ISSN 2278-3091
Volume 9, No.1.3, 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse0391.32020.pdf
https://doi.org/10.30534/ijatcse/2020/0391.32020

Ontology Model for C-Overflow Vulnerabilities Attack

Nurul Haszeli Ahmad[1], Syed Ahmad Aljunid[2], Normaly Kamal Ismail[3], Muthukkaruppan Annamalai[4], Shaiful Bakhtiar bin Rodzman[5]

[1] Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia, haszeli.ahmad@yahoo.com

[2] Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia, aljunid@fskm.uitm.edu.my

[3] Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia, normaly@fskm.uitm.edu.my

[4] Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia, mk@fskm.uitm.edu.my

[5] Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia, shybug_2@yahoo.com

ABSTRACT

The researcher propose the ontology model for C overflow vulnerabilities (COV) Ontology which include the relationship between vulnerabilities and its properties. Many current ontology were developed similar to constructing taxonomies or classifications whereby meaning of relationship were ignored resulting in ineffectiveness in describing the relationship between vulnerabilities and its properties. Eventually, the ineffectiveness affect the tools efficiencies. Studies on current ontology in static analysis of COV also shown that most current ontology focuses on symptoms rather than the root cause of COV occurrences. Therefore the designed of the propose model for C overflow vulnerabilities (COV) will cater this limitation. The Ontology Model consist of sixteen new classes and four new object properties. Based on the evaluation, the new Ontology model could supply and retrieve the right information and consenquently will be reliable to use in the semantic analysis of COV.

Key words: Ontology Model, C overflow vulnerabilities (COV)

1. INTRODUCTION

Program analysis is the key important to understand the computer systems. Studies in the area started in the '70s focusing on debugging, verifying and understanding programs [1]. In the year 2000 onwards, the program analysis capability further extended to security analysis tool according to Viega et al in [2], after the first unintended exploitation on software vulnerabilities according to One in [3].Since then, software vulnerabilities have become a common platform in exploiting computer system along with other Data Security issue such as in [4][5].

Among all vulnerabilities, overflow vulnerabilities (OV) is the most prominent and predicted to continue its existence in the future [6]. It occur in almost all systems that are poorly developed. It is potentially been produced in program languages like Java and PHP. Compared to other programming languages, OV are prominent in C [7]. Due to its behaviour and nature, lack of defensive and preventive mechanism [8]. From this onward, overflow vulnerabilities in C is referred as COV.

There are many ways to inject COV invasion. Morris Worm, as an example, abused vulnerabilities exist in sendmail, fingerd, and rsh/exec command in UNIX platform by overflowing the memory stack [3]. The vulnerability was in C printf() function. When a string longer than the buffer is used, the function replaces to subsequent stack address, allowing an attacker to force the system to run his/her function stored in the following address in the computer system.

There are numbers of C functions e.g. scanf(), gets(), and sprint() that are considered unsafe, which, if mishandled, will become vulnerable. Memory related functions such as free() in [9], mismatch variable conversion and arithmetic operation in [10], null termination in [11], and uninitialized variable in [12], are few examples of overflow vulnerabilities. These COV, if not identified and detected, may cause unnecessary consequences and serious mishaps.

Recently, ontology approaches was brought into software security domain such as by H. Gomes in [13] and specifically for program analysis by Harshal et al in [14]. One of the reason was to improve the semantic-based method by ensuring the methods understand the relation within the code [14]. Ontology approach were used together with classifications or taxonomies as using either ontology alone or only taxonomy is insufficient. The ontology model gives meaning to each classes, hence improve the understanding of taxonomy's user [13]. The use of ontology will help to capture the relationship between the classes in the taxonomy and its characteristics, and provide a readable specifications between the taxonomy and source code in a structure model [14]. This

will enhance the analysis capability such as in [15] especially in a complicated source code or application in [16].

To date, ontology was implemented in education such as by H. Gomes et al in [13], static analysis on web vulnerabilities by Harshal et al in [14], static analysis on Java vulnerabilities by Lian et al [16] and security analysis on requirement by Souag et al in [17]. With regards to static analysis on COV, the initial recorded works utilizing ontology is identified in 2012 by Ellison & Rosu in [18] and further improvised by Hatthorn in [19].

However, regardless of many improvement on analysis method including implementing ontology into semantics static analysis to detect COV, the success rate of detection and preventing COV from occurring is still low. It is either the limitation of the method such as causing overhead, limited COV coverage, flow in-sensitive, require annotation and require extensive vulnerability definition. These current issues has impact on the implementation of the method causing ineffectiveness and inefficiency of analysis.

Even though being a promising approach to improve semantic analysis capability, ontology approach seem to have shortcoming too. Many current ontology were developed similar to constructing taxonomies or classifications whereby meaning of relationship were ignored resulting in ineffectiveness in describing the relationship between vulnerabilities and its properties such as the of Alqahtani et al. in [20]. Eventually, the ineffectiveness affect the tools efficiencies. Studies on current ontology in static analysis of COV also shown that most current ontology focuses on symptoms rather than the root cause of COV occurrences [20]. Therefore the designed of the propose model in this for C overflow vulnerabilities (COV) will cater this limitation.

2. RESEARCH METHOD

2.1 Ontology Framework Design

The following diagram depicted the activities in this phase.

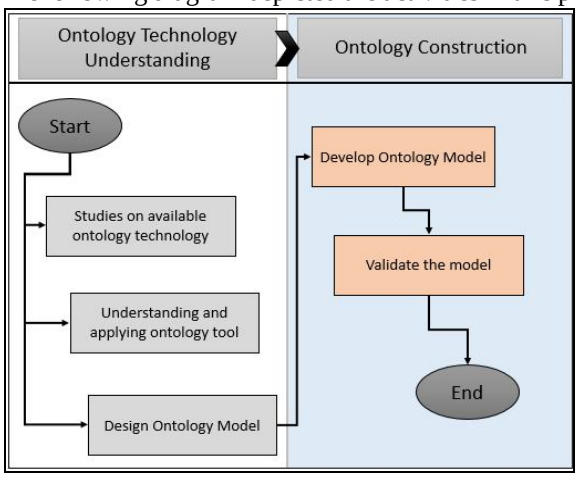


Figure 1: Ontology Framework Design Phase

In this phase, the activities start with studies on various ontology technology including semantics web to understand the technology for ontology development. Upon understanding the technology, the friendliest technology will be used to design the ontology model. Friendliest technology is defined based on the ease of used, available community support and easily accessible by researcher.The activities continue with ontology development based on the design before informal validation is done on the model. Upon the design is completed, subsequent phase is activated without the need to fulfill the model validation activity. The reason is because the development of the tool will help in the refinement and validation process.

2.2 Ontology Construction

At this stage, the pre-processing of ontology indexed was developed. The ontology for domain of C-Overflow Vulnerabilities was created consists of 16 topics. The ontology was classified under the language expressivity and formality and the focus in on software ontology. The focus was chosen as the ontology is used to develop a computer system. According to [21], the Ontology must be designed in intention to meet the purpose and reasons of their development and it involves the sequence of stage before it being evaluated such as, first, planning phase, where the domain of the ontology research area is determined. Through the previous research, the researcher has identified the 10 types C-Overflow that can be the basis of the Classes with additional frequent class that always appeared and related in the domain such as Activity, Function, Vulnerable Criteria, Location, Other Attack and Situation. Each class may have their own unique sub class according to the knowledge of specific topic of classes. The example of date can be seen in table 1

Table 1: The example data of C-Flow Vurnerabilities

Class	Vulnerable Criteria	Function	Activity	Location	Other Attack	Situation
Array Out of Bound	✔		✔			
Unsafe Function	✔	✔			✔	✔
Memory Function	✔	✔		✔		

The second phase is the data collection and analysis, where, the data and suggested class were analyzed according to the previous taxonomy in [22]. The third phase, is to design the ontology according to the step that suggest by [23], as shown in Figure 2 is followed:

:

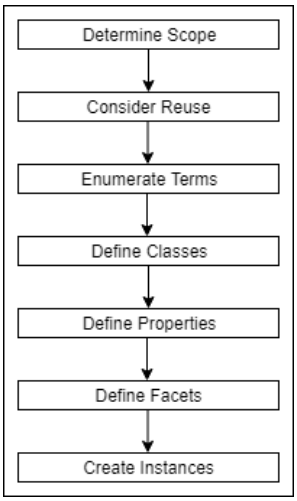


Figure 2: The steps of Ontology design according to N. F., & McGuinness, (2001)

The first step is to determine the scope, which is the C Overflow Vulnerabilities Attack. The Second step is reuse, in this issue, C Overflow Vulnerabilities Attack have no previous Ontology, and however the research have the previous taxonomy in [22], to be used as the guideline. The third step is to enumerate the term that related to C Overflow Vulnerabilities Attack, which are divided to section such as class and subclass similar as in the previous hierarchical taxonomy in [22], where the class of Vulnerable Attack has subclasss such as for Class Unsafe Function like Criteria, Most Attack and Similar Attack. Then, the researchers need to define the properties and facets that suitable for the class. In the last step the researcher then define the individual instances for the class. After the construction model have been done, the evaluation phase can be conducted using Protégé and SPARQL query to test that the ontology may produce the expected information or not. All this steps will be presented in the Result and Analysis Section.

2.3 Result and Analysis

In this section, the researchers will be presented the classes and object properties creation in 2.3.1, the relationships between the classes and object properties in 2.3.2 and the evaluation of of the ontology in section 2.3.3.

2.3.1 The Classes and Object Properties

In this research, 16 new classes such as *Activity, Array Out of Bound, Function, Function Pointer, IntegerRange/Overflow, Location, Memory Function, Null Termination, Other Attack, Pointer Scaling or Mixing, Return intolibc, Situation, Uninitialized Variable, Unsafe Function, Variable Type Conversion,* and *Vulnerable Criteria* have been added which can be seen in the Figure 3.

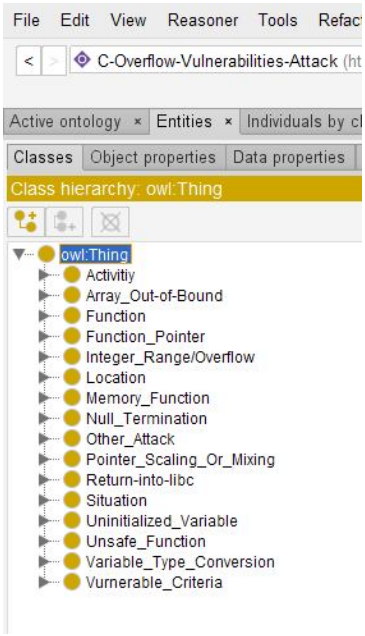


Figure 3: New classes inside the C-Overflow Vulnerabilities Attack

For some individual class may have its own subclasses according to its taxonomy. An example, the Unsafe Function class that contain three subclasses such as Criteria, Most Attack and Similar Attach. Formally, the development of ontology also use the knowledge reference from the previous taxonomy by the researcher in [22].

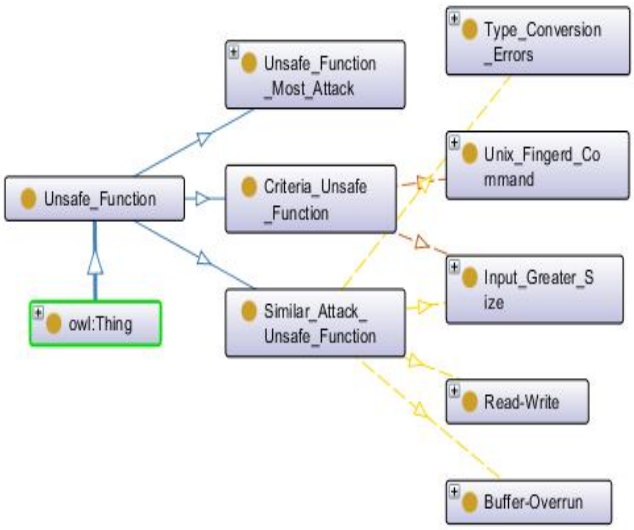


Figure 4: Ontograph of *Unsafe Function* class in C-Overflow Vulnerabilities Attack

The ontology model contains 4 object properties such as: *afffecFunction, hasCriteria, hasPart,* and *hasSituation.*

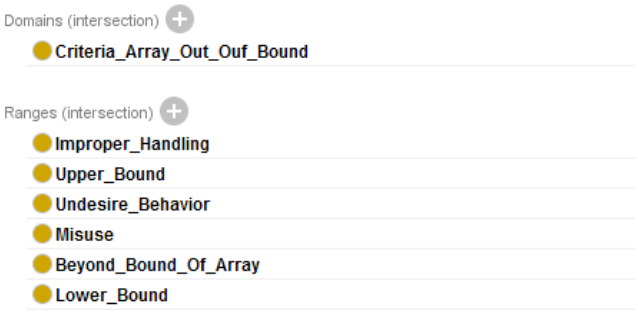


Figure 5: *hasCriteria* object property in C-Overflow Vulnerabilities Attack Ontology

According to Figure 5, the object property of *hasCriteria* include *Criteria_Array_Out_of_Bound* as its domain and *Improper Handling, Upper Bound, Undesire Behavior, Misuse, Beyond Bound of Array* and *Lower Bound* as its range.

2.3.2 Classes and Object Properties Relationship

Technically, all the new classes and object properties must be connected to each other to have a connection which we called as ontology. For example, a criteria of vulnerability of Array out of
Bound.

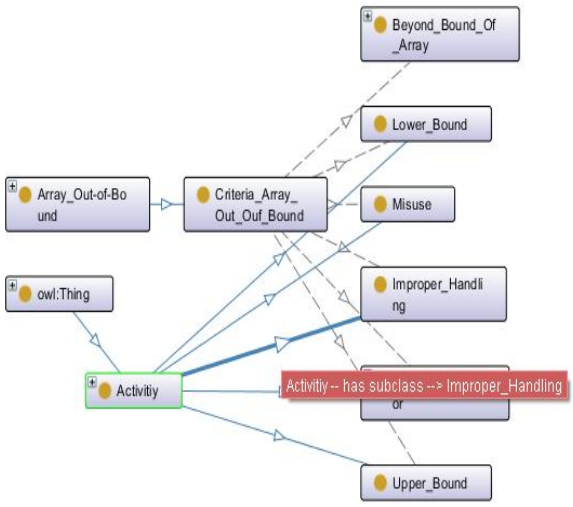


Figure 6. Array Out of Bound and Activity classes are linked through object properties

2.3.2 Evaluation of the Ontology

To evaluate whether the ontology can supply and provide the retrieve the right information or not, Protégé and SPARQL query is utilized to be executed inside the ontology model.

This method has been agreed by Dr. Hazrina binti Sofian from Faculty of Computer Science & Information Technology, Universiti Malaya and also from the literature review such as in Hamiz et al in [24]. The list of SPARQL query for different purposes to evaluate the ontology have been identified as follows:

1. Find by Properties

Table 2: The example of SPARQL query for Find by Properties

Query	Statement
Query 1	SELECT ?dataRange WHERE { ?subClass rdfs:subClassOf ?restriction. ?restriction owl:onProperty mo:hasCriteria;
Query 2	SELECT ?object WHERE { mo:hasCriteria rdfs:range ?object}
Query 3	SELECT ?domain ?range WHERE { mo:hasCriteria rdfs:domain ?domain; rdfs:range ?range .}

2. Find by Domain

Table 3: The example of SPARQL query for Find by Domain

Query	Statement
Query 1	SELECT ?domain ?properties ?range WHERE { mo:Criteria_Array_Out_Ouf_Bound rdfs:subClassOf* ?domain. ?properties rdfs:range ?range. ?properties rdfs:domain ?domain}

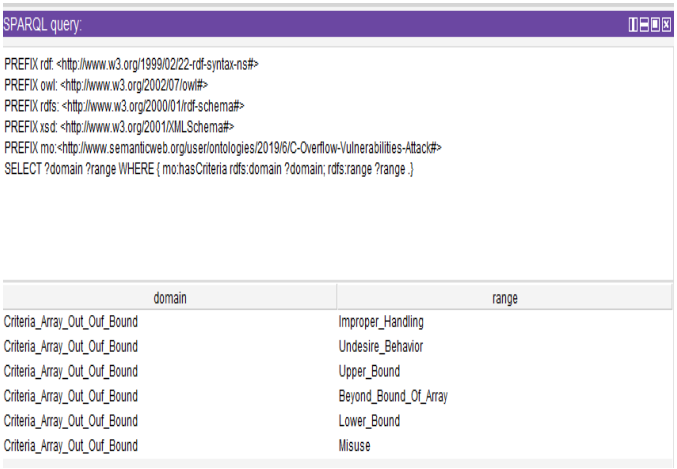
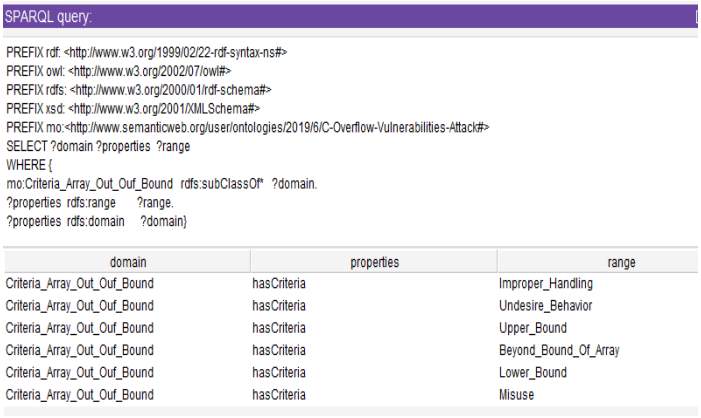


Figure 7: The SPARQL query results of Criteria Unsafe Function.

From Figure 7, Based on the result, the Ontology Model are capable to provide reliable information, which is in this case the Range and Object Property of Domain, Criteria Unsafe Function attack with the specific SPARQL that had been used.



SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX mo:<http://www.semanticweb.org/user/ontologies/2019/6/C-Overflow-Vulnerabilities-Attack#>
SELECT ?domain ?properties  ?range
WHERE {
mo:Criteria_Array_Out_Ouf_Bound  rdfs:subClassOf*  ?domain.
?properties  rdfs:range     ?range.
?properties  rdfs:domain    ?domain}
```

domain	properties	range
Criteria_Array_Out_Ouf_Bound	hasCriteria	Improper_Handling
Criteria_Array_Out_Ouf_Bound	hasCriteria	Undesire_Behavior
Criteria_Array_Out_Ouf_Bound	hasCriteria	Upper_Bound
Criteria_Array_Out_Ouf_Bound	hasCriteria	Beyond_Bound_Of_Array
Criteria_Array_Out_Ouf_Bound	hasCriteria	Lower_Bound
Criteria_Array_Out_Ouf_Bound	hasCriteria	Misuse

Figure 8: The SPARQL query results of Bound Attack.

From Figure 8 above, the same reliable results have been produced, this time the ontology model were successful in retrieving the Range and Object Property of Domain, Criteria Array out of Bound attack, such as Improper Handling, Undesire Behavior, Upper Bound, Beyond Bound Of Array, Lower Bound and Misusse. Consequently, shown to us the classes and Object Properties that have been added on the C-Overflow Vulnerabilities Attack Ontology Model have its own purpose and can be utilized in further research and analysis due to it can provide the correct and right information.

3. CONCLUSION

This article presents the C-Overflow Vulnerabilities Attack Ontology Model was created consists of 16 topics. The ontology was classified under the language expressivity and formality and the focus in on software ontology. Various step has been done in the Construction of this Ontology Model such as the data collection, the analysis according to the previous taxonomy in [22] and the agreement from the expert. Through the previous research, the researcher has identified the 10 types C-Overflow that can be the basis of the Classes with additional frequent class that always appeared and related in the domain such as Activity, Function, Vulnerable Criteria, Location, Other Attack and Situation. Each class may have their own unique sub class according to the knowledge of specific topic of classes. Furthermore, four object properties such as; *afffecFunction*, *hasCriteria*, *hasPart*, and *hasSituation* also have been added to link and provide the association among the classes. Based on the evaluation using the SPARQL queries this classes and object properties in C-Overflow Vulnerabilities Attack ontology model may provide the correct and reliable information for the further use. For the enhancements, the individuals and example of the code must be put inside the ontology. In overall, this constructed ontology model may reliable to use and represent the knowledge and information of C-Overflow Vulnerabilities attack especially in it classification. It might be valuable and useful for the further analysis of C-Overflow Vulnerabilities attack.

ACKNOWLEDGEMENT

This research is funded by the Ministry of Education (MOE) Malaysia under FRGS Research Grant at Universiti Teknologi MARA, Shah Alam (600-IRMI/FRGS 5/3 (021/2017)).

REFERENCES

1. P. Cousot, and R. Cousot, **Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints**, in *Proc. of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of programming languages (POPL), Los Angeles, California*, 1977. https://doi.org/10.1145/512950.512973
2. J. Viega, J. T. Bloch, Y. Kohno and G. McGraw, **ITS4: a static vulnerability scanner for C and C++ code**, in *16th Annual Conf. of Computer Security Applications (ACSAC), New Orleans, LA, USA*, 2000.
3. One, A. **Smashing the Stack for Fun and Profit,** in *Phrack Magazine*, vol. 7, no. 49, 1996.
4. M. T. Basu and J.K.R. Sastry. **Enhancing Data Security under Multi-Tenancy within Open Stack**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 1, pp. 533-544, January – February 2020. https://doi.org/10.30534/ijatcse/2020/73912020
5. K. Ahmad, Al Hwaitat, M. H. Qasem and R. A. Fabozzi. **Security of Data Access in Fog Computing using Location-based Authentication**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 1, pp. 247-254, January – February 2020. https://doi.org/10.30534/ijatcse/2020/37912020
6. L. Constantin, **Attackers can turn Microsoft's exploit defense tool EMET against itself**, in *InfoWorld*. Retrieved from http://www.infoworld.com/article/3036850/security/attackers-can-turn-microsofts-exploit-defense-tool-emet-against-itself.html, 24 February 2016.
7. D. Chechik, **Prevalent Exploit Kits Updated with a New Java Exploit**, in *M86 Security Labs*. Retrieved from http://labs.m86security.com/tag/java/, 16 December 2011.
8. Oracle Corporation. **Java SE Security**, in *ORACLE*, Retrieved from http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html, 2012.

9. P. Akritidis, C. Cadar, C. Raiciu, M. Costa and M. Castro, **Preventing Memory Error Exploits with WIT,** in *Proc, of the 2008 IEEE Symposium on Security and Privacy (18 May – 21 May 2008), Washington, DC, USA*, 2008. https://doi.org/10.1109/SP.2008.30
10. D. Pozza, and R. Sisto, **A Lightweight Security Analyzer Inside GCC**, in *3rd International Conf. on Availability, Reliability and Security (ARES '08), Barcelona, Spain*, 2008. https://doi.org/10.1109/ARES.2008.26
11. L. A. Grenier, **Practical Code Auditing**, 2002.
12. J. J. Tevis, and J. A. Hamilton (2004). **Methods for the prevention, detection and removal of software security vulnerabilities**, in *Proc. of the 42nd annual South East regional conf. (2nd - 3rd Aprill 2004), Huntsville, Alabama, USA*, 2004.
13. H. Gomes, A. Zúquete and G. P. Dias, G, **An Overview of Security Ontologies**, 2015.
14. H. A. Karande, P. A. Kulkarni, S. S. Gupta, D. Gupta. **Security against Web Application Attacks Using Ontology Based Intrusion Detection System**, *International Research Journal of Engineering and Technology (IRJET),* vol. 3, no. 1, pp. 89-92, 2016.
15. E. Atilla, **Standard Ontology of Security of Information and Networks?** in *7th International Conf. on Security of Information and Networks (SIN'14), Glasgow, UK*, 2014.
16. Y. Lian, W. Shi-Zhong, G. Tao, et al. **Ontology Model-Based Static Analysis of Security Vulnerabilities**, *ICICS - Lecture Notes of Computer Sciences*, vol. 7043, pp. 330 – 344, 2011. https://doi.org/10.1007/978-3-642-25243-3_27
17. A. Souag, C. Salinesi, I. Wattiau, and H. Mouratidis, (2013). **Using Security and Domain Ontologies for Security Requirements Analysis**, in *37th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, Kyoto, Japan, 2013.
18. C. M. Ellison, and G. Rosu, **Defining the undefinedness of C**, 2012.
19. C. Hathhorn, C. Ellison and G.Roşu, **Defining the undefinedness of C**, in *ACM SIGPLAN Notices*, 2015. https://doi.org/10.1145/2737924.2737979
20. S. S. Alqahtani, E. E Eghan, and J. Rilling, **Tracing known security vulnerabilities in software repositories–A Semantic Web enabled modeling approach**, *Science of Computer Programming*, vol. 121, pp. 153-175, 2016. https://doi.org/10.1016/j.scico.2016.01.005
21. A. S. A. Latiff, H. Haryani and M. Annamalai, **Software Engineering Approach for Domain Ontology Development: A Case Study of Islamic Banking Product**, Journal *of Information Retrieval and Knowledge Management*, vol. 3, pp. 36-53. 2017.
22. N. H. Ahmad, S. A. Aljunid and J. A. Manan. **Taxonomy of C Overflow Vulnerabilities Attack**, in *Conf. Communications in Computer and Information Science*, June 2011. https://doi.org/10.1007/978-3-642-22191-0_33
23. N. F. Noy, and D. L. McGuinness, **Ontology Development 101: A Guide to Creating Your First Ontology**, Stanford Knowledge Systems Laboratory, vol. 25, 2001.
24. M. Hamiz, H. Haron, M. Bakri, N. L. M Lazim. **Ontology model for intake suggestion and preparation for Malay confinement dietary recipes**, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 1,, pp. 481-488, January 2020. https://doi.org/10.11591/ijeecs.v17.i1.pp481-488