International Journal of Innovative Computing, Information and Control
Volume **11**, Number **1**, February **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 309–325

ONTOLOGY-BASED BOTNET TOPOLOGY DISCOVERY APPROACH WITH IP FLOW DATA

Ci-Bin Jiang[1,2] and Jung-Shian Li[1,2]

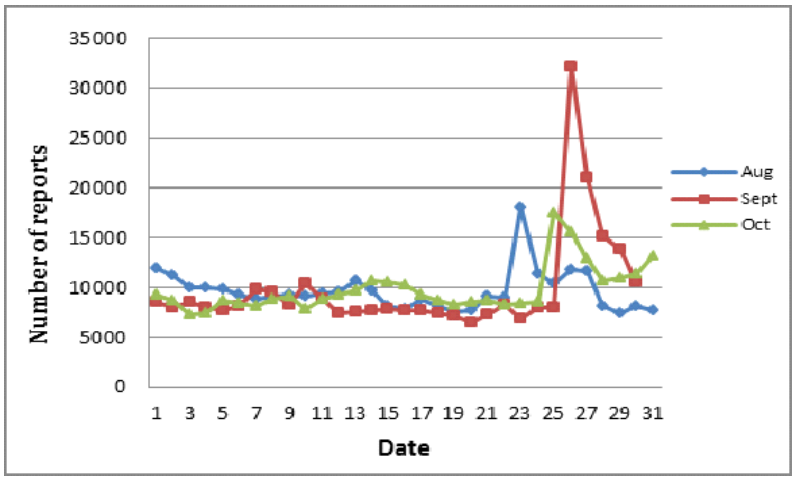
[1]Department of Electrical Engineering
[2]Institute of Computer and Communication Engineering
National Cheng Kung University
No. 1, University Road, Tainan 70101, Taiwan
{ q38981276; jsli }@mail.ncku.edu.tw

Received March 2014; revised July 2014

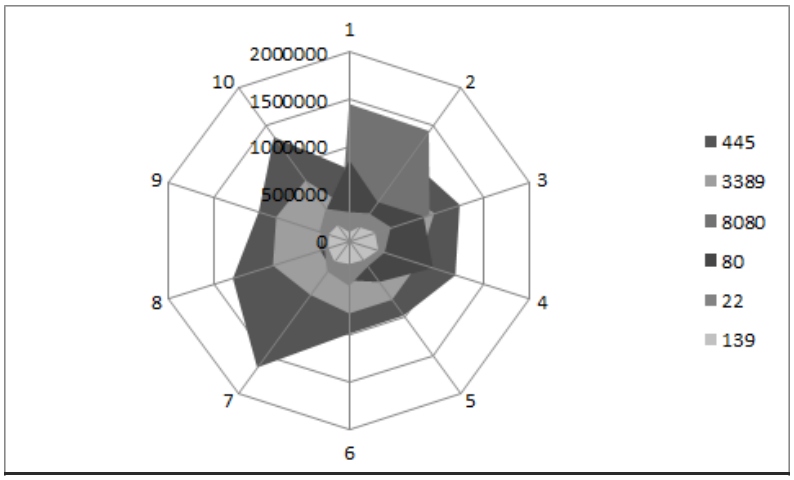
Abstract. *Botnet activity continues to grow at an alarming rate and poses a major threat to the security of networked systems around the world. Botnet malfeasance is quite devastating, such as credit card stealing or DDoS. So it is important to understand the botnet behavior, topology and structure. If botnet communication can be tracked, the C&C server can be identified and infection routes detected, allowing for takedown of botnets. Hence, we propose a new ontology and a set of inference rules to facilitate the automatic identification of the botnet topology by means of a machine learning algorithm. The validity of the proposed approach is demonstrated utilizing blacklisted IP flow data collected over three plus months. The inference time and system convergence performance obtained when using the proposed ontology and inference rules are systematically examined. Overall, the results presented in this paper indicate that the proposed methodology provides a viable means of determining botnet topology with low inference time and high degree of accuracy compared to previous research works, thereby enabling appropriate security measures to be put in place.*
Keywords: Botnet, Topology, Inference rules, IP flow data

1. **Introduction.** In a botnet, hundreds or even thousands of compromised computers are controlled by a malicious user (referred to as a botnet herder) for nefarious purposes such as stealing personal information, propagating spam, network attacks. Botnet activity has risen alarmingly in recent years [1] and is not confined to simple geographic boundaries, but is endemic around the world [2]. As a result, botnets are now recognized as major threats to the security of modern day networked systems. The data logs from *Dshield.org* [36] can be used to understand the patterns in detail. Figure 1 shows insights from the *Dshield,org* dataset [36]. We analyzed over 3 months of *Dshield.org* data and found that a peak appears at the end of each month, as depicted in Figure 1(a). Figure 1(b) offers visualization for port analysis. The most often attacked port collected by *Dshield.org* is port 445. Figure 1(c) presents temporal behavior of botnets collected by *Dshield.org*. Attack interval time is defined as the average time between attacks on the target IP address. The dataset exhibits an average attack interval between 2 and 6 minutes, as shown in Figure 1(c).

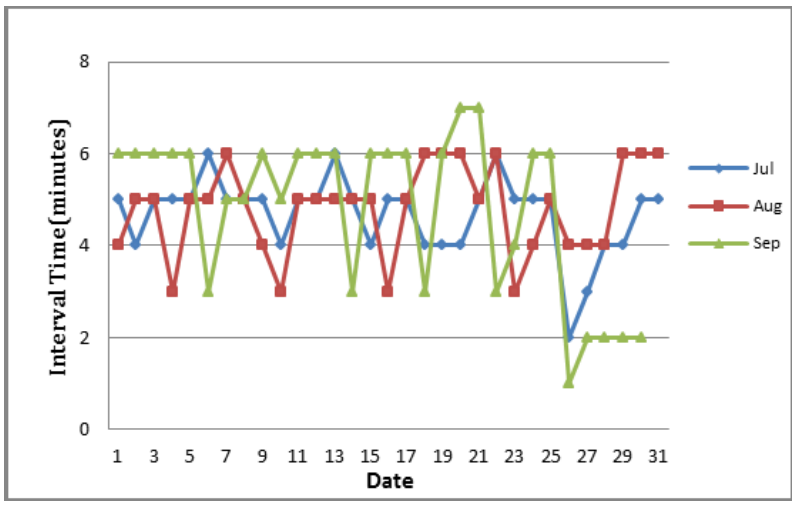
To safeguard the security of computer networks, it is necessary to react rapidly and efficiently to any perceived attack. Accordingly, many methods based on inspection and analysis of IP flow data have been proposed for intrusion detection [5,7-9,14] and various methods have been presented for analyzing communication behavior between bots [10-12,14,15]. Understanding botnet topology and structure is important. If we can track their communication, find the C&C server and know the infection vectors, we can shut them down. Hence, we propose a new ontology and a set of inference rules to facilitate automatic



(a) A sample of malicious activity



(b) Port report analysis



(c) Target IP interval time

FIGURE 1. Insights from *Dshield.org*

identification of botnet topology using machine learning on IP flow data (containing packet header information) collected over three plus months. The main contributions of this work are listed as follows:

- This paper proposes a methodology to facilitate the identification of the botnet topology by analyses of the IP flow data using a machine learning (ML) algorithm. The methodology combines knowledge-based and behavior-based models (signature-based and anomaly detection methods).
- The inference time and system convergence performance obtained when using the proposed ontology and inference rules are systematically examined in this work.

We design a set of inference rules and a bot behavior modelling mechanism based on IP flow data using three different reasoners (Jena, Pellet and Protégé) and compare the rule evaluation of the reasoners to show the superiority of the proposed scheme.

Traditional rule-based correlation engines [44] did not identify inherent semantic relationships and did not try to explain findings by examining correlations. Compared to existing work, our method explores the correlations of different botnet attackers, shown in Section 3, for each IP address to perform reasoning among the rules.

The remainder of this paper is organized as follows. Section 2 reviews background and botnet topology detection mechanisms in the literature. Section 3 describes the inference rules developed in the present study. Section 4 examines the inference time and system convergence properties when using the proposed ontology and rule evaluation. Finally, Section 5 provides some brief conclusions and indicates the intended direction of future research.

2. **Background and Related Work.** Today, many possible cyber-attacks threaten the Internet, such as DDoS (Distributed denial of Service) attacks [16] or personal information thefts by Zeus and Spyeye Tracker [42]. The hacker may join a FFSN (Fast-flux Service Network) to evade tracking. Hence, knowing the routes of infection is very important. Digital fingerprints, such as firewall logs and IDS (Intrusion Detection System) reports [17,18] may generate hundreds or thousands of security alerts, events or IP flow data. It is hard to read and understand the attack techniques or communication relationship in the detailed investigation.

In [10], the author presents a rule-based reasoning approach for analyzing security alerts using semantic web technology. In [21], the authors used a machine learning (ML) technique to analyze network anomaly detections via inspection of IP flow data. In [11,19-23,39,40,43], various ML-based methods were proposed for intrusion detection. Although the results showed high accuracy rate and low detection time, botnet topology determination often fails due to lack of rule-based reasoning. In [41], the authors proposed a method to choose and execute the optimum response using SWRL [24-27]. However, their approach is unsuitable for detecting botnet topology with long inference times in big datasets. The related works discussed above have limitations in rule-based reasoning.

Accordingly, we propose a new ontology method with a set of inference rules to facilitate the automatic identification of the botnet topology by machine learning on a repository of IP flow data. We also provide performance evaluation about inference time and system convergence. We analyze the IP datasets from *Dshield.org* [36], Zeus Tracker [42], and our own datasets [45], which contains data traces for over 3 months. The goal of this paper is to identify botnet topology with low inference time and high degree of accuracy compared to previous works.

3. **Botnet Ontology and Inference Rules.** In the present study, both the botnet ontology and the inference rules are constructed using Semantic Web technology [28]

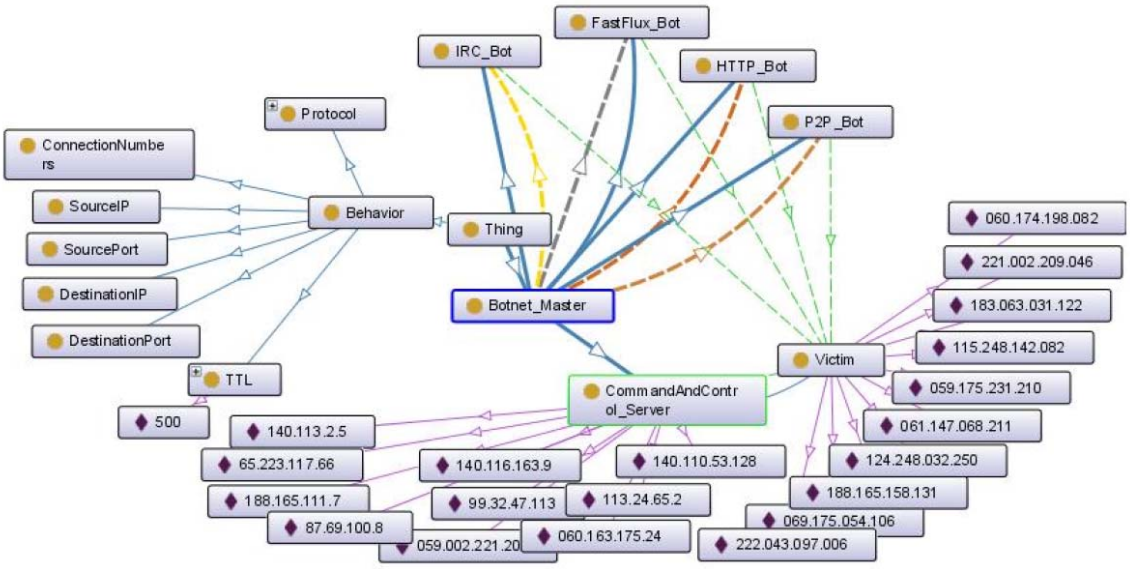


FIGURE 2. Proposed botnet ontology

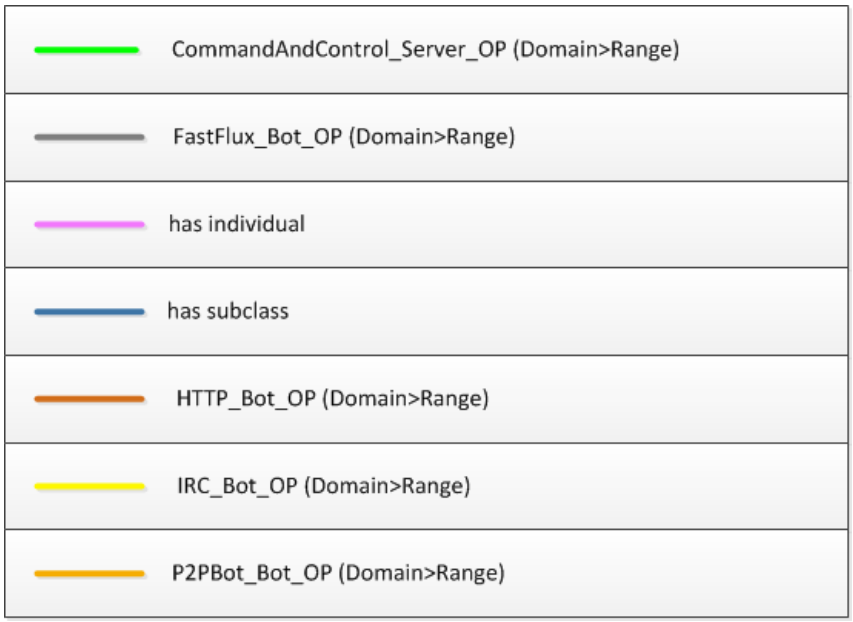


FIGURE 3. Ontology relationships

in order to facilitate the use of an ML technique in interrogating the IP flow data and deriving inferences regarding botnet topology. The proposed ontology is illustrated in Figure 2. Basically, the ontology comprises a set of attributes, multiple instances of each attribute, and a set of statements describing the possible relationships between a source and a destination node in the network (Figure 3). As shown, the input IP flow data is mapped to the seven attributes of DNS_Name, Protocol, Destination Port, Source IP, Destination IP, Source Port and Connection Numbers. The IP flow data is then analyzed using the query language SPARQL (SPARQL Protocol and RDF[1] Query Language [33]) in order to determine the type of botnet (e.g., Fast Flux), the bot master, C&C Servers and bots.

[1]RDF [33] stands for Resource Description Framework which is a standard database format.

TABLE 1. Bot behaviors

Statement	Domain (Subject)	Property (Predicate)	Range (Object)
S1	BotMasters	BotMaster_Bots	Bots
S2	BotMasters	BotMaster_CCserver	CCServers
S3	CCServers	CCserver_Bots	Bots
S4	SourceIP SourcePort	hasHTTP	DestinationIP DestinationPort
S5	SourceIP SourcePort	hasICMP	DestinationIP DestinationPort
S6	SourceIP SourcePort	hasP2P	DestinationIP DestinationPort
S7	SourceIP SourcePort	hasTCP	DestinationIP DestinationPort
S8	SourceIP SourcePort	hasUDP	DestinationIP DestinationPort
S9	IP_Flow_Based	IP_FlowBased_Information	DestinationIP SourceIP Protocol DestinationPort SourcePort
S10	Thing	CCserver	String
S11	ConnectionNumbers	ConnectionNumber	int
S12	TTL_time	TTL	time
⋮	⋮	⋮	⋮

In implementing the ontology, the botnet is modelled by a set of characteristic botnet behavior patterns. As shown in Table 1, each behavior pattern (relationship) is a statement and has the form of an RDF-triple [33] (subject, predicate, object). For example, statement S1 shows a behavior pattern where the bot master controls a bot. Similarly, statement S2 shows a behavior pattern where the bot master controls a C&C Server. Statement S4 shows a behavior pattern where the source node establishes an HTTP connection with the destination node. There are over 50 behavior pattern statements in this model, only the first 12 are shown.

Having mapped the input IP flow data into the seven ontology relationships, the statements are applied sequentially to the database in order to identify the botnet topology. Figure 4 below shows the outcome of the integration process following the application of statements S1 and S2.

In this particular case, the results indicate that the bot master controls five bots by means of four C&C Servers. Since the input IP flow data is reliable (i.e., collected from real-world network activities), it is highly likely that the identified nodes are true bots. Having applied each of the fifty plus statements to the database, inference rules are then used to test for the existence of additional bots. The output of the completed integration and inference processes is shown below in Figure 5. The inference results suggest that the bot master may actually control eight bots rather than initially known five. Note that having run the inference process, the new (inferred) information is added to the database in order to increase the domain knowledge of the search algorithm.

3.1. **Inference process rules.** The developed system assumes the bot master, C&C Servers, IP addresses, and domain name are already known, and the objective is to identify the botnet topology. The inference process used to identify the botnet topology utilizes a set of inference rules written in Semantic Web Rule Language (e.g., Protégé [29], Pellet [30], or Jena [32]). In total, around 50 rules were constructed; with each rule independent of the others. As shown in Table 2, each of the rules has the format C(x), P(x,y), differentFrom(x,y), sameAs(x,y) and builtin(r,x), where C is an OWL (OWL Web Ontology Language) class, P is an OWL property, r is a built-in relation and x, y are variables. The definition of Rules-Botnet means that the combination of the BotMaster and Bots properties implies the BotMaster_Bots property, allowing the atomic formula to be written as:

$$\text{BotMaster}\ (?x, ?y) \wedge \text{Bots}\ (?y, ?z) \rightarrow \text{BotMaster\_Bots}\ (?x, ?z).$$

In accordance with this rule, if IP address 10.1.1.2 has 10.1.1.1 as a bot master and 10.1.1.1 has 10.1.1.5 as a C&C Server, then IP address 10.1.1.2 is controlled by 10.1.1.5. Similarly, Rule-HTTP in Table 2 means that HTTP communication is established between the source IP address and the destination IP address.

In general, two forms of inference methods exist, namely forward chaining and backward chaining [3]. The present study utilizes a forward chaining approach since the inference process is data driven rather than goal driven. The inference steps are shown in the following.

The inference results indicate the communication topology of the botnet. (Note that four basic types of botnet topology exist [15]). By understanding the botnet topology, appropriate countermeasures can be put in place to protect the network until the botnet can be taken down. Once the inference results have been obtained, further SPARQL queries can be performed based on parameters provided by Protégé. For example, to determine the number of victims in a particular domain affected by the botnet within a certain period of time, the following query can be performed:

SELECT ? dns_name

WHERE { ?subject rdfs:comment ? dns_name filter (?dns_name) }

3.2. **Modeling bot behavior.** In the present study, bot behavior is modelled in terms of the source and destination IP addresses, the source and destination port numbers, the IP protocol, the number of bytes, and the number of sessions, i.e.,

```
<BotMasters rdf:ID="BotMasters_3">
<BotMaster_Bots rdf:resource="#Bots_1"/>
<BotMaster_Bots rdf:resource="#Bots_2"/>
<BotMaster_Bots rdf:resource="#Bots_6"/>
<BotMaster_Bots rdf:resource="#Bots_8"/>
<BotMaster_Bots rdf:resource="#Bots_9"/>
<BotMaster_CCserver rdf:resource="#CCServers_1"/>
<BotMaster_CCserver rdf:resource="#CCServers_3"/>
<BotMaster_CCserver rdf:resource="#CCServers_4"/>
<BotMaster_CCserver rdf:resource="#CCServers_5"/>
<rdfs:comment
rdf:datatype="&xsd;string">88.117.175.114</rdfs:comment>
< /BotMasters>
```

FIGURE 4. Statement integration of S1 and S2

```
<BotMasters rdf:ID= "BotMasters_3">
        <BotMaster_Bots rdf:resource="#Bots_1"/>
        <BotMaster_Bots rdf:resource="#Bots_2"/>
        <BotMaster_Bots rdf:resource="#Bots_4"/>
        <BotMaster_Bots rdf:resource="#Bots_5"/>
        <BotMaster_Bots rdf:resource="#Bots_6"/>
        <BotMaster_Bots rdf:resource="#Bots_7"/>
        <BotMaster_Bots rdf:resource="#Bots_8"/>
        <BotMaster_Bots rdf:resource="#Bots_9"/>
        <BotMaster_CCserver rdf:resource="#CCServers_1"/>
        <BotMaster_CCserver rdf:resource="#CCServers_3"/>
        <BotMaster_CCserver rdf:resource="#CCServers_4"/>
        <BotMaster_CCserver rdf:resource="#CCServers_5"/>
        <rdfs:comment
rdf:datatype="&xsd;string">88.117.175.114</rdfs:comment>
< /BotMasters>
<owl:Class rdf:ID="Bots">
        <rdfs:subClassOf rdf:resource="#Botnets"/>
< /owl:Class>
    <Bots rdf:ID="Bots_1">
        <rdfs:comment
rdf:datatype="&xsd;string">10.16.11.24</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_10">
        <rdfs:comment
rdf:datatype="&xsd;string">10.16.9.12</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_2">
        <rdfs:comment
rdf:datatype="&xsd;string">10.11.22.14</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_3">
<rdfs:comment
rdf:datatype="&xsd;string">10.16.9.15</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_4">
        <rdfs:comment
rdf:datatype="&xsd;string">10.11.31.17</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_5">
        <rdfs:comment
rdf:datatype="&xsd;string">10.22.12.24</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_6">
        <rdfs:comment
rdf:datatype="&xsd;string">10.24.10.13</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_7">
        <rdfs:comment
rdf:datatype="&xsd;string">10.24.10.31</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_8">
        <rdfs:comment
rdf:datatype="&xsd;string">10.16.11.31</rdfs:comment>
    < /Bots>
    <Bots rdf:ID="Bots_9">
        <rdfs:comment
rdf:datatype="&xsd;string">10.24.10.18</rdfs:comment>
< /Bots>
```

FIGURE 5. Model results after integration and the inference processes

Rule1: Bots(X) → CCServers(X)
Rule2: CCServers(X) → BotMasters(X)
Facts: 10.1.2.1 is a bot.
Inference steps:
1. Bot (10.1.2.1) is true
2. X = 10.1.2.1 (*unification*)
3. Bots(X) → CCServers(X) → CCServers(10.1.2.1) is true
X' = X = 10.1.2.1
4. CCServers(X') → BotMasters(X')
5. BotMaster(10.1.2.1) is true

*Note that the process of finding substitutions for variables (X) such that the arguments match is referred to as unification

FIGURE 6. Inference example

$$(sip,\ spt,\ dip,\ dpt,\ ptcl,\ bytes,\ sessions)$$
$$M = \{B_1, B_2, B_3, \ldots, B_n\}$$
$$B = \{sip,\ spt,\ dip,\ dpt,\ ptcl,\ bytes,\ sessions\}$$

where M is a set of behaviors and B comprises the properties given above. The bot behavior is modelled through a set of rules having form

$$r:\ (M_1 \rightarrow b_j, \beta_{1j})$$

where r is the degree of reliability of the rule. The following example illustrates the inference rule used to distinguish malware from a fast-flux botnet:

$$\text{if } (ttl < 500)\ \&\&\ \{[(sip,\ dip) \mathrel{!}== (sip,\ dip)] > 5\}$$
$$\text{then } bot(X) = \text{fastflux\_bot}$$

In applying the inference rules, the properties of the causal operator are as follows:

a. Conditional: If a causes b, then a → b.
b. Transitivity: If a → IS-b and b → IS-c, then a → IS-c.
c. Reflexivity: c → IS-c.

TABLE 2. SWRL inference rules

Rule	Expression
Rule-HTTP	SourceIP(?x) ∧ SourcePort(?y) ∧ DestinationIP(?z) ∧ DestinationPort(?a) → hasHTTP(?x, ?y)
Rule-ICMP	SourceIP(?x) ∧ SourcePort(?y) ∧ DestinationIP(?z) ∧ DestinationPort(?a) → hasICMP(?x, ?y)
Rule-P2P	SourceIP(?x) ∧ SourcePort(?y) ∧ DestinationIP(?z) ∧ DestinationPort(?a) → hasP2P(?x, ?y)
Rule-TCP	SourceIP(?x) ∧ SourcePort(?y) ∧ DestinationIP(?z) ∧ DestinationPort(?a) → hasTCP(?x, ?y)
Rule-UDP	SourceIP(?x) ∧ SourcePort(?y) ∧ DestinationIP(?z) ∧ DestinationPort(?a) → hasUDP(?x, ?y)
Rule-Botnets	BotMasters(?x) ∧ CCServers(?y) ∧ Bots(?z) → Botnets(?x)
Rule-FastFluxBot	DNS_Name(?x) ∧ ConnectionNumbers(?y) ∧ swrlb:lessThan(TTL, 500) → FastFlux(?x)
⋮	⋮

The steps in the inference process are as follows:

$$\text{if } (\alpha \text{ causes } \beta) \text{ then } \alpha \text{ explains } \beta \text{ because\_possible } \{\alpha\}.$$

For example:
Consider the causal operator

$$C = \{\text{CCserver(bot) causes Bots(bot)}\}$$

Here, the atom CCserver(bot) explains Bots(bot) because_possible {CCserver(bot)} is inferred. That is, CCserver(bot) is an explanation for Bots(bot).

4. Performance.

4.1. **Performance evaluation.** In the present study, the efficiency of the proposed ontology is evaluated by measuring the inference time obtained by different reasoners for the given databases containing a different number of alerts. The performance evaluation experiments were performed on a PC equipped with an Intel Core 2 Duo E2140 processor (1.6 GHz) and 2 G RAM. The performance results are presented in Figure 7.

As shown in Figure 7, the inference time increases approximately linearly with an increasing number of alerts for all three reasoners. In addition, the Protégé-SWRL reasoner is slower than the Jena and Pellet reasoners since Protégé is converted to Jess [31] (i.e., the rule engine for the Java platform) before inferencing can be performed.

Figure 8 illustrates the system convergence properties of the three reasoners given repeated inference processes. Note total time comprises load time, transfer time and inference time:

$$\text{Load time (lt)} + \text{transfer time (tt)} + \text{inference time (it)} = \text{total time.}$$

The results show the Jena reasoner provides the best overall inference efficiency of the three schemes. The total numbers of axioms inserted by the Protégé, Jena and Pellet reasoners are presented in Tables 3, 4 and 5, respectively, together with a breakdown of the corresponding total inference time.

Note that in the tables above, the second column indicates the number of new axioms inserted into the database following the inference process, while the third column indicates the number of alerts related to botnet behavior.

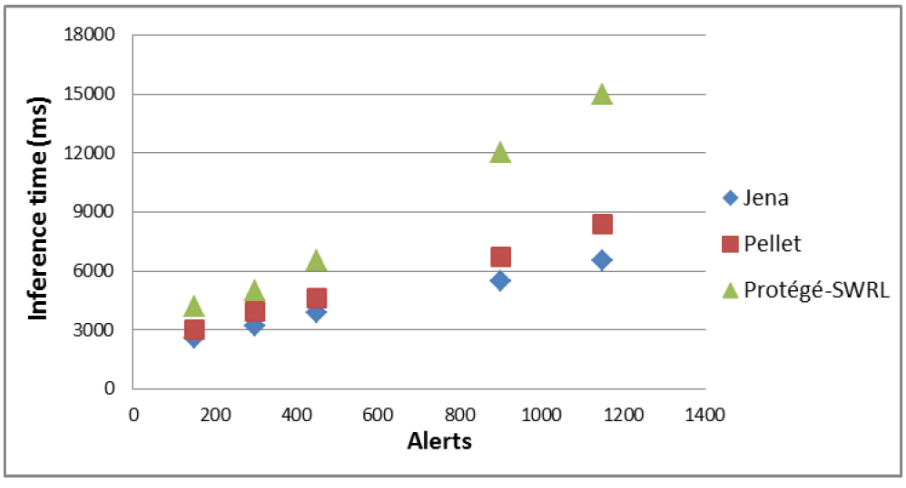


FIGURE 7. Inference times of Jena, Pellet and Protégé reasoners

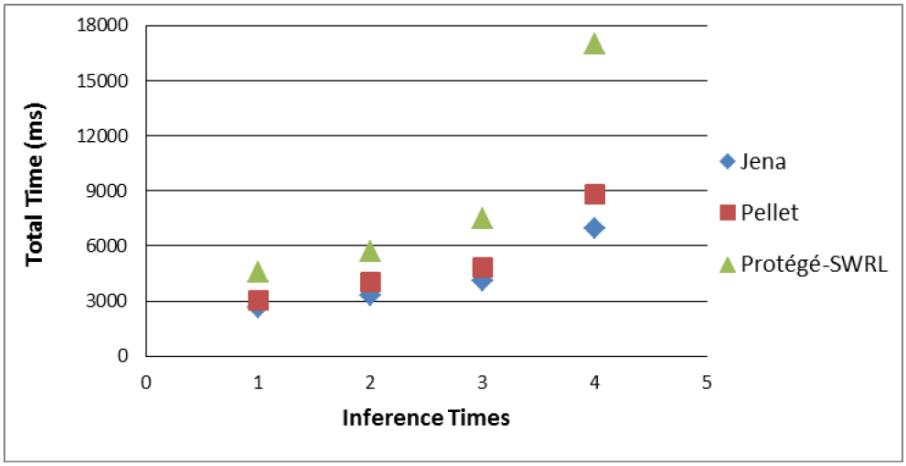


FIGURE 8. System convergence properties of Jena, Pellet and Protégé reasoners

4.2. **Performance comparison of three reasoners.** Figure 9 compares the total inference times obtained by the proposed ontology with the Jena, Pellet and Protégé reasoners with the results presented in [10] for the Jena and Protégé reasoners and in [41] for the SWRL reasoner. Note for the present study, [10] and [41], the Jena reasoner provides in the lowest inference time. Moreover, it is seen that the ontology proposed in the present study yields a more efficient inference process than that in [10] or [41] given a database containing more than approximately 440 alerts.

The performance differences between our work versus [10] and [41] are in the rule amounts and the computational complexity. The computational complexities of the three reasoners are as follows:

$$\text{Protégé-SWRL}: f(n) = O(g(n)) = 3898n - 1051$$
$$\text{Jena:}\ f(n) = O(g(n)) = 1382n + 798$$
$$\text{Pellet:}\ f(n) = O(g(n)) = 1814n + 645$$

In other words, all reasoners have a complexity of $O(T)$, where T is the number of inference steps.

The performance evaluation results suggest that Protégé-SWRL should be chosen as the reasoner for the inference process since it has the best integration. Moreover, neither Jena nor Pellet has a common communication method to invoke a remote knowledge base.

Pellet supports punning (vocabulary sharing) with a minor caveat [30]. No additional atoms are needed in the rules used to perform inferencing. The SWRL rules can be loaded directly into Pellet and parsing and processing operations then performed. The difference between the Protégé reasoner and the Pellet reasoner lies in that Pellet only supports SWRL built-ins for data, time and duration, but does not support built-ins such as swrlb:first, swrlb:empty. The present evaluation results show that Protégé results

TABLE 3. Axioms inserted and convergence time details – Protégé-SWRL

Alerts	Number of axioms inserted	Proportion of botnet axioms	Load time (ms)	Transfer time (ms)	Inference time (ms)	Total time (ms)
170	32	0.8416	469	53	4117	4589
300	47	0.8646	587	117	4993	5697
450	62	0.8789	794	215	6504	7513
1150	39	0.9672	1518	488	14971	16977

TABLE 4. Axioms inserted and convergence time details – Jena

Alerts	Number of axioms inserted	Proportion of botnet axioms	Load time (ms)	Transfer time (ms)	Inference time (ms)	Total time (ms)
170	32	0.8416	0	48	2595	2643
300	47	0.8646	0	106	3182	3288
450	62	0.8789	0	207	3897	4104
1150	39	0.9672	0	483	6495	6978

TABLE 5. Axioms inserted and convergence time details – Pellet

Alerts	Number of axioms inserted	Proportion of botnet axioms	Load time (ms)	Transfer time (ms)	Inference time (ms)	Total time (ms)
170	32	0.8416	0	51	3006	3057
300	47	0.8646	0	112	3911	4023
450	62	0.8789	0	211	4593	4804
1150	39	0.9672	0	487	8359	8846

in a longer inference time than Pellet. The difference between the Pellet reasoner and the Jena reasoner lies in the SPARQL query engine. The Pellet query engine provides better optimization performance than Jena, and yields more rapid query response. In addition, the Pellet reasoner produces results using inferred individuals, whereas the Jena reasoner does not. In the system proposed in this study, RDF statements are combined with customized rules and used in a forward inferencing process. The syntax of the Jena rules is shown in Table 6.

Note Jena and Protégé rules are the same in SWRL. However, the inference engines of the two reasoners are very different. Table 7 shows comparisons among different systems.

As depicted in Table 8, many proposed methods are based on inspection and analysis of IP flow data and the communication behaviors between bots. The existing studies are abundant but identifying the topology of a botnet is rarely discussed in previous literature. Some existing papers developed ontology construction and explained build concepts. However, our contributions lie in botnet ontology identification by a set of interference rules from examining IP flow data. How accurate the topology can be estimated by the

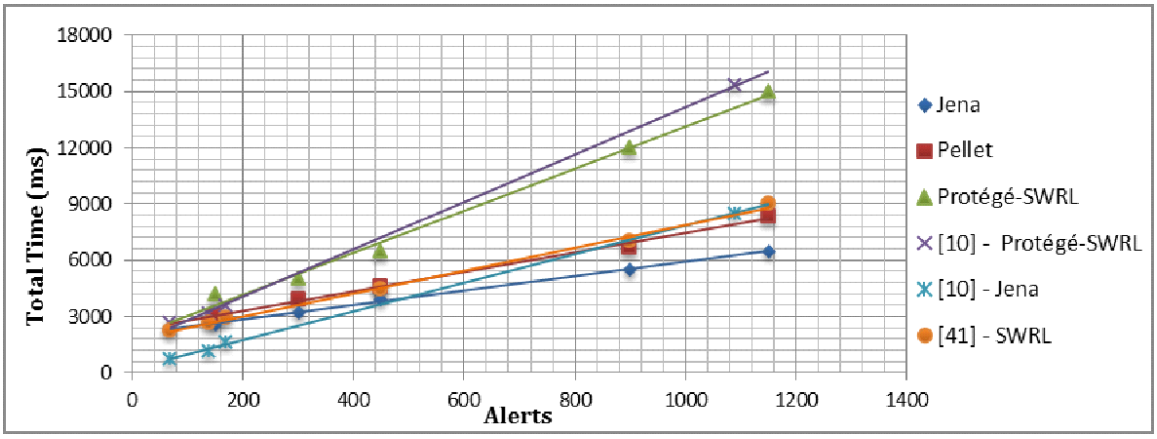


FIGURE 9. Comparison of present results with those presented in [10,41] for inference times of Jena, Pellet and Protégé reasoners

limited input information for the three reasoners is shown in Figure 10. Jena has better performance (92.13%) than the other reasoners due to *hybrid mode* to balance the reasoning. To get more accuracy, we run in hybrid mode. See the following example:

[rdf: (?x ?y ?a), SourceIP(?x ?y) ? ConnectionNumbers(?a > 100) → [DDoS(?x ?y) ← notEqual(?x,?y)]]

The above example shows that if we want to know the DDoS attack map of the botnets, we can use hybrid mode at alerts up to 20,000. Both ?x and ?y are different alerts (instances) as well as the connections ?a are more than 100 times. However, the computation complexity is over 20,000 (ms).

After considering the malware behaviors, payload information and sophisticated bot communication, topology accuracy could be improved by the hybrid mode. To find the conjunction, we run in *hybrid mode*, the forward engine maintains a set of inferred statements, then queries are answered by the backward engine. As shown in Figure 11, when number of alerts is more than 20,000, the CPU usage is exhausted due to the large amount of IP flow data. Our datasets were loaded on disk instead of memory to maximize space for computation.

4.3. **Rule evaluation.**

4.3.1. *Relationship richness.* In [13,40], metrics such as population, cohesion or relationship richness are defined. We choose the metric relationship richness (RR), which is defined as $RR = \frac{|P|}{|SC|+|P|}$. In our domain knowledge, $|P|$ is the number of properties (including object and data properties), and $|SC|$ is the total number of subclasses (which is the same as the number of *IS-A* relationships). According to the definition, this metric reflects the diversity of relations in ontology [35]. Considering the special case, which only contains classes which used disjoint unions or equivalent classes, $|SC|$ is 0 and no properties have been introduced, $|P|$ is 0 as well, leading to $RR = \frac{|0|}{|0|+|0|}$, which is undefined. When we add a property to the ontology, RR will become $\frac{|1|}{|0|+|1|} = 1$, denoting very good

TABLE 6. Syntax of Jena rules

Rule	Expression
Rdfs1	[rdfs1: (?x ?y ?z), (SourceIP(?x) ? SourcePort(?y) ? DestinationIP(?z) ? DestinationPort(?a) → hasHTTP(?x, ?y))]
Rdfs2	[rdfs2: (?x ?y ?z ?a), (SourceIP(?x) ? SourcePort(?y) ? DestinationIP(?z) ? DestinationPort(?a) → hasICMP(?x, ?y))]
Rdfs3	[rdfs3: (?x ?y ?z ?a), (SourceIP(?x) ? SourcePort(?y) ? DestinationIP(?z) ? DestinationPort(?a) → hasP2P(?x, ?y))]
Rdfs4	[rdf4: (?x ?y ?z ?a), (SourceIP(?x) ? SourcePort(?y) ? DestinationIP(?z) ? DestinationPort(?a) → hasTCP(?x, ?y))]
Rdfs5	[rdf5: (?x ?y ?z ?a), (SourceIP(?x) ? SourcePort(?y) ? DestinationIP(?z) ? DestinationPort(?a) → hasUDP(?x, ?y))]
Rdfs6	[rdf6: (?x ?y ?z), (BotMasters(?x) ? CCServers(?y) ? Bots(?z) → Botnets(?x))]
Rdfs7	[rdf7: (?x ?y), (DNS_Name(?x) ? ConnectionNumbers(?y) ? swrlb:lessThan(TTL, 500) → FastFlux(?x))]
⋮	⋮

TABLE 7. System comparison

System	Detection Method	Rule Metric	Performance Comparison	Data Collection
Wang et al. [34]	anomaly	no	no	distributed
Cuppens-Boulahia et al. [14]	anomaly	no	no	centralized
Geyik and Szymanski [37]	misuse	yes	yes	centralized
Undercoffer et al. [28]	misuse	no	no	distributed
Guerrero et al. [29]	not specified	no	no	not specified
Holgado et al. [10]	misuse	no	yes	distributed
Martimiano and Moreira [4]	misuse	no	no	not specified
Vergara et al. [24]	compound	no	no	centralized
Szymczyk [38]	compound	no	no	centralized
Chen et al. [12]	not specified	no	no	distributed
Qi et al. [39]	not specified	yes	yes	not specified
Li et al. [25]	compound	yes	yes	distributed

TABLE 8. Detail of computing process

Training data: 20 whitelists and 100 blacklists, $k = 8$	
Rule r1: covers 10 whitelists and 15 blacklists	$f_w = 10,\ f_b = 15,\ \theta_w = 5,\ \theta_b = 20.8$
Rule r2: covers 10 whitelists and 13 blacklists	$f_w = 10,\ f_b = 13,\ \theta_w = 3.83,\ \theta_b = 19.17$
Rule r3: covers 8 whitelists and 10 blacklists	$f_w = 8,\ f_b = 10,\ \theta_w = 3,\ \theta_b = 15$
Rule r4: covers 8 whitelists and 30 blacklists	$f_w = 8,\ f_b = 30,\ \theta_w = 6.33,\ \theta_b = 31.67$
Rule r5: covers 7 whitelists and 42 blacklists	$f_w = 7,\ f_b = 42,\ \theta_w = 8.17,\ \theta_b = 40.83$
Rule r6: covers 14 whitelists and 73 blacklists	$f_w = 14,\ f_b = 73,\ \theta_w = 14.5,\ \theta_b = 72.5$
Rule r7: covers 16 whitelists and 84 blacklists	$f_w = 16,\ f_b = 84,\ \theta_w = 16.67,\ \theta_b = 83.33$
Rule r8: covers 2 whitelists and 0 blacklists	$f_w = 2,\ f_b = 0,\ \theta_w = 0.33,\ \theta_b = 1.67$
⋮	⋮

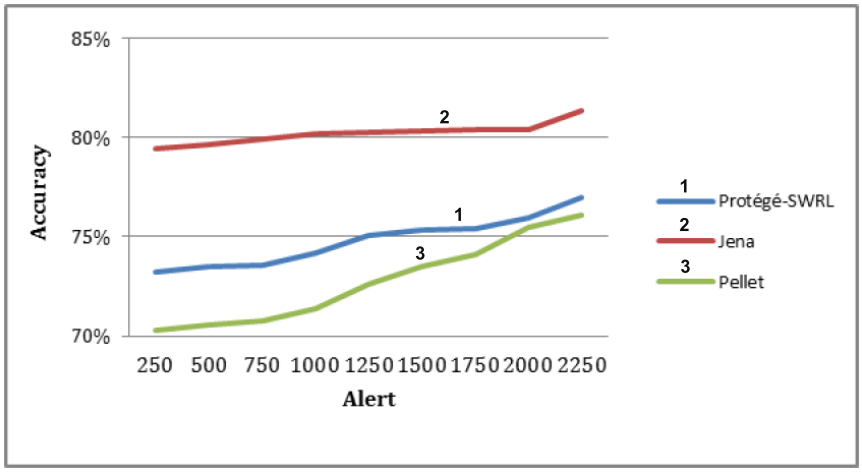


FIGURE 10. Botnet topology accuracy

relationship richness. Therefore, relationship richness [35], is not a useful measure, since the number of subclasses converges to be $|H(N(O))|$ [35].

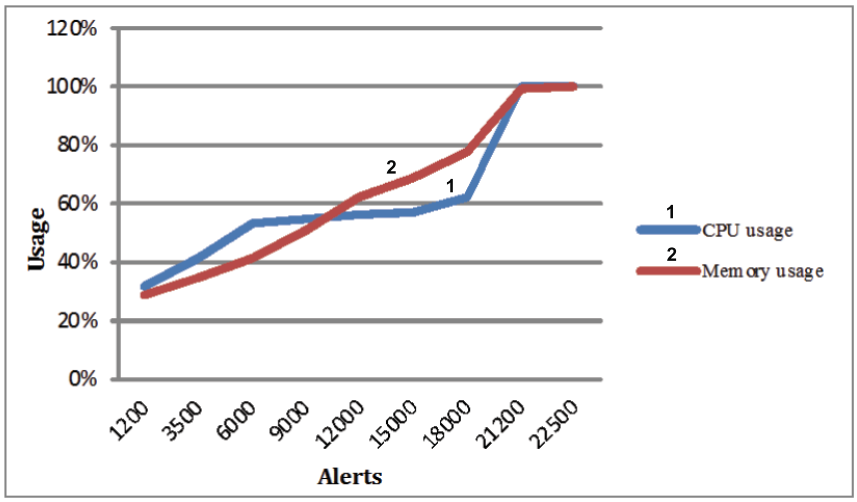


FIGURE 11. The processing ability test

The relationship richness metric from [35] is:

$$RR(O) = \frac{|p_n(O)|}{|C_n(O)| + |p_n(O)|}$$

where $|C_n(O)|$ is the number of classes. Since we understand the number of subclasses converges to be $|H(N(O))|$ [35], we use the number of classes rather than the number of subclasses. We define the *treelikeness* of the class hierarchy as follows:

$$T(O) = \frac{|C_n(O)/R(O)|}{|H(N(O))|}$$

where $|R(O)|$ is the set of leaf classes that have no subclass and is given by:

$$R(O) = \{C | C \in C_n(O) \wedge \forall D \in C_n(O) : SubClassOf(CD) \notin H(N(O))\}$$

The closer the value is to 1, the more treelike the class hierarchy is. For the example ontology in Figure 2, the $T(O) = 0.42$, the $RR(O) = 0.5$, and the original $RR(O) = 0.56$. We confirm that the number of classes is a better choice, since the *treelikeness* is independent of the relationship richness.

4.3.2. *Rule evaluation metrics.* To understand what rules we predefined are available, we use an evaluation metric to determine which causal operator should be added (or

TABLE 9. Computing the R value

Training data: 20 whitelists and 100 blacklists, $k = 8$	
R(r1)	$2(10 \cdot \log_2 10/5 + 15 \cdot \log_2 15/20.8) = 15.74$
R(r2)	$2(10 \cdot \log_2 10/3.83 + 13 \cdot \log_2 13/19.17) = 3.95$
R(r3)	$2(8 \cdot \log_2 8/3 + 10 \cdot \log_2 10/15) = 3.30$
R(r4)	$2(8 \cdot \log_2 8/6.33 + 30 \cdot \log_2 30/31.67) = 0.21$
R(r5)	$2(7 \cdot \log_2 7/8.17 + 42 \cdot \log_2 42/40.83) = 0.09$
R(r6)	$2(14 \cdot \log_2 14/14.5 + 73 \cdot \log_2 73/72.5) = 0.02$
R(r7)	$2(16 \cdot \log_2 16/16.67 + 84 \cdot \log_2 84/83.33) = 0.004$
R(r8)	$2(2 \cdot \log_2 2/0.33 + 0 \cdot \log_2 0/1.67) = 3.13$
⋮	⋮

removed) during the inference process. For example, consider training data that contains 20 whitelists and 100 blacklists. Then, we use the following likelihood ratio statistic to prune rules (Tables 8 and 9) that have poor coverage:

$$R = \sum_{i}^{k} f_i \log_2 \left(\frac{f_i}{e_i} \right),$$

where, k is the number of rules, and f_i is the number of instances with class value i covered. According to the R value, we suggest that R(r1) is a better rule than the others.

5. **Conclusion and Future Work.** This paper aims to identify botnet topology with low inference time and high accuracy using a set of inference rules to facilitate automatic identification of the botnet topology via a machine learning algorithm. The effectiveness of the proposed approach has been demonstrated using three different reasoners (Jena, Pellet and Protégé) and compared to [10] and [41] (see Figure 9). Our approach has the lowest inference time and highest degree of accuracy (see Figure 10). Future study will extend the proposed ontology and inference rules to the case of an Advanced Persistent Threat (APT).

Acknowledgment. This work was supported by the Testbed@TWISC (Taiwan Information Security Center) under contracts numbers NSC 100-2219-E-006-001 and NSC 100-2218-E-006-029-MY3.

REFERENCES

[1] *DAMBALLA: Top 10 Botnet Threat Report*, http://www.damballa.com/downloads/r_pubs/Damballa_2010_Top_10_Botnets_Report.pdf, 2013.
[2] *The Anti-Botnet Project of TAnet*, http://www.anti-botnet.edu.tw, 2013.
[3] *Forward Chaining and Backward Chaining*, http://knut.hinkelmann.ch/lectures/ke2011/KE-4_FC_VS_BC.pdf, 2013.
[4] L. A. F. Martimiano and E. Moreira, The evaluation process of a computer security incident ontology, *Proc. of the Workshop on the 2nd Workshop on Ontologies and Their Applications Co-located with the International Joint Conference IBERAMIA-SBIA-SBRN'06*, 2006.
[5] Y. Gao, Z. Li and Y. Chen, A dos resilient flow-level intrusion detection approach for high-speed networks, *Proc. of the 26th IEEE International Conference on Distributed Computing Systems*, p.39, 2006.
[6] H. Lai, S. Cai, H. Huang, J. Xie and H. Li, A parallel intrusion detection system for high-speed networks, *Proc. of the 2nd International Conference Applied Cryptography and Network Security*, pp.439-451, 2004.
[7] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagl, K. Levitt, C. Wee, R. Yip and D. Zerkle, GrIDS – A graph based intrusion detection system for large networks, *Proc. of the 19th National Information Systems Security Conference*, pp.361-370, 1996.
[8] N. Duffield, P. Haffner, B. Krishnamurthy and H. Ringberg, Rule-based anomaly detection on IP flows, *IEEE INFOCOM*, pp.424-432, 2009.
[9] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras and B. Stiller, An overview of IP flow-based intrusion detection, *IEEE Communications Surveys & Tutorials*, vol.12, no.3, 2010.
[10] P. Holgado, J. E. L. de Vergara, V. A. Villagra, I. Sanz and A. Amaya, Sharing information about security alerts using semantic web technologies, *International Conference on Network and Service Management*, pp.270-273, 2010.
[11] C. Livadas, B. Walsh, D. Lapsley and T. Strayer, Using machine learning techniques to identify botnet traffic, *The 2nd IEEE LCN Workshop on Network Security*, pp.967-974, 2006.
[12] C.-S. Chen, T.-C. Liu and C.-H. Su, Exploring the correlation between P2P applications and network architecture, *Proc. of ICACT*, vol.1, pp.171-175, 2009.
[13] D. Vrandečić, Handbook on ontologies, *International Handbooks on Information Systems*, pp.293-313, 2009.

[14] N. Cuppens-Boulahia, F. Cuppens, J. E. Lopez de Vergara, E. Vazquez, J. Guerra and H. Debar, An ontology-based approach to react to network attacks, *Risks and Security of Internet and Systems*, pp.27-35, 2008.
[15] *DAMBALLA: Botnet Communication Topologies*, http://www.damballa.com/downloads/r_pubs/WP%20Botnet%20Communications%20Primer%20%282009-06-04%29.pdf, 2013.
[16] *Internet2: About Internet2 Report*, http://www.internet2.edu/resources/AboutInternet2.pdf, 2013.
[17] M. Roesch, *Snort, Intrusion Detection System*, http://www.snort.org, 2013.
[18] *WinSnort: Windows Intrusion Detection System (WinIDS)*, http://www.winsnort.com, 2013.
[19] A. Lakhina, M. Crovella and C. Diot, Mining anomalies using traffic feature distributions, *SIGCOMM 05*, pp.217-228, 2005.
[20] T. Shon and J. Moon, A hybrid machine learning approach to network anomaly detection, *Information Science*, vol.18, pp.3799-3821, 2007.
[21] T. Ahmed, B. Oreshkin and M. J. Coates, Machine learning approaches to network anomaly detection, *The 2nd Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, Cambridge, MA, USA, 2007.
[22] A. W. Moore and D. Zuev, Internet traffic classification using Bayesian analysis techniques, *Proc. of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, New York, USA, p.50, 2005.
[23] Z. Li, A. Goyal, Y. Chen and V. Paxson, Towards situational awareness of large-scale botnet probing events, *IEEE Trans. on Information Forensics and Security*, vol.1, no.6, pp.175-188, 2011.
[24] J. E. López de Vergara, E. Vázquez, A. Martin, S. Dubus and M. N. Lepareux, Use of ontologies for the definition of alerts and policies in a network security platform, *Journal of Networks*, vol.4, no.8, pp.720-733, 2009.
[25] J. E. López de Vergara, V. A. Villagrá, P. Holgado, E. de Frutos and I. Sanz, A semantic web approach to share alerts among security information management systems, *Communications in Computer and Information Science*, vol.72, pp.27-38, 2010.
[26] J. Undercoffer, A. Joshi and A. Pinkston, Modeling computer attacks: An ontology for intrusion detection, *The 6th International Symposium on Recent Advances in Intrusion Detection*, pp.113-135, 2003.
[27] A. Guerrero, V. Villagr'a, J. B. J. E. L'opez de Vergara and A. S'anchez-Maci', An ontology-based policy refinement using SWRL rules for management information definitions in OWL, *Lecture Notes in Computer Science*, vol.3775, pp.12-23, 2005.
[28] A. Maedche, B. Motik and L. Stojanovic, Managing multiple and distributed ontologies on the semantic web, *Very Large Data Bases Journal*, vol.12, no.4, pp.286-302, 2003.
[29] *Protégé*, http://protege.stanford.edu, 2013.
[30] *Pellet*, http://clarkparsia.com/pellet, 2013.
[31] *Jess – The Rule Engine for the Java Platform*, http://www.jessrules.com, 2013.
[32] *Jena*, http://jena.apache.org/index.html, 2013.
[33] E. Prud'hommeaux and A. Seaborne, SPARQL query language for RDF, *W3C Recommendation*, 2008.
[34] B. Wang, X. Ju and S. Zhang, A distributed intrusion detect model based on alert data correlation analysis, *International Conference on Computer Application and System Modeling*, vol.3, pp.669-673, 2010.
[35] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth and B. Aleman-Meza, OntoQA: Metric-based ontology quality analysis, *IEEE Computer Society*, pp.45-53, 2005.
[36] *Dshield.org*, http://dshield.org, 2013.
[37] S. C. Geyik and B. K. Szymanski, Event recognition in sensor networks by means of grammatical inference, *IEEE INFOCOM*, pp.900-908, 2009.
[38] M. Szymczyk, Detecting botnets in computer networks using multi-agent technology, *The 4th International Conference on Dependability of Computer Systems*, pp.192-201, 2009.
[39] Y. Qi, L. Xu, B. Yang, Y. Xue and J. Li, Packet classification algorithms: From theory to practice, *IEEE INFOCOM*, pp.648-656, 2009.
[40] C. A. Catania and C. G. Garino, Automatic network intrusion detection: Current techniques and open issues, *Computers and Electrical Engineering*, vol.38, no.5, pp.1062-1072, 2012.
[41] V. Mateos, V. A. Villagrá, F. Romero and J. Berrocal, Definition of response metrics for an ontology-based automated intrusion response systems, *Computers and Electrical Engineering*, vol.38, no.5, pp.1102-1114, 2012.
[42] *Zeus and Spyeye Tracker*, http://www.abuse.ch, 2013.

[43] C.-B. Jiang and J.-S. Li, IP flow data correlation with inference rules, *Advanced Materials Research*, vol.403, no.408, pp.1211-1213, 2012.
[44] *McAfee Advanced Correlation Engine*, http://www.mcafee.com/us/resources/data-sheets/ds-advanced-correlation-engine.pdf, 2011.
[45] *IP Flow Datasets*, http://www.hsnet.ee.ncku.edu.tw/Project/botd-dataset, 2010.