
Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Computers in Human Behavior 139 (2023) 107520

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

# An ontology-driven framework for knowledge representation of digital extortion attacks

Masoudeh Keshavarzi, Hamid Reza Ghaffary *

*Department of Computer Engineering, Islamic Azad University of Ferdows Branch, Iran*

ARTICLE INFO

*Keywords:*
Ransomware
Cyber-ontology
Conceptual modeling
Knowledge base
Knowledge graph
Philosophy of computer science

ABSTRACT

With the COVID-19 pandemic and the growing influence of the Internet in critical sectors of industry and society, cyberattacks have not only not declined, but have risen sharply. In the meantime, ransomware is at the forefront of the most devastating threats that have launched the lucrative illegal business. Due to the proliferation and variety of ransomware forays, there is a need for a new theory of categories. The intricacy and multiplicity of components involved in digital extortions entails the construction of a knowledge representation system that is able to organize large volumes of information from heterogeneous sources in a formal structured format and infer new knowledge from it. This paper suggests and develops a dedicated ontology of digital blackmails, called Rantology, with a particular focus on ransomware assaults. The logic coded in this ontology allows to assess the maliciousness of programs based on various factors, including called API functions and their behaviors. The proposed framework can be used to facilitate interoperability between cybersecurity experts and knowledge-based systems, and identify sensitive points for surveillance. The evaluation results based on several criteria confirm the adequacy of the suggested ontology in terms of clarity, modularity, consistency, coverage and inheritance richness.

## 1. Introduction

Malware has always been a weapon in the hands of cybercriminals. Ransomware is one of the ghastliest malware, designed with the mindset of extorting from users by blocking their access to data or other computing resources. It is actually a data or resource hijacking in the cyber world analogous to kidnapping in the real-world. Since 2012, almost two decades after the emergence of AIDS Trojan as the first sample, ransomware variants have become more complex and destructive. At the end of 2016, the Justice Department reported an average of more than 4000 ransomware attacks per day since January 1, 2016, a 300 percent increase over 2015 (U.S. Government, 2016). Although extortion-based threats in 2018 seemed lower on the identified cyber risk scale due to the rise of cryptocurrency mining malware, it was only a turning point for ransomware that led to the "Big Game Hunting", a phenomenon aimed at generating higher revenue at lower attack volume (Frankoff & Hartley, 2018; Osena, 2018). Despite a slight slowdown in the growth trend of the number of ransomware assaults in late 2019, since the beginning of 2020, these cyber blackmails have once again made headlines by aggressing high-profile targets (Frankoff & Hartley, 2018; Freed, 2021; Johnson, 2014; Logan et al., 2021; Osena, 2018). An example of this is the Energias de Portugal (EDP)- one of the largest European energy sector operators-being hit by the Ragnar Locker in April 2020, in which the attackers claimed to have stolen 10 TB of the company's sensitive information and demanded an exorbitant payment of 1580 bitcoins (approximately more than $ 10 million) to free up resources (Kaspersky, 2018).

A report by Cybersecurity Ventures predicted that the damage caused by ransomware attacks would reach $ 20 billion by 2021, up from $ 325 million in 2015 (Morgan, 2020). In the same report, the costs imposed by this cyber threat in 2018 and 2019 were estimated at $ 8 billion and $ 11.5 billion, respectively. These costs are not limited to the ransom, but also include other ancillary expenses such as data recovery, downtime, and so on. Fig. 1 shows the estimated financial loss from blackmail attacks along with the approximate number of ransomware onslaughts derived from (Johnson, 2014). However, the actual number of attacks and the costs incurred will certainly be higher, as many individuals and organizations may not report the offence for a variety of reasons. As can be seen from Fig. 1, despite the decline in the number of attacks from their peak in 2016, the damage rate has increased

* Corresponding author.
*E-mail addresses:* Masoudeh_k@yahoo.com (M. Keshavarzi), keshavarzighafari@gmail.com (H.R. Ghaffary).

https://doi.org/10.1016/j.chb.2022.107520
Received 6 February 2022; Received in revised form 2 October 2022; Accepted 7 October 2022
Available online 13 October 2022

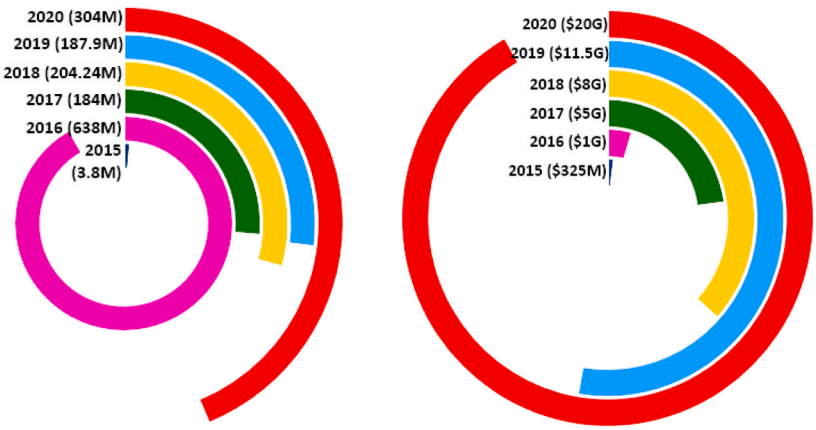
dramatically. This uptrend indicates that ransomware forays have become more targeted and are seeking larger prey for higher extortion instead of trapping the general public. Over the past two years, the advent of extortionist malware families such as LockerGoga, Doppel-Paymer, Sodinokibi, NetWalker, Maze, Ryuk, MegaCortex, Conti, Nephilim, ProLock, and CLOP is proof of this concept, reflecting the fact that ransomware invasions have not only not subsided, but have shifted to larger, more sophisticated attacks aimed at crippling giant corporate entities (Cimpanu, 2020; Frankoff & Hartley, 2018; O'Brien et al., 2019; Santos, 2021; Trend Micro Research, 2022; Walter, 2020). Moreover, recently released security reports show no indication that these extortionate threats will disappear in 2022 (Trellix., 2022).

Given the proliferation of digital extortion attacks in terms of diversity, volume and intricacy, having a defense strategy and countering this lucrative criminal business is of great importance. Despite efforts made to improve security solutions, ransomware remains a problem. Early antivirus and anti-malware software were signature-based and identified malware using a single metadata entity, such as a file hash. Such approaches were extremely weak against obfuscation and encryption techniques and could easily be bypassed by malware developers. Following this issue, behavioral solutions as well as heuristic methods and machine learning have been proposed. These systems operate by running malware instances in controlled environments and capturing their behavioral patterns. Current antiviruses utilize various detection methods and operate based on multiple engines that are activated according to the type of scan and the requested context (Botacin et al., 2022). A comprehensive longitudinal analysis of these security solutions is presented in (Botacin et al., 2020), which evaluates them based on six metrics proposed by the authors from different aspects. The issue that arises in these recent approaches is how to display behaviors and also to select the best distinguishing behavioral features. This problem is exacerbated by ransomware species because of the nature and similarity of their functionalities to benign software, which deals mainly with files.

Several research studies have been conducted to detect, prevent and classify ransomware families based on static and dynamic analysis, each of which considers different aspects to circumvent such incursions (Ahmed et al., 2020; Akbanov et al., 2019; Al-rimy et al., 2019; Andronio et al., 2015; Chen & Bridges, 2017; Cimitile et al., 2018; Continella et al., 2016; Gómez-Hernández et al., 2018; Hampton et al., 2018; Homayoun et al., 2017; Maiorca et al., 2017; Mehnaz et al., 2018; Morato et al., 2018; Scalas et al., 2019; Xiaofeng et al., 2019; Xu et al., 2017; Zhang et al., 2019). However, despite the high importance of this cyber resource hijacking, no common knowledge base of extortionate malware, especially ransomware, is available. Selecting the most critical parts for monitoring necessitates a cognitive insight into the attack behavior and the interactions between the program and the system components. For this purpose, it is crucial to extract prominent terms in the domain and determine the relationships among them. The development of an ontology for extortion attacks can provide a terminology for extortionist software, system components, and other elements in the chain of such cyber threats. In fact, it defines a common glossary for researchers who need to share information in a particular domain (Noy & McGuinness, 2001). Indeed, formally defined semantics will enable detailed searches and complex queries (Obrst et al., 2012).

"A branch of metaphysics that deals with the nature of being" can be considered a primitive definition of an ontology. At present, ontology has many applications in scientific disciplines and multiple definitions of it are available. According to the several explanations of the ontology, this paper uses the following definitions. An ontology is a representation of the types of entities and concepts (also known as classes) within a given domain and the relationships (known as properties) between them. In other words, ontology is a hierarchy of taxonomic classes and the associations among them. The hierarchy is defined by a single root concept, referred to as Thing to which other classes are related. These concepts and sub-concepts are related to each other by the relation "is-a" or "kind-of". Ontology has been successful in various sciences, especially biology, and has been attracting the attention of cybersecurity researchers for several years. Although ontologies have been proposed in the field of cyber security and malware, none of them can be effective in the scope of extortion onsets and cover eminent terms in this area.

Given the similar functionality of many extortionist malware, especially ransomware that blocks access to resources, with benign software used for encryption, compression, batch renaming, and data wiping, it is important to determine the distinguishing features. This can be achieved through a large knowledge base of such programs and their interactions with system components, which will lead to the inference of new knowledge about the similarities and differences between them. By using an ontology-based similarity measure and reasoning on the knowledge base, it is possible to discover ransomware samples that are semantically similar to the ones of interest. To the best of our knowledge, none of the available ontologies present an exhaustive set of concepts in cyber extortion attacks. In this study, we develop an ontology of ransomware attacks and other components involved in the field of digital blackmail, and utilize its ability to take advantage of the relative stability of natural language to enhance interoperability across heterogeneous data systems. This ontology can then be used as a basis for some applications in defense systems and to gain cognitive insights from different species and families. As a matter of fact, this paper is part of a larger research project, and for the first time applies the concept of




**a) Estimated number of ransomware attacks** **b) The amount of damages caused by ransomware**

**Fig. 1.** Approximate statistics of the number of cyber extortion attacks and the financial costs imposed by them.

ontology in the context of digital extortion attacks to build a knowledge base of ransomware and provide a framework for conceptual modeling of such threats. The acquired knowledge must be machine-readable and reusable. Using this knowledge, not only can suspicious ransomware-related behaviors be inferred and distinguished from normal behaviors, but also ransomware threat intelligence can be achieved through relationships and dependencies between elements involved in an extortionate aggression. Overall, the contributions of the paper can be summarized as follows:

- Design and implementation of a dedicated ontology for digital extortion attacks with a specific focus on Windows ransomware
- Creation of ransomware knowledge base by populating the proposed ontology with individuals
- Evaluation of the suggested ontology in terms of clarity, modularity, consistency, coverage and inheritance richness.

The remainder of the paper is organized as follows. Section 2 reviews related work. Design and implementation of Rantology, the proposed ontology of ransomware attacks, is expounded in Section 3. This section begins by stating the assumptions. Subsections 3.1 and 3.2 deal with determining the building blocks of Rantology and how to determine the hierarchy of classes. Subsection 3.3 addresses design details. In subsection 3.4, the issue of keeping the ontology consistent as the domain evolves, as well as design considerations, are discussed. Subsection 3.5 deals with the process of creating a knowledge base of digital extortion attacks. The details of the data set used are provided in this section. The proposed framework is evaluated based on several criteria in Section 4. Section 5 is dedicated to the discussion and expression of existing challenges and elaborate the impact of human behavior in cyber security and its application in ontology. Finally, the paper concludes in Section 6.

## 2. Related work

One of the areas that has begun to receive a lot of attention from academia and industry is the concept of establishing an ontology in the field of cybersecurity. Ontology is an increasingly predominant strategy for organizing scientific information about the world in a computer-interpretable form (Arp et al., 2015). In other words, the ontology is a window on reality and a representation of the types of entities in a domain of interest and the relationships between them. It is a computable lexicon that is used in many fields including knowledge engineering, artificial intelligence, semantic web, information security and many more. This paper aims to conceptualize an ontological representation of digital extortion attacks, which is a specific domain of cybersecurity. Therefore, despite the efforts and studies conducted in the field of identifying and preventing ransomware threats, this section only reviews research work related to cyber security ontologies, especially extortionate malware.

Several ontologies have been proposed for conceptual modeling of attacks and providing knowledge of cyber security. An almost preliminary research paper on the use of ontologies in cybersecurity was conducted by Undercoffer et al. (Undercoffer et al., 2003), in which they produced an ontology for modeling computer attacks and intrusion detection systems (IDSs) in a descriptive logic language. The researchers utilized DAML + OIL to implement their proposed ontology and applied DAMLJessKB as the reasoning system. Huang et al. (Huang et al., 2010) presented an ontology for malware behavioral analysis. Based on that, they designed a platform called TWMAN, which consisted of three layers: knowledge, communication and application. Their initial work was limited to the four main classes Malware_Type, Malware_Impact_Target, Malware_Behavioral and Malware_Sample. Although the authors later continued their research and in the next work their ontology was divided into three classes Malware_Type, File_Type and Malware_Behavioral (Huang et al., 2014), but there were still ambiguities in the defined concepts, and on the other hand, due to the time of presentation, many of the current malware were not included. Another study was done by Obrst et al. (Obrst et al., 2012) in which an ontology of the cyber security domain based on the Diamond model was discussed to describe malicious activity. The authors divide the ontologies into three categories according to the level of abstraction: upper, mid-level and domain. They claim that cyber ontology encompasses some concepts such as Time, Geospatial, or Person that go beyond cybersecurity and can be derived from mid-level ontologies called utility. Their ultimate goal is to integrate data from a variety of resources and reuse existing ontologies. Gao et al. (Gao et al., 2013) developed an ontology-based attack model to evaluate system security and the effect of offenses. To do this, they first proposed a taxonomy of attacks containing five dimensions: Attack impact, Attack vector, Attack target, Vulnerability, and Defense. Then they designed their own ontology based on the suggested taxonomy as well as some existing ontologies. CRATELO is a three-level ontology for cyberspace, designed to improve the situational awareness of cyber defenders (Oltramari et al., 2014). It consists of a domain ontology of cyber operations called OSCO, a security-related mid-level ontology called SECCO, and a foundational or top-level ontology called DOLCE. USpam was proposed by Shoaib and Farooq as a spam detection system that applied an ontology to model users' interests based on their profile (Shoaib & Farooq, 2015). An ontology-based system for predicting and classifying web application attacks based on their severity was suggested in (Salini & Shenbagam, 2015), in which the three classes Threat, Vulnerability, and Attack formed the building blocks of the proposed ontology. Another effort to integrate cybersecurity concepts and topics into an overall ontology belongs to Iannacone et al. (Iannacone et al., 2015). Their developed ontology, STUCCO, included data from 13 structured sources in a variety of formats. The purpose of designing this ontology was to facilitate the integration of data from structured and unstructured sources into a knowledge graph and to organize information. Unified cybersecurity ontology (UCO) was designed to support information integration and cyber situational awareness (Syed et al., 2016). It is mapped to a number of existing cybersecurity ontologies as well as concepts in the Linked Open Data cloud. UCO is an extension of Undercoffer et al.'s intrusion detection system ontology, and most definitions of classes and relationships are based on Structured Threat Information eXpression (STIX). The goal of UCO developers is to provide a common understanding of the cyber security domain and to unify the most widely used security standards.

Navarro et al. (Navarro et al., 2018) propose an ontology-based framework for modeling relationships between Android applications and system elements, and use a machine learning approach to analyze this complex network of relations and dependencies. The authors' goal is to analyze which permissions and resources provided in the manifest files are related to malicious apps. Hence, they focus on the manifest files of apps as a source of information to extract the proposal ontology's terms, and do not seek to find resources and relationships that are directly involved in an attack itself. Narayanan et al. (Narayanan et al., 2018) describe the design of a framework called cognitive cybersecurity (CCS) to collect and ingest information derived from textual resources and host and network based sensors that use the power of semantically rich knowledge representation to assist security analysts. However, the authors do not present a new ontology, and their graph utilizes only terms from an extended version of the UCO, and therefore does not include all the components involved in extortion attacks. IoTSec, proposed by Mozzaquatro et al. (Mozzaquatro et al., 2018), is an ontology consisting of 228 classes, 24 object properties and 7 data properties designed to address the security aspects of the Internet of Things (IoT) within industrial environments. Jia et al. (Jia et al., 2018) introduced a cybersecurity knowledge base and detection rules based on a quintuple model. Assets, Vulnerability and Attack are the three main entities of the ontology presented by them. The Assets class consists of two subclasses, Software and OS. The researchers applied the Stanford named entity recognizer (NER) to extract cybersecurity-related entities. Ding et al.

(Ding et al., 2019) also conducted a study on the use of ontologies for knowledge representation of malware and their families. However, the prototype model developed by them included only a limited number of malware-related classes. Rastogi et al. (Rastogi et al., 2020) introduced MALOnt, an ontology for malware threat intelligence, in order to extract structured information and generate knowledge graph. The authors stated that they instantiated the knowledge graph with hundreds of annotated threat reports. The CyberTwitter framework for analyzing tweets about cybersecurity vulnerabilities and issuing timely threat alerts was explained by Mittal et al. (Mittal et al., 2016). The designers link the entities extracted from the tweets to real-world concepts using UCO ontology and publicly available knowledge bases, including DBpedia and YAGO, and then store this data in the resource description framework (RDF) triples. In another similar research effort, a domain ontology called cybersecurity vulnerability ontology (CVO) was presented (Syed, 2020), which is a formal knowledge representation of the realm of vulnerability management. In addition to the concepts in NIST, CERT/CC and CVSS, CVO considers vulnerability information extracted from Twitter, and eventually uses it to design the cyber intelligence alert (CIA) system to issue cyber alerts about vulnerabilities and countermeasures. The APTMalInsight framework is proposed using system call information and ontology knowledge to detect APT malware (Han et al., 2021). The ontology model offered in APTMalInsight consists of three core classes: APT malware, System component and Behavior. The developers of APTMalInsight believe that this knowledge framework realizes the systematic mapping of APT malware behaviors and ultimately enables the recognition of APT attacks. Finally, the attention to the MITRE ATT&CK[1] also deserves to be considered, which is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. Over the past two decades, MITRE has provided standard languages and formats to capture cyber security information, including Common Attack Pattern Enumeration and Classification (CAPEC), Malware Attribute Enumeration and Characterization (MAEC), the Common Vulnerabilities and Exposures (CVE), the Cyber Observable eXpression (CybOX) and many more. The D3FEND framework (Kaloroumakis & Smith, 2021), which has received a positive feedback from MITRE experts, is in fact a precise semantic model of cyber security countermeasures. The authors show how this knowledge graph supports queries that can inferentially map cyber security countermeasures to offensive tactics, techniques, and procedures (TTPs).

The ontologies presented in the field of malware and computer systems security only cover the general concepts of cyber security and lack the context of digital extortion threats. To the best of our knowledge, there is currently no framework that models digital extortion onslaughts and displays them in the form of knowledge graphs. On the other hand, due to the irreversible nature of destructive ransomware attacks, the process of analyzing them is a tedious and costly task. Therefore, collecting and integrating reports related to extortion offenses and the results of ransomware analysis from various heterogeneous sources can play a significant role in facilitating the study of this type of notorious cyber threat. In this study, we intend to fill this gap by designing a framework to conceptualize the scope of extortionist malware. For this purpose, we propose Rantology, which is an ontology in the field of fear-based attacks focusing on ransomware. The main purpose of this paper, which is actually part of a larger research work, is to develop an ontology of the digital extortion domain, expressed in the Web ontology language (OWL), which enables the integration of data from heterogeneous sources. As part of the W3C's Semantic Web technology stack, OWL is a computational logic-based language that, thank to richer vocabulary, enables and facilitates the processing of information contents and the assessment of knowledge compatibility by providing greater machine-interpretable capabilities than that supported by the extensible markup language (XML), RDF, and RDF Schema (RDFS) (Arp et al., 2015; W3C, 2021).

[1] https://attack.mitre.org/.

## 3. Design of digital extortion ontology

A domain ontology is a structured controlled representation of entities pertinent to a domain of discourse that organizes data to make it comprehensible, accessible, and computer-analyzable (Arp et al., 2015). While taxonomy is a representational artifact that is organized hierarchically with nodes representing classes and edges that delineate a "is-a" relation, an ontology goes beyond it and encompasses connections other than the simple "is-a" relation. In this sense, ontologies allow the inference of new knowledge and the extraction of more complex relationships. As mentioned in the previous section, despite the rich literature on the use of ontologies in cybersecurity, there is no ransomware-specific ontology designed to build a knowledge base of this thriving criminal trade. This paper proposes a dedicated ontology, called Rantology, for nefarious extortionist malware, designed to explore the interactions of ransomware and benign software with the system and to perceive the relationships between system API calls and the behaviors resulting from the execution of these binaries.

Before dealing with the design of Rantology, it is important to mention some assumptions used. As suggested by Botacin et al. (Botacin et al., 2021), it is appropriate that the design phase of a security solution consists of reasoning about its design aspects, such as the definition of a threat model, assumptions, target platform, and so on. A threat model specifies which, why and how resources will be protected (Botacin et al., 2022). The ultimate goal of our research work is to design an ontology and then produce a knowledge base for digital extortion-based onslaughts. We separate these threats into three categories: Rogueware, Ransomware, and Leakware, and in this paper, we focus on the development of this ontology for the second group, specifically Windows ransomware. Rantology is a domain ontology that incorporates concepts related to the scope of cyber extortion attacks. It contains all the entities associated with fear-based malware and digital blackmailers, various types of ransomware, system calls, ransom and payment methods, cyber threat actors, and other agents involved in this realm. Although Rantology consists of several sections, in order to achieve the purpose of this study, we focus more on Windows ransomware and legal software with similar functionalities and their associations to the system API calls and behaviors. One of the important assumptions in our work is that the most characteristic feature of ransomware, which distinguishes it from other general malware, is to notify the victim of its presence. This behavior is usually accompanied by showing a ransom note to the user. In addition, ransomware has the behavior of denying access to resources, which can be data or non-data. The use of these constraints to define ransomware class is described in detail in subsection 3.3. Of course, it is important to note that not all ransomware are designed with monetary intents in mind, and many of them, such as RanRan (Falcone & Grunzweig, 2017), have non-monetary goals. However, in both cases, they demand a ransom from the victim, which in the second case will not necessarily be financial payment. Also, this type of cyber threat targets a wide range of home users, organizations and large businesses, so we put them in a main class of Rantology. Peoples, whether individual victims, organizations and even software developers use different tools and software that can be deduced with the help of defined relationships to what extent these programs increase the attack surface or which industrial sectors are more vulnerable to these forays or which group of cybercriminals carry out more targeted attacks. In this research, we distinguish between software developers and attackers and place them in two disjoint classes. We assume that a software developer only develops a goodware, whereas an attacker produces a malware or uses specific threat components in cyber attacks. Although one might think that a software developer might be an attacker, this assumption is necessary to organize some classes and relationships in Rantology. A summary of the adopted assumptions is given in Table 1.

Various methodologies have been proposed for designing ontologies

**Table 1**
Assumptions adopted in the design of Rantology.

| Assumption | Type |
|---|---|
| Currently, there are three general categories of malware that include known extortion threats. | general assumption |
| In general, all ransomware, regardless of the family they belong to, must go through the same attack chain to achieve success and obtain the ransom from the victim. | general assumption |
| Ransomware, especially those that encrypt files or overwrite them with junk data, exhibit similar behaviors to legitimate software, especially in the categories of archivers, compressors, and the like. | general assumption |
| Two prominent features of ransomware are denying users legal access to resources and notifying them of the attack in order to demand a ransom. | general assumption |
| Ransomware victims are a wide range of users and organizations, but some ransomware campaigns tend to target specific sectors of industry and society. | general assumption |
| A software developer develops a goodware. | special assumption |
| A malware is produced by an attacker. | special assumption |

(Arp et al., 2015; Noy & McGuinness, 2001; Uschold & Gruninger, 1996). In this paper, an iterative approach is applied to develop the Rantology. This seven-step approach includes determining the scope of the ontology, reviewing existing ontologies for reuse, enumerating important terms in the domain, defining classes and their hierarchy, identifying features and relationships between classes, setting restrictions on properties, and finally creating instances. After these steps, the initial ontology will need to be refined, and this repetitive process will continue throughout the lifecycle of the ontology design. Fig. 2 illustrates the seven steps used in this study.

Requirement analysis is one of the vital stages in any design process. Ontology engineering is no exception to this rule. Competency questions identify the knowledge that should be entailed in the ontology. Asking competency questions is part of the requirements analysis phase for developing an ontology. In fact, the scope of an ontology is outlined by these natural language questions. Although there is no specific standard for designing such questions, it is desirable that competency questions be posed in a stratified way. These informal questions should be designed so that they can be converted into formal queries, for example in the SPARQL form, and used for the initial evaluation of the ontology (Uschold & Gruninger, 1996; Wiśniewski et al., 2019). The scope of the ontology presented in this study is digital extortion attacks. To demarcate the domain or the subject matter of the proffered ontology, some competency questions are asked so that the knowledge base built on this ontology is able to answer them. Competency questions are actually user-oriented inquiries that only determine the scope of the ontology, and considering that the audience of Rantology can be both ordinary users and security experts, we have included some prior knowledge to design them. These questions are just a sketch of the ontology and do not need to be comprehensive (Noy & McGuinness, 2001). Table 2 synopsizes the competency questions planned for Rantology.

**Table 2**
Rantology's competency questions (CQs).

| CQ number | Competency Question |
|---|---|
| CQ1 | What is the set of behaviors that qualify a sample as a member of ransomware? |
| CQ2 | What is the set of descriptive behaviors for a ransomware category X? |
| CQ3 | What is the set of behaviors that make a sample eligible for membership in a ransomware family X? |
| CQ4 | Which ransomware samples have doxing capability? |
| CQ5 | Which crypto ransomware samples use symmetric cryptographic algorithms? |
| CQ6 | Which ransomware species manipulate the master boot record (MBR)? |
| CQ7 | Is there a decryptor for a ransomware family X and if so, from which URL can it be downloaded? |
| CQ8 | Is a ransomware X available as Ransomware-as-a-Service (RaaS)? |
| CQ9 | What is the difference between the versions of a ransomware family X? |
| CQ10 | Which ransomware families are cross-platform? |
| CQ11 | What software changes a key/value X in the registry? |
| CQ12 | Which ransomware samples delete the volume shadow copies? |
| CQ13 | What API functions are related to manipulating or deleting volume shadow copy service (VSS)? |
| CQ14 | What common actions do instances that have the ability to delete VSS perform in the registry? |
| CQ15 | What system calls characterize a behavior X? |
| CQ16 | Which system components get similar effects from ransomware and benign software? |
| CQ17 | Which system components are only influenced by ransomware? |
| CQ18 | What is the relationship between an encryption algorithm X and system calls? |
| CQ19 | What behavior in the system are suspicious and should be monitored? |
| CQ20 | Which crypto ransomware samples require a connection to a C&C server to perform encryption? |
| CQ21 | Vulnerability X in which ransomware families have been exploited? |
| CQ22 | Which ransomware instances use a threat tool X? |
| CQ23 | What is the relationship between the operating system version and the attack tools used? |
| CQ24 | What is the relationship between vulnerabilities and ransomware release year? |
| CQ25 | Which ransomware samples charge ransom in a payment system X? |
| CQ26 | Which ransomware instances are associated with a Bitcoin address X? |
| CQ27 | Which ransomware families only target organizations? |
| CQ28 | Which organizations are more vulnerable to extortion offenses? |
| CQ29 | What threat tools does an attacker X utilize? |
| CQ30 | Which person or group is responsible for an extortion attack X? |
| CQ31 | Is it possible to achieve a ransomware threat intelligence through conceptual modeling? |

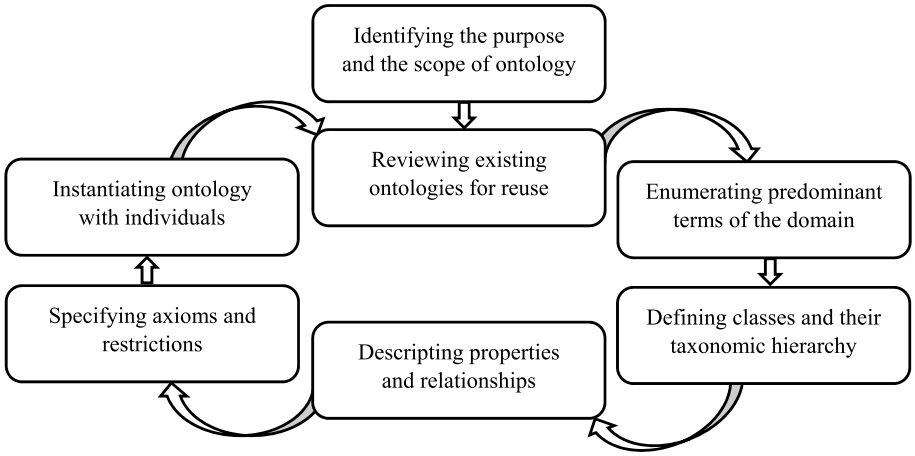



**Fig. 2.** Ontology design methodology.

After delimiting the scope of the ontology, it is time to consider the related ontologies in this domain. For this purpose, it should be checked whether the competency questions presented in Table 2 have already been solved by the existing ontologies. We searched for ransomware ontologies to refine and expand them (if any). To the best of our knowledge, no related ontology has been provided for extortion assaults and their relationship to system behaviors and components that can answer the aforementioned competency queries. Given that our goal is different from the ontologies presented in software (Hilario et al., 2009; Keet et al., 2015; Malone et al., 2014; Oberle et al., 2009), cybersecurity (Gao et al., 2013; Huang et al., 2010, 2014; Iannacone et al., 2015; Jia et al., 2018; Mozzaquatro et al., 2018; Narayanan et al., 2018; Navarro et al., 2018; Oltramari et al., 2014; Rastogi et al., 2020; Salini & Shenbagam, 2015; Shoaib & Farooq, 2015; Syed et al., 2016; Undercoffer et al., 2003), and vulnerability management (Mittal et al., 2016; Syed, 2020), we start developing the ontology from scratch. Although there were slight overlaps in some of the concepts and specifications between the proposed ontology and the research work mentioned, due to the small number, we manually merged them into the Rantology.

### 3.1. Determining the constituent elements of Rantology

The main building blocks of an OWL ontology are classes. As a matter of fact, classes are some of the most prominent terms in the domain of interest. Elicitation of notable terms requires that raw data be collected and analyzed in the target area. To understand and derive the terms of the domain, we studied a wide range of research papers related to ransomware attacks as well as existing threat reports in this regard (a number of them are cited throughout the paper, although there are many more security reports reviewed than those cited). After identifying the most salient concepts based on a comprehensive survey of the literature as well as competency questions posed in the previous phase, another important step in designing an ontology is determining the hierarchy of classes and ascertaining the properties and relationships. During the process of modeling and designing the Rantology, we tried to define the main classes and their hierarchy in such a way that they are not too nested with a large number of subclasses and not too flat with very few subclasses with a lot of information encoded in the properties. To make the paper more readable, we write the names of the classes and properties defined in Rantology in italic type.

According to the concepts extracted in the scope of cyber extortion attacks and the application of this ontology, the terms *Software*, *Behavior*, *OperatingSystemAPI*, *PaymentSystem, CyberActor* and *ThreatComponent* are defined as the main classes. Core classes are located at the root level of the hierarchy, which are defined as subclasses of the most general class, Thing, and do not necessarily have to be similar concepts. They actually depict the major parts of the scope of cyber extortion forays. All of these core classes are disjointed, except for the *Software* and *ThreatComponent*. Because many malware, such as droppers, are engaged in digital blackmail attacks. In addition, the attackers are abusing legitimate tools to facilitate their incursions. These strategies of exploiting legal programs, known as living off the land (LotL), are common in many ransomware families, including WastedLocker.

The goal of Rantology is to identify and classify suspicious behaviors related to ransomware and to comprehend the dependencies between system API calls and those behaviors. Ransomware is a malicious software that demands ransom in exchange for the emancipation of hostage resources by impeding users from accessing the computing device or data. The *Software* class is a general concept that can surround ransomware or any benign software with similar functionalities to this malware, including file encryption tools, compressors, file shredders, convertors, batch renaming programs, and so on. Given that this paper focuses on ransomware targeting Microsoft Windows platforms, we will omit the definition of a separate platform class and instead consider a slot called *hasPlatform* for the *Software* class. The cardinality of this feature is set to multiple, as some ransomware strains and even legitimate software may be cross-platform. On the other hand, a program may run on different architectures and versions of Windows operating system. Ransom32, for example, is a multi-platform ransomware written in JavaScript that affects Windows, Linux, and Mac OS X. Also, the Ryuk ransomware family targets both 32-bit and 64-bit Windows operating systems. According to the mission of this research to investigate Windows ransomware, the primary programming environment for such software programs is the Windows application programming interface (API), which provides an exhaustive set of functions for managing processes, threads, memory, and peripherals (Silberschatz et al., 2018). Windows NT is a modified microkernel architecture and instead of supporting one operating system API, it implements several operating environment subsystems in user mode that provide special APIs to the client applications. The most widely used API of NT is Win32, which most software written for the Windows platform utilize the functions provided in its client-side dynamic-link libraries (DLLs). We define the main *OperatingSystemAPI* class to extend Rantology so that it can later cover other ransomware families that target platforms other than Windows. Nevertheless, we need a more absolute concept in this work. Therefore, we define the *WindowsAPI* class (subclass of *OperatingSystemAPI*), which includes several sub-concepts that cover the various functions of Windows API and pertaining system call services. Also, instead of assigning a class to system components, we define the *Behavior* class so that the suggested ontology can be easily expanded to support extortionist malware targeting non-Windows platforms, which may have fundamental structural differences. In this way, for example, any file modifying behavior can be defined regardless of the file system structure. Another example to justify such a design is registry behaviors. The registry is a hierarchical database in Microsoft Windows operating systems that maintains a set of configuration settings and critical data for Windows operations, applications, and services. Such a system component with this name does not exist in other platforms such as Linux, Mac OS and Android.

There is a rational for designing the other three core classes. Digital extortion attacks similar to other good or evil cyber activities have actors such as developer, attacker, and victim (the last two are the two main vertices of the Diamond model (Obrst et al., 2012)). Accordingly, we define a new entity called *CyberActor*, which is a participant (person or group) in an action or process using computers, devices, systems, or networks. It includes the *Person* and *Organization* sub-concepts that can take on the roles of victim, attacker, software developer, and even Ransomware-as-a-Service (RaaS) operators and their affiliate. For this purpose, each of these roles is specified as defined subclasses of *CyberActor* through axioms. Also, a concept called *ThreatGroup* is defined as a subclass of *Attacker* that contains information about cybercriminal groups. On the other hand, extortionist malware must go through a six-step attack chain to succeed (Keshavarzi & Ghaffary, 2020). This chain consists of infection, installation, communication, execution, extortion and emancipation. Various components such as spam emails, botnets, droppers, vulnerabilities or exploit kits in the infection phase may be used by ransomware campaigns. Cybercriminals also use a variety of tools and tactics to achieve their goals in other stages of the attack, including command and control (C&C or C2) servers, all of which can be incorporated into an entity called *ThreatComponent*. Unlike many types of malware, digital extortion threats inform victims and ask them to follow instructions to get rid of the attack. In fact, the extortion phase is one of the most important steps in the chain of ransomware offenses. Samples designed for monetary purposes utilize a variety of payment methods, mainly through cryptocurrencies such as Bitcoin, to collect ransom from the victims. This step is so important that several studies have been done on it (Hernandez-Castro et al., 2017, 2020; Laszka et al., 2017; Paquet-Clouston et al., 2019; Sokolov, 2021). Therefore, allocating a separate class for this purpose can lead to the production of knowledge of financial transactions related to extortionate attacks. An overview of Rantology is shown in Fig. 3. Since ransomware is one of the key concepts that Rantology focuses on, Fig. 3 is based on this object in

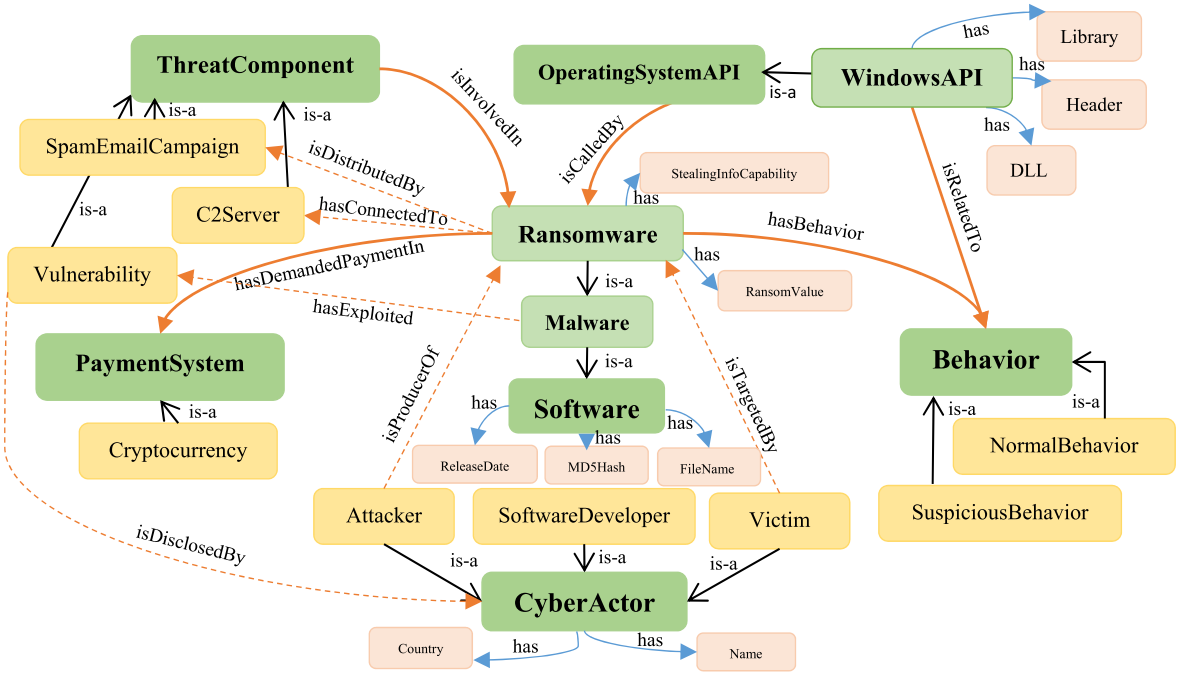



**Fig. 3.** A high-level view of Rantology.

which the relationships of ransomware with other components are described in an aggressive extortion threat. Some classes are derived from other super-concepts, which will be elucidated in the following. However, many concepts, properties and relationships are not delineated in the figure.

### 3.2. Hierarchizing the classes

Each of the designated main classes must be arranged in a hierarchy of more and less general ones. There are several approaches to define a class hierarchy (Noy & McGuinness, 2001; Obrst et al., 2012). Due to the fact that the ontology presented in this study, Rantology, consists of elements involved in digital extortion attacks, a middle-out (or combined) methodology has been used to design it. We utilize a top-down approach for the *Software* and *CyberActor* class hierarchies. In the case of the *Software* class, we begin by creating the general concepts of software, namely *Goodware* and *Malware*. This may differ from the types provided in some of the literature in which the software embraces application software, system software and programming tools. But, many benign programs that behave similarly to ransomware are in the category of utility software or specialized applications. After all, this paper only looks at programs and malware that run on the Microsoft Windows platform. Therefore, according to the purpose of the presented ontology, we do not separate system software from application. The *Goodware* class can embody application software as well as subsets of system software, including utility programs. The data properties of the *Software* class are defined in such a way that this ontology is modular and can be easily expanded depending on the application, for example by adding the *ProgrammingLanguage* and *OperatingSystem* subclasses. Efforts have been made to apply this approach to all classes throughout the ontology design process.

The *Malware* concept comprises subclasses depending on the hierarchy specified for it. Each of these malicious programs, in turn, can contain subcategories that describe concepts that are more specific to their superclass. There are several taxonomies of malware and its types that classify these noxious software programs in different ways (Al-rimy et al., 2018; Bajpai et al., 2018; Grégio et al., 2015; Keshavarzi & Ghaffary, 2020; Luo & Liao, 2007; Qamar et al., 2019). Given that the main focus of this ontology is digital extortion attacks, we use the taxonomy offered in (Keshavarzi & Ghaffary, 2020) to cover all modern samples of ransomware and other extortion-based malware. Creating the *Scareware* class and its subclasses, *Rogueware*, *Leakware* (aka Doxware) and *Ransomware*, instead of listing all of these malignant software as direct subclasses of the *Malware*, reflects different types of extortionate and fear-based attacks in a more organized way. The most distinctive feature of ransomware from other kinds of scareware is the confiscation of valuable data and device owned by the user and the denial of access to them. This feature, along with the characteristic of having a ransom note, is defined by axioms and quantifier restrictions for the *Ransomware* class. Fig. 4 illustrates a part of the hierarchy and different levels of granularity for the *Software* concept. The *Goodware* and *Malware* classes include many more sub-concepts, which are not shown in the figure for brevity.

All siblings in the hierarchy, except those at the root level, are at the same level of granularity. Here, *Software*, *Goodware*, and *Malware* are the most general concepts in the Software hierarchy, while *CryptoRansomware*, *WiperRansomware*, and *LockerRansomware* are the most specific ones. Although these bottom-level concepts also contain subclasses that belong to different families of ransomware. Therefore, ransomware families in this hierarchy are defined at the lowest level as disjoint classes, where each sample with a unique hash is an individual of the respective class.

Ransomware species, like any software program, require access to certain operating system services to perform their tasks. These extortionist malware, especially those belonging to the Crypto and Wiper-Ransomware categories, need actions such as creating and managing new processes, creating and deleting files, listing directories, reading and writing to files, establishing network connections for communication, and much more. System calls provide a programmatic way in which a computer program can invoke such services from the operating system kernel. Computer programs mainly utilize APIs to access these services. The choice of API term for the class name is too general and may be confusing, as web APIs are now the most common meaning of the term API. In this research, API is actually an interface between an application and the operating system. So to clear up the ambiguity in Rantology, we use the term *OperatingSystemAPI*. This can be pursued by further specializing this concept to encompass POSIX APIs in future work, in addition to the Windows APIs.

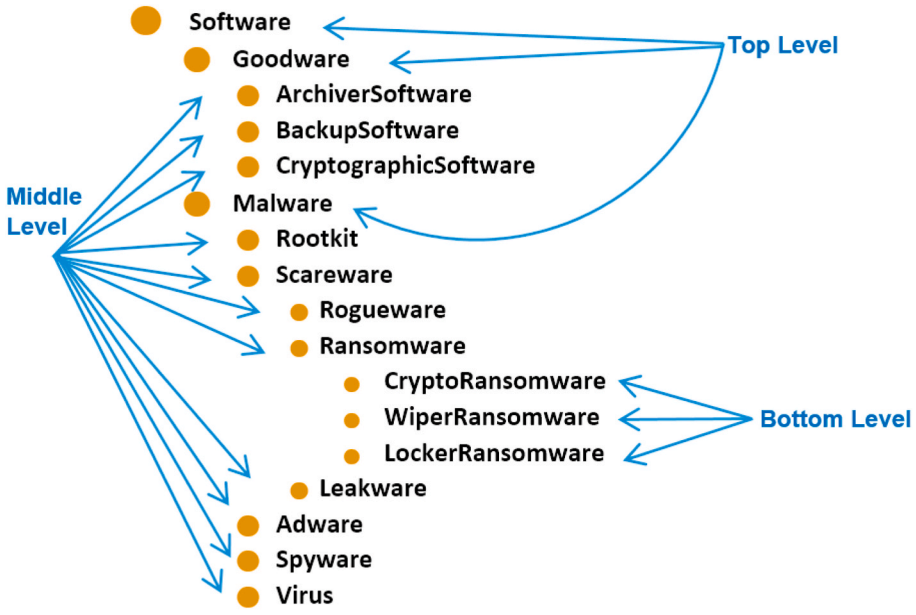



**Fig. 4.** Part of the Software taxonomic class hierarchy in Rantology.

In much literature, six major categories are considered for system calls (Silberschatz et al., 2018): process control, file management, device management, information maintenance, communication, and protection. Nevertheless, API calls are classified in several ways, including the version or services embedded in the respective DLLs. Due to the multitude of API functions as well as the lack of a clear categorization, we utilize a hybrid approach to the *WindowsAPI* class hierarchy. In this way, the most salient concepts are first defined and then appropriately generalized or specialized (Noy & McGuinness, 2001). Because of the strong correlation between a function in the API and its dependent system call within the kernel, we apply the aforesaid classification. So the most general concepts are identified, but given the multiplicity of functions, we analyze 108 ransomware strains and 84 benign programs and then locate the extracted API calls in the corresponding classes. To better organize the functions, some of these subclasses are subdivided into more detailed categories according to Microsoft documentation. Table 3 presents part of the hierarchy specified for *WindowsAPI*, along with some functions that are instances of each class. These subclasses are not disjoint due to the overlap of different categories. For example, the *GetProcessInformation* function is in both the *ProcessControl* and *InformationMaintenance* classes. Also, many of the asserted subclasses are not listed in the table for succinctness. For instance, many functions that accept the string as a parameter may contain the suffix "A" or "W" in their name. An example of such a function can be seen in the first row of Table 3, in which the two functions *CreateProcessA* and *CreateProcessW* have a superclass called *CreateProcessFunction*.

**Table 3**
Part of the WindowsAPI class hierarchy with some representative individuals.

| Class name | Subclass name | Some of individuals |
|---|---|---|
| ProcessControl | ProcessThreadFunction | CreateProcessA, CreateProcessW, ExitProcess, ExitThread, |
| | ObjectFunction | DuplicateHandle |
| | SynchronizationFunction | WaitForSingleObject, WaitForMultipleObjectsEx |
| FileManagement | DataAccessFunction | CreateFileA, DeleteFileA, FlushFileBuffers |
| | BackupFunction | IsVolumeSnapshotted |
| DeviceManagement | PowerFunction | GetSystemPowerStatus |
| | MemoryFunction | GlobalLock, GlobalReAlloc, VirtualAllocEx |
| | ConsoleFunction | ReadConsole |
| InformationMaintenance | RegistryFunction | RegCreateKeyA, RegEnumValueA |
| | TimerFunction | SetTimer |
| | SysInfoFunction | GetTickCount |
| Communication | NetworkFunction | NetApiBufferSize |
| Protection | SecurityFunction | AdjustTokenGroups, DuplicateToken, SetUserObjectSecurity |

Since each of the suspicious behaviors has specific implications to their relationship to other objects in the ontology, we define them as separate classes (accommodated in the general *Behavior* concept) and not as properties. Because in fact, the purpose of Rantology, in addition to classifying and separating ransomware samples, is to create a knowledge representation system of their behavior and interactions with system components and a deeper understanding of the functionalities of the APIs extracted from them to infer new knowledge. In the case of the *Behavior* class, we used a bottom-up approach. The two main sources for extracting behaviors are analyzing available samples and grabbing information and threat reports about extortionate assaults from reputable security websites. To relate the reports in practice, 108 ransomware samples were analyzed and the results were compared with this technical literature. In this way, the individuals that are actually events or actions that occurred during the execution of ransomware samples are first identified. These behaviors then together form higher-level classes and eventually constitute the more general *Behavior* class.

For example, the behavior of adding an entry to the registry Run key was observed in species belonging to many ransomware families, including Avaddon, BTCWare, Ryuk, and BitPaymer. However, some of the analyzed instances imported their own entry into the "HKLM \SOFTWARE\Microsoft\Windows\CurrentVersion\Run" path and some added it to the "HKCU\Software\Microsoft\Windows\CurrentVersion \Run". These are the two individuals that make up the *RegSettingValue* class. The *RegSettingValue* concept, along with the *RegSettingKey*, form the larger *CreatingReg* class. Likewise, the *CreatingReg* class, together with the *DeletingReg*, *ReadingReg*, and *QueryingReg* classes, build the more general *RegistryBehavior* class, which will be a subclass of the *Behavior*. Other subclasses of *Behavior* are also defined by analyzing samples, and are generalized to more general concepts. In addition to the *RegistryBehavior*, *FileSystemBehavior*, *NetworkBehavior*, and *ProcessBehavior* classes obtained by observing ransomware behavior, there are other classes that are defined as direct subclasses of *Behavior* and

include individuals of those classes and derived from them. The *PersistenceBehavior* concept is an example of such a class. Ransomware families, like many malware, need to maintain persistence in the system to ensure execution after each reboot. They apply a variety of techniques for this purpose, such as adding the program to the startup folder or appending an entry to the Run Keys in registry. As a result, the *PersistenceBehavior* class can contain instances of the *RegistryBehavior* and *FileSystemBehavior*. More details about the defined classes (or equivalent classes) for separating behaviors will be explained in Section 3.3.

Given that the ultimate goal of most digital extortions is to make money, the payment system is an important aspect of such attacks, and a separate class in Rantology was defined for it. To identify the hierarchy of *PaymentSystem*, we collected ransom notes for 287 ransomware samples and extracted payment methods. These methods ranged from gift cards, vouchers and online payment services to cryptocurrencies. For example, the Yandex online payment service was used by WannaCash and TeleCrypt (Balaban, 2016; Cimpanu, 2016; TrendMicro, 2016). In addition to this online payment service, TeleCrypt also accepted payments via Qiwi service. However, in 261 of the examined samples (approximately 91%), ransom payment was demanded in the cryptocurrencies. Many ransomware families also have several payment methods. CryptoLocker is a representative example that wants ransom to be paid in the form of Bitcoin, UKash, CashU, Paysafecard, MoneyPak or prepaid cash vouchers. A bottom-up approach has been used to determine the *ThreatComponent* class hierarchy, as in the *Behavior* and *PaymentSystem* classes. The elements engaged in the extortion attack chain, such as droppers, loaders, spammers, botnets, exploited vulnerabilities, C2 servers, and the elicited domain generation algorithm (DGA) from binaries (Almashhadani et al., 2020) compose subclasses of *ThreatComponent*. Many of these sub-concepts are derived from the results of static and dynamic analysis of ransomware executables.

After defining the classes and their hierarchy, the characteristics of each class and relationships between them should be extracted from the identified key terms. For each property elicited in the Rantology, there is a class that is described by this feature. Each class can be associated with other classes, which are denoted by object properties. On the other hand, individuals belonging to each class can have characteristics that are specified by data properties. Data properties can be set with built-in datatypes. Datatypes are a collection of literals, such as strings, integers, and so on (W3C, 2021). We define these relationships and properties from other conspicuous terms extracted in the third phase of the seven-step approach adopted for the ontology design process. Table 4 shows a summary of statistical information related to the Rantology. However, this is an ongoing research and the numbers displayed in the table will increase over time.

### 3.3. Design details

We utilized Protégé[2] as a development environment to implement the ransomware ontology. Protégé is a free, open-source ontology editor that maintains a single namespace for all its frames and is case-sensitive.

**Table 4**
Rantology metrics.

| Metrics | Number |
|---|---|
| Class count | 473 |
| Object property count | 24 |
| Data property count | 77 |
| Axiom | 1495 |
| Logical axiom count | 813 |
| Declaration axioms count | 641 |

Logical reasoning is also done on the knowledge base using FaCT++ reasoner. We used the following general rules for naming classes and relationships. This naming system is followed throughout the ontology.

- Using PascalCase naming convention for class names
- Using camelCase naming convention for property names
- Using singular form for all concepts

We also used Process Monitor[3] and IDA Pro[4] tools to observe the behavior of ransomware and benign software, and extract API functions from them. Process Monitor is a tool from Windows Sysinternals that has the ability to monitor and display real-time file system, registry and process/thread activities, and many other features. It connects to a filter driver to capture system events. IDA Pro is also a powerful disassembler and debugger used in static and dynamic analysis of a variety of executable formats. In addition, the research utilizes Python programming language to scrap web pages to gather threat information and extract terms of interest.

As seen in the previous section, the classes defined in Rantology are connected to each other by the relationships called object properties. Indeed, these properties are the binary relationships between two individuals that link them together. Table 5 exhibits these relations along with the domain and range specified for them. If several classes are specified as the range of a property, the range will be interpreted as the intersection of these classes. Many of the classes in the domain and range columns are subclasses of the main concepts in Rantology. Also, some of the defined relationships are inverse to each other, which are listed in a separate column to avoid enlarging the table. The domain and range of such properties are the opposite. The considerations for determining the domain and range of object properties will be further elaborated in Section 3.4.

In contrast to object properties, data properties link an instance to an XML schema or literal. We have defined multiple data properties for each class of Rantology. In general, all specified properties for each class are inherited to its subclasses. As can be seen from Fig. 4, *CryptoRansomware*, *LockerRansomware* and *WiperRansomware* are subsumed by the *Ransomware* class. Therefore, the relationships and data properties defined for the *Ransomware* class are inherited to all its subclasses. This applies to all classes defined in Rantology. In fact, a new class is defined in the ontology when it either has a property or participates in a relationship that its superclass does not have or dose not participate in. Therefore, some features such as *hasFileName*, *hasVersion*, *hasSize*, *hasMD5Hash*, *hasSHA256Hash* and so on, are assigned to the more general *Software* class. For the *CryptoRansomware* class, in addition to data properties such as *hasMD5Hash*, *hasReleaseDate* and many others that are inherited from the parent classes, slots describing the encryption algorithm and key generation and management are also defined. This will further aid to classify ransomware based on the type of algorithm or key generation strategies (online or offline). By knowing the details of the cryptographic system, memory forensics tools can be used for memory dumping and gaining the key.

Leakware, also known as Doxware, is a new evolution of digital hijacking and cyber extortion (Keshavarzi & Ghaffary, 2020). Unlike ransomware, leakware does not prevent users from accessing their legal resources, but instead threatens to make public the personal and sensitive data of the victims if they do not pay a ransom. Since only some types of ransomware have the ability to doxing, or are likely to utilize such a behavior in the future, to distinguish them from the *Leakware* category, we define a data property called *hasStealingInfoCapability* for the *Ransomware* class that has a Boolean value. Depending on the version of Windows API, its functions can be deployed in.exe or.dll files. For instance, Win32 functions reside primarily in the core system DLLs

[2] https://protege.stanford.edu/.

[3] https://docs.microsoft.com/en-us/sysinternals/downloads/procmon/.

[4] https://hex-rays.com/ida-pro/.

**Table 5**
Object properties in Rantology.

| Object Property | Inverse of | Domain | Range |
|---|---|---|---|
| hasTargeted | isTargetedBy | Malware | Victim |
| hasDeveloper | isDeveloperOf | Goodware | SoftwareDeveloper |
| hasProducer | isProducerOf | Malware | Attacker |
| hasBehavior | isBehaviorOf | Software | Behavior |
| hasCalled | isCalledBy | Software | WindowsAPI |
| hasExploited | isExploitedBy | Malware | Vulnerability |
| hasVulnerability | – | Software | Vulnerability |
| isDisclosedBy | – | Vulnerability | CyberActor |
| hasUsed | isUsedBy | ThreatGroup | ThreatComponent |
| hasDemandedPaymentIn | isPaymentMethodOf | Ransomware, Leakware | PaymentSystem |
| hasRansomNote | – | Ransomware, Leakware | RansomNoteFile |
| isInvolvedIn | – | ThreatComponent | Malware |
| isRelatedTo | – | WindowsAPI | Behavior |
| hasConnectedTo | – | Malware | C2Server |
| hasString | – | Ransomware, Leakware | ThreateningMessage |
| isDistributedBy | – | Malware | ExploitKit, SpamEmailCampaign, Malware |

of Windows such as kernel32.dll, advapi32.dll, user32.dll, netapi32.dll, and gdi32.dll. This is defined as a data property named *hasDLL* for individuals belonging to the *WindowsAPI* class. Likewise, other classes of Rantology have data properties with values specified for them. In order to achieve clarity, it has been tried that the names selected for the properties correspond to the terms extracted from the threat reports in the scraped pages.

After going through the fifth step of the methodology used to design the ontology, at this stage the axioms and constraints must be defined (see Fig. 2). In reality, there may be instances that participate in certain relationships defined in the ontology but the class to which they belong is unclear. In OWL, such classes can be defined through restrictions. As mentioned earlier, two types of classes can be distinguished in OWL: primitive and defined. Primitive classes have only the necessary conditions, while defined classes are specified with the necessary and sufficient conditions. In Rantology, some classes are designed to be primitive and others to be defined. The principal classes located at the root level, as well as some of their subcategories, including *Person*, *Organization*, subclasses of *WindowsAPI*, and many others, are primitive concepts, which allow inference in only one direction. Such classes are introduced with subclass axioms. Other classes in Rantology, which are mostly in the middle or bottom levels, such as *Attacker*, *Victim*, *NormalBehavior*, *SuspiciousBehavior*, *Ransomware* and its categories, are defined concepts that are determined by various restrictions using equivalent classes axioms.

The distinguishing feature of ransomware from other fear-based malware is the obstruction of the user's legal access to resources. On the other hand, unlike other legitimate software that encrypts, erases and locks data, this denial of access will be accompanied by a threat in the form of a ransom note. For this purpose, we apply constraints to define the *Ransomware* class. We add an existential restriction along the property *hasBehavior* with a filler of *DenyingAccessBehavior* (a subclass of *Behavior*). This class contains instances of the primitive classes specified in the *Behavior*. The challenge in defining the second characteristic of ransomware is how to add the condition of having a ransom note. The ransom note, which comprises information about the attack on the victim and instructions for releasing the hostage resources, can be in the form of background image,.txt,.html files and the like. These files either come with the ransomware itself or are downloaded from C2 servers in the attack lifecycle. They mostly contain threatening phrases. Moreover, extortionist malware exhibits similar behaviors to display ransom note and inform the victim. For example, several files with the same name and different extensions may be left in each directory containing files tampered by ransomware. These files may also be dropped in the Desktop and Startup directories. In many instances, registry keys are edited to show the ransom note on each system reboot. We define these behaviors in a class called *ShowingRansomNoteBehavior*, which includes individuals of several subclasses of *Behavior*, including *FileSystemBehavior*, *NetworkBehavior*, and *RegistryBehavior*. Fig. 5 represents the definition of *Ransomware* class in the Manchester syntax. According to this definition, the conditions are not only necessary for membership in *Ransomware*, but also sufficient to determine if something satisfies these conditions is a member of the *Ransomware* class.

The main categories of ransomware are defined in a similar way by axioms. Many types of crypto-ransomware delete the original files when they perform the encryption operation. In addition, in many extortion onslaughts, the malware binary itself and some log files may be removed from the system by the attacker. Therefore, in the case of the *Wiper-Ransomware* class, since not every ransomware that has the behavior of deleting files is a wiper-ransomware, we specify this category only with the necessary conditions. As can be seen from Fig. 6, if something is a member of the *WiperRansomware* class, it is necessary for it to be a member of the *Ransomware* and it is necessary for it to be a member of the anonymous class of things that are linked to at least one individual of the class *DeletingFileBehavior* via the *hasBehavior* relationship. Thus, if a ransomware has a file deletion behavior, it is not a sufficient condition for this instance to be a member of the *WiperRansomware* class.

Classes related to ransomware families are disjoint and have more restrictive membership requirements. These constrains range from affected file extensions, threat group behind the attack, type of ransom demanded, and the tools used by the ransomware to other distinguishing behaviors. Given that reasoning in OWL ontology is based on the open world assumption (OWA), we used closure axioms on the properties to define classes associated with ransomware families. Also for the two defined classes *SuspiciousBehavior* and *NormalBehavior*, we utilized quantifier and cardinality restrictions by considering behaviors that participate in the *isBehaviorOf* relationship with instances of *Malware* and *Goodware* classes. Consequently, behaviors that satisfy the specified restrictions will become members of one of these two classes, leading to the separation of suspicious behaviors from normal ones. With cardinality constraints, even if really destructive behavior is observed in a small number of ransomware instances but exists in none or at most in a certain number of legal programs, it will be correctly classified in the *SuspiciousBehavior* class. This is also true for common behaviors between extortionist malware (especially Crypto and Wiper-Ransomware) and benign software, and prevents normal behaviors from being categorized in the wrong class. The greater the number of analyzed samples fed to the ontology, the higher the accuracy of behavior separation. Such restrictions will help classify the individuals that will later be added to the ontology.

In addition, other behavioral classes were asserted based on the relations they had with some of the Windows APIs, which assist in distinguishing malicious samples from harmless ones through the *hasBehavior* property. API calls are not malicious in themselves and can

```
Class: Ransomware

  Annotations:

        rdfs:label "Ransomware"@en,

        rdfs:comment "Ransomware is a type of malware that blocks users access to
        their resource, whether data or non-data, with the intent of extortion."

  EquivalentTo:

        Scareware
        and (hasBehavior some DenyingAccessBehavior)
        and ((hasBehavior some ShowingRansomNoteBehavior)
        or (hasRansomNote some RansomNoteFile)
        or (hasString some ThreateningMessage))
```

**Fig. 5.** Definition of Ransomware class in the Manchester syntax.

```
Class: WiperRansomware

  Annotations:

        rdfs:label "WiperRansomware"@en,

        rdfs:comment "WiperRansomware is a kind of ransomware aimed at preventing
        access to data resources, which will not return the hostage data even after
        the ransom has been paid by the victim."

  SubClassOf:

        Ransomware,

        hasBehavior some DeletingFileBehavior
```

**Fig. 6.** Definition of WiperRansomware class in the Manchester syntax.

be found in both legitimate and destructive software. What distinguishes malware from benign software based on these functions and provides insight into how they work is to gather a large volume of these API calls and understand their relationships to system components and the correlations between them. This will be achieved with Rantology. Ransomware, like many skillfully designed malware, uses a variety of camouflage techniques such as process injection, DLL injection and process hollowing. These behaviors are malevolent in nature and consequently software related to these classes is considered malicious. For such behaviors, several APIs may be exerted by extortionist malware authors. For example, a sequence of Windows API calls CreateToolhelp32Snapshot, Process32First, Process32Next, OpenProcess, CreateRemoteThread, VirtuallAllocEx and WriteProcessMemory are commonly seen in a DLL injection behavior (Sikorski & Honig, 2012). The corresponding classes of these behaviors are defined by the axioms and constraints on the *isRelatedTo* property in Rantology.

### 3.4. Design considerations

The challenge that any ontology may encounter is maintaining a consistent class hierarchy as the domain evolves (Noy & McGuinness, 2001). This section summarizes the design considerations used in the development of Rantology. Also, errors that may occur when defining classes are investigated. As mentioned earlier, classes and properties are the most prominent terms in a domain. In designing the Rantology, a lot of effort was made to differentiate these segments in such a way that in addition to covering the goals of the proposed ontology and competency questions, the most stability is achieved. For example, in the case of threat components, the IP should not be a class because it may change continuously and its individuals will have to be moved and superseded. So we define C2 server as a class and specify IP as a data property for it. The same is true for spam distribution servers as well as exploit kit hosting sites. Classes are organized in a hierarchy so that the relation among superclass and subclass is transitive. To specify the domain and range of the properties, some of the general rules should be taken into account. When defining the range or domain of a property, if there are several classes and they have a hierarchical relationship, we leave out the subclasses. For example, since the range of *hasDemandedPaymentIn* property can include cryptocurrencies (such as Bitcoin, Monero, Dash, and etc.), vouchers, and so on, we omit these sub-concepts and define only the superclass *PaymentSystem* as a range. Because these classes do not add any new information, and the parent class specified in the range will implicitly encompass these subclasses. Also, since all ransomware categories have ransom note (in the forms of html, text or image files), instead of adding the *hasRansomNote* property to these subclasses, we append this slot to the more general *Ransomware* class. Further generalization of the domain of this property would not be correct, as there is no ransom note in all malware or other types of scareware such as Rogueware. Rogueware mostly contains annoying popups and does not comprise threatening message. However, leakware also has such a feature, and hence this class has been added as a domain of the *hasRansomNote* property. In addition to running ransomware, such messages can also be obtained by analyzing the strings in the binary file itself. This knowledge can help determine the harmfulness of statically analyzed samples. All properties specified for *Ransomware* will be inherited to its subclasses. Similar considerations apply to other relationships defined in Rantology.

The name of a class can be changed according to the selected terminology, but this concept will still display the same objective reality in the domain. This issue can occur in naming classes related to ransomware families. To accelerate the process of malware detection, the vendors of antivirus tools are mainly using signature-based or heuristic approaches, which will lead to malware naming issues (Laszka et al., 2017). This is evident by submitting a sample to VirusTotal. Concepts related to ransomware families in Rantology, regardless of the name assigned to them, are identified by the certain properties that describe them. To prevent misinterpretation, we enter other names devoted to it by various antivirus engines in the documentation of each class of ransomware family (in the form of annotation properties). This issue has also been addressed for the aliases of some criminal gangs in the

*ThreatGroup* class. Many of these issues can be resolved by using the owl: sameAs property for different names that refer to the same entity.

### 3.5. Creation of ransomware knowledge base

The three main parts of an ontology building are generation, population and maintenance (or refinement). After defining the classes, properties, and the corresponding restrictions, a knowledge base is created at this stage by populating the ontology with individuals. In order to determine individuals, the most specific concepts must be identified, depending on the application of the ontology and competency questions. In our work, the most specific objects that we intend to display in the knowledge base and are at the lowest level of granularity include ransomware samples, instances of benign programs with similar behaviors to ransomware, diverse API functions, and suspicious events and activities in various parts related to the system components. For this purpose, we gathered 84 legal programs in 5 subclasses of *Goodware* along with 108 ransomware belonging to 79 different families from various repositories available on the Internet (https://virusshare.com/; https://sourceforge.net/; https://software.informer.com/). At this stage, the goal is to instantiate the ontology with individuals so that by creating different queries, the ontology can be evaluated in advance and refined if necessary. On the other hand, in order to reach a knowledge base that has the least amount of missing information, we chose this number of samples so that later, if incomplete data is added, we will have the ability to predict relationships and discover new knowledge well. However, the more instances are fed into the ontology, the more comprehensive the knowledge base and consequently the knowledge graph will be. Therefore, although we downloaded a total of more than 38,000 crypto-ransomware and more than 1000 locker samples from the VirusShare repository, we selected only 108 samples that almost we had scraped reports about them from web pages. To complete many of the sub-concepts of classes such as *Behavior* and *CyberActor*, we needed to fully analyze the samples and also have reports about them, including responsible threat groups or victims, published by security agencies and companies. In order to get the correct labels for each malicious instance, we submitted all samples with their MD5 hash to the VirusTotal website.[5] Since various antivirus engines assigned different names to each instance, we put the name that most closely matched the security reports inside the *hasFileName* data property and documented the other names in the annotation properties. Although we tried to select individuals in all three ransomware categories balanced, wiper-ransomware only accounted for less than 19% of the total malicious samples, which were downloaded in the crypto-ransomware collection.

We also collected 84 benign programs, which were mainly in the category of software that behaves similar to ransomware, such as archiver, cryptographic, backup software, and the like (https://sourceforge.net/; https://software.informer.com/). A significant part of these downloaded instances were compression and file archiver programs that placed in the *ArchiverSoftware* subclass, such as 7-Zip, ALZip, PeaZip, B1FreeArchiver, FreeArc, etc. However, we also submitted these samples to the VirusTotal to ensure they are safe. These instances were utilized only for analyzing and extracting API functions and finally deducing normal behaviors. Given that the two classes *NormalBehavior* and *SuspiciousBehavior* are specified through quantifiers and cardinality restrictions by considering the behaviors that participate in the *isBehaviorOf* relation with *Goodware* and *Malware* classes respectively, we needed these programs.

After analyzing the collected samples and extracting the desired features from them, the proposed and designed ontology was populated with these individuals. The properties defined in Rantology were filled with the corresponding values obtained from the analysis of the samples. Table 6 presents a number of crypto-ransomware instances imported into the Rantology along with some of their data properties. Not all properties are listed in the table. We use the MD5 hash of each ransomware as its unique name in the respective family class.

[5] https://www.virustotal.com/.

## 4. Evaluation

As with many research technologies, the development of ontologies will not be error-free, and the possibility of inconsistencies and redundancies during the design process is inevitable. Ontology assessment is the process of determining quality or accuracy according to a set of evaluation criteria that depends on the purpose for which the ontology is designed. Ontology evaluation is an indispensable part of ontology development. In the literature, ontology evaluation has been considered from different perspectives. In a taxonomy presented in (Brank et al., 2005), there are four categorical schemes of ontology assessment methods: Approach based on comparison with a golden standard; Approach based on comparison with a source of data about the domain to determine amount of coverage by the ontology; Approach based on using the ontology in an application and evaluating the results; And finally, manual evaluation by human experts based on a set of specific standards and criteria. In addition to aforementioned approaches, due to the relatively complex structure of the ontologies, Brank et al. (Brank et al., 2005) also considered different levels of evaluation. Therefore, each evaluation approach can verify different levels of ontology. Such level-based approaches will be more practical than those that evaluate ontologies as a whole. In other sources, the ontology assessment process is explored in two separate sections (Gómez-Pérez, 2004; Vrandečić, 2009): ontology verification and ontology validation. Amith et al. (Amith et al., 2018), who conducted a comprehensive review of ontology evaluation methods, also considered two interrelated steps for assessing biomedical ontologies: Ontology evaluation, which includes measurements for quality assessment, deals with the issue of goodness; Quality assurance (or auditing), which seeks to improve the quality of ontologies by focusing on the detection of modeling errors and inconsistencies, and is mainly done after the public release of ontologies by third parties.

The choice of an appropriate approach to assessment depends on the application in which the ontology is to be used as well as the aspects of the ontology we are trying to evaluate. To the best of our knowledge, no dedicated ontology has been designed for cyber extortion attacks and no golden standard has been defined for it. Due to the lack of previous work in the field of ransomware (or any extortion-based attacks) ontology, it is not possible to compare the performance of the proposed framework in this paper. Hence, we evaluate the suggested ontology in terms of clarity, coverage, modularity, semantic relations and consistency. Table 7 outlines these criteria along with a description of them.

In addition to the abovementioned evaluation criteria, the assessment of the quality of ontologies has also been studied in various aspects (Beydoun et al., 2011; Duque-Ramos et al., 2014; Mc Gurk et al., 2017; Roldán-Molina et al., 2021; Tartir et al., 2005, 2010; Zhu et al., 2017). For this purpose, several frameworks have been published in the literature, including OntoQA (Tartir et al., 2005), OQuaRE (Duque-Ramos et al., 2011) and OntoQualitas (Rico et al., 2014). Tartir et al. (Tartir et al., 2005, 2010) believe that the quality of ontology can be evaluated in different dimensions. They categorize the quality of ontologies into three groups: schema, knowledge base and class. Therefore, in addition to the criteria stated in Table 7, we also measure the quality of Rantology based on merely the schema metrics provided in OntoQA. These metrics can highlight key characteristics of Rantology schema.

Assuming that an ontology schema is a 6-tuple $O := \{C, P, A, H, Prop, att\}$ , where C, P and A are respectively the set of concepts, object properties (relationships) and data properties (attributes) defined in the ontology; $H \subseteq C \times C$ is a directed transitive relation to represent the concept taxonomy; And the functions $Prop : P \longrightarrow C \times C$ and $att : A \longrightarrow C$ relate concepts non-taxonomically and concepts to literal values, respectively. According to these assumptions, Eq. (1), Eq. (2), and Eq.

**Table 6**
Some of CryptoRansomware individuals.

| Individual (MD5) | hasFileName | hasSize | hasEncryptionAlgorithm |
|---|---|---|---|
| 21a563f958b73d453ad91e251b11855c | "Maze"^^string | 754176 | "RSA-2048"^^string |
| b7ad5f7ec71dc812b4771950671b192a | "Sekhmet"^^string | 709632 | "ChaCha"^^string<br>"RSA-2048"^^string |
| e9454a2ff16897e177d8a11083850ec7 | "Pysa"^^string | 516608 | "AES-256"^^string<br>"RSA"^^string |
| d32ff14c37b0b7e6c554ce3de5a85454 | "VCrypt"^^string | 794801 | "7Zip"^^string |
| 8f90539c405672016c0dec7ac3574eea | "Nefilim"^^string | 72672 | "AES-128"^^string<br>"RSA-2048"^^string |
| 53dddbb304c79ae293f98e0b151c6b28 | "MegaCortex"^^string | 745408 | "AES-256"^^string<br>"RSA-4096"^^string |
| 9d418ecc0f3bf45029263b0944236884 | "DarkSide"^^string | 60416 | "Salsa20"^^string<br>"RSA-1024"^^string |

**Table 7**
Rantology evaluation criteria.

| Criterion | Description |
|---|---|
| Clarity | It refers to the absence of undefined or ambiguous concepts in the ontology. |
| Modularity | This criterion determines the extensibility and reuse of the ontology. |
| Consistency | It points to the correct classification of objects and the absence of contradictions in the ontology. |
| Coverage | Coverage (or more comprehensively, completeness) indicates the appropriateness of an ontology for modeling and representing the domain of discourse. |

(3) measure Relationship Richness (RR), Attribute Richness (AR), and Inheritance Richness (IR), respectively.

$$RR = \frac{|P|}{|H| + |P|} \tag{1}$$

$$AR = \frac{|att|}{|C|} \tag{2}$$

$$IR = \frac{|H|}{|C|} \tag{3}$$

RR has a value between 0 and 1; The closer this value is to zero, the more relations in the ontology are class-subclass relationships, and vice versa. A high value for AR points out that each class has a large number of attributes on average, while a lower value may indicate that less information is provided about each class. Finally, the IR metric specifies the distribution of information among different levels of the ontology. This measure can be used to distinguish a horizontal ontology (where classes have a large number of direct subclasses) from a vertical one (where classes have a small number of direct subclasses) (Tartir et al., 2005, 2010). An ontology with low inheritance richness will be a vertical ontology, indicating that it covers a specific domain in a precise way, while an ontology with high IR is a horizontal ontology, representing that it covers a wide range of knowledge with a low level of detail. According to the statistical information depicted in Table 4, the values of RR, AR and IR were measured for Rantology and are shown in Table 8.

Rantology passes the criterion of clarity since the definitions of selected terms have been tried to be objective and extracted from a large number of relevant scraped threat reports. Also, as mentioned before, annotation properties were specified to remove the ambiguity in the names of ransomware families and other concepts, and this information was documented in natural language. In addition, the modular design of Rantology allows it to be utilized in other applications or to add new concepts to it with minimal modifications. For example, the definition of the *PaymentSystem* class allows the use of this knowledge representation system in applications aimed at analyzing the economics of ransomware. On the other hand, applying the taxonomic hierarchy presented in our previous work (Keshavarzi & Ghaffary, 2020) makes this ontology extensible to other modern extortion-based malware.

**Table 8**
Evaluation of the quality of Rantology schema.

| Quality Schema Metrics | Value |
|---|---|
| Relationship Richness (RR) | 0.08 |
| Attribute Richness (AR) | 0.16 |
| Inheritance Richness (IR) | 0.55 |

Evaluating semantic relationships other than "is-a" between concepts is another important aspect of ontology assessment. We describe ransomware families by their relationships to other classes, and then utilize reasoner to deduce and classify the individuals we fed into the Rantology (instances in the *Uncategorized* class) to see which category of ransomware they belong to. The individuals added in the *Uncategorized* class were selected from the same dataset described in subsection 3.5, and some of them were downloaded samples that had not previously been imported into the Rantology. Reasoning capability is utilized for subsumption testing and consistency checking. To do this, we ran FaCT++ reasoner on the knowledge base. The inferred class hierarchy was free of any inconsistency and the individuals asserted in the *Uncategorized* category were correctly classified. For better evaluation and validation of Rantology, classes corresponding to ransomware families are defined as disjoint. By doing so, there is no sample that belongs to different families, and if such a thing is inferred, there is a modeling error that the system is able to show. But this is not the case for individuals of *Goodware* subclasses. Because there may be a program with the ability to encrypt and archive data (e.g., PowerArchiver) that is a subclass of the two classes *CryptographicSoftware* and *ArchiverSoftware*, and therefore inherits from both. Finally, the completeness of an ontology is determined by the competency questions raised at the beginning of the design process, which will show the extent of its coverage. To verify the capability of Rantology in answering competency questions, we used the description logic (DL) query interface available in Protégé. For example, query "hasStealingInfoCapability value true" is designed to answer competency question 4 (CQ4) and returns all ransomware individuals that have doxing capability. Similarly, other queries tailored to other competency questions were designed to assess ontology coverage. This study is part of a larger research work, and certainly with its dissemination and widespread use in various applications, some missing or redundant concepts and relationships may be discovered.

## 5. Discussion

Despite the many efforts made by the security sector, ransomware continues to claim victims around the world. All humans with any role (attacker, defender, user, etc.) are an integral part of computer networks and all of them, not only attackers, can be considered a threat to the system. User awareness training should be the priority of any organization's security practice because many researchers believe that the

weakest link in the cyber security chain is the human user. Describing human behaviors that have a higher probability of attracting digital extortion attacks provides new opportunities to identify and defend sectors and hosts that are more exposed to these threats. Accordingly, research on the importance of human behavior in the development of security products has grown increasingly. Ovelgönne et al. (Ovelgönne et al., 2017) conducted a study on 1.6 million hosts of Symantec's WINE dataset belonging to an 8-month period to reveal the relationship between user behavior and cyber-attacks against their machines. The surprising result of their research showed that software developers are more prone to malware attacks than other user categories defined in their work. Also, the development of a comprehensive and predictive risk assessment model requires the characterization of human factors to understand how the actions of users, defenders and attackers affect cyber security risk (Oltramari et al., 2015). Therefore, the integration of ontologies into cyber security science helps decision makers build the foundations needed for quantitative and predictive risk assessments. In essence, cyber defense is highly dependent on human analysts involved in the process of data fusion and situational awareness (Oltramari et al., 2014, 2015).

Ransomware attacks continue to progress and by preventing legal access to resources and setting a deadline for ransom payment, they have created a space in cyber security that requires dynamic decision making. Situational awareness is one of the main pillars and key skills for making correct decisions. Basically, the lack of situational awareness is one of the main causes of accidents with the root of human error. Endsley defines situational awareness as follows (Endsley, 2017): Percepting the environmental elements within a volume of time and space, comprehending their meaning and projecting their status in the near future. Although the main mission of Rantology is to discover the relationships between APIs and system behaviors and consequently to identify and categorize ransomware and build a knowledge base of them, but due to the importance of human role in risk management and cyber security chain, a separate class was considered for this concept. This will help expand Rantology to encompass more goals, including extracting TTPs of attackers, ranking victims and vulnerable sectors based on the probability of being infected with this type of malware, increasing situational awareness, and ultimately generating ransomware threat intelligence.

In addition to considering the human factor as a core class, Rantology also addresses payment systems. This can inspire the development of another ontology to display knowledge pertaining to illegal financial transactions related to Internet crimes, especially ransomware in the domain of cryptocurrencies and other payment systems. Such a knowledge base will be of great help in tracking down cybercriminals based on ransom payments.

One of the important challenges in security solutions is generating false alarms. False Positive is one of the common problems in computational approaches including machine learning techniques in detecting malware, especially ransomware, which have a lot of functional similarities with legitimate programs. Many research works exploit API calls to identify and distinguish malicious samples from harmless ones. However, only the presence or absence or the sequence of API functions is considered and its meaning is neglected. Mapping API functions to normal and suspicious behaviors can help to better understand their semantics and more optimal selection of distinguishing features and subsequently reduce the false positive rate. However, we encountered this problem in our work, especially in the definition of *NormalBehavior* and *SuspiciousBehavior* classes. The false positive rate was inversely related to the quantifiers and cardinality restrictions used in these equivalent classes. This issue will be resolved by filling the ontology with more examples and creating richer knowledge of cyber extortion threats. Also, extracting more object and data properties will help in inferring new knowledge from existing complex relationships and optimizing queries, thereby reducing the false positive rate.

It is important to note that Rantology is not a real-time monitoring solution. It is simply a semantic model of the elements involved in digital extortion attacks, which by providing a knowledge base, is able to reason on the classes and relationships defined in it. Finally, ontologies usually suffer from assessment issues. Therefore, we do not claim to have done a comprehensive evaluation of Rantology. Definitely, applying it in different applications can reveal more fine-grained details.

## 6. Conclusion and open issues

Cybercrime has become a growing, thriving and highly profitable industry. Meanwhile, digital extortion threats are one of the most lucrative attacks, which not only have not diminished, but with the COVID-19 pandemic, they have accounted for a larger share of these illicit revenues. Researchers and security experts have been constantly fighting these extortionate offenses, especially ransomware, and dissecting and analyzing them in order to understand how they work and provide appropriate countermeasures against these threats. Given the vast amount of data from ransomware analysis as well as numerous threat reports about the ecosystem of cyber extortion attacks on the web, we need a model to organize structured and unstructured information through entities, features, and how they relate to each other. Understanding the behavior of ransomware and recognizing critical points for monitoring is essential for a successful defense. To this end, having an explicit formal specification of the components involved in cyber extortion attacks and the relations between them seems crucial. In this paper, an ontology for ransomware attacks, called Rantology, is proposed that can be used by researchers and security experts to share and annotate information in the context of digital extortion onslaughts. In addition to sharing a common understanding of information as well as inferring domain knowledge to generate ransomware threat intelligence, this research will pursue another goal. The current study focuses on digital extortion attacks. By expanding this ontology and knowledge base to other malicious programs and integrating them, a reference malware ontology can be achieved that covers a wider domain.

This research was conducted with the aim of gaining insight into the behavior of ransomware and legal software. For this purpose, the relationships between these programs, system calls and behaviors were scrutinized. In fact, Rantology is a formalized vocabulary of terms in the field of digital extortion attacks that focuses on the relations between ransomware, API functions and behaviors. It is a deductive and incremental framework, not just a passive repository of asserted instances, and can accept imperfect descriptions of ransomware strains and then complete and refine them. In this research work, a combined methodology has been used to design the ransomware ontology. First, the most prominent concepts in the scope of cyber extortion attacks are identified, and then each of them is appropriately generalized or specialized. The stages of the development process of this ontology and how to determine the hierarchy of classes, specifications, relationships and individuals belonging to each of them were described in detail. Since this is a research work in progress and the information of many collected samples has not been entered into the Rantology, this knowledge base is not yet publicly available. As soon as the population phase is completed with more gathered individuals, the resulting knowledge base will be made public so that other researchers can help evaluate, improve and eliminate the possible shortcomings of this ontology by using it in different scenarios.

However, the proposed conceptual modeling is extensible, and this ontology can also be used to understand the economic structure of ransomware and track financial transactions related to such extortions. This can be considered as an open issue for future work, despite the fact that a separate class was provided for payment systems in Rantology. Furthermore, identifying attacker groups or comprehending their TTPs is not the main focus of this paper, although pertained classes have been incorporated into the ontology, but have not been further explored. The next task we are doing is to infer knowledge from this ontology-based framework to build recommender systems using machine learning

techniques and artificial intelligence (AI) to monitor events and various parts of the operating system for the sake of ransomware detection.

## Author contribution

Conceptualization, Masoudeh Keshavarzi, Methodology/Study design, Masoudeh Keshavarzi, Software, Masoudeh, Keshavarzi, Validation, Masoudeh Keshavarzi, Hamid Reza Ghaffary, Formal analysis, Masoudeh Keshavarzi, Investigation, Masoudeh Keshavarzi, Resources, Masoudeh Keshavarzi, Hamid Reza Ghaffary, Data curation, Masoudeh Keshavarzi, Hamid Reza Ghaffary, Writing – original draft, Masoudeh Keshavarzi, Writing – review and editing, Masoudeh Keshavarzi, Visualization, Masoudeh Keshavarzi, Supervision, Masoudeh Keshavarzi, Hamid Reza Ghaffary, Project administration, Hamid Reza Ghaffary, Funding acquisition.

## Data availability

The data that has been used is confidential.

## References


Ahmed, Y. A., Koçer, B., Huda, S., Al-rimy, B. A. S., & Hassan, M. M. (2020). A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *Journal of Network and Computer Applications, 167*, Article 102753.

Akbanov, M., Vassilakis, V. G., & Logothetis, M. D. (2019). Ransomware detection and mitigation using software-defined networking: The case of WannaCry. *Computers & Electrical Engineering, 76*, 111–121.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2018). Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security, 74*, 144–166.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2019). Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems, 101*, 476–491.

Almashhadani, A. O., Kaiiali, M., Carlin, D., & Sezer, S. (2020). MaldomDetector: A system for detecting algorithmically generated domain names with machine learning. *Computers & Security, 93*, Article 101787.

Amith, M., He, Z., Bian, J., Lossio-Ventura, J. A., & Tao, C. (2018). Assessing the practice of biomedical ontology evaluation: Gaps and opportunities. *Journal of Biomedical Informatics, 80*, 1–13.

November Andronio, N., Zanero, S., & Maggi, F. (2015). Heldroid: Dissecting and detecting mobile ransomware. In *International symposium on recent advances in intrusion detection* (pp. 382–404). Cham: Springer.

Arp, R., Smith, B., & Spear, A. D. (2015). *Building ontologies with basic formal ontology*. Mit Press.

May Bajpai, P., Sood, A. K., & Enbody, R. (2018). A key-management-based taxonomy for ransomware. In *2018 APWG symposium on electronic crime research (eCrime)* (pp. 1–12). IEEE.

Balaban, D. Tripwire (2016). *November 2016: The month in ransomware* [Online] Available: https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/november-2016-month-ransomware/.

Beydoun, G., Lopez-Lorca, A. A., Garcia-Sanchez, F., & Martinez-Bejar, R. (2011). How do we measure and improve the quality of a hierarchical ontology? *Journal of Systems and Software, 84*(12), 2363–2373.

Botacin, M., Ceschin, F., de Geus, P., & Grégio, A. (2020). We need to talk about antiviruses: Challenges & pitfalls of av evaluations. *Computers & Security, 95*, Article 101859.

Botacin, M., Ceschin, F., Sun, R., Oliveira, D., & Grégio, A. (2021). Challenges and pitfalls in malware research. *Computers & Security, 106*, Article 102287.

Botacin, M., Domingues, F. D., Ceschin, F., Machnicki, R., Alves, M. A. Z., de Geus, P. L., & Grégio, A. (2022). AntiViruses under the microscope: A hands-on perspective. *Computers & Security, 112*, Article 102500.

October Brank, J., Grobelnik, M., & Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)* (Vol. 17)Slovenia: Citeseer Ljubljana.

December Chen, Q., & Bridges, R. A. (2017). Automated behavioral analysis of malware: A case study of wannacry ransomware. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)* (pp. 454–460). IEEE.

Cimitile, A., Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2018). Talos: No more ransomware victims with formal methods. *International Journal of Information Security, 17*(6), 719–738.

Cimpanu, C. (2016). *Telecrypt ransomware uses telegram as C&C server* [Online] Available: https://www.bleepingcomputer.com/news/security/telecrypt-ransomware-uses-telegram-as-candc-server/.

Cimpanu, C. (2020). *ProLock ransomware - everything you need to know* [Online] Available: https://www.zdnet.com/article/prolock-ransomware-everything-you-need-to-know/.

December Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., & Maggi, F. (2016). Shieldfs: A self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd annual conference on computer security applications* (pp. 336–347).

Ding, Y., Wu, R., & Zhang, X. (2019). Ontology-based knowledge representation for malware individuals and families. *Computers & Security, 87*, Article 101574.

Duque-Ramos, A., Boeker, M., Jansen, L., Schulz, S., Iniesta, M., & Fernández-Breis, J. T. (2014). Evaluating the good ontology design guideline (GoodOD) with the ontology quality requirements and evaluation method and metrics (OQuaRE). *PLoS One, 9*(8), Article e104463.

Duque-Ramos, A., Fernández-Breis, J. T., Stevens, R., & Aussenac-Gilles, N. (2011). OQuaRE: A SQuaRE-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology, 43*(2), 159–176.

Endsley, M. R. (2017). Toward a theory of situation awareness in dynamic systems. In *Situational awareness* (pp. 9–42). Routledge.

Falcone, R., & Grunzweig, J. (2017). *Targeted ransomware attacks middle eastern government organizations for political purposes* (Accessed June 2018) https://researchcenter.paloaltonetworks.com/2017/03/unit42-targeted-ransomware-attacks-middle-eastern-government-organizations-political-purposes.

Frankoff, S., & Hartley, B. (2018). *Big game hunting: The evolution of INDRIK SPIDER from dridex wire fraud to BitPaymer targeted ransomware. Crowdstrike* [Online] Available: https://www.crowdstrike.com/blog/big-game-hunting-the-evolution-of-indrik-spider-from-dridex-wire-fraud-to-bitpaymer-targeted-ransomware/. (Accessed 28 June 2019) Accessed on.

Freed, A. M. (2021). *A brief history of ransomware evolution* [Online] Available Accessed on December 5, 2021 https://www.cybereason.com/blog/a-brief-history-of-ransomware-evolution.

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

Gómez-Hernández, J. A., Álvarez-González, L., & García-Teodoro, P. (2018). R-Locker: Thwarting ransomware action through a honeyfile-based approach. *Computers & Security, 73*, 389–398.

Gómez-Pérez, A. (2004). Ontology evaluation. In *Handbook on ontologies* (pp. 251–273). Berlin, Heidelberg: Springer.

Grégio, A. R. A., Afonso, V. M., Filho, D. S. F., Geus, P. L. D., & Jino, M. (2015). Toward a taxonomy of malware behaviors. *The Computer Journal, 58*(10), 2758–2777.

Hampton, N., Baig, Z., & Zeadally, S. (2018). Ransomware behavioural analysis on windows platforms. *Journal of Information Security and Applications, 40*, 44–51.

Han, W., Xue, J., Wang, Y., Zhang, F., & Gao, X. (2021). APTMalInsight: Identify and cognize APT malware based on system call information and ontology knowledge framework. *Information Sciences, 546*, 633–664.

Hernandez-Castro, J., Cartwright, A., & Cartwright, E. (2020). An economic analysis of ransomware and its welfare consequences. *Royal Society Open Science, 7*(3), Article 190023.

Hernandez-Castro, J., Cartwright, E., & Stepanova, A. (2017). *Economic analysis of ransomware*. Available at: SSRN 2937641.

September Hilario, M., Kalousis, A., Nguyen, P., & Woznica, A. (2009). A data mining ontology for algorithm selection and meta-mining. In *Proceedings of the ECML/PKDD09 workshop on 3rd generation data mining* (pp. 76–87) (SoKD-09).

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., & Khayami, R. (2017). Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing, 8*(2), 341–351.

July Huang, H. D., Chuang, T. Y., Tsai, Y. L., & Lee, C. S. (2010). Ontology-based intelligent system for malware behavioral analysis. In *International conference on fuzzy systems* (pp. 1–6). IEEE.

Huang, H. D., Lee, C. S., Wang, M. H., & Kao, H. Y. (2014). IT2FS-based ontology with soft-computing mechanism for malware behavior analysis. *Soft Computing, 18*(2), 267–284.

April Iannacone, M., Bohn, S., Nakamura, G., Gerth, J., Huffer, K., Bridges, R., … Goodall, J. (2015). Developing an ontology for cyber security knowledge graphs. In *Proceedings of the 10th annual cyber and information security research conference* (pp. 1–4).

Jia, Y., Qi, Y., Shang, H., Jiang, R., & Li, A. (2018). A practical approach to constructing a knowledge graph for cybersecurity. *Engineering, 4*(1), 53–60.

Johnson, J. "Number of ransomware attacks per year 2014-2020". Statista, 13 April 2021. Accessed April 2021. https://www.statista.com/statistics/494947/ransomware-attacks-per-year-worldwide/.

Kaloroumakis, P. E., & Smith, M. J. (2021). *Toward a knowledge graph of cybersecurity countermeasures*. Corporation.

Kaspersky. Ransomware 2018–2020 Accessed on 11 July 2020 https://media.kaspersky.com/wp-content/uploads/sites/100/2020/05/12075747/KSN-article_Ransomware-in-2018-2020-1.pdf.

Keet, C. M., Ławrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., … Hilario, M. (2015). The data mining optimization ontology. *Journal of web semantics, 32*, 43–53.

Keshavarzi, M., & Ghaffary, H. R. (2020). I2CE3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion. *Computer Science Review, 36*, Article 100233.

October Laszka, A., Farhang, S., & Grossklags, J. (2017). On the economics of ransomware. In *International conference on decision and game theory for security* (pp. 397–417). Cham: Springer.

Logan, M., Mendoza, E., Maglaque, R., & Tamaña, N. (2021). *The state of ransomware: 2020's catch-22. Trend Micro* [Online] at https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-state-of-ransomware-2020-s-catch-22. (Accessed 24 May 2021) Accessed on.

Luo, X., & Liao, Q. (2007). Awareness education as the key to ransomware prevention. *Information Systems Security, 16*(4), 195–202.
April Maiorca, D., Mercaldo, F., Giacinto, G., Visaggio, C. A., & Martinelli, F. (2017). R-PackDroid: API package-based characterization and detection of mobile ransomware. In *Proceedings of the symposium on applied computing* (pp. 1718–1723).
Malone, J., Brown, A., Lister, A. L., Ison, J., Hull, D., Parkinson, H., & Stevens, R. (2014). The software ontology (SWO): A resource for reproducibility in biomedical data analysis, curation and digital preservation. *Journal of Biomedical Semantics, 5*(1), 1–13.
Mc Gurk, S., Abela, C., & Debattista, J. (2017). *Towards ontology quality assessment*.
September Mehnaz, S., Mudgerikar, A., & Bertino, E. (2018). Rwguard: A real-time detection system against cryptographic ransomware. In *International symposium on research in attacks, intrusions, and defenses* (pp. 114–136). Cham: Springer.
August Mittal, S., Das, P. K., Mulwad, V., Joshi, A., & Finin, T. (2016). Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *2016 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM)* (pp. 860–867). IEEE.
Morato, D., Berrueta, E., Magaña, E., & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications, 124*, 14–32.
13 November Morgan, S. (2020). *Cybercrime to cost the world $10.5 trillion annually by 2025*. Cybersecurity Ventures Accessed April 2021 https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/.
Mozzaquatro, B. A., Agostinho, C., Goncalves, D., Martins, J., & Jardim-Goncalves, R. (2018). An ontology-based cybersecurity framework for the internet of things. *Sensors, 18*(9), 3053.
Narayanan, S., Ganesan, A., Joshi, K., Oates, T., Joshi, A., & Finin, T. (2018). *Cognitive techniques for early detection of cybersecurity events*. arXiv preprint arXiv:1808.00116.
Navarro, L. C., Navarro, A. K., Gregio, A., Rocha, A., & Dahab, R. (2018). Leveraging ontologies and machine-learning techniques for malware analysis into Android permissions ecosystems. *Computers & Security, 78*, 429–453.
Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*.
Oberle, D., Grimm, S., & Staab, S. (2009). An ontology for software. In *Handbook on ontologies* (pp. 383–402). Berlin, Heidelberg: Springer.
O'Brien, D., DiMaggio, J., & Nguyen, H. G. (2019). *Targeted ransomware: An ISTR special report*. Whitepaper, Symantec Corporation.
October Obrst, L., Chase, P., & Markeloff, R. (2012). Developing an ontology of the cyber security domain. In *STIDS* (pp. 49–56).
November Oltramari, A., Cranor, L. F., Walls, R. J., & McDaniel, P. D. (2014). Building an ontology of cyber security. In *STIDS* (pp. 54–61).
Oltramari, A., Henshel, D. S., Cains, M., & Hoffman, B. (2015). Towards a human factors ontology for cyber security. *Stids*, 26–33, 2015.
Osena, M. (2018). Cryptocurrency-mining malware: 2018's new menace? *Trend Micro blog, 28*.
Ovelgönne, M., Dumitraş, T., Prakash, B. A., Subrahmanian, V. S., & Wang, B. (2017). Understanding the relationship between human behavior and susceptibility to cyber attacks: A data-driven approach. *ACM Transactions on Intelligent Systems and Technology (TIST), 8*(4), 1–25.
Paquet-Clouston, M., Haslhofer, B., & Dupont, B. (2019). Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity, 5*(1), tyz003.
Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems, 97*, 887–909.
August Rastogi, N., Dutta, S., Zaki, M. J., Gittens, A., & Aggarwal, C. (2020). MALOnt: An ontology for malware threat intelligence. In *International workshop on deployable machine learning for security defense* (pp. 28–44). Cham: Springer.
Rico, M., Caliusco, M. L., Chiotti, O., & Galli, M. R. (2014). OntoQualitas: A framework for ontology quality assessment in information interchanges between heterogeneous systems. *Computers in Industry, 65*(9), 1291–1300.
Roldán-Molina, G. R., Ruano-Ordás, D., Basto-Fernandes, V., & Méndez, J. R. (2021). An ontology knowledge inspection methodology for quality assessment and continuous improvement. *Data & Knowledge Engineering, 133*, Article 101889.
Salini, P., & Shenbagam, J. (2015). Prediction and classification of web application attacks using vulnerability ontology. *International Journal of Computer Application, 116*(21).
Santos, D. (2021). *Threat assessment: Clop ransomware* [Online] Available: https://unit42.paloaltonetworks.com/clop-ransomware/.
Scalas, M., Maiorca, D., Mercaldo, F., Visaggio, C. A., Martinelli, F., & Giacinto, G. (2019). On the effectiveness of system API-related information for Android ransomware detection. *Computers & Security, 86*, 168–182.
January Shoaib, M., & Farooq, M. (2015). USpam–A user centric ontology driven spam detection system. In *2015 48th Hawaii international conference on system sciences* (pp. 3661–3669). IEEE.
Sikorski, M., & Honig, A. (2012). *Practical malware analysis: The hands-on guide to dissecting malicious software*. no starch press.
Silberschatz, A., Gagne, G., & Galvin, P. B. (2018). *Operating system concepts* (10th ed.). John Wiley & Sons.
Sokolov, K. (2021). Ransomware activity and blockchain congestion. *Journal of Financial Economics*.
Syed, R. (2020). Cybersecurity vulnerability management: A conceptual ontology and cyber intelligence alert system. *Information & Management, 57*(6), Article 103334.
Syed, Z., Padia, A., Finin, T., Mathews, L., & Joshi, A. (2016). *UCO: A unified cybersecurity ontology*. UMBC Student Collection.
Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-Meza, B. (2005). *OntoQA: Metric-based ontology quality analysis*.
Tartir, S., Arpinar, I. B., & Sheth, A. P. (2010). Ontological evaluation and validation. In *Theory and applications of ontology: Computer applications* (pp. 115–130). Dordrecht: Springer.
January 19 Trellix. (2022). *Trellix 2022 threat predictions* [Online] Available: https://www.trellix.com/en-us/about/newsroom/stories/threat-labs/2022-threat-predictions.html.
Trend Micro Research. (2022). *Ransomware spotlight clop* [Online] Available: https://www.trendmicro.com/vinfo/us/security/news/ransomware-spotlight/ransomware-spotlight-clop.
TrendMicro. (2016). *Ransomware recap: Nov. 7- 18, 2016* [Online] Available: https://www.trendmicro.com/vinfo/se/security/news/cybercrime-and-digital-threats/ransomware-recap-nov-7-18-2016.
September Undercoffer, J., Joshi, A., & Pinkston, J. (2003). Modeling computer attacks: An ontology for intrusion detection. In *International workshop on recent advances in intrusion detection* (pp. 113–135). Berlin, Heidelberg: Springer.
Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The Knowledge Engineering Review, 11*(2), 93–136.
U.S. Government. (2016). *How to protecting your networks from ransomware, 2–8*. Retrieved September 17, 2017, from https://www.justice.gov/criminal-ccips/file/872771/download.
Vrandečić, D. (2009). Ontology evaluation. In *Handbook on ontologies* (pp. 293–313). Berlin, Heidelberg: Springer.
W3C. (2021). *Web ontology language (OWL)*. Retrieved from https://www.w3.org/OWL/.
May 4 Walter, J. (2020). *Meet NEMTY successor, nefilim/nephilim ransomware* [Online] Available: https://www.sentinelone.com/labs/meet-nemty-successor-nefilim-nephilim-ransomware/.
Wiśniewski, D., Potoniec, J., Ławrynowicz, A., & Keet, C. M. (2019). Analysis of ontology competency questions and their formalizations in SPARQL-OWL. *Journal of Web Semantics, 59*, Article 100534.
Xiaofeng, L., Fangshuo, J., Xiao, Z., Shengwei, Y., Jing, S., & Lio, P. (2019). ASSCA: API sequence and statistics features combined architecture for malware detection. *Computer Networks, 157*, 99–111.
May Xu, D., Ming, J., & Wu, D. (2017). Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping. In *2017 IEEE symposium on security and privacy (SP)* (pp. 921–937). IEEE.
Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F., & Sangaiah, A. K. (2019). Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Generation Computer Systems, 90*, 211–221.
Zhu, H., Liu, D., Bayley, I., Aldea, A., Yang, Y., & Chen, Y. (2017). Quality model and metrics of ontology for semantic descriptions of web services. *Tsinghua Science and Technology, 22*(3), 254–272.
https://software.informer.com/.
https://sourceforge.net/.
https://virusshare.com/.