Ontologies for Vulnerability Terrain Mapping and Attack Reasoning

Anonymous Author(s)

ABSTRACT

The characteristics of cyber-attacks, e.g., nefarious IP addresses and malware hashes, are relatively easy to detect and blacklist. However, to perform a comprehensive analysis, establishing a *context* is required to facilitate system defenders in understanding the manifestation of the vulnerability across the system. Furthermore, the context assists in exploring common attack characteristics among the vulnerabilities. For instance, revealing attack mechanisms prevents attackers from exploiting the same attack mechanism across vulnerabilities. We argue that an ontology-based approach may be followed to help in building context to explore *vulnerability terrain* based on the contextual similarity among the vulnerabilities. To build the context, we extract data from the National Vulnerability Database (NVD) to map the data to the respective ontology class(es), which can be partially automated for usability. We perform reasoning on the ontology to identify different characteristics of the vulnerability terrain. Our results show that recurring actions operated by an attacker in a Cloud environment are associated with manipulating the combination of legitimate Virtual Machine (VM) actions that adversely impact the Cloud functionality.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

threat modelling and analysis, security assessment

ACM Reference Format:

Anonymous Author(s). 2018. Ontologies for Vulnerability Terrain Mapping and Attack Reasoning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Cyber-attacks have two basic actors: an attacker and a defender. An attacker's goal is achieved by targeting system vulnerabilities, design/operational deficiencies, mis-configurations and so on. The defender's objective is to protect the system against attacks. This involves comprehensively understanding when to apply appropriate security countermeasures to successfully mitigate the attacks.

A defender needs to monitor the disclosed vulnerabilities as a proactive measure to formulate and validate threat hypotheses in order to maintain system defenses. However, the increasing number of vulnerabilities, their high rate of discovery coupled with the complexity of vulnerability information requires significant manpower for a detailed analysis. Further, the volume, rate of production, and information complexity of vulnerabilities present difficulties in managing and analyzing the relevant information to protect the system under consideration [8]. This results in a window of opportunity for an attacker to exploit newly disclosed vulnerabilities during the time a defender takes to analyze new vulnerability information [31].

Among the commonly used repositories of vulnerability data is the National Vulnerability Database (NVD) [21]. Other vendor-specific databases such as Microsoft's Security Bulleting [19] or TrendMicro [28] are also available, though their value is limited as they primarily report vulnerabilities pertinent only to their specific products. Such repositories complement vulnerability assessment tools such as Nessus [3], OpenVAS [24] by exposing an external view of the potential attack surface of a system.

These databases are typically organized based on the consequence of the vulnerability being exploited and the effected platform (software or hardware and version). Given this structure, it is difficult to identify potential linkages with other disclosed vulnerabilities so a defender may build up a holistic view of the system's potential attack surface. We define the *potential attack surface* as the **Vulnerability Terrain**. We hold that understanding the vulnerability terrain is of particular importance in complex systems, such as Cloud platforms when evaluating multiple attack hypotheses or scenarios to best allocate defensive resources.

Problem Statement: The challenge for defenders is to holistically understand the Vulnerability terrain of complex systems in such a way as to allow the defender to develop multiple, plausible attack hypotheses to better plan the system defense. The necessary condition is to have a rich interlinking of vulnerability contexts that reveal potential attack options and alternatives open to a malicious agent in the system under consideration. Fulfilling such a condition would support the defenders to identify common attack mechanisms aiding in the diagnosis of attacks. We hypothesize that it is possible to automatically construct a rich, interlinked, vulnerability context space enabling holistic vulnerability terrain analysis and defend decision making.

Approach and Contributions: Our work herein describes the creation of a rich, structured, interlinked vulnerability terrain using a vulnerability context extracted by a semi-automated process based on natural language processing techniques. An ontology is presented based on a systematic review of the vulnerability information held in CVE format used by the NVD. Importantly, the context analysis work identifies two key characteristic classes; *Structural* describing attack surface features and *Behavioural* describing attacker actions.

The collection of the extracted vulnerability context ontologies provides the basis to use a reasoning process to map the collected vulnerability terrain. A key innovation in this reasoning is the introduction of an inference process to find "similar" vulnerabilities based on their context. The approach measures the strength of the linkage using a tuple of grouped characteristics and natural language

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

processing, in this case, a cosine similarity technique [18]. The combination of ontological reasoning, with similarity interlinking, maps the vulnerability terrain based on defender defined attack hypothesis and scenarios.

Our contributions can be summarized as follows.

(1) Development of an ontology-based approach that defines a rich vulnerability context and contextual relationships.
(2) Development of a structured reasoning approach to map the vulnerability "terrain" for a system and attack hypothesis under consideration
(3) A novel natural language processing based approach to provide vulnerability context interlink to support attack hypothesis analysis
(4) Demonstration of the presented techniques in a practical evaluation using an open-source Cloud computing platform.

The remainder of the paper is organized as follows. Section 2 reviews contemporary usage of ontology's in the cyber-security domain and Section 3 details the construction of an ontology-based vulnerability context analysis. Section 4 demonstrate the validation of the approach on Cloud-specific vulnerabilities.

2 RELATED WORK

Vulnerability information can be obtained from a variety of structured and unstructured sources, collectively forming a knowledge base. The research challenge is to make sense of this knowledge base to derive meaningful intelligence that a defender can act upon. A fundamental approach to making sense of the available information is to construct ontology's to define information abstractions and information relationships as a foundation for analysis.

A number of prior works have attempted to interpret multiple knowledge sources to provide a structure for subsequent analysis. In [15], the usefulness of multiple diverse sources, such as the Facebook threat exchange [9], is evaluated as to the efficacy of generating actionable intelligence. Importantly this work defines a series of metrics for characterizing intelligence sources as a mechanism for comparison. The concept of threat intelligence computing is demonstrated in [26] which uses graph computation for mapping threat information and illustrating malicious behavior. Natural language processing techniques have been used to automate the extraction of machine-readable Indicators of Compromise (IoC) from unstructured data sources in [16] to directly support defensive systems (intrusion detection systems for example).

The use of an ontology to structure complex data is a well-understood approach in computer science and has been used for security analysis for a range of purposes. In [27], an ontology is used to elicit and analyze security requirements to predict potential threats arising from the requirements themselves. The ontology defines three dimensions: organization, risk and security, and identifies relationships among these dimensions during the requirement engineering process. In [10], a generic information security ontology is presented supporting a wide range of risk management approaches. Based on an analysis of a NIST publication [11] and German IT security manual [5] high-level entities and relationships in the ontology were developed. Oltramari et al. [1] proposed a multi-layer cyber-security ontology named "CRATELO". CRATELO provides a semantic representation of the cyber-security domain and targets improvements in the situational awareness of security analysts.

Obrst et al. [23] proposed a methodology for creating an ontology using existing ontologies by using them as sub-ontologies. Their approach relies on the usefulness of off-the-shelf schemas, vocabularies, and reports for the acquisition of the domain knowledge and enables them to identify and analyze concepts, hierarchies, and classes' attribute that is used in defining the ontology. In [6], instead of an ontology, the authors use a diamond model to express the relationship between attacker, their capabilities, target infrastructure and the target organization. Undercoffer et al. [29] laid the foundation for transitioning from taxonomies to ontologies for intrusion detection. They presented an ontology capable of intrusion detection via a distributed set of sensors contributing to the ontology in knowledge acquisition enabling the efficient identification of attacks. Using this approach, More et al. [20] proposed to build a knowledge base to reason about different aspects of the captured data to identify threats and vulnerabilities. However, incorporating heterogeneous data into the knowledge base is complex and reasoning on such data is challenging. Work exists that focuses on the use of vulnerability data directly, such as ontologoies developed by Wang and Guo [30], the National Institute of Science and Technology (NIST) [4], and Kanakogi et al. by [14]. They propose an ontology (OVM) to analyze vulnerabilities' cause and impact using NVD's existing vulnerability data. Their ontology also captures the relationships between defined elements from the CVE format such as vulnerability, affected product, and consequence. Similarly, an ontology was developed in [17] to infer attacks and create cyber threat intelligence. They used the same concepts for ontology as introduced in [2].

Although, the existing approaches provide useful insights into the cyber-attacks. However, a key aspect of understanding the correlation among the attacks is still obscure. Therefore, an attacker can utilize a variation of the attack to compromise the system or utilize the same mechanism on a different service/component of the system. Thus, we focus on exploring the linkages among the different vulnerabilities to understand the potential attack landscape by creating a context based on the CVE data. This allows defenders to take informed decisions not only on the primary vulnerabilities but also on the peripheral vulnerabilities.

3 CONSTRUCTING AN ONTOLOGY-BASED VULNERABILITY CONTEXT ANALYSIS

This section outlines the approach taken to form the vulnerability context ontology and details the resultant ontology structure. It concludes with a description of the reasoning process that is used to map the vulnerability terrain for defender analysis.

To develop the analytical and reasoning approaches presented herein, a multi-stage process was undertaken. Initially, CVE data was analyzed to identify features that would give rise to common classes of information to use in the ontology. This forms the basis of the ontology's classes and the relationships defined between them. The ontology forms the basis of the vulnerability context which is then used to analyze and reason about the inter-relationships among the vulnerabilities, thus revealing the complex attack terrain.

To aid the discussions, a typical vulnerability disclosure report is depicted in Table 1. Vulnerabilities are reported to the public through

databases such as NIST's National Vulnerability Database [21] or vendor-specific databases e.g., Microsoft security bulletin [19]. For our analysis, we focus on and utilize the NVD given its being the single largest public data source.

Table 1: An excerpt of a vulnerability disclosure report.

Vulnerability ID	CVE-2016-5363
Date Published	06/17/2016
Type	Denial Of Service/ Bypass a restriction
Summary	The IPTables firewall in OpenStack Neutron before 7.0.4 and 8.0.0 through 8.1.0 allows remote attackers to bypass an intended MAC-spoofing protection mechanism and consequently cause a denial of service or intercept network traffic via (1) a crafted DHCP discovery message or (2) crafted non-IP traffic.
Characteristics	Attack Vector (AV): Network, Attack Complexity (AC): Low Privileges Required (PR): None, User Interaction (UI): None, Scope (S): Unchanged, Confidentiality (C): Low, Integrity (I): None, Availability (A): High
Impact	CVSS Score: 8.2, Impact Score: 4.2, Exploitability Score: 3.9,
Products Affected	Product Type: Application, Vendor, OpenStack, Product Type: Neutron, Version 7.0.0
References	https://security.openstack.org/ossa/OSSA-2016-009.html
Exploit Available	No
Metasploit Module Available	No

The disclosure report is structured as follows:

- **Vulnerability ID**: This field uniquely identifies each vulnerability.
- **Date Published**: The date on which the vulnerability is disclosed to the public.
- **Type**: This represents potential consequence(s) of a vulnerability exploit. For example, a vulnerability belongs to the DoS category if it causes a denial of service.
- **Summary**: The summary describes, in natural language, vulnerability characteristics, exploit mechanism, root cause and preconditions of the exploit. Therefore, the summary is significant to comprehend the vulnerability and identify high-level indicators of compromise.
- **Characteristics**: This field constitutes further vulnerability characteristics such as attack vector, attack complexity, etc. The attack vector identifies a potential attack surface while attack complexity reflects the difficulty level of the exploit.
- **Impact**: The impact score, assigned on a scale from 0 to 10, represents the severity of the vulnerability while the exploitability score indicates the likelihood of the exploit.
- **Products Affected and Product Type:** This field lists products and their versions affected by the vulnerability.
- **References:** Pointers to further information about the vulnerability are listed in the references.
- **Exploit Available:** This field indicates if an exploit is publicly available.

3.1 Defining the Ontology Classes

In the first instance, the creation of ontology classes for a vulnerability seems trivial. However, report attribute heterogeneity and subjectivity of the information in free form fields, such as the summary, presents challenges for information extraction. While there are standards for vulnerability reporting (ISO 29147 [12]) and vulnerability handling (ISO 30111 [13]), even these may be interpreted differently, resulting in a diversity of application. Additionally, the standard CVE attributes are too restrictive to completely capture the rich context needed for subsequent analysis. Further, by developing an abstraction in the ontology classes, away from the CVE attributes, it provides flexibility to provide alternate mappings to different vulnerability data sources.

Upon review of the vulnerability report, two types of features become evident: *Structural* and *Behavioral*. Structural features include the attack surface, vulnerability types, products affected and so on. Therefore, these features reflect the structural composition of a vulnerability. The behavioral features are typically embodied in the vulnerability summary and include mechanisms, issues, tactics. As such this describes the approaches taken by the attacker to exploit the vulnerability. The desired objective is to generate classes from these features to create a coherent representation of a vulnerability and obtain its context.

It is clear that many of the CVE attributes identify features in the data set that can be mapped directly as ontology classes. For instance, the vulnerability attribute 'type' represents the consequence of the vulnerability and hence it is mapped to the 'consequence' class of the ontology. However, diverse information such as attack mechanisms, preconditions, etc., are embodied in the summary attribute of the vulnerability. As typically these are free form text fields, the automated extraction of these characteristics is challenging.

A review of the CVE attributes, and subsequent analytical iterations, gave rise to the following ontology classes and CVE attribute to ontology class mapping. Table 2 gives an overview of the attribute to class mapping along with example instances of the data held in each class As our main intent is to illustrate the process, the class hierarchy is only provided to a secondary level with other levels modeled similarly.

3.1.1 Structural Attributes to Class Mapping. The **Type** attribute is mapped to the *Consequence* class of the ontology. Furthermore, the class has further sub-classes to capture the potential impact on confidentiality, integrity, or availability. These sub-classes contain members with respect to the potential impact. For instance, DoS is a member of the availability subclass while the information leakage belongs to the confidentiality subclass.

This **Product** attribute is mapped to the *Infrastructure* class of the ontology. This class contains an affected list of products and versions. Furthermore, the attacker can exploit issues in the hardware,

Table 2: Extracting classes from the vulnerability report.

Vulnerability Data	Ontology Class	Secondary levels	Instance
Type	Consequence	Confidentiality	Bypass
		Integrity	Bypass
		Availability	DoS
Summary	Vulnerability	–	–
	Precondition	Software Version	v1.2
		Configuration Settings	Service disabled
	Issues	Token Mishandling	–
		Improper Security Policy	–
	Action	Brute Force	Number of requests
		Error Action	Storage
Products	Infrastructure	Hardware	
		Software	–
		Network	DHCP
Characteristics	Attack Expertise	High	–
		Medium	–
		Low	–
	Attack Surface	Remote	–
		Physical	–

software, etc., therefore, the infrastructure has sub-classes to identify the effect on the software, service, hardware, network, etc.

The **Characteristics** attribute reflects further properties of the vulnerability in terms of its severity, complexity, etc. We extract the respective information and create different classes to illustrate these characteristics.

- *Attack Surface:* This class explains whether physical access to the system is required or the vulnerability can be exploited remotely. This can be directly mapped from the characteristics attribute of the report as shown in Table 2.
- *Attacker:* The attacker class is required in our ontology to represent the information associated with the attack. For example, the respective actions/issues correspond to this class as well as attacker expertise to substantiate the attack. Moreover, this class is further used to elaborate on the attack pattern and mechanism to exploit a certain system.

3.1.2 Behavioural Attribute to Class Mapping. The information to populate classes related to the behavioral attributes of the vulnerability is held in the free text **Summary** field, making their automated extraction non-trivial. We utilized natural language techniques to extract data from the vulnerability report to populate ontology class instances. Our approach is as follows. In order to remove superfluous information from the Summary field, we create a vocabulary by removing stop words and generate n-grams sequence illustrating exploit criterion, precondition, and attacker's mechanism. We choose the n-gram model over the Bag-of-Words (BoG) model because the former provides the following benefits. First, the order and sequence are maintained in the n-gram model. Second, the n-gram model can be used to extract the syntactical semantics of textual description. These benefits are critical in creating context representing the characteristics of the vulnerability. Class selection and instance assignment is based on the longest token matching between the class and the information.

The following classes have been identified as deriving from the information held in the summary attribute:

- *Preconditions:* This class captures the necessary prerequisites for a vulnerability to be exploitable. For example, specific software versions or system configurations are required for vulnerability and in the absence of such requirements, vulnerability is not exploitable. Therefore, this class captures preconditions for a vulnerability manifestation.
- *Issues:* The root cause of the exploit can be either an inherent issue in the service/component or an attacker ascertains action(s) to circumvent normal behavior of the service/component. In lieu of this information, we create two disparate classes. The issues class in the ontology exhibits inherent service issues/bugs exploitable by an attacker. For example, using an incorrect precision method for authentication token comparisons leads to bypassing the authentication mechanism. We further categorize the information with respect to the issue specificity, e.g., a subclass 'token mishandling' (cf., Table 2) captures issues related to the token mismanagement.
- *Action:* The class "action" elaborates on the common actions performed by an attacker to exploit a vulnerability and enables the system defenders to postulate the potential threats and consequently, render mitigation mechanism against the common actions. This class complements the issues class. Thus, in this class, the action of the attackers with a primary objective to circumvent the security is focused. For example, a trivial action from an attacker is a brute force attack. Alternatively, an attacker sends crafted messages to check the system's behavior. This class captures the action by the attacker. Furthermore, this class has further sub-classes to identify the action specificity. For instance a subclass (cf., Table 2) 'error action' captures attacker tactics that deal with testing the error handling of a system.
- *Vulnerability:* The reason for creating vulnerability as an independent class is due to its relationship with other (sub)classes e.g., issues, attack surfaces. This is necessary to create a complete context of a vulnerability (cf., 3.2). This is necessary as our objective is to infer further vulnerabilities satisfying a similar context.

3.2 Defining the Ontology Relationships

The previous section detailed the classes inferred from the CVE data. However, building a complete ontology requires establishing relationships among the classes. We define two primary relationships that govern the development of the ontology. These are:

- *is_a:* This relationship defines the relationship between a class and its sub-classes. For instance, the consequence has three sub-classes to indicate whether the consequence of the vulnerability is on confidentiality, integrity, or availability.
- *has:* The "has" relationship defines the association among the classes for making available their respective data to the participating classes. For instance, vulnerability "has" a precondition or an infrastructure "has" an issue.

Figure 1 depicts the classes involved in the ontology and their relationships. For instance, the attacker class relates to the action class and expertise to identify the attack action and the respective expertise required for the attack mechanism. Similarly, vulnerability class connects with different classes to exhibit the respective characteristic of the vulnerability. For instance, the interaction between the vulnerability class and the precondition indicates the corresponding requirements of the vulnerability. The vulnerability class also relates to the action and issues class (through the infrastructure class) to illustrate the vulnerability exploit mechanism concerning the issues in the software and the corresponding action of the attacker. Thus, the ontology serves the objective to create a complete context of a vulnerability reflecting its characteristics.

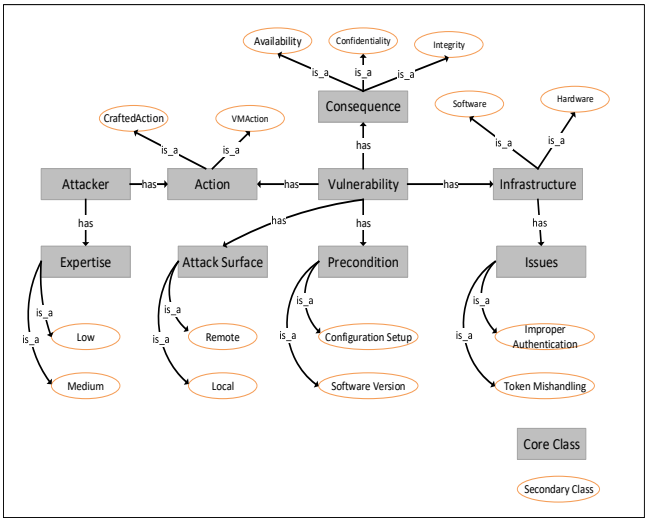

Figure 1: Representing the vulnerability information

Given the completed ontology (classes and relationships) it is possible to derive a structured context for any vulnerability report. An example of the vulnerability context using the ontology classes/subclasses is shown in Figure 2. For instance, attacker_1 is a member of the attacker class that can potentially utilize action_1 from the action class. Furthermore, vul_1 is a member of the vulnerability class and relates with precond_1 from the precondition and has a potential attack_surface_1 from the attack surface class. Moreover, the software_1 is a member of the software class that has the issue exploited by the respective vulnerability.

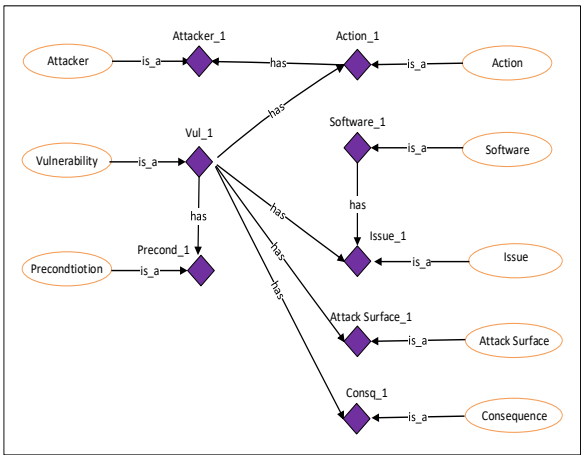

Figure 2: Context of a vulnerability using the Ontology

3.3 Creating a Vulnerability Terrain

Given a data set of vulnerability reports mapped into ontology instances, it is possible to map the given vulnerability terrain. It is assumed here and demonstrated in the next section, an ontology reasoning tool is used to help build the terrain and understand it. To perform the reason a series of properties are defined which help map the vulnerability terrain for a given system under consideration.

Property 1 - Attack mechanism: This property identifies mechanisms that an attacker can utilize to achieve his/her respective goal. In other words, a defender can ascertain the plausible attack mechanisms against the services that an attacker can utilize to exploit a system.

Property 2 - Prevalent issues and actions: This facet of the cyber-attacks entails elaborating on the legacy issues in the service/component exploitable by an attacker. Additionally, recurrent attacker's actions that stipulate the vulnerability manifestation are also identified. The advantage of such an analysis is to support defenders to focus on mitigating attacker's action which consequently patches a set of vulnerabilities employing the same action.

Property 3 - Attack inference: A key objective of this work is to identify potential linkages between different vulnerabilities. This enables the defender to infer further vulnerabilities that may satisfy a similar context. This property forms the basis to evaluate the extent to which vulnerabilities are related to each other. We describe in detail how we extend the reasoning process to provide these linkages in 3.3.1

Property 4 - Multi-stage attack: In this property, we leverage the ontology to identify service(s) with the highest number of associated vulnerabilities and the degree of service's connectivity. This enables to identify possible threat propagation paths by identifying the vulnerable service's interaction with other services.

3.3.1 Linking Vulnerability Contexts. Given a context for a vulnerability defined by an ontological representation, a method needs to be found to build interlinks between the vulnerability contexts. This

method then enables the construction of the vulnerability terrain for subsequent analysis. To perform this comparison construct a tuple consisting of three distinct elements: **structure**, **behavior** and the **consequence**.

- *structure* represents the attributes of the feature set that characterize the structural semantics of the vulnerability.
- *behavior* embodies the exploit mechanism from the summary that represents how the vulnerability can be exploited
- *consequence* defines the impact of the vulnerability such as a denial of service, information leakage, etc.

This provides the basic unit of analysis for comparing the strength of the connections between vulnerabilities. In order to achieve this, we apply the cosine similarity [18] technique to create a similarity matrix showing the distance among the vulnerabilities' analysis tuple. The cosine similarity is widely used to estimate text similarity and since the contexts contain text/strings comparison, therefore cosine similarity is an appropriate choice. It works by evaluating the angular differences between the texts rather than their magnitudes. Thus, texts with the maximum similarity will have a cosine similarity of 1, and the texts that have no commonality will have a cosine similarity value of 0. Since the context is largely textual and thus, cosine similarity is an optimal choice in this scenario. For alternate similarity algorithms, we refer the reader to [7] for more details.

The result of applying cosine similarity is a matrix with values ranging between 0 and 1. A Similarity Matrix (SM) between the vulnerabilities is shown in the matrix. 1.

$$SM = \begin{array}{c} \\ Vul_1 \\ Vul_2 \\ \vdots \\ Vul_n \end{array} \begin{array}{c} \begin{array}{cccc} Vul_1 & Vul_2 & \dots & Vul_n \end{array} \\ \begin{pmatrix} V_{1,1} & V_{1,2} & \dots & V_{1,n} \\ V_{2,1} & V_{2,2} & \dots & V_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n,1} & V_{n,2} & \dots & V_{n,n} \end{pmatrix} \end{array} \quad (1)$$

The cosine similarity ranks the potential variants of a vulnerability based on the similarity among their contexts. However, the analysis can also be utilized to assess the similarity among the vulnerabilities for an individual tuple of the context. For instance, the analysis can aid in providing the similarity among the attack mechanism across a set of vulnerabilities to identify common attack mechanisms or to align the vulnerabilities by their common consequence.

In the next section, we select apply our approach to generate a vulnerability terrain of OpenStack [25], an open-source Cloud computing platform.

4 PRACTICAL VALIDATION USING OPENSTACK

The complete process for the analysis of a vulnerability report data set through to reason is depicted in Figure 3. This process was applied to the OpenStack [25] platform. OpenStack was chosen due to its wide deployment as a cloud environment coupled with the complexity, diversity and volume of services required in an OpenStack installation.

All vulnerabilities relating to OpenStack were extracted from the NVD and initially passed through a semi-automated process as described in Section 3. This produced a series of vulnerability context ontology instances. These instances then formed the basis

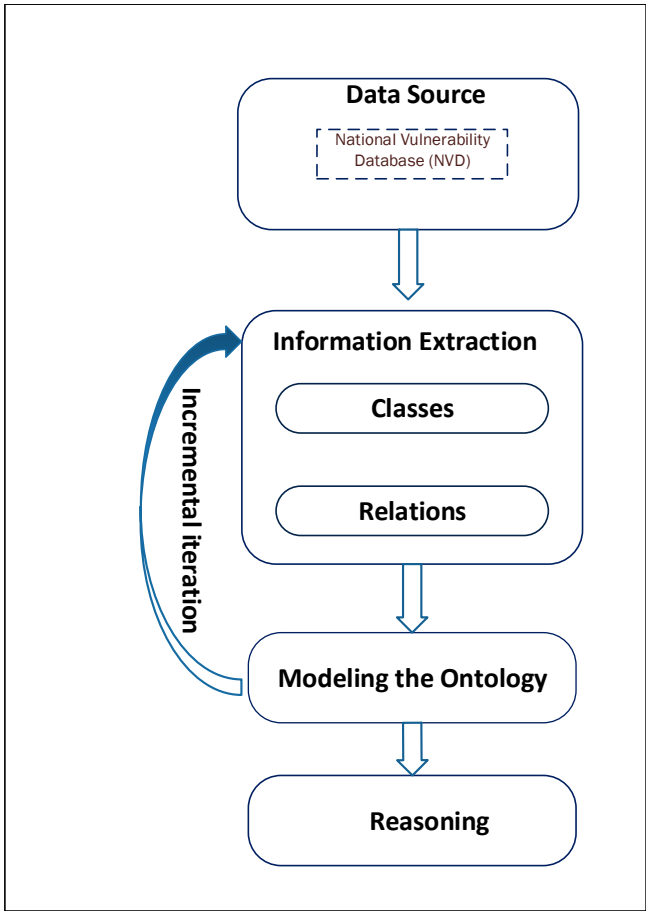

Figure 3: Stages in the approach

for the creation of the vulnerability terrain for further reasoning. We perform reasoning on the ontology data using protégé [22] which is an open-source tool. This is supplemented with our analytical engine for attack inference.

As Figure 3 illustrates, there is an incremental iteration between the ontology and the information extraction process. This is due to the fact that the summary is highly subjective and the same report may be written in a different style depending on the author. Therefore, to ratify the influence of subjectivity, an incremental iteration is performed between the classes and the information to make the ontology classes consistent with the extracted information.

An example of mapping the vulnerability terrain to answer and support the development of attack hypothesis and scenarios is given next.

4.1 Exploring the vulnerability terrain

The objective of the paper is to allow system defenders to hypothesize attack scenarios with respect to the system under investigation. In the following sections, the application of the approach in analyzing different aspects of vulnerability is explored.

4.1.1 Property 1: Attack mechanism. The security defense of a system relies on information that is actionable and leads to revealing patterns, and mechanisms used by the attacker. Therefore, this property explores the methods and techniques that an attacker can utilize to achieve his/her objective considering his/her respective capability. We ran a query for OpenStack's existing vulnerabilities with different constraints, i.e., considering the attacker's objectives what are the possible actions by the attacker. Table 3 shows different objectives and how an attacker can achieve such an objective.

Table 3: Examples of different attack objectives.

Objective	Primary Action	Secondary Action
Accessing Confidential Data	Error Action	Storage Error
		Process Execution Error
		API Error
Preventing Security Policy	Error Action	Overlapping Port ranges
		Unsupported Protocol
Denial of Service	Brute force action	VM Instance
		API requests
		DHCP discovery message

An example from Table 3 is shown graphically in Figure 4 illustrating possible scenarios that lead to accessing the confidential information.

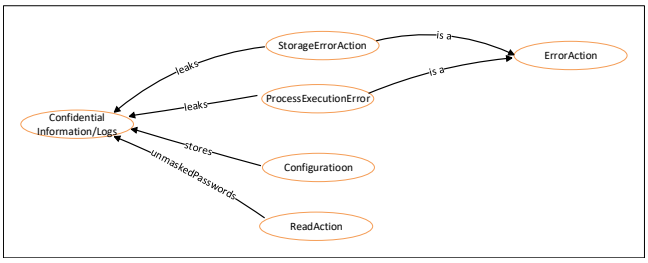

Figure 4: Attack mechanism to access logs

As Figure 4 depicts, a trivial approach would be to directly access the logs and if the service stores passwords unmasked then the password are accessible through the logs. Interestingly, the result shows that causing errors in multiple storage-related processes allows the attacker to access the confidential information provided the error statements are not correctly processed.

The property aims to explore possible methods to access confidential information and therefore, we do not mention the precondition(s) of the methods. The advantage of using such reasoning is twofold. First, these methods can be used as additional security test cases for their respective services. Second, the security analyst learns from these mechanisms and establishes proper mitigation techniques against them. This results in altering the behavior of the attacker and challenges the attacker to find new and innovative methods to achieve his/her goals. For example, fixing the configuration option or inspecting the information entailed in error messages eliminates the possibility of accessing logs and limits the attacker's capacity to steal passwords.

4.1.2 Property 2: Prevalent Issues and Actions. As mentioned in Section 3.1, we classify the root cause into either an inherent service/-component issue or malignant action(s) by the attacker. Therefore, this property answers the root cause of the OpenStack vulnerabilities. A few examples of the root causes and the recurring attacker's actions for the Cloud vulnerabilities are shown in Table 4.

Table 4: Examples of prevalent issues and actions.

Parent	Issue	Consequence
Improper handling	authentication token	Bypass
	token expiration	Bypass/DoS
VM Action	VM migration + VM suspend	DoS
	VM resize + VM delete	DoS

As presumed, the frequent target of attackers is authentication service which is responsible for authenticating and authorizing a user. Our investigation reveals that the legacy issues exploited in the service stem from the mismanagement and improper handling of the authentication tokens and token expiration method. Thus, these issues allow attackers to exploit the service and consequently compromise the system's security. For instance, a wrong precision method in the token expiration comparator allows an attacker to use an expired token for successfully authenticating him/herself. These results are not specific to Cloud and hence, can be utilized to assess an authentication service in other systems such as e-commerce, banking, etc.

On the other hand, the prevalent malignant actions from the attacker emphasize on manipulating the VMs. The VMs manipulations exhibited in the OpenStack vulnerabilities combine different "normal" actions to yield an abnormal action that leads to the system/service crash. For example, a VM resize action followed by a VM delete action crashes the service. Alternatively, suspending a VM while it is migrating to another host also leads to the malfunctioning of the service.

The property reveals legacy issues to provide support in safeguarding the service from these malignant actions by disallowing certain combinations of actions to preserve the proper functioning of the system.

4.1.3 Property 3: Attack Inference. The lack of cross-reference among the vulnerabilities hampers a holistic security assessment of the system under investigation. Therefore, in this property, we infer further vulnerabilities that are semantically equivalent to limit an attacker's ability to exploit a variation of the vulnerability. To explore

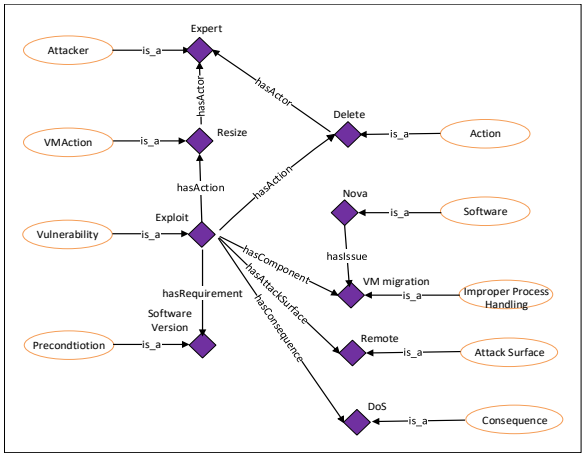

Figure 5: context of a CVE-2015-3241 using the Ontology

vulnerability variants, we create a context of a vulnerability using the ontology and classify vulnerabilities that follows a similar context. The context is a coherent view of the vulnerability's preconditions, attack surface, etc. An example of a vulnerability context is shown in Figure 5.

Figure 5 describes a vulnerability instance using the ontology classes/sub-classes. This particular vulnerability instance requires an attacker to perform multiple actions to cause a denial of service. Therefore, the attacker initially resizes a VM and simultaneously deletes a VM. These actions are represented using the respective classes. The reason for the exploit is due to an issue on the Nova component of the OpenStack for mishandling conflicting instructions by the user. Furthermore. the vulnerability can be exploited remotely.

We evaluated OpenStack to explore vulnerabilities that are contextually equivalent and a potential result is shown in matrix 2.

$$SM = \begin{array}{c} \\ CV...8914 \\ CV...5362 \\ CV...5363 \end{array} \begin{array}{c} \begin{array}{ccc} CV...8914 & CV...5362 & CV...5363 \end{array} \\ \begin{pmatrix} 1 & 0.73 & 0.66 \\ 0.73 & 1 & 0.77 \\ 0.66 & 0.77 & 1 \end{pmatrix} \end{array} \quad (2)$$

The matrix 2 shows that the vulnerabilities are closely related and in the presence of a vulnerability the existence of remaining vulnerabilities should be checked. Thus, this property enables reasoning on a class of vulnerabilities that could also be exploited by the attacker. In this property, we target the vulnerabilities belonging to a single service. However, an attacker can trigger vulnerabilities belonging to different services leading to a multi-stage attack. We tackle this in the next property.

The advantage of such a classification results in revealing further vulnerabilities that have the potential to undermine the security of the system. This leads to patching a class of vulnerabilities instead of individual vulnerability patches.

We evaluate the similarity among OpenStack vulnerabilities in Figure 6. As mentioned before, we use OpenStack as a use case but the methodology is a technology and product independent. The X and Y-axis are data points representing vulnerabilities and the colors represent the extent to which the vulnerabilities are related to each other. As the cosine similarity of 1 represents the maximum similarity, therefore, the diagonal of the heatmap shows this evidence since it represents self-comparison. From Figure 6, it is evident that the vulnerabilities between 42 to 52 [1] have higher similarities. We can further fine-tune the analysis to investigate the similarity among the vulnerabilities in terms of attack attack mechanisms, preconditions, etc. For instance, Figure 7 illustrates the similarity among the vulnerabilities, in terms of the attack mechanism used in the vulnerabilities. This essentially enables the security analysts to better understand the attack mechanism used in multiple vulnerabilities. Consequently, an effective countermeasure against the attack mechanism patches a class of vulnerabilities and thus, limits the reuse of the attack mechanism.

4.1.4 Property 4: Multi-stage attacks. This property targets the capability of an attacker to compromise multiple services with the same vulnerability. In the previous property, we reason on the vulnerabilities that exhibit similar contexts. This property extends the scope by assessing an impact of a vulnerability on multiple services. On the other hand, we can tweak the property to answer the actions that could trigger the transition effect in the system. An example of the transition pattern is shown in Figure 8.

As can be seen from Figure 8, the vulnerability exploit affects two different components (Nova and compute) although the exploit has almost similar actions. Therefore, it is critical to understand the impact of the actions and exploits beyond a single service to understand a multi-stage attack and the propagation of the threat in a system.

In this property, we try to answer possible ways of launching a multi-stage attack. This is critical in understanding how the vulnerability propagates from one service to another. In other words, an attacker can exploit multiple services to causes severe damage to the system. In contrast to exploiting a single service with higher risk, an attacker can exploit multiple services with low risks to cause a higher cumulative impact on the system.

In the next section, we summarise our key findings and discuss the advantages of performing a multi-faceted analysis of the vulnerability.

4.2 Results and Discussion

The application of our approach on the OpenStack vulnerabilities reveal interesting results which can be summarised as follows:

- The initial observation our analysis shows is that there exists a similarity among the vulnerabilities that go beyond the common consequence of the vulnerabilities. We investigated OpenStack vulnerabilities and observe that around 10 percent (cf., Figure 6) of the reported vulnerabilities have a varying degree of correlation between them. Moreover, the degree of correlation is higher among the vulnerabilities exploiting the same service. This is partially due to the deployment of a "similar" attack to exploit the functionality of the service. On the other hand, the similarity between the vulnerabilities exploiting different services shows a lower correlation.
- Similarly, we explore the degree of attacker's reliance on utilizing the same mechanism in different vulnerabilities. Out

[1]The list of vulnerabilities can be accessed from the link

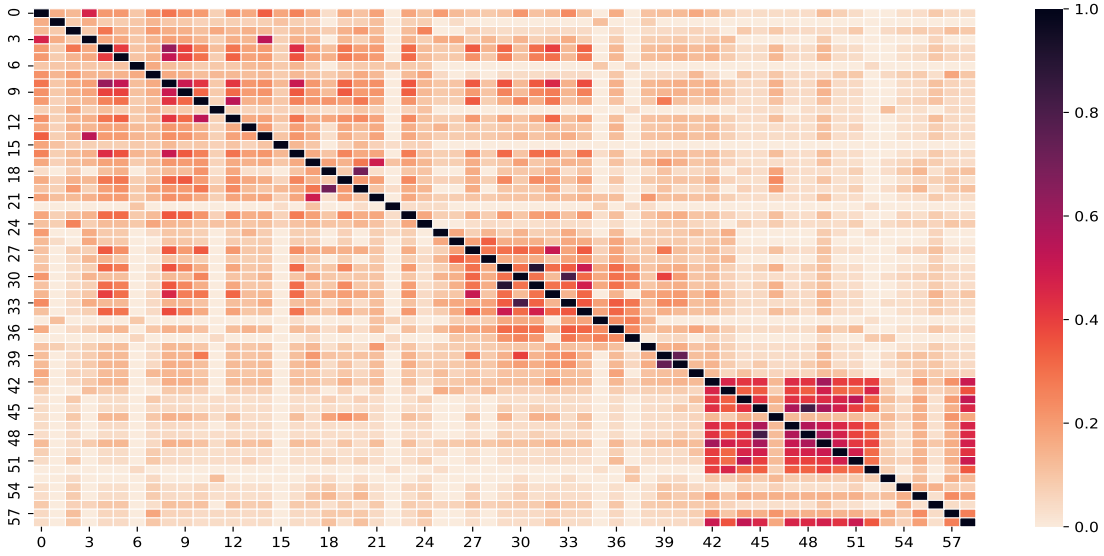

Figure 6: Context-Based Similarity among Cloud vulnerabilities

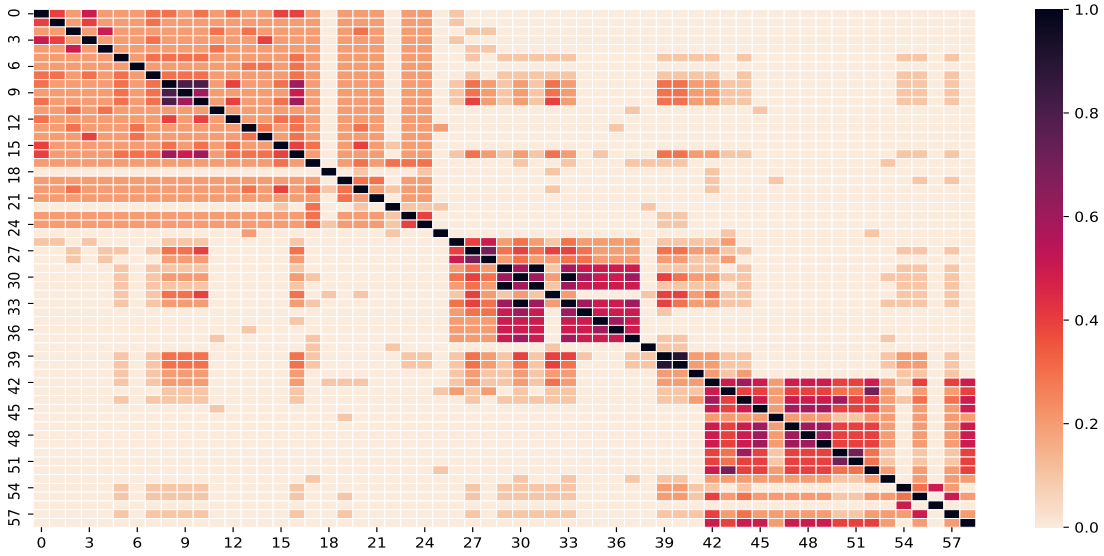

Figure 7: Attack mechanism Similarity among Cloud vulnerabilities

of the vulnerabilities we investigated, around 15 percent of the vulnerabilities have strong similarity (cf., Figure 7), i.e., the cosine similarity is above 0.5. Furthermore, we investigated the applicability of an attack mechanism across different services. The insight into this investigation is similar to the results of contextual similarity. We see a high degree of

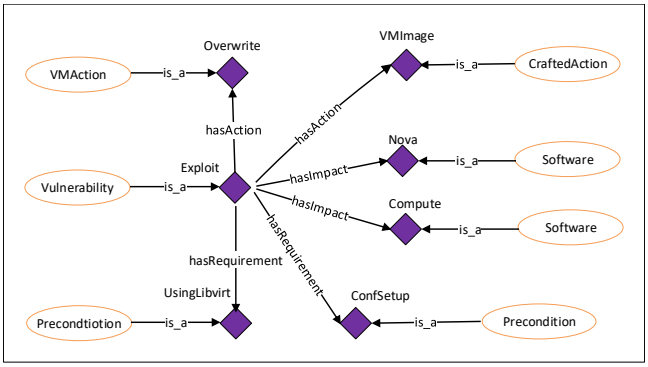

Figure 8: Multi-stage attack

reusability of the attacks for the same service. For instance, there is a high chance of using cross-site scripting attacks with different payloads for authentication service. However, the same attack is less effective on other services in the Cloud platform.

- We also investigate the possible attack surface for both the authenticated and non-authenticated attackers. In the former case, we observe that most of the vulnerabilities or the attacker's actions were related to the manipulations of the VM that could lead to either a denial of service or cause the VM to behave incorrectly. In the latter case, since the attacker is not authenticated therefore, the primary target of the vulnerabilities is the authentication service.

In the following sections, we discuss the broader aspects of our analysis and its limitations.

Fine-grained Vulnerability Classification: We assert that by revealing more elaborate patterns among the vulnerabilities, we allow system defenders to patch subsequent vulnerabilities in the presence of a primary vulnerability. This ensures that the same vulnerability cannot be exploited on a different system or a variant of the vulnerability cannot be used to compromise the system. Furthermore, a system defender can proactively hypothesize different threats that could potentially damage the critical services of the system.

Changing the attacker's behavior: Reusing old exploits is significantly important for an attacker. It speeds up the process of exploiting the system since the attacker does not invest time to create new exploits. Therefore, it is essential to understand common actions in the exploits and provide a safe-guards against the actions to limit their reuse. This is achieved in our approach by identifying common actions from the existing vulnerabilities and enable system defenders to take an informed decision about such actions. This will ensure that an attacker has to find new methods to compromise the system and therefore, enforces the attacker to change his/her behavior for future exploits.

Learning from the vulnerability terrain: Among the roles of the system defenders is to patch each vulnerability and consequently, mitigate the fault in the service/component. This process has an inherent limitation, i.e., this process is reactive and the patch is limited to fixing the reported service/component. On the other hand, learning from a wide spectrum of existing vulnerabilities can lead to proactively assess the security of the system holistically by including the peripheral vulnerabilities in the process. Moreover, learning gives a tactical advantage to the system defenders by allowing them to assess threats across different services. This allows them to learn the progression of vulnerability and also a consequence of an attack mechanism on a different service. For example, a vulnerability (CVE-2015-8914) was reported in 2015 and a year later two new vulnerabilities were reported (CVE-2016-5362 and CVE-2016-5363) with the same attack mechanism applied to a different protocol. Thus, it is critical to understand the attack mechanism and its application to different services to prevent future vulnerabilities. Thus, mitigating attack mechanisms lead to patching multiple vulnerabilities and it also compels the attacker to find new and innovative mechanisms instead of using recurring attack mechanism.

The fundamental objective of this paper is to reveal the correlation among the vulnerabilities. This entailed revealing common attack mechanisms across the vulnerability terrain exhibiting a similar context and establishing a context from the CVE is the basic premise behind the approach. Additionally, property 1 and property 4 demonstrate that the analysis could be extended to hypothesizing different attack mechanisms or explore multi-stage attacks. On the one hand, these properties are limited to the results shown in the paper but on the other hand, these could be extended to cover a wide range of attack hypotheses with respect to the system under investigation.

5 CONCLUSION

We have explored the utility of an ontology-based vulnerability terrain to analyze different facets of the vulnerability. The obtained model constitutes the basis to create the context of the vulnerability by coherently representing it using its CVE data. The context provides a basis to enumerate further vulnerabilities across the multiple operational layers of the system, and consequently, enables the system defenders to make an informed decision on the peripheral vulnerabilities.

Our investigation reveals that there exists a similarity among the vulnerabilities that go beyond the common consequence. For instance, our analysis showed that the correlation among the attack mechanisms exploiting the same service is higher than the attack mechanism exploiting different services. Moreover, an attacker's rely on reusing the "similar" attack mechanism to exploit the same service. Therefore, if the attack mechanism is mitigated then it will compel the attacker to find new and innovative ways to compromise the system which is a challenging task.

REFERENCES

[1] Oltramari Alessandro, Cranor Lorrie, Walls Robert, and McDaniel Patrick. 2014. Building an Ontology of Cyber Security. In *Proceedings of the International Conference on Semantic Technologies for Intelligence, Defense, and Security*. 54–61.

[2] Sean Barnum. 2012. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation* 11 (2012), 1–22.

[3] Jay Beale, Haroon Meer, Charl van der Walt, and Renaud Deraison. 2011. *Nessus Network Auditing: Jay Beale Open Source Security Series*. Elsevier.

[4] Harold Booth and Christopher Turner. 2016. *Vulnerability description ontology (vdo): a framework for characterizing vulnerabilities*. Technical Report. National Institute of Standards and Technology.

[5] BSI. Accessed on: March 01 2020. IT Grundschutz Manual. https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html. [online].

[6] Sergio Caltagirone, Andrew Pendergast, and Christopher Betz. 2013. *The Diamond Model of Intrusion Analysis*. Technical Report. Center For Cyber Intelligence Analysis and Threat Research Hanover Md.

[7] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Gaia Varese. 2011. Ontology and Instance Matching. In *Proceedings of the International Conference on Knowledge-driven Multimedia Information Extraction and Ontology Evolution*. 167–195.

[8] CVE. Accessed On: 01 March 2020. Vulnerability Distribution over the years. https://www.cvedetails.com/browse-by-date.php. [Online].

[9] Facebook. Accessed on: March 01 2020. Facebook Threat Exchange. https://developers.facebook.com/programs/threatexchange.

[10] Stefan Fenz and Andreas Ekelhart. 2009. Formalizing Information Security Knowledge. In *Proceedings of the International Symposium on Information, Computer, and Communications Security*. 183–194.

[11] Barbara Guttman and Edward Roback. 1995. *An introduction to Computer Security: the NIST handbook*. Diane Publishing.

[12] ISO. Accessed on: March 01 2020. ISO Standard 29147. Information Technology – Security Techniques – Vulnerability Disclosure, Edition 2, Year 2018. [online].

[13] ISO. Accessed on: March 01 2020. ISO Standard 30111. Information Technology – Security Techniques – Vulnerability Handling Processes, Edition 3, Year 2019. [online].

[14] Kenta Kanakogi, Hironori Washizaki, Yoshiaki Fukazawa, Shinpei Ogata, Takao Okubo, Takehisa Kato, Hideyuki Kanuka, Atsuo Hazeyama, and Nobukazu Yoshioka. 2022. Comparative Evaluation of NLP-Based Approaches for Linking CAPEC Attack Patterns from CVE Vulnerability Information. *Applied Sciences* 12, 7 (2022), 3400.

[15] Vector Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey Voelker, and Stefan Savage. 2019. Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence. In *Proceedings of the Usenix Security Symposium*. 851–867.

[16] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. 2016. Acing the IoC Game: Toward Automatic Discovery and Analysis of Open-source Cyber Threat Intelligence. In *Proceedings of the Conference on Computer and Communications Security*. 755–766.

[17] Yazid Merah and Tayeb Kenaza. 2021. Proactive Ontology-based Cyber Threat Intelligence Analytic. In *2021 International Conference on Recent Advances in Mathematics and Informatics (ICRAMI)*. IEEE, 1–7.

[18] Steinbach Michael, Karypis George, and Vipin Kumar. 2000. A Comparison of Document Clustering Techniques. In *Proceedsings of the International Conference on Knowledge Discovery and Data Mining*.

[19] Microsoft. Accessed on: March 01 2020. Microsoft Security Bulleting (MSB). https://technet.microsoft.com/en-us/security/bulletins.aspx. [online].

[20] Sumit More, Mary Matthews, Anupam Joshi, and Tim Finin. 2012. A knowledge-based Approach to Intrusion Detection Modeling. In *Proceedings of the Symposium on Security and Privacy workshops*. 75–81.

[21] NIST. Accessed on: March 01 2020. National Vulnerability Database (NVD). https://nvd.nist.gov/. [online].

[22] Natalya Noy, Monica Crubézy, Ray Fergerson, Holger Knublauch, Samson Tu, Jennifer Vendetti, and Mark Musen. 2003. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment.. In *Proceedings of the American Medical Informatics Association Symposium*. 953–953.

[23] Leo Obrst, Penny Chase, and Richard Markeloff. 2012. Developing an Ontology of the Cyber Security Domain.. In *Proceedings of the Conference on Semantic Technologies for Intelligence, Defense, and Security*. 49–56.

[24] OpenVas. Accessed on: March 01 2020. Open Vulnerability Scanner Assessment System. http://www.openvas.org/. [online].

[25] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. 2012. OpenStack: Toward an Open-source Solution for Cloud Computing. *Proceedings of the International Journal of Computer Applications*, 38–42.

[26] Xiaokui Shu, Frederico Araujo, Douglas Schales, Marc Stoecklin, Jiyong Jang, Heqing Huang, and Josyula Rao. 2018. Threat Intelligence Computing. In *Proceedings of the Conference on Computer and Communications Security*. 1883–1898.

[27] Amina Souag, Camille Salinesi, Raúl Mazo, and Isabelle Comyn-Wattiau. 2015. A Security Ontology for Security Requirements Elicitation. In *Proceedings of the International Symposium on Engineering Secure Software and Systems*. 157–177.

[28] TrendMicro. Accessed on: March 01 2020. TrendMicro threat encyclopedia. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/vulnerability/all-vulnerabilities. [online].

[29] Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. 2003. Modeling Computer Attacks: An Ontology for Intrusion Detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. 113–135.

[30] Ju Wang and Minzhe Guo. 2009. OVM: an Ontology for Vulnerability Management. In *Proceedings of the Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. 34–40.

[31] Su Zhang, Xinwen Zhang, and Xinming Ou. 2014. After we knew it: Empirical Study and Modeling of Cost-effectiveness of Exploiting Prevalent known Vulnerabilities across IaaS Cloud. In *Proceedings of the Symposium on Information, Computer and Communications Security*. 317–328.