Ontology-based Automation of Penetration Testing

Ge Chu and Alexei Lisitsa
Department of Computer Science, University of Liverpool, Liverpool, U.K.

Keywords: Automated Penetration Testing, Information Security, Ontology, Protege, SWRL, BDI Agent.

Abstract: Ontology is a widely used knowledge representation formalism in artificial intelligence area in recent years. In this paper, we propose an ontology-based automated penetration testing approach. We use protege to create the ontology, which is constructed based on attack taxonomy. SWRL rules are used to create penetration testing knowledge base and reasoning. We use agent-based(BDI) framework to achieve the process of automated penetration testing.

1 INTRODUCTION

Penetration testing is a popular method used to evaluate system security from the perspective of an attacker. According to PTES (penetration testing execution standard) (Nickerson et al., 2014), the process of penetration testing includes seven stages: Pre-Engagement Interaction, Information Gathering, Threat Modelling, Vulnerability Analysis, Exploitation, Post Exploitation and Reporting. In general, the success of penetration testing highly relies on the expertise, technology and experience of the executive team. With this in mind, it is worth pointing out that only a minority of people possess knowledge about penetration testing and the knowledge of penetration testing is challenging to share and reuse (Shah and Mehtre, 2015). Due to these reasons, the efficiency of penetration testing has been adversely affected. Similarly, establishing and subsequent-managing of the knowledge base also poses a challenge in the research on automated penetration testing.

Ontology (Guarino, 1995) is a concept from the field of philosophy, which has been widely used for knowledge representation in the field of artificial intelligence in recent years. It can be used to describe concepts and their relationships in a certain domain. With the assistance of domain experts, researchers have currently established ontology in many areas. For example: SENSUS ontology (Knight et al., 1995) provided a conceptual structure for machine translation, UMLS ontology (Bodenreider, 2004) is a medical language systems , CYC ontology (Lenat and Guha, 1989) is used to establish human common sense, and an English dictionary is based on cognitive linguistics Word-Net ontology (Miller, 1995).

Ontology not only allows the domain knowledge to be shared and reused through formalisation, but it also has an excellent conceptual hierarchy and support for logical reasoning. In this paper, we firstly build a penetration testing ontology according to attack taxonomy and secondly we use SWRL (Horrocks et al., 2004) which is a semantic web rule language combining OWL and RuleML, to create a knowledge base and reasoning in our automated penetration testing approach. Finally, we use the BDI agent structure (Rao et al., 1995) to implement an automated penetration testing attack process.

The remainder of this paper is organised as follows: Section 2 presents related work regarding ontology in the information security field. Section 3 outlines our proposed ontology-based on attack taxonomy. Section 4 concerns the knowledge base by SWRL and reasoning. Section 5 describes the process of automated penetration testing using a BDI agent structure. Section 6 illustrates a case study of an attack scenario. Finally, we summarise this paper in section 7.

2 RELATED WORK

There have been some research investigating taxonomy and ontology in the security field and related to automated penetration testing. Pinkstion et al.(Pinkston et al., 2004) produced an ontology specifying a model of computer attacks, based on over 4,000 classes of computer attacks for intrusion detection. It is categorized according to the system compo-

Chu, G. and Lisitsa, A.
Ontology-based Automation of Penetration Testing.
DOI: 10.5220/0009171007130720
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 713-720
ISBN: 978-989-758-399-5; ISSN: 2184-4356

nent targeted, means of attack, a consequence of the attack and location of the attacker.

Similarly, Herzog et al.(Herzog et al., 2007) put forward an ontology of information security which models assets, threats, vulnerabilities, countermeasures and their relations. This ontology covered general knowledge and can be used as the vocabulary, roadmap, and an extensible dictionary of the domain of information security.

In addition to the ontology built for attack or information security concepts, some ontologies have been based on vulnerabilities. For example, Wang et al. (Wang and Guo, 2009) built an ontology for vulnerability management (OVM), and it has been populated with all vulnerabilities in NVD such as CVE, CWE, CVSS, CAPEC and the relationships among them.

In order to evaluate the security of a system, (Stepanova et al., 2015) proposed a novel penetration testing ontology to achieve semi-automatic knowledge extraction from vast amounts of data, which would yield a holistic view on security analysis. (Gao et al., 2013) proposed a taxonomy which consists of five dimensions including attack impact, attack vector, attack target, vulnerability and defence. They also provided an ontology-based model to assess the security of the system from an attacker's of view.

Most other research models the penetration testing as a planning problem expressed in terms of an attack tree (Ning et al., 2008), attack graph (Kotenko and Doynikova, 2014) or other forms of planning such as PDDL (Planning Domain Definition Language) (Obes et al., 2013).The major problem with this approach is that it is computationally expensive, and this approach faces challenges in decision-making environments where an agent must plan and act in real-time (Bordini et al., 2007). Some research attempted to achieve automated penetration testing by using Markov Decision Processes (MDP) or Partially Observable Markov Decision Processes (POMDP). However, scalability has been reported in (Sarraute et al., 2013) as a primary issue limiting applications of POMDP. To solve these problems, (Moga et al., 2015) presented two patterns of massive type attacks applicable in cyberspace using BDI (belief-desire-intention) agents. Furthermore, (Chu and Lisitsa, 2018) proposed an automated penetration testing approach based on BDI agents model.

Our approach is based on these previous research and builds an ontology from the perspective of penetration testing in practice. Our approach also has the reasoning ability to achieve automation using BDI agent-based architecture.

3 ONTOLOGY DESIGN

An ontology is a formal, explicit specification of a shared conceptualisation (Studer et al., 1998). The main components of ontology are classes, relations, functions, axioms and instances. We can build ontology based on taxonomy in the domain.

3.1 Taxonomy for Penetration Testing

Based on taxonomies of attack in previous work and ATT&CK knowledge base (Strom et al., 2018), we establish an attack taxonomy from the perspective of penetration testing. By penetration testing in practice (Broad and Bindner, 2013), we summarise the attacks into the following categories: information gathering attack, configuration attack, buffer overflow attack, password attack, web attack, sniffer attack, social engineering attack, and denial of service attack.

3.1.1 Information Gathering Attack

Information gathering is the most critical step in penetration testing. Typically, we need to collect information about the target including IP address, opened port, application information, OS information, human or organisation information, network information, defence mechanism, configuration information, vulnerability information, physical environment information and so on. The collection of the above information determines whether the penetration test is successful or not.

3.1.2 Configuration Attack

Such attacks are usually based on an administrator's misconfiguration of the system. For example, the robot.txt file usually exposes the structure information of the website, or the directory that allows users to upload files has executable permission so that attackers can upload a malicious file and execute it.

3.1.3 Buffer Overflow Attack

A buffer overflow is a typical software coding mistake that an attacker could exploit to gain access to the target system. When a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations. It allows attackers to change the program flow and execute their commands or programs. Buffer overflow is a widespread and very dangerous vulnerability, which widely appears in various operating systems and application software. It is a famous attack in penetration testing.

3.1.4 Password Attack

Password attack is an essential part of penetration testing. Usually, an attacker can gain some specific permission from the target system if a password attack is successful. The most commonly used password attack is based on a dictionary, which consists of all possible passwords.

3.1.5 Web Attack

WEB attack is an attack against web applications. The most common attacks include Injection, Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF). The Open Web Application Security Project (OWASP) (Pandya and Patel, 2016) publishes the TOP 10 vulnerabilities every year to raise awareness amongst developers and managers, which becomes the application security standard. However, these vulnerabilities tend to be exploited by attackers.

3.1.6 Sniffer Attack

If there is no vulnerability or the target system is well-protected, a human experienced penetration tester would typically attempt to perform a sniffer attack. Firstly, they break into other systems which are under the same sub-network as the original target, and the tester can monitor and analyse all network flow to gain sensitive information such as a password.

3.1.7 Social Engineer Attack

Social engineer attacks are against humans, such as administrators who have weak security awareness. Social engineering is the term used for a broad range of malicious activities accomplished through human interactions. In remote penetration testing, usually perform Spear-Phishing Attack by email or link, Website forge Attack or Spoofing Attack.

3.1.8 Denial of Service Attack

A denial-of-service (DoS) is an attack where the attackers (hackers) attempt to prevent legitimate users from accessing the service. In a DoS attack, the attacker usually sends excessive data flow to the network or server to exhaust target resources. In penetration testing, a DoS attack is not typical and usually leads to the reboot of the target system for some purposes. These kinds of attacks include SYN flood, TCP/UDP attack, SMTP attack, ICMP attack, and so on. If the attack source comes from a different device, we call it a distributed denial-of-service attack(DDoS attack).

3.2 Our Ontology

According to our proposed taxonomy above, we build an ontology for penetration testing by Protege (Musen et al., 2015) which is a widely-used open-source ontology editor and a knowledge management system. Classes describe concepts in a domain. Object properties represent the relations between instances, data properties represent a property of the instance, restrictions represent axioms, and individuals represent instances of a class. Figure 1 illustrates our ontology for penetration testing.

In the ontology, we build *attackers*, *targets* and *attack methods* classes as top level concepts. *Attackers* class includes a set of attacker instances, such as *attacker1*.

The *targets* class includes a set of target instances, and for the presentation purposes we create *target1* and *target2*. These target instances are described by data properties such as *IP address*, *port*, *OS*, *application* , *configuration* , *vulnerability* and *current permission*. Due to the limited space in this paper, we only present two target instances and indicate their same subnet relation. Through the ontology, we can easily understand the network topology of the target.

The *attack methods* class consists of multiple levels of attack methods, based on our proposed penetration testing taxonomy, such as buffer overflow attack or password attack. Under the class of the attack method, the specific attack actions are defined as instances which include data properties such as *action*, *precondition* and *postcondition*. Similarly, due to the limited space in this paper, we only present one instance which indicates to perform a buffer overflow attack with CVE number MS08-067. Property characteristics and descriptions are used to represent axioms and restrictions in our ontology. For example, the object property *isSameSubnet* is *transitive* and *symmetric*. That is, if target1 and target2 are in the same subnet, target2 and target3 are in the same subnet. Then we can know that target1 and target3 are also in the same subnet. This knowledge usually comes from insights gained during penetration testing.

To describe the relations between instances, we define five object properties below:

- **hasPermission:** represents the attacker has specific current permission in a target.
- **isConnected:** represents the attacker instance can be connected to target instance.
- **isNotConnected** represents the attacker instance cannot be connected to target instance.

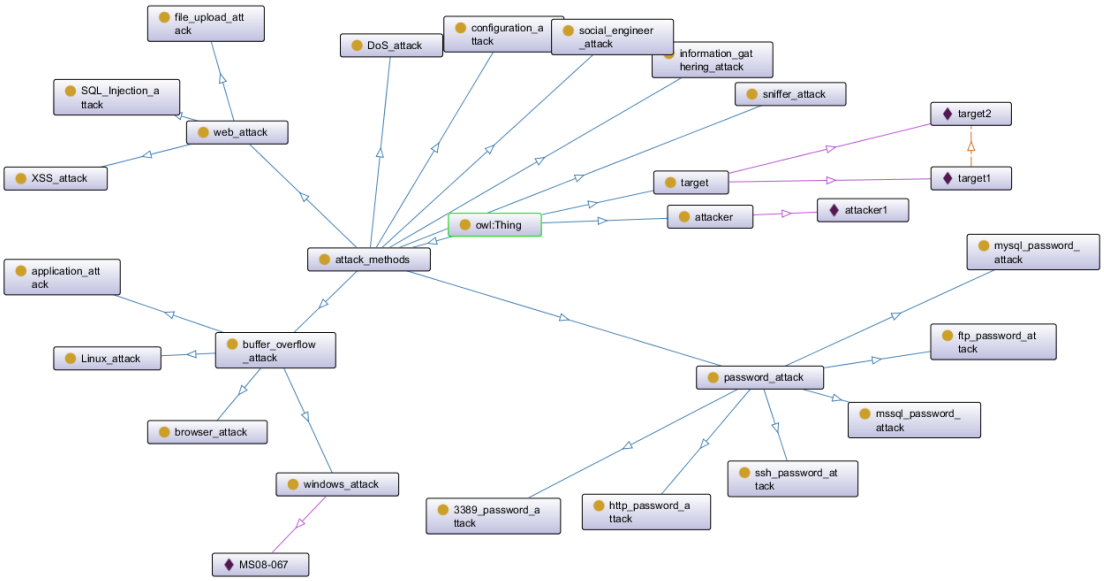

Figure 1: Ontology for penetration testing.

- **isSameSubnet:** represents the target instances which in the same subnet.
- **exploitBy:** represents attacker can perform specific attack.

All attack methods are pre-defined and extensible in our ontology; in other words, the ontology will continue to expand.

4 SWRL AND REASONING

After creating the ontology for penetration testing, we add SWRL rules to implement the inference function. SWRL is regarded not only as a combination of rules and ontology but also as being able to use relationships and vocabulary in an ontology directly. SWRL rules include two parts, body part and head part, which represent precondition and postcondition. The body and head consist of a set of atoms. Informally, a rule can be read as meaning that if the body part is true, then the head part must also be true.

SWRL rules are used to determine the attack actions when certain preconditions are satisfied as well as obtain new knowledge. Therefore, the SWRL rule base is a vital part of making decisions in the automation of penetration testing. We take the buffer overflow vulnerability MS08-067 as an example:

Rule1:
attacker(?attacker)$\bigwedge$ isConnected(?attacker, target1)
$\bigwedge$ vulnerability(target1,"MS08-067")
$\rightarrow$ exploitBy(?attacker,MS08-067)

Rule1 presents the process of an attack on vulnerability MS08-067: when the attacker can connect to the target2, and it has MS08-067 vulnerability, it then performs the actions of an MS08-067 attack. After reasoning by Inference Engine in Protege, a new relation occurs between the attacker instance and the MS08-067 instance, which is represented by object property *exploitBy*. Figure 2 shows how the attacker1 will perform the MS08-067 attack.

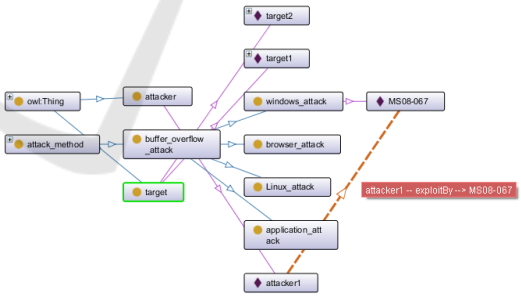

Figure 2: Example for SWRL rule-based reasoning.

SWRL rules can not only make a decision but are also able to produce new knowledge. For example, ontology can discover potential attack paths through a combination of SWRL rules, property characteristics and descriptions. The rule2 presents a penetration testing scenario into the internal network. That is, if the attacker can connect to target2 and successfully gains root permission, then target2 becomes an internal attacker. In this way, we can get the path to reach its final target.

Rule2:
attacker(?attacker)⋀ isConnected(?attacker, target2)
⋀ exploitBy(?attacker, target2) ⋀
currentPermission(target2,"root")
→ attacker(target2)

5 AUTOMATION

Based on our ontology and SWRL rules, we make use of the BDI agent framework to achieve the process of automated penetration testing. In general, ontology is used to store knowledge of targets and attacks, while the BDI agent is used to perform specific attack actions. Owlready2 (Lamy, 2017) is a module for ontology-oriented programming in Python, which is used to interact between ontology and the BDI agent by performing actions such as load, query, create, update classes, instances, properties and perform reasoning. In addition, BDIPython is used to implement the BDI mechanism, which is a Python library used to support BDI-style Programming (Bremner et al., 2019). Our approach includes two knowledge bases, which SWRL rules and BDI plans. The SWRL rules are used to make decisions about an attack by inference in ontology, while plans are used to perform multiple steps of attack in the BDI agent. In fact, only the result of an attack will be updated to the ontology rather than the information generated by the intermediate process of an attack. The advantage of using an SWRL-based ontology knowledge base is that the agent can infer new knowledge rather than relying on pre-defined plans.

5.1 BDI Agent-based Interaction

The BDI agent is one of the classical models of cognitive architecture, which include three essential logic components: belief, desire and intention. The BDI agent is defined as a tuple <Ag, B, D, I, P>,where:

- **Ag:** is an agent name.
- **B:** is a belief set which represents the informational state of the agent. Usually, they refer to environmental information.
- **D** is a desire set which represents the motivational state of the agent. They represent objectives or situations that the agent would like to accomplish or bring about.
- **I:** is an intention set which represents the deliberative state of the agent. They show what the agent has chosen to do.
- **P:** is a plan set which consists of available actions. Plans may include other plans that an agent can perform to achieve one or more of its intentions.

The structure of a plan is shown below:

Trigger Event: context <- body.

The interaction between the BDI agent and the ontology can be illustrated by the following pseudocode. In this example, we first call a Nmap port scanner as a function to get port information from the target. Secondly, the *updateontology* function assigns the result of the port scan into the ontology and performs reasoning to obtain new knowledge. Finally, the *sshattack* function updates the result of ssh attack into the ontology data property if there is an *exploitBy* relation between the instance of the *attacker* and *sshAttack*.

Algorithm 1: Interaction between agent and ontology.

```
function PORTSCAN(ip)
    result ← Nmap(ip)
    return result
end function
function UPDATEONTOLOGY(portscan)
    onto ← ontology.load
    onto.target1.port ← portscan
    syncReasoner
end function
function SSHATTACK(ip, port)
    if onto.attacker.exploitBy = onto.sshAttack
then
        result ← sshAttackRule(ip, port)
    end if
    onto.target1.ssh ← result
end function
```

5.2 Automation Process

(Chu and Lisitsa, 2018) proposed an approach for the automation of penetration testing based on the BDI agent. They proposed using a belief set and plan set to store target information and penetration testing knowledge, respectively. Our approach is based on their BDI agent framework. However, the difference is that target information, and penetration testing knowledge is stored in the ontology. In our approach, target information is stored as instances under target classes with data properties such as IP address, port, OS, application, configuration, vulnerability and current permission. The object property *isSameSubnet* represents the relation between targets. From the prospects of attackers, attack actions are stored as instances under the attacker methods class with data properties such as action, precondition and post-condition. SWRL rules are used to determine the

attack actions and their preference according to target information, in the meantime, updating the information or relations in the ontology. After the ontology determines the specific attack action, the BDI plan will perform the attack. At this point, the belief set in the BDI agent is used to store new information generated during the attack. Finally, the ontology will be updated, including the object property *exploitBy* and the data property *current property* after the attack is finished in the BDI agent by Owlready2. The process of our ontology-based automation of penetration testing is presented in Figure 3:

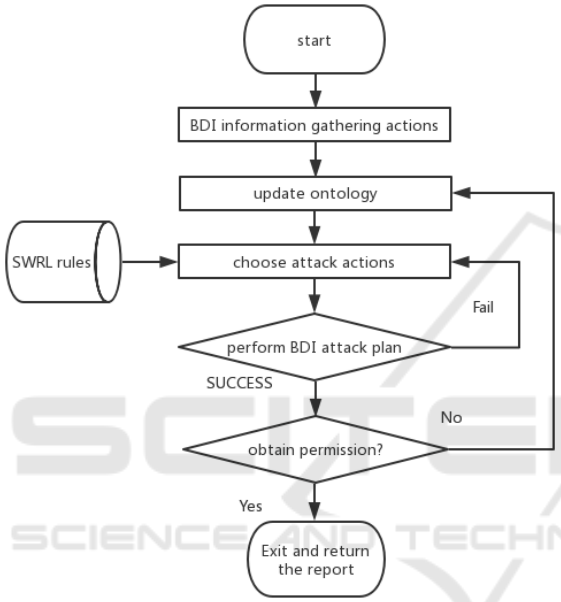

Figure 3: Process of automation of penetration testing.

6 ATTACK SCENARIO

To validate our approach, we present an attack scenario in a virtual environment. Our demo runs on the Kali Linux virtual machine, while the target runs on a WindowsXPSP3 virtual machine. We use Metasploit, a popular penetration testing framework (Kennedy et al., 2011) as the attack action space. In BDIPython, we use functions to define attack actions and plan rules to active the BDI reasoning cycle. The target information is shown below :

- `Ip address: 192.168.1.162`
- `Ports: 135, 139, 445, 3389`
- `Operating system: WindowsXP SP3`
- `Vulnerabilities: MS08067,Weak password`
- `Current permission: None`

In our demo, we pre-define MS08-067 buffer overflow attack and ssh password attack. We can see from Figure 4: firstly, after we input the target IP address and the attacker's IP address, our demo start to probe the target port.

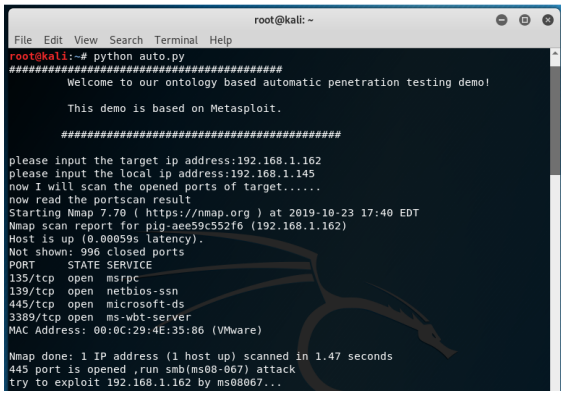

Figure 4: Probe target's port.

Then, according to the port results, there is no ssh port running on the target. Thus, our demo performs MS08-067 attack.

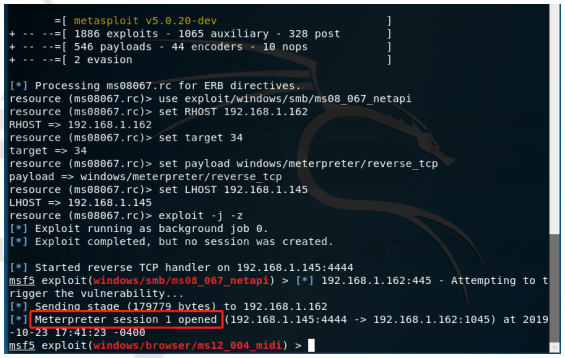

Figure 5: MS08-067 attack.

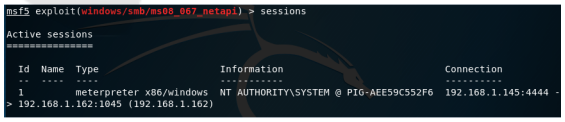

Figure 6: System permission.

Finally, the MS08-067 attack is successful (see Figure 5), and we obtain the highest permission in the target system. Figure 6 shows a connection has been created between the attacker and target machine with system privilege through 4444 port.

We can see the update of data properties in target1 instance and the relation between attacker instance and MS08-067 instance in our ontology(see Figure 7). In target1 data properties, current permission is changed to *system*.

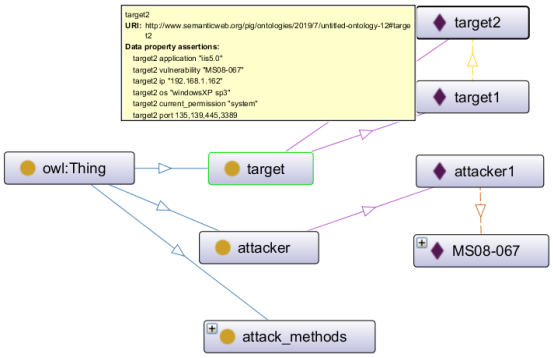

Figure 7: Properties update in ontology.

7 CONCLUSION

In this paper, we proposed an ontology-based automated penetration testing approach. According to our proposed penetration testing attack taxonomy, we built an ontology by Protege. To help make decisions and produce new knowledge, we make use of SWRL rules to create a penetration testing knowledge base as well as to achieve reasoning functions. Finally, we use agent-based(BDI) framework and Owlready2 to achieve the process of automated penetration testing. To validate our approach, we used BDIPython library to implement an attack scenario in a virtual environment. The ontology and BDI agent are extendable, while we are going to find ways to extend it automatically in the future.

REFERENCES

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.

Bremner, P., Dennis, L. A., Fisher, M., and Winfield, A. F. (2019). On proactive, transparent, and verifiable ethical reasoning for robots. *Proceedings of the IEEE*, 107(3):541–561.

Broad, J. and Bindner, A. (2013). *Hacking with Kali: practical penetration testing techniques*. Newnes.

Chu, G. and Lisitsa, A. (2018). Poster: Agent-based (bdi) modeling for automation of penetration testing. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–2. IEEE.

Gao, J.-b., Zhang, B.-w., Chen, X.-h., and Luo, Z. (2013). Ontology-based model of network and computer attacks for security assessment. *Journal of Shanghai Jiaotong University (Science)*, 18(5):554–562.

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5-6):625–640.

Herzog, A., Shahmehri, N., and Duma, C. (2007). An ontology of information security. *International Journal of Information Security and Privacy (IJISP)*, 1(4):1–23.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31.

Kennedy, D., O'gorman, J., Kearns, D., and Aharoni, M. (2011). *Metasploit: the penetration tester's guide*. No Starch Press.

Knight, K., Chander, I., Haines, M., Hatzivassiloglou, V., Hovy, E., Iida, M., Luk, S. K., Whitney, R., and Yamada, K. (1995). Filling knowledge gaps in a broad-coverage machine translation system. *arXiv preprint cmp-lg/9506009*.

Kotenko, I. and Doynikova, E. (2014). Security assessment of computer networks based on attack graphs and security events. In *Information and Communication Technology-EurAsia Conference*, pages 462–471. Springer.

Lamy, J.-B. (2017). Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28.

Lenat, D. B. and Guha, R. V. (1989). *Building large knowledge-based systems; representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Moga, H., Boscoianu, M., Ungureanu, D., Lile, R., and Erginoz, N. (2015). Massive cyber-attacks patterns implemented with bdi agents. In *Applied Mechanics and Materials*, volume 811, pages 383–389. Trans Tech Publ.

Musen, M. A. et al. (2015). The protégé project: a look back and a look forward. *AI matters*, 1(4):4.

Nickerson, C., Kennedy, D., Smith, E., Rabie, A., Friedli, S., Searle, J., Knight, B., Gates, C., and McCray, J. (2014). Penetration testing execution standard.

Ning, Z., Xin-yuan, C., Yong-fu, Z., and Si-yuan, X. (2008). Design and application of penetration attack tree model oriented to attack resistance test. In *2008 International Conference on Computer Science and Software Engineering*, volume 3, pages 622–626. IEEE.

Obes, J. L., Sarraute, C., and Richarte, G. (2013). Attack planning in the real world. *arXiv preprint arXiv:1306.4044*.

Pandya, D. and Patel, N. (2016). Owasp top 10 vulnerability analyses in government websites. *International Journal of Enterprise Computing and Business Systems*, 6(1).

Pinkston, J., Undercoffer, J., Joshi, A., and Finin, T. (2004). A target-centric ontology for intrusion detection. In *In proceeding of the IJCAI-03 Workshop on Ontologies and Distributed Systems. Acapulco, August 9 th*. Citeseer.

Rao, A. S., Georgeff, M. P., et al. (1995). Bdi agents: from theory to practice. In *ICMAS*, volume 95, pages 312–319.

Sarraute, C., Buffet, O., and Hoffmann, J. (2013). Penetration testing== pomdp solving? *arXiv preprint arXiv:1306.4714*.

Shah, S. and Mehtre, B. M. (2015). An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques*, 11(1):27–49.

Stepanova, T., Pechenkin, A., and Lavrova, D. (2015). Ontology-based big data approach to automated penetration testing of large-scale heterogeneous systems. In *Proceedings of the 8th International Conference on Security of Information and Networks*, pages 142–149. ACM.

Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., and Thomas, C. B. (2018). Mitre att&ck: Design and philosophy. *MITRE Product MP*, pages 18–0944.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197.

Wang, J. A. and Guo, M. (2009). Ovm: an ontology for vulnerability management. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 34. ACM.