place by these entities to safeguard their cyber assets. Intelligent Systems needs to be employed to buttress the cyber security protocols established in cloud computing for proper decision-making, which may depend on the effective knowledge representation. However, as one of the dominant industry standards for knowledge representation, Web Ontology Language (OWL) has limitations, such as the lack of support for custom relations. Pace University has extended OWL to support Knowledge Graph as a replacement to better support knowledge representation and decision making. This paper examines using KG as the basis in the design of a knowledge-representation system that drives the filtration process of a company's cyber security ecosystem in cloud computing by employing a use case of cyber security communications in-order to identify the entity relations of threat types for the filtration process.**

*Index Terms*—**Cyber security, knowledge graph, filtration process, cloud computing**

*Claude Asamoah is with Department of Computer Science, Pace University, New York, NY, 10038, USA, ca86749w@pace.edu;*
*K. Gai is with Department of Computer Science, Pace University, New York, NY, 10038, USA, kg71231w@pace.edu;*
*Jiang is with Software School, Henan University, Henan, China, 475001, China, and Department of Computer Science, Pace University, New York, NY, 10038, nj55195w@pace.edu.*
*∗ Dr. Lixin Tao is the corresponding author, ltao@pace.edu.*

## 1. Introduction

Although the semantic Web has tremendous promise of revolutionizing the World Wide Web by enabling machines to discern and make decisions spontaneously on vast information on the internet superhighway, there is a limitation to ontology in that while OWL is the dominant industry standard for knowledge representation, it has some limitations which include the lack of support for custom relations, it's reliance on the single "is-a" relation, and the emulation of other relations via complex object and data properties. Knowledge Graphs, on the other hand, extends Web Ontology Language (OWL) by relating two classes with custom relations. Custom relations present flexibility in relating two classes in an expressive way. This paper is focused on the domain ontology space by using knowledge graph to define the entity relations of the cyber security threat entity hierarchy. Some challenges have to be surmounted before a knowledge graph can be used to facilitate the cyber security threat filtration system based on knowledge-representation. Two of these challenges are the syntax validation of knowledge graph that should be generic enough to validate any knowledge graph and the provision of visual navigation capabilities to any knowledge graph so as to support easy navigation since real-life knowledge graph can easily contain hundreds or thousands of classes with complex inter-relations that can pose a major challenge for the review and validation of knowledge representation.

Knowledge graph is an extension of OWL and is serialized in the RDF/XML format. The capability of creating custom relations in addition to the standardized "is-a" relation extends OWL into a knowledge graph. Syntax validating knowledge graph is crucial because Knowledge graph uses special syntax for easier declaration of custom relations. Also, knowledge graphs could be huge and complex. In addition, knowledge graph correctness is the foundation of any knowledge-based decision-making The RDF/XML serialized knowledge graph cannot be syntax validated using the W3C XML OWL Schema because it supports only owl documents serialized in OWL/XML format. Third parties RDF schemas will not work because it has only "rdf" namespace and will not recognize Pace "rel" namespace. The next challenge is how to make the XSD schema be generic enough that it can validate any knowledge graph with custom relations. To solve these problems, we have designed a custom Pace Knowledge Graph Syntax Validator (PaceKGSV) to validate any inputted knowledge graph before it is parsed by Pace Jena to facilitate visual navigation capabilities to the knowledge graph. We created three functions in Pace Jena to facilitate visual navigation capabilities of knowledge graph: (1) load and return an array list with all unique classes in knowledge graph; (2) load and return all unique relations in knowledge graph; (3 takes a class name and a relation name as arguments and return all instances of the triple namely a class instance as the subject, the custom

978-1-5090-0946-6/16 $31.00 © 2016 IEEE
DOI 10.1109/CSCloud.2016.36

relation as the predicate, and the related class instance as the object.

The main contributions of this paper include:

1) We provide an algorithm that uses knowledge graph in the cyber security threat filtration system.

2) We provide syntax validation for any inputted RDF/XML serialized knowledge graph with custom relations.

The roadmap of this paper is as follows: Section 2 shows the related work of using OWL for cyber security domain ontology. Furthermore, we provide the main concepts of knowledge graph syntax validation, and visual navigation and the descriptions of the proposed function in Section 3. Moreover, major algorithm is given in Section 4. Next, we show the implementation of knowledge graph in cyber security threat filtration system as well as a visual navigation use case in Section 5. Finally, we conclude the Paper in Section 6.

## 2. RELATED WORK

*Cyber Ontology Architecture*

According to Leo Obrst *et al* of MITRE Corporation in their paper "Developing an Ontology of the Cyber Security Domain", cyber ontology architecture can be grouped into three distinct subgroups namely upper, mid-level, and domain ontologies. The upper ontologies are high-level and domain-independent that provides common knowledge bases from which more domain-specific ontologies may be derived while the mid-level ontologies also includes the set of ontologies that represent commonly used concepts, such as Time and Location. The domain ontologies stipulates concepts specific to a domain of interest and represent those concepts and their relationships from a domain specific perspective [18]. Fig. 2 depicts using OWL to define domain specific cyber security threat entities by identifying four main entities which are "ThreatActor", "Threat", "ThreatType", and "ThreatVector" respectively.


Fig. 2 Threat Entities Relation Definitions Using OWL

*B. Knowledge Graph Syntax Extension*

Pace University opted to extend RDF/XML to knowledge graph instead of OWL/XML because OWL/XML serialized documents are not suitable for people to read or write since it scatters information into many small structures. Also, OWL/XML format has no extension for supporting custom relations but rather emulates custom relations with complex object and data properties. In addition, most Knowledge Representation (KR) applications need custom relations. Another reason is that RDF/XML document is more concise and used by most researchers.

Custom relation declaration in a knowledge graph can be broken into two parts namely:

- Definition of custom relation
- Application custom relations to Classes

Fig. 3 depicts an example of custom relation definition and Fig. 4 shows an application custom relations to classes example.

```
// Relations
 -->
<rel:NewRelation rdf:about=http://pace.edu/claude#partOf>
 <rdf:type  rdf:resource="TransitiveRelation">
  <rdf:type  rdf:resource="AsymmetricRelation">
>
```

Fig. 3 "partOf" Custom Relation Definition

```
// Classes     -->
<!-- http://pace.edu/claude#Anatomy -->
<owl:Class rdf:about="http://pace.edu/claude#Finger">
  <rel:partOf rdf:resource="http://pace.edu/Hand"/>
</owl:Class>

<owl:Class rdf:about="http://pace.edu/claude#Hand">
  <rel:partOf
rdf:resource="http://pace.edu/claude#Body"/>
 </owl:Class>
```

Fig. 4 Application Custom Relations To Classes Example.

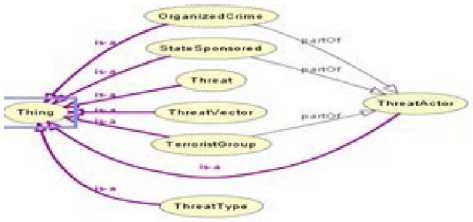


Fig. 5 Cyber Security Threat Entities Relation Definition Using Knowledge Graph

Fig. 5 portrays using knowledge graph to define domain specific cyber security threat entities. Before the knowledge graph can be used in the cyber security filtration system it has to be syntax validated to make sure that it is well formed and OWL syntactically correct.

The following sections will give detailed descriptions about the proposed function as well as main algorithms.

## 3. CONCEPTS, PROBLEMS, PROPOSED FUNCTION

This section introduces the main concepts, problems as well as the proposed function's basic mechanism. We first design the

algorithm for the cyber security threat filtration system. We then tackle the syntax validation and visual navigation challenges of the knowledge graph. Two crucial aspects are covered in this section, namely DOM parse and the generic *Application Program Interface* (API) for custom relations.

A. *Algorithm for Cyber Security filtration System*

Algorithm I Cyber Security Threat Filtration Algorithm

**Require:** Knowledge Graph

**Ensure:** Cyber Security threat Types filtered correctly

**1:** Get incoming traffic into Company IT Infrastructure in chunks into List A

**2:** Send List A to Knowledge Graph for identification

**3: for** ∀ cyber events in List *A* **do**

**4: if** event name found in technical terminology **then**

**5:** mark event as name identified

**6: else**

**7:** check layman terminology if name can be found there

**8: if** found

**9:** mark event as name found

**10: else**

**11:** mark event as not identified and to be sent to behavioral agent for analysis

**12: end if**

**13: end if**

**14: if** event name is found

**15:** identify threat type in knowledge graph

**16: else**

**17:** mark event as not identified and to be sent to behavioral agent for analysis

**18: end if**

**19:** Determine Threat Mitigation type for event from Knowledge graph

**20:** Sent Event to appropriate channel for mitigation and resolution

*B. DOM Parse*

The syntax validation is a function designed in Pace Jena. It is used to validate the knowledge graph before employing other functions designed in Pace Jena to parse the classes and relations of the knowledge graph. Our function presents a visual navigation capability. Furthermore, it is prudent that it must first validate knowledge graph and confirm whether it is well-formed before the Pace Jena parses it. The DomParse.java written as an XMLValidator is used to validate the knowledge graph in conjunction with a designed Pace RDF/XML centric schema. The RDF/XML serialized KG has six distinct namespaces, including "rdf", "rdfs", "owl", "pace", "xml", and "rel". A unique function is needed to be devised for all six namespaces to work together as a single schema because an ".xsd" schema can only have one targeted namespace. The main schema will be the "rdf" namespace called main.xsd and will import the "rel.xsd", "rdfs.xsd", "pace.xsd", "xml.xsd", and "owl.xsd" schemas. Likewise the other schemas will import each other into themselves just as the main.xsd schema did. By this configuration, all the five namespaces persist in the main.xsd schema. Fig 7 depicts the Pace Custom Schema.

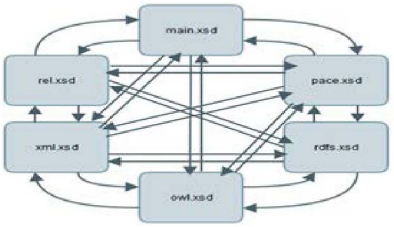


Fig. 7 Custom Pace Schema

C. *Generic Application Program Interface for Custom Relations*

The validation of any knowledge graph will require that its relation elements be declared in the Pace Knowledge Graph schema. To solve this problem, we designed a custom PaceKGSV to validate any inputted knowledge graph before it is parsed by Pace Jena to facilitate visual navigation capabilities for the knowledge graph. Four functions were created in Pace Jena and are: The first function will load and return an array list with all unique classes in the knowledge graph. The second function will load and return all unique relations in the knowledge graph. The third function which takes a class name and a relation name as arguments and return all classes related by the relation. Finally, the forth function will take the knowledge graph as an argument and syntax validate the knowledge graph using the PaceKGSV and DomParse. We have extended the Pace Jena to provide visual navigation capabilities for the knowledge graph that can display all Classes and Relations of knowledge graph and provide navigation in Web and Java application models.

## 4. ALGORITHMS

A. *Knowledge Graph Syntax Validation and Challenges in Syntax Validation*

For the knowledge graph to be fully utilized, the input owl document needs to be validated. This validation is necessary to make sure that before Pace Jena parses the owl document it is determined that it is well formed and syntactically correct.

Next, validating knowledge graph is accomplished by running the ValidateKnowledgeGraph.java created by us. The ValidateKnowledgeGraph.java program takes a knowledge graph document, "owl.xsd" schema

document, and a "rel.xsd" schema document as arguments. It then uses a getRelations() function we created in Pace Jena and retrieves all customs relations in the owl document into an array list and checks the "rel.xsd" schema if any of the custom relations is already declared in the relation schema ("rel.xsd"). If the custom relation is not registered, it updates both the "rel.xsd" and "owl.xsd" schemas with the new relations using appropriate syntax and saves the schema documents. It then calls the DomParse.java program to validate the knowledge graph document. Once the validation concludes and the resultant verdict is acted upon, the "owl.xsd" and "rel.xsd" are reverted to their original state when it was first created. In order to validate the syntax of the knowledge graph with custom relations, the problem of validating elements first needs to be addressed since those elements are used for validation in the same document. Also the elements are supposed to be declared in a XSD schema or DTD that will be used to validate the knowledge graph. To compound the problem, the elements are custom relations serialized in RDF/XML format that is not recognized by the W3C OWL Schema so a custom schema needs to be created that will recognize the custom relations element. Another challenge that has to be overcome is to have a way to check the custom created schema if the elements have been declared and if not they should be declared instantaneously before the knowledge graph syntax is validated. Another problem to be resolved that can come up when declaring the custom relation element is that a situation can manifest when any of the custom relation element retrieved for declaration is already declared in the "owl.xsd" schema resulting in the order of the declaration to be out of order and no longer declared in a "First In First Out" (FIFO), causing the syntax validation to fail. Through research it has been identified that for "owl.xsd", the order of which the custom relations are declared in the knowledge graph matters and must be adhered to when declaring new custom relation elements in "owl.xsd". Consider a Use Case of two custom relations namely "partOf" and "typeOf" in a new knowledge graph named KG_B. The "partOf" custom relation relates two classes "Engine" and "Car" in a triple as [Engine partOf Car]. The "typeOf" custom relation is used to relate two other classes "Cessna", and "Airplane" respectively in the triple as [Cessna typeOf Airplane]. So in the array list of the custom relations named array list1, "partOf" is the first element and "typeOf" is the second element of array list1. A knowledge graph named KG_A was used as the initial knowledge graph to create the "rel.xsd" and "owl.xsd" schemas and syntax validated successfully with two custom relations namely "typeOf" and "includedIn" respectively and are declared in "owl.xsd" in that order. When KG_B is syntax validated, the "partOf" is written to "owl.xsd" and "rel.xsd" because it is a new relation and needs to be declared. The "typeOf" relation is not written to "owl.xsd" or "rel.xsd" because it has already been declared by KG_A. However during the syntax validation, an error will occur because the "typeOf" is encountered first before "partOf" in the "owl.xsd" schema (it is expecting the FIFO order of "partOf" then "typeOf" in the "owl.xsd"). To illustrate further the "owl.xsd" schema will now have the relations declared in this order ("typeOf", "includedIn" and "partOf").To solve this problem, when a new custom relation encountered in a loop is flagged to be declared in "owl.xsd" and "rel.xsd" schemas, it checks the "owl.xsd" schema if the following custom relations has already been declared and if so it gets is position number, decrement that number by one and inserts the current custom relation in that line and thus synchronizing the ordering of the custom relations in the knowledge graph to the ordered declaration in "owl.xsd" which will now be ("partOf", "typeOf", and "includedIn" These problems are resolved by using the Multiple Pass Syntax Validation Algorithm.

A. *Multiple Pass Syntax Validation Algorithm*

This section mainly describes an important algorithm used in our proposed approach, Multiple Pass Syntax Validation (MPSV) Algorithm. This algorithm is with DOM Parsing used by ValidateKnowledgeGraph.java. When an inputted knowledge graph document is sent for knowledge graph syntax validation, the getRelations() function created in Pace Jena is used to retrieve all custom relations into an array list collection. Algorithm II illustrates the pseudo codes of MPSV algorithm.

---

Algorithm II. Multiple Pass Syntax Validation (MPSV) Algorithm

---

**Require:** Knowledge Graph
**Ensure:** Updated Knowledge Graph
**1:** Read Knowledge Graph
**2:** Get all relations into list *A*
**3: for** ∀ relations in List *A* **do**
**4: if** Relation tag is in owl.xsd **then**
**5:** Put in list *B*
**6:** Get line number in owl.xsd and put in List *C*
**7: end if**
**8: if** Relation tag is not in "owl.xsd" **then**
**9:** Put relation tags into list *D* to be declared as new relation
**10: end if**
**11: end for**
**12: while** List *D* is not empty **do**
**13: if** the adjacent relation name to the intended relation to be updated is in List *B*
**14:** Use the corresponding index to get the lineNo from list *C*, decrement lineNo by 1 to be used to update relation in "owl.xsd"
**15: else** locate last element tag in "owl.xsd"

and get the lineNo to be used to update relation in “owl.xsd”

**16: end if**

**17:** Locate the last element tag in “rel.xsd” and get line number to be used to update relation in “rel.xsd”

**18:** Update “rel.xsd” and “owl.xsd” schemas by declaring new relation element using determined lineNos.

**19: end while**

**20:** Call “DomParse.java” with “main.xsd” schema, KG document, -print flag as arguments

**21:** Validate Knowledge Graph against XSD

As depicted in Algorithm II, relation element in the array list is fetched and checked if it is listed in the “owl.xsd”. If it is determined that the relation is not declared and therefore not found in “owl.xsd”, the relation is put in a new relation array list collection. This process is repeated until the first array list is exhausted. A second pass is initiated if new relations are needed to be declared before the knowledge graph is validated. This is determined by testing if the new array list’s size is greater than zero. If the new array size is greater than zero, then in a loop, it checks the “owl.xsd” if the following relation element has been declared. If it is declared, it uses that position number decrement by one, and inserts the current relation element in that position in “owl.xsd” to maintain the order it was declared in the knowledge graph but declare the current relation element at the next available position. If the follow-up relation element is not declared the current relation element is declared in the “rel.xsd” and “owl.xsd” documents. Before the new relations element in the “rel.xsd” document is declared by updating the schema document, the position (line number) to insert and declare the new relation had to be determined and a function of validating knowledge graph for getting token number is used to return the line number that the relation element to be declared is to be inserted. It then updates the “rel.xsd” document using a function to insert the element with appropriate syntax in the “rel.xsd” document.

## 5. USE CASES

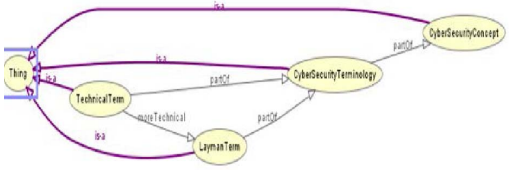


Fig. 8: Example of the Layman Terms, Professional Terms, and Cyber Security Terminology Relations

A. *Use Case 1: Building Cyber Security Threat Filtration System*

The engineer posits that a same cyber threat can have different terminologies and a good filtration system should recognize and reconcile these threat terminologies and pipe the threat to the appropriate channel for resolution. Cyber security has threats, networking concepts, and cyber security terminology. Threats are cyber security concepts that have threat types displayed in Fig.10, and threat mitigations portrayed in Fig. 11. In order to define and create the classes, a class hierarchy is first constructed of the cyber security terminology using a top down approach as shown in Fig.9. The benefit of reconciling these different versions of terminologies is to use the information collected to build a powerful cyber security filtration process based on knowledge representation. Fig. 8 shows the layman terms, professional terms, and cyber security terminology relations. Based on Fig. 10, the main blocks of the threat mitigation process has been identified as *Detection*, *Deterrence*, and *Defense* respectively. These main blocks may have sub divisions with further specializations. Filtered threats via knowledge graph will be sent to these three blocks that specializes in certain types of threat mitigation and threat resolution. The Class relations provided by KG via custom relations will fetch all related entities to a threat type and facilitate the proper classification, filtration, and analysis of cyber security threats and send these threats to the appropriate channel for mitigation or resolution by executing the Cyber Security Threat Filtration Algorithm and the process flow portrayed in Fig. 12.

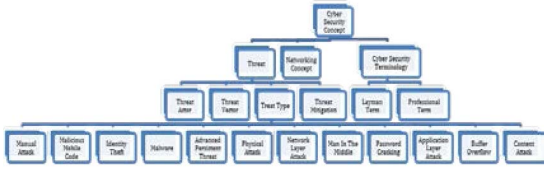

Fig. 9: Threat Type’s Hierarchical Tree

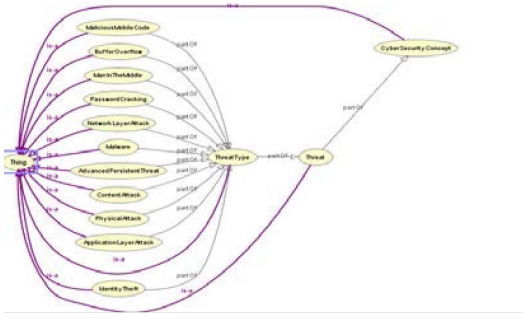

Fig. 10: Threat Types Class Relations

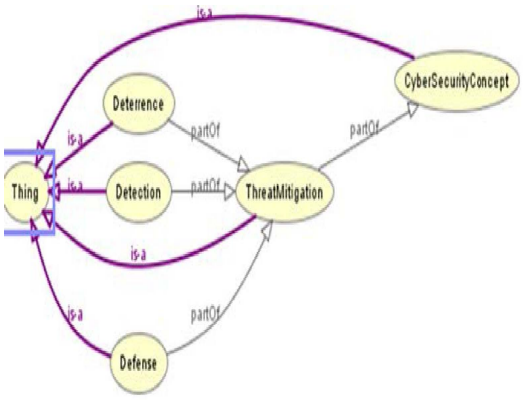


Fig. 11: Threat Mitigations Class Relations

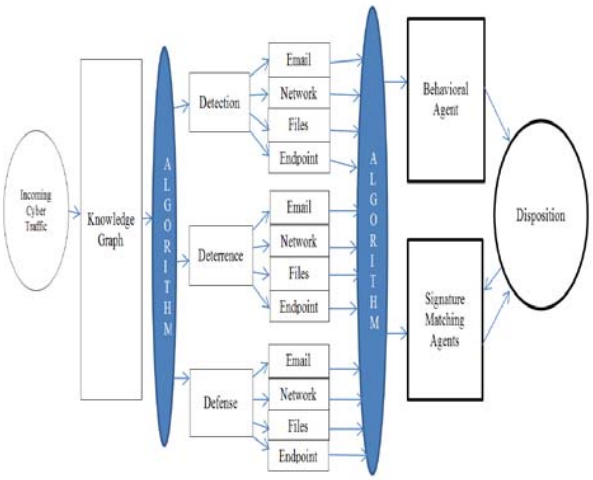


Fig. 12: Cyber Security Threat Filtration System

B. *Use Case 2: Web-Based Knowledge Graph Navigation and Syntax Validation*

This program, WebBrowser.java, takes as an input any well-formed knowledge graph document and uses functions we created in a revised Pace Jena to retrieve all classes and relations in the knowledge graph document and display them in an html page using a java program that creates the class relationships on the fly. This program presents to the user, the ability to navigate the classes and relations of knowledge graph. Fig. 14 portrays the Classes page of the Html Visual Navigation application and Fig. 15 displays all relations of Advanced Persistent Threat. Clicking on the classes' link will take you to that class page with all its linked classes displayed.

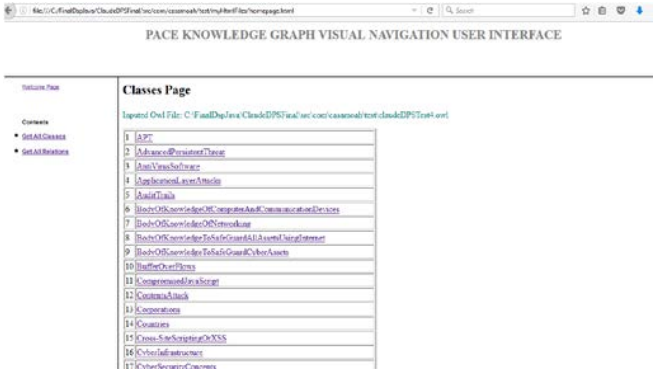


Fig. 13 KG Visual Navigation Classes Page

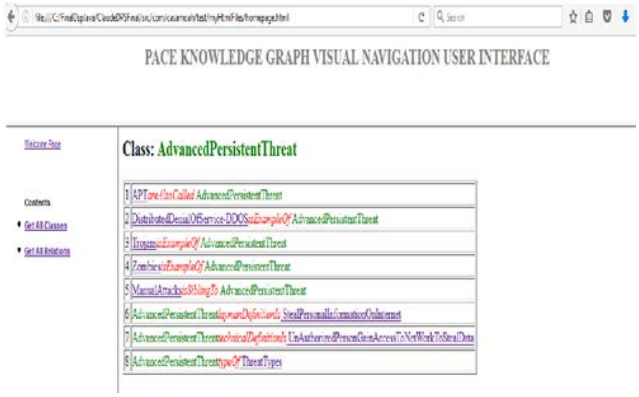


Fig.14: Advance Persistent Threat Relations

## 6. SUMMARY

We evaluated our proposed knowledge graph-based function in multiple application scenarios. The outcomes had proven that the function could reach our expected goal.

CONCLUSION

This paper proposed an approach using knowledge graph for the filtration process of the cyber security ecosystem. We showed the algorithm for classifying and filtering cyber events and sending them to the appropriate channel for mitigation and resolution. We extended ontology to knowledge graph which simplified the relation of classes. We addressed the problem that an "xsd" schema could only have one targeted namespace prompting the design to import the other five namespaces into each other for the five namespace to persist in the "main.xsd" schema that has the "rdfs", "owl", "pace", "xml", and "rel" namespaces imported into it. Our approach accomplished this by designing and implementing the Multiple Pass Syntax Validation algorithm with Dom Parsing to declare any new custom relations in the Pace custom schema

before the knowledge graph is syntax validated. A few use cased had been provided by this paper to prove the executions.

## 7. References


[1] K.Gai, M. Qiu, S. Jayaraman, and L.Tao. Ontology based knowledge representation for secure self-diagnosis in patient-centered telehealth with cloud systems. In The 2nd IEEE Int'l Conf. on Cyber Security and Cloud Computing, pages 98–103, New York, USA, 2015. IEEE.

[2] M. Qiu, M. Zhong, J. Li, K. Gai, and Z. Zong. Phase-change memory optimization for green cloud with genetic algorithm. IEEE Transactions on Computers, 64(12):3528 – 3540, 2015.

[3] K. Gai and S. Li. Towards cloud computing: a literature review on cloud computing and its development trends. In 2012 Fourth Int'l Conf. on Multimedia Information Networking and Security, pages
142–146, Nanjing, China, 2012.

[4] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong. Dynamic energy-aware
cloudlet-based mobile cloud computing model for green computing.
Journal of Network and Computer Applications, 59:46–54, 2016.

[5] L. Tao, S. Golikov, K. Gai, and M. Qiu. A reusable software component for integrated syntax and semantic validation for services computing. In 9th Int'l IEEE Symposium on Service-Oriented System
Engineering, pages 127–132, San Francisco Bay, USA, 2015.

[6] K. Gai, M. Qiu, B. Thuraisingham, and L. Tao. Proactive attribute-based secure data schema for mobile cloud in financial industry. In The IEEE International Symposium on Big Data Security on Cloud;
17th IEEE International Conference on High Performance Computing and Communications, pages 1332–1337, New York, USA, 2015.

[7] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, and H. Zhao. Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry. Future Generation
Computer Systems, PP:1, 2016.

[8] D. Pancho, J. Alonso, O. Cordo´ n, A. Quirin, and L. Magdalena. FINGRAMS: visual representations of fuzzy rule-based inference for expert analysis of comprehensibility. IEEE Transactions on Fuzzy
Systems, 21(6):1133–1149, 2013.

[9] X. Wang, Z. Ma, J. Chen, and X. Meng. f-RIF metamodel-centered fuzzy rule interchange in the semantic web. Knowledge-Based Systems, 70:137–153, 2014.

[10] M. Lytras and R. Garc´ıa. Semantic web applications: A framework for industry and business exploitation-what is needed for the adoption of the semantic web from the market and industry. International Journal of Knowledge and Learning, 4(1):93–108, 2008.

[11] Y. Li, K. Gai, Z. Ming, H. Zhao, and M. Qiu. Intercrossed access control for secure financial services on multimedia big data in cloud systems. ACM Transactions on Multimedia Computing Communications and Applications, PP(99):1, 2016.

[12] A. Altowayan and L.Tao. Simplified approach for representing part-whole relations in OWL-DL ontologies. In The IEEE International Symposium on Big Data Security on Cloud; 17th IEEE International Conference on High Performance Computing and Communications, pages 1399–1405, New York, NY, USA, 2015. IEEE.

[13] A. Formica. Similarity reasoning for the semantic web based on
Fuzzy concept lattices: An informal approach. Information Systems Frontiers, 15(3):511–520, 2013.

[14] M. Minsky. A for representing knowledge. MIT Publishing, 1974.

[15] N. Nilsson. Logic and artificial intelligence. Artificial Intelligence, 47(1-3):31–56, 1991.

[16] K. Gai, M. Qiu, L. Chen, and M. Liu. Electronic health record error prevention approach using ontology in big data. In 17th IEEE International Conference on High Performance Computing and Communications, pages 752–757, New York, USA, 2015.

[17] K. Patel, I. Dube, L. Tao, and N. Jiang. Extending OWL to support custom relations. In IEEE 2nd International Conference on Cyber Security and Cloud Computing, pages 494–499, New York, NY, USA, 2015. IEEE.

[18] L. Obrst, P. Chase, R. Markeloff, Developing an Ontology of the Cyber Security Domain, The MITRE Corporation, McLean, VA, Bedford, MA