Ontology-Driven Framework for Trend Analysis of Vulnerabilities and Impacts in IoT Hardware

Charan Bandi*§, Soheil Salehi†§, Rakibul Hassan*§, Sai Manoj P D*, Houman Homayoun†, and Setareh Rafatirad*
†School of Electrical and Computer Engineering, University of California Davis, Davis, CA 95616
*School of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030
*{vbandi, rhassa2, spudukot, srafatir}@gmu.edu; †{ssalehi, hhomayoun}@ucdavis.edu

2021 IEEE 15th International Conference on Semantic Computing (ICSC) | 978-1-7281-8899-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSC50631.2021.00045

***Abstract*—The number of publicly known cyber-security vulnerabilities submitted to the National Vulnerability Database (NVD) has increased significantly. However, it is cumbersome to explore useful information from this large corpus of unstructured data to find meaningful trends over time without proper tools. Prior works with this purpose have mainly focused on the software vulnerabilities and fail to provide a storytelling framework that can extract useful information about the relationship and trends within the CVE and CWE databases over time. Additionally, hardware attacks on IoT devices are evolving very rapidly due to the recent proliferation of computing devices in mobile and IoT domains. Thus, herein, we focus on IoT hardware vulnerabilities and develop an Ontology-driven Storytelling Framework (OSF) which aims to identify similar patterns of vulnerabilities over time, to help mitigate the impacts of vulnerabilities or predict and prevent future vulnerabilities.**

***Index Terms*—National Vulnerability Database (NVD), Common Vulnerability and Exposure (CVE), Common Weakness Enumeration (CWE), Internet of Things (IoT), Hardware Vulnerability, Ontology Learning, Natural Language Processing (NLP).**

I. Introduction

The increasing complexity of modern computing systems has promised enhanced processing capabilities, but has also resulted in the growth of security vulnerabilities, making such systems an appealing target for evolving sophisticated security threats. In contrast to the traditional software-based exploits, the hardware-based attacks exploit the hardware vulnerabilities, and attack the underlying architecture with the aid of software applications [1]–[6]. According to the National Vulnerability Database (NVD) by National Institute of Standards and Technology (NIST)[1], the number of CVEs submitted to the NVD in 2019 were 17,302. This calls for investigating and developing defenses to this class of vulnerabilities widely knows as Common Weakness Enumeration (CWE)[2], and not just their CVE instances. While CWEs and CVEs encompass both software and hardware vulnerabilities, herein, we focus on the software-assisted hardware vulnerabilities, as they have been increasing in recent years. Furthermore, the number of Internet of Things (IoT) devices has been estimated to surpass 75 billion devices by 2025 [7]. Moreover, state-of-the-art hardware attacks on resource constrained IoT devices are evolving very rapidly [7], [8]. According to the 2020 McAfee threats report[3], 58% new IoT malware's were discovered during the first quarter of 2020. Additionally, the recent proliferation of computing devices in mobile and IoT domains exacerbates the security threats and attacks, which calls for effective mitigation techniques. Thus, there is an increasing demand in addressing this emerging challenge of detecting and mitigating threats involving resource constrained IoT devices.

Most of the emerging vulnerabilities and exposures within IoT systems are due to incomplete or outdated software applications and hardware systems. Thus, it is crucial to know the trends and evolution of various vulnerabilities and impacts to be able to provide effective and secure design solutions. However, it is cumbersome to extract useful information from the large corpus of data provided by NIST and MITRE without the proper means to analyze the vulnerabilities and exposures in extensive details. Most prior works have shifted their focus on the software vulnerabilities and fail to provide a storytelling framework that can extract useful information about the relationship and trends within the CVE and CWE databases over time. Thus, herein we utilize the state-of-the-art Natural Language Processing (NLP) and Machine Learning (ML) techniques to provide a detailed analysis of the hardware vulnerabilities in IoT within CWEs and CVEs datasets, study their relationships, and provide insight into how to prevent emerging vulnerabilities and impacts. We develop a semi-automated Ontology-driven Storytelling Framework (OSF) which utilizes NLP to extract useful information from the CWE database and NVD with the focus on hardware vulnerabilities in IoT to find trends and meaningful relationships among various classes of vulnerabilities and their impacts.

II. Background

Despite the performance benefits achieved with emerging computing architectures, security attacks such as Distributed Denial-of-Service, malware and side-channels are ever increasing, due to increased surface attacks as a result of more complexity and heterogeneity. While there has been numerous works that studied the behavior at the software-level, there are only recent works that attempt to understand their behavior at the hardware-level that can be exploited and propose a mitigation solution.

§Equal contribution
[1]https://nvd.nist.gov/vuln/full-listing
[2]https://cwe.mitre.org/
[3]McAfee Labs, "COVID-19 Threats Report," Technical report, 2020.

978-1-7281-8899-7/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSC50631.2021.00045

NVD Database hosted by the National Institute of Standards and Technology (NIST) provides data feeds that contain several key concepts and a subset of concepts that we found most relevant to this work including: a description of a particular vulnerability, CVE-ID, CWE-ID, CVSS score, CVSS severity impact, references to the websites that describe this vulnerability, modified and created date, etc. In this work we also use CWE Dataset for connecting root causes for each instance of the vulnerability input. On the CWE website, each CWE is described as "Weaknesses are flaws, faults, bugs, vulnerabilities, or other errors in software or hardware that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack." We use this community contributed Dataset to describe vulnerabilities in our ontology-built relationships.

Authors in [9] propose a classification of device-related hardware vulnerability data for IoT and Industrial IoT equipment. In [9], authors divide the CVE records from NVD database into 7 distinct categories based on market and scope of IoT applications and the database samples were hand-classified by authors based on expert knowledge. After hand-classification, authors utilize a Support Vector Machine (SVM) classifier on device and vulnerability data to predict and mitigate threats resulting from vulnerabilities. Furthermore, authors in [10] propose an integrated data mining framework to automatically lay out how vulnerabilities develop over time and detect the evolution of a specific cyber-security threat. Authors in [10] use a Supervised Topical Evolution Model (STEM), which discovers temporal themes from a text corpus with the aim to discover trends in vulnerabilities based on latent structure of corpus and by incorporating temporal feature of vulnerability reports. Then, they use a diffusion-based storytelling technique that sifts through past vulnerability reports to describe how a current threat has evolved and study the evolution of a vulnerability as a series of connected vulnerability reports. Moreover, in [11] authors propose a classification approach using Latent Dirichlet Allocation (LDA), an unsupervised learning technique, on the description texts of CVE entries in order to develop a classification system, to identify prevalent topics and emerging trends in an automated fashion. Authors in [12] investigate use of historical patterns of vulnerabilities to predict future vulnerabilities in software applications.Moreover, authors report that they did not find any statistical significance in trends of occurrence of vulnerabilities over time. An interesting finding of this study is that sequential patterns of vulnerability events follow a first order Markov property that is, they can predict next vulnerability using only the previous vulnerability.

All prior works have chosen the software vulnerabilities as the focus of their study and most prior works despite their valuable effort, fail to provide a storytelling framework that can find useful information on the relationship and trends within the CVE and CWE datasets and offer detailed insight as to how a vulnerability and impacts have evolved over time. In this paper, our aim is on explore the trends and relationship between vulnerabilities over a period of time with a focus on IoT hardware vulnerabilities.

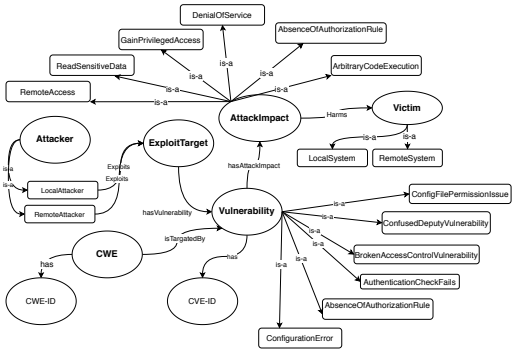


Fig. 1: Hardware Security Ontology Framework

III. Proposed Ontology and Framework

In this section we describe our proposed ontology framework. In order to capture the vulnerability threat evolution in the IoT hardware-security domain, our ontology has six concept classes named Vulnerability, CWE, AttackImpact, Attacker, ExploitTarget, and Victim. Here we provide a brief description of our ontology classes: **1) Vulnerability**: A vulnerability is a software or hardware related flaw of a system that could be exploited by an attacker that might lead to a certain kind of attack; **2) Victim**: An entity that is susceptible to an attack due to the presence of the vulnerability within the system; **3) AttackImpact**: Once the attacker exploits a vulnerability, it will result in an impact on the system. The "AttackImpact" class captures the concept of several possible impacts on a system as a consequence of the attack; **4) Attacker**: An adversary who tries to exploit a vulnerability to gain access of the victim machine; **5) CWE**: The CWE class represents the weakness type associate to a particular vulnerability in a system using a unique ID. **6) ExploitTarget**: A victim system has one or many vulnerabilities that open up a door to the attacker to exploit these vulnerabilities and harm the victim, which are defined as ExploitTarget class. Fig. 1 shows the entities and relationship between them. This relationship is termed as the "object property" that connects a subject with an object.

To capture the evolution process of NVD vulnerabilities, in this work, we capture the relationship between vulnerabilities and the impact that each vulnerability has on victim systems once being exploited by the attacker. We make these connections using Ontology technique to map one entity instance to another to capture the relationship between vulnerabilities and their impacts. Our ontology maps each vulnerability with a corresponding CVE-ID. If a design has a fault that is responsible for opening a door for the attacker to exploit that faulty part is a subclass of a vulnerability. Each of the vulnerability class is linked to a CWE-ID that defines the type of the vulnerability. We capture all of the possible impacts caused by a certain vulnerability in the AttackImpact class. We update our model by automatically labeling the entities of the new data and map each entity to equivalent classes

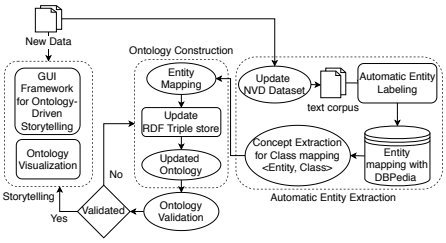


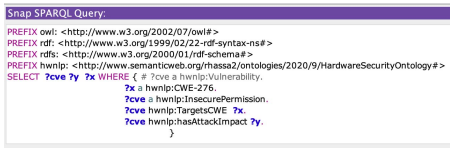
Fig. 2: Ontology-Driven Storytelling Framework (OSF)

and assign the relationship between the entities and the object properties dynamically and store them as RDF triple state.

The entire corpus to model the problem is data from CWE and CVE datasets. We use all the individual CVEs from the time period of 2010 to Present as initial classes in our model. We use an ngram formulation function from NLTK to form bigrams, trigrams, and quadgrams of our original data from the dataset available from CVE and CWE to form a set of combinations. We propose an algorithm that identifies the main subject and the specific target that is being affected after experimenting with the CVE data and observe that the target to these vulnerabilities are either organizations or names that could usually be identified as people. We use part-of-speech tagging using the spaCy NLP library to explore all named entity recognition to identify a "real-world object" that has an assigned name. This includes finding all 'PERSON' and 'ORG' initially. A fail-safe measure we noticed after analysing all of the vulnerability descriptions is to also get the NN noun - singular & NNS noun - plural from the NLTK POS-tagger instead to also identify the most probable noun that can represent the target. To find the frequency distribution of these occurring ngrams for each specific vulnerability, we formulate the output of the previous function using ngrams and apply Collocations Association Measures function to rank the scores.

We link each CVE to a CWE and calculate the similarity metric of the input description and each CVE description using 'cosine similarity' between each vector to identify the most relevant topics to the input. To properly utilize these analysis methods and improve visibility of the cyber-security threats, we have used PyQt5 to implement a GUI that provides an interactive experience, as shown in Fig. 5. Using our proposed GUI, the user can provide a description of a vulnerability and receive the data in an organized view. This allows the user to understand the type of vulnerability and the distribution of CWEs that could explain the scenario. Our Ontology-Driven Storytelling Framework (OSF) is shown in Fig. 2.

We use the NLTK module to remove stop-words, punctuation, nonsensical data, and then finally tokenize and perform stemming to standardize the meaning of the description. We use our developed ontology to connect these occurrences and show connections of these instances to provide a story of development to the user and display this using OntoSpy to visualize the RDF models and to interact with documentation as shown in Fig. 4, only most important classes are expanded here due to the space constraint.



(a)

?cve	?impact	?cwe
hwnlp:CVE-2018-12441	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2018-12441	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276
hwnlp:CVE-2019-14737	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2019-14737	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276
hwnlp:CVE-2019-14925	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2019-14925	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276
hwnlp:CVE-2018-17860	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2018-17860	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276
hwnlp:CVE-2012-5578	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2012-5578	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276
hwnlp:CVE-2018-19592	hwnlp:WorldReadableFiles	hwnlp:CWE-276
hwnlp:CVE-2018-19592	hwnlp:GainAccessSensitiveInformation	hwnlp:CWE-276

12 results

(b)

Fig. 3: Attack Impacts of Vulnerabilities related to CWE-276

IV. Data Analysis and Discussion

We develop our proposed ontology in Protege 5.5.0 using Ontology Web Language (OWL), version-2. Our current ontology has 1,460 Axioms with 652 Logical Axioms and 801 Declaration Axioms. There are 252 Classes, 32 object properties, and 518 Individuals.We consider the following question for our case-study, *What kind of possible Attack Impacts can a victim system experience for a particular vulnerability that links to a specific CWE-ID (hardware weakness)?* To answer this question, first, we need to know the vulnerability class for queried vulnerability instance. Second, we need to know the possible impacts for exploiting that vulnerability. As each vulnerability is mapped to a single and/or multiple instances of the "AttackImpact" class by "hasAttackImpact" relationship, this object property will lead us to all of the possible Attack Impacts for that vulnerability. At last, each vulnerability targets a specific CWE-ID using "targetsCWE" object property, which links our target vulnerability to a particular CWE-ID. Analyzing the attack pattern starts by querying a text input that maps to a certain vulnerability. A sample input text is given to our topic modeling framework which maps that input text to a vulnerability named "InsecurePermission" using cosine similarity matching. Our goal is to find the corresponding CVE-ID, CWE-ID, and the AttackImpact corresponding to that vulnerability. Fig. 3a shows the SPARQL query for extracting the connection between the "InsecurePermission" vulnerability and the corresponding CWE-276. The query returns 12 responses shown in Fig. 3b. In this case, every CVE-ID links to the weakness type CWE-276.

After the algorithm computes the results, the user is provided with a dashboard of analytic that provide detailed explanation of the different metrics and values. A tabular view shows the most relevant CVEs and their respective specifications is provided. This table lists a metric to show how similar each vulnerability description is to the user input, a CWE and a CVE ID, a description to explain how the vulnerability is relevant, and the CVSS V3 score to access and classify levels of threat in terms of exploit and impact score.

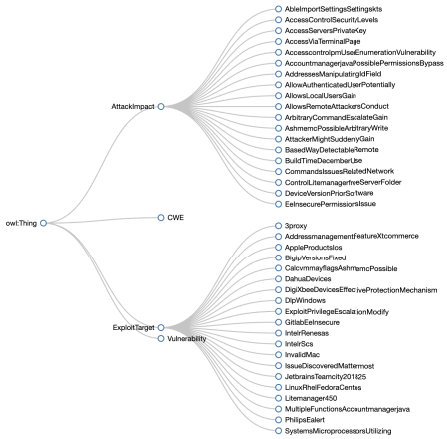


Fig. 4: OntoSpy Visuals of the Ontology Interaction

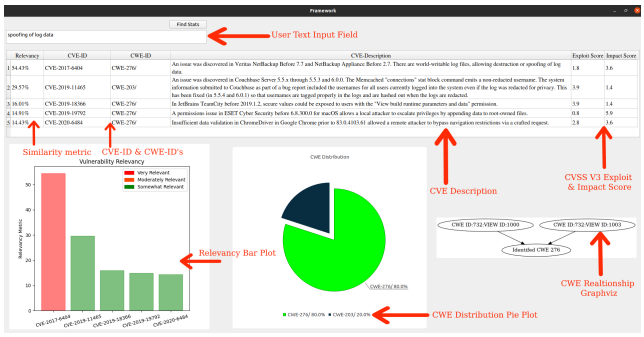


Fig. 5: GUI Demo

A graph shows the relevancy of these calculated CVEs with color coding to show similarity level in the GUI according to the similarity scores calculated by cosine similarity algorithm using the following Similarity Score (SS) thresholds determined after experimenting results: SS≥40%: Very Relevant, 40%>SS≥30%: Moderately Relevant, and 30%>SS≥20%: Somewhat Relevant. A pie chart shows the most occurring CWE that could directly connect to the cause of this common weakness. The biggest CWE slice will be used in the storytelling process to connect the matching instances. A graphViz rendering of the selected CWE that shows the relationship with other CWEs such as parent, child, and peer. We use a tool called OntoSpy to visualize these occurrences with the selected CWE and CVE IDs. This tools allows us to view the documentation and visuals in many relevant forms such as: HTML documentation, interactive bubble chart, partition table, rotating cluster, force directed graph, etc.

To validate our proposed OSF, we use "spoofing of log data" as the user input to the GUI for an observable vulnerability we are facing. The GUI starts making connections among instances of CVE that are available in our corpus. Then the GUI cleans the string and calculates the relevance among the vulnerabilities using a similarity function such as cosine distance and rank the closest match found from our database.

We can observe the first result showing '54.43%' similarity with the input. The second and third rows display the CVE-IDs and CWE-IDs of the entities that match with our input. The fourth row shows the description of the CVE we matched against from the NVD dataset. In our example, GUI shows "An issue was discovered in Veritas NetBackup Before 7.7 and NetBackup Appliance Before 2.7. There are world-writable log files, allowing destruction or spoofing of log data." From our result, we can observe how 'spoofing of log data' is a clear exploit that occurred from an issue discovered in 'Veritas NetBackup' for a certain version. We then query our ontology to tell us the story regarding the possible attack impact and root cause associated to the CVE-ID that has the highest relevance to the user text input.

V. Conclusion

In an effort to better analyze the vulnerabilities and impacts provided by NVD, we developed a novel Ontology-driven Storytelling Framework (OSF). Proposed OSF is capable of highlighting a range of relevant stories on the fly only from a plain text input. Furthermore, OSF can readily identify possible attack impacts and scope as well as related weakness for each vulnerability. Additionally, we have developed a GUI that provides all of the vulnerability and impact analysis information in an organized and meaningful fashion. To the best of our knowledge this is the first ontology-driven storytelling framework focusing on IoT hardware vulnerabilities.

Acknowledgment

This work was supported in part by the National Science Foundation (NSF) through Computing Research Association for CIFellows Project 2030859.

References

[1] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3. IEEE Press, 2014, pp. 361–372.

[2] Y. Yarom *et al.*, "FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack." in *USENIX Security Symposium*, vol. 1, 2014, pp. 22–25.

[3] P. Kocher *et al.*, "Spectre Attacks: Exploiting Speculative Execution," in *IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[4] F. Liu *et al.*, "Last-level cache side-channel attacks are practical," in *SP*. IEEE, 2015, pp. 605–622.

[5] D. Gruss *et al.*, "Flush+ Flush: a fast and stealthy cache attack," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 279–299.

[6] Z. He *et al.*, "How secure is your cache against side-channel attacks?" in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, 2017.

[7] W. Zhou *et al.*, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, 2019.

[8] K. Yang *et al.*, "Hardware Designs for Security in Ultra-Low-Power IoT Systems: An Overview and Survey," 2017.

[9] G. J. Blinowski *et al.*, "CVE Based Classification of Vulnerable IoT Systems," in *Advances in Intelligent Systems and Computing*, 2020.

[10] M. A. Williams *et al.*, "Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database," in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2019.

[11] S. Neuhaus *et al.*, "Security trend analysis with CVE topic models," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2010.

[12] S. S. Murtaza *et al.*, "Mining trends and patterns of software vulnerabilities," *Journal of Systems and Software*, 2016.