A Cross-Hypervisor Analysis of Recurrent Vulnerability Categories in Virtualization Subsystems

ALEXANDRU DAN CULDA, University of Twente, The Netherlands

Virtualization is a cornerstone technology in modern computing, underpinning everything from cloud services to enterprise IT infrastructure. Despite proprietary architectures or vendor-specific implementations, certain categories of vulnerabilities might be present across multiple hypervisors, underlying a possible systemic flaw. This study does a quantitative meta-analysis of 1,536 vulnerabilities through means of secondary research, spanning over four major hypervisors - VMware, Xen, Hyper-V, and KVM. It uses the National Vulnerability Dataset (NVD) as the sole resource for dataset creation, gathering vulnerabilities, also referred to as Common Vulnerabilities and Exposures (CVE), from 2009 up to 2024. Moreover, this paper proposes a 6-category taxonomy based on hypervisor functionalities - CPU, Device I/O, Control & Execution, Interrupts, and Timer Mechanisms (ITM), Soft Memory Management Unit (Soft MMU), and Add-ons & Management - paired with the industry-standard Common Weakness Enumeration (CWE) classification in order to outline possible conceptual or systemic flaws present across the four hypervisors. The 1,536 vulnerabilities were classified into the six aforementioned categories using an implementation of the Self-Attention Deep Neural Network (SA-DNN) presented by Vishnu et al. [6] because it demonstrates superior performance in vulnerability classifications. Additionally, further analysis is performed on the vulnerability distribution by category (Fig. 1), temporal trends across the 16 years (Fig. 2), and distribution of the 6-category taxonomy across hypervisors (Fig. 3). The contribution of the study to the state of the art is that it introduces a novel dual-stratified classification meant to bridge the gap between granular and vendor-specific vulnerabilities and high-level systemic insights across the four major players in the virtualization market. The intended audience of the present paper is fellow students and researchers that present interests in the field of cybersecurity, while having prior, Bachelor-level knowledge of Natural Language Processing models, virtualization technologies, and vulnerability reporting and documenting.

CCS Concepts: • **Security and privacy → Virtualization and security**.

Additional Key Words and Phrases: Virtual Machine; virtual security; cloud security; cross-Hypervisor, meta-bug

1 INTRODUCTION

1.1 Context and Motivation

Virtual Machines are virtual systems that imitate physical ones, with the exception of the hardware being virtualized. This virtualization of the underlying infrastructure is managed by a Virtual Machine Monitor (VMM), also known as a hypervisor. A VMM is a virtualization technology that runs either alongside (Type 2) or under (Type 1) an operating system [2]. Given the highly sensitive environment in which processes related to the VMM are running, the hypervisors represent a technology with potentially high rewards, albeit difficult to exploit due to the inherent technological knowledge required about computer architecture, organization, and proprietary, closed-source software. There are two types of VMMs: *type 1*, which refers to hypervisors that are directly running onto the hardware, also known as bare-metal hypervisors, and *type 2*, which runs alongside the existing operating system (OS), also known as hosted. The present study is motivated by the importance of virtualization in modern computing services and the high stakes of possibly discovering hidden cross-hypervisor patterns. Although future work is needed for a statistically proven correlation of identical vulnerability exploitations between hypervisors, this study serves as an excellent starting point for subsequent research.

1.2 Specific Problem

The specific problem stems from the assumption that vulnerabilities present in hypervisor implementations are unique, isolated, and vendor-specific, despite the shared technological features and concepts between virtualization solutions. This study challenges that assumption, measuring and answering if systemic flaws recur across the four hypervisors, despite the varied proprietary and technological capabilities. If flaws categorized based on hypervisor functionalities are deemed to be easy to pivot from one hypervisor to another, the security implications could be considerable for all stakeholders. The four hypervisors were selected due to a survey showing that 93% of the market share is represented by four main hypervisors - two closed-source (VMware and Microsoft Hyper-V) and two open-source (Xen and KVM) [4]. As such, others would represent too small of a market portion, deeming them unimportant for the analysis.

1.3 Research Questions

The aim of this study is to identify systemic and conceptual vulnerabilities within VMware, Xen, KVM, and Hyper-V, aiming to assess the ease with which attackers can pivot from one hypervisor to another by exploiting identical flaws. As a guide, the following research questions are formulated, each of which is addressed in section 6:

(1) Which hypervisor functionalities account for the largest share in virtualized environments?
(2) How are vulnerabilities distributed across Hyper-V, KVM, VMware and Xen when categorized based on hypervisor functionalities?
(3) Do conceptually similar vulnerabilities recur across Hyper-V, KVM, VMware and Xen?

1.4 Approach

This study employs a quantitative meta-analysis of 1,536 vulnerabilities from 2009 to 2024, acquired from NVD. The approach begins

Author's address: Alexandru Dan Culda, a.culda@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

TScIT 42, January 29, 2025, Enschede, The Netherlands

with collecting, organizing, preparing, and providing vulnerabilities to the SA-DNN model, which labels the vulnerabilities into six distinct functionality-based categories. Vulnerabilities are participating in a pre-processing stage of Tokenization, Case Normalization, Curation, and Lemmatization, forming a Vocabulary Matrix with padded vectors. The vectors are mapped to pre-trained, 6 billion parameters, 100-dimensional GloVe embeddings and given to the SA-DNN model. The model is comprised of an Embedding layer, aforementioned in this paragraph, a Bi-directional Long Short-Term Memory (Bi-LSTM) layer, an Attention Layer, one Flatten, and two Dense rounds [6]. A standard 80/20 train-test split is performed on the labelled data, with Precision, Recall, and F1-scores being provided for each category. The process, coupled with a standardized CWE categorization, depicts that, whilst attackers can pivot some attacks from one hypervisor to another, the degree to which it is possible vastly depends on the category and involved hypervisors. Certain categories lead to no evidence of cross-hypervisor meta-bugs, whereas others do.

1.5 Structure

The study commences with explanations in regards to Section 3.1 and Section 3.2, whilst explaining the source from which the proposed 6-category taxonomy stems. Afterwards, the SA-DNN model is detailed. Accuracy metrics are provided for the text-mining model. Section 4 regarding vulnerability trends and patterns follows. Finally, Section 6 rounds the study, offering potential explanations for the results and answering the 3 Research Questions.

2 RELATED WORK

Perez-Botero et al. [4] conducted an early analysis on hypervisor vulnerabilities in 2013, albeit the study focused solely on Xen and KVM, prioritizing the characteristic of being open-source. Their research introduced, among classification systems based on the attack source and target, 11 functionalities of traditional hypervisors as attack vectors. Whilst the work provided a solid knowledge base and taxonomy for subsequent research, their analysis was limited to just two of the four major players in the market. Moreover, their vulnerability dataset was significantly smaller, spanning over 97 vulnerabilities, limiting the generalization aspect that my paper considers. Hence, this research takes upon the research and does an exhaustive research based on all four hypervisors, while abstractifying the 11 categories into 6 due to the necessities of NLP models, all while enhancing the existing taxonomy with a doubled-layer classification that embeds CWE IDs.

Russo et al. [5] addressed the challenge of processing vulnerability descriptions through means of natural language processing. The study proposes CVErizer, a method to automatically extract summaries of vulnerabilities and categorize them based on a clearly defined taxonomy system [5]. Whilst proving itself to be of much use, especially due to the comparison between multiple ML algorithms, the model was trained on 3,369 vulnerabilities. Moreover, no emphasis was paid to a cross-hypervisor analysis, incapacitating the formulation of conclusions in regards to meta-bug patterns. As such, my work builds upon their classification, employing deep neural networks for a better semantic understanding. By combining the aforementioned improvement with the novel 6-category taxonomy, the current research underlines how vulnerabilities manifest across multiple virtualized environments, rather than solely improving the readability of CVEs.

Hypervisor	**Products**
Microsoft Hyper-V	Hyper-V on Windows
Xen	-
KVM	-
VMware	VMware vSphere Foundations Workstation Suite Fusion Suite Cloud Foundation View/Horizon Suite VMware Cloud Other

Table 1. List of Covered Hypervisors and Their Products

Vishnu et al. [6] presented a novel model for categorizing vulnerabilities— a Self-Attention Deep Neural Network. The results, which depict better metrics than existing state-of-the-art models, are sufficiently convincing for subsequent research that involves vulnerability classification based on text-mining techniques. The paper built upon the results of Russo et al. [5] by employing deep neural techniques and expanding the dataset with the work of Vishnu et al. [6]. The research undertaken in the work of Vishnu et al. [6] did not emphasize a cross-hypervisor analysis, but rather a generic approach to all existing vulnerabilities in the dataset. Further on, a generic taxonomy has been deployed, which, whilst proving itself the right approach for the given problem, would not have brought novel conclusions into this analysis. My study builds upon the presented SA-DNN model by, firstly, suggesting lemmatization methods instead of word stemming, and, secondly, by applying a taxonomy better suited to the requirements of the present dataset.

3 METHODOLOGY

3.1 Data Collection

The preliminary dataset used for the research was acquired from the National Standardized Vulnerability (NVD) database of the United States of America [1]. The NVD offers downloadable data feeds from 2002 to 2025.

The research is based upon data from 2009 to 2024. Data from earlier years are not included in the present research, as some hypervisors could accumulate more documented vulnerabilities simply due to being on the market for a longer time. As Microsoft Hyper-V was released in 2008, KVM in 2007, Xen in 2003, and VMware in 1998, the bias that stems from uneven release dates is reduced by ensuring the initial year is 2009.

Moreover, the dataset doesn't contain CVEs that are rejected or disputed. It is considered that the dataset must be concise, up-to-date, and, last but not least, a valid list of vulnerabilities.

Attribute	Description
CVE ID	Uniquely identifies each dataset entry. Example: `CVE-2014-1776`.
Year	Used for further research based on dataset trends. This attribute supports longitudinal analysis
CWE ID	Classification of software vulnerabilities into granular categories. IDs are standardized and maintained by NVD
Target	High-level overview of affected hypervisors. Each item is represented by a vendor and its product(s)
Description	Used for classification via Natural Language Processing. Provides contextual understanding of vulnerabilities
Category	Broad classification of vulnerabilities for cross-hypervisor analysis

Table 2. List of Attributes and Their Meaning

3.2 Data Organization

In total, 1536 vulnerabilities were gathered. 561 (36,52%) targeted VMware products, 383 (24,93%) targeted Xen, 235 (15,29%) targeted KVM and 213 (13,86%) Hyper-V. Note that some vulnerabilities target two or more hypervisors, so that there might be overlaps in the counting. The list of CVE files went under a curation process. For clarity purposes, the attributes are presented and described in Table 2.

The dual-stratified categorization has two levels of granularity and purposes. The first type of categorization lies on the Common Weakness Enumeration (CWE), the industry's standard proposed by the MITRE Organization[1]. The community-driven taxonomy is presented as a hierarchical tree of low-level vulnerabilities, where each item has a base, class, and variant. The second type of categorization lies on the work of Perez-Botero et al. [4], which outlines 11 functionalities offered by traditional hypervisors.

The 11 functionalities would have formed an overly granular classification for the present 1536 vulnerabilities that used a corpus for the SA-DNN, so an abstraction has been performed as follows:

- **CPU**
  - Virtual CPUs
  - Symmetric Multiprocessing (SMP)
- **Device I/O**
  - I/O and Networking
  - Paravirtualized I/O
- **Control & Execution**
  - VM Exits
  - Hypercalls
- **Add-ons & Mgmt**
  - VM Management (configure, start, pause, and stop VMs)
  - Remote Management Software
  - Hypervisor Add-ons
- **Interrupt Timer Mechanisms (ITM)**
- **Soft MMU**

As such, hereby a 6-category taxonomy is adhered to. The CWE taxonomy is kept as-is, ensuring standardized vulnerability identification, whereas the 6-category taxonomy is present due to providing a way to reach the research goals.

3.3 Data Preparation

The number of 1536 vulnerabilities was deemed too high for a manual classification; hence, a Natural Language Processing (NLP) model was employed. In those regards, the work of Vishnu et al. [6] is present, which proposes a Self Attention - Deep Neural Network (SA-DNN) to identify the category of vulnerabilities from their description via text mining approaches. The DNN was chosen because it proves itself to be more effective at categorizing vulnerabilities based on their description, with the model outperforming other DNN models - Convolutional Neural Networks Long Short-Term Memory Networks (CNN-LSTM), Graph Convolutional Networks (GCN), and Support Vector Machine (SVM). Moreover, the ambiguity and complex security contents of hypervisor descriptions require a supervised learning model as a way to learn precise mappings to the proposed 6-category taxonomy. The following steps have been performed after the architectural description of the SA-DNN presented in [6], with one improvement - this research replaces word stemming with lemmatization.

3.3.1 Corpus Pre-Processing. The preprocessing stage involved several intermediary steps: Tokenization, Case Normalization, Description Curation, and, building upon the presented architecture and improving it, Lemmatization. The final step preceding *Corpus Training, Testing and Modelling* is characterized by Feature Extraction. For a better representation of the meaning and purpose of each step, an example is provided;

- **'Windows Hyper-V, as in Windows 11, has a Remote Code Execution Vulnerability!'**.

Tokenization. The tokenization of vulnerability descriptions represents a method through which sentences are divided into smaller units, usually words which are also known as *tokens*. Text mining models require granular representations of paragraphs to better process and understand the semantic correlation between different words. The architectural implementation included, at this step, the removal of punctuation or special characters, such as *!*, *@*, *,* or *.*, through RegexpTokenizer from NLTK.

- **'Windows', 'Hyper-V', 'as', 'in', 'Windows', '11', 'has', 'a', 'Remote', 'Code', 'Execution', 'Vulnerability'**.

Case Normalization. Case normalization is needed to ensure consistency and avoid mistreating the words due to case variations. This step is crucial due to the current text mining implementation assigning weights to the meaning of words based on the context [6]. Using native Python built-in functionalities, all tokens were transformed into their lower case form.

- **'windows', 'hyper-v', 'as', 'in', 'windows', '11', 'has', 'a', 'remote', 'code', 'execution', 'vulnerability'**.

[1] https://cwe.mitre.org/about/index.html

Description Curation. In order to exclude irrelevant noise, this process implied removing stopwords, numbers, hexadecimals, and special cases. Stopword tokens, such as "my", "on", "during", or "this" do not add meaningful context to vulnerability descriptions. The NLTK built-in list of stopwords was used for filtering. Further on, strings of words representing hyperlinks, memory addresses, and byte packets were removed by only including words with a length of at least 3 characters (to exclude instances such as *a3 80 00 00 2f 9b* and no more than 2 repeating letters (to exclude instances such as *ffffffffda*). However, a check for words with a digit at the end was included for meaningful words such as *vmxnet3*.

- **'windows', 'hyper-v', 'has', 'windows', 'remote', 'code', 'execution', 'vulnerability'**.

Lemmatization. As an improvement brought to the architecture presented in the work of Vishnu et al. [6], lemmatization is used instead of word stemming. Lemmatization is similar to stemming, although is considered the superior method due to morphologically analysing the words, resolving the ambiguousness of English[2]. For example, the verb "have" might as appear as "had", "having" or "haven't", and lemmatization converges all variations into "have". The WordNetLemmatizer library from NLTK has been used, such that each token of each description is processed to acquire the nltk_tag [3]. The 36 categories of NLTK tags are mapped to 4 broad categories (verbs, nouns, adjectives, adverbs) considering the first NLTK tag letter (E.g., The NLTK tag starts with the letter **J**, therefore it is mapped to an *adjective*). The resulting WordNet tag is analysed and the vast majority of words are categorized, albeit technical words specific to vulnerability descriptions are kept as-is if the tag is *None*.

- **'window', 'hyper-v', 'have', 'window', 'remote', 'code', 'execution', 'vulnerability'**.

Feature Extraction. The curated corpus served as a basis for the process of building a Vocabulary Matrix, a matrix of all unique words that appear across all 1536 vulnerabilities. Each word in the vocabulary was assigned a unique index, and these indices were then organized into the Vocabulary Matrix, where each row corresponds to a vulnerability description and each cell represents the index of a word. Each row was then padded with 0's up to size 315, as the SA-DNN model requires equal-size vectors. The 315 represents the maximum size of a description in the corpus. Then, a different matrix was created during the process called Word Embedding, the methodology undertaken in the Embedding layer. Word Embedding refers to the process of attributing numerical values to the correlation between words. The co-existence of words in a corpus might reveal meaningful correlations that are especially useful in the context of textual mining for vulnerability classifications. GloVe (Global Vectors for Word Representation) [3] creates a matrix of word vectors by calculating the probability of each word occurrence across the entire text dataset. Although a train on any given corpus is possible, the restricted size of the present corpus prompted the research to use GloVe's 6-billion pre-trained embedding represented as 100-dimensional vectors.

[2]https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/
[3]https:/www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Category	**Keyword**
Soft MMU	["page table", "shadow page", "TLB", "paging"]
Control & Execution	["Remote Code Execution"]
CPU	["vCPU", "SMP", "CPU scheduling", "multi-threading", "Intel", "AMD"]
Device I/O	["paravirtual", "emulator", "PCI", "IOMMU", "network stack", "I/O buffer overflow", "vmxnet3", "NIC"]
ITM	["APIC", "timer", "pit", "irq", "idt", "trap", "interrupt"]
Add-ons & Mgmt	["vCenter", "plug-in", "management interface", "configuration file"]

Table 3. Category-to-Keywords mapping for training-test dataset

Therefore, every cell in Vocabulary Matrix was replaced with the corresponding GloVe vector. Then, the result was presented to the Bi-LSTM layer. The 6 target labels underwent a one-hot encoding process to enable multi-class classification [6].

3.3.2 Corpus Training, Testing and Modelling. 536 vulnerabilities out of 1536 were selected based on keywords for each category, as can be seen in Table 3. The 536 vulnerabilities were preprocessed as described in the aforementioned section and processed into a Bidirectional-Long Short-Term Memory Layer (Bi-LSTM) and an Attention Layer. Then, the results are flattened into a 1D vector, as the last 2 steps, the Dense steps, expect a 1D input. The third-to-last and second-to-last steps are paired with a Dropout layer in order to overcome overfitting. The final step, the output layer, is activated with the Softmax function. All layers are abstracted with the aid of the Keras library.

Out of the 536 vulnerabilities, 80% were assigned to training and 20% for testing. After multiple fine-tunes of hyperparameters, it seems that the optimal configuration for the present 6-class problem is: Epochs=15, Batch size=32, Attention Layer Dropout=0.3, Dropout Layer=0.2, Early stopping=5.

4 RESULTS

	Precision	Recall	F1-Score	Support
CPU	0.75	0.75	0.75	32
Soft MMU	0.77	0.83	0.80	12
ITM	0.36	0.36	0.36	14
Device I/O	0.71	0.67	0.69	18
Control & Execution	1.00	1.00	1.00	16
Add-ons & Mgmt	1.00	1.00	1.00	15
Accuracy			0.77	107
Macro Avg	0.76	0.77	0.77	107
Weighted Avg	0.77	0.77	0.77	107

Table 4. Classification Report for Vulnerability Categories

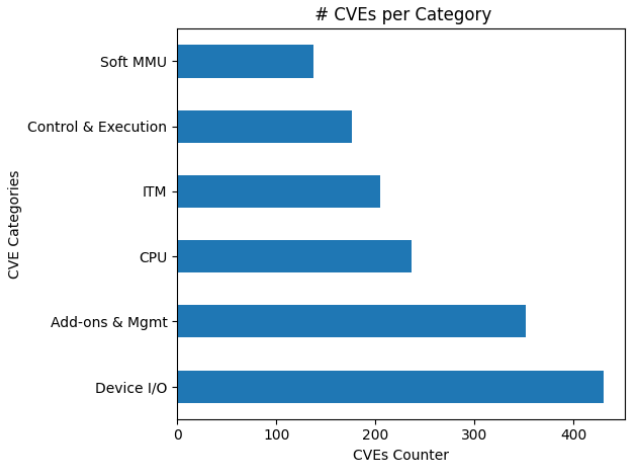

Fig. 1. Number of Vulnerabilities per Category

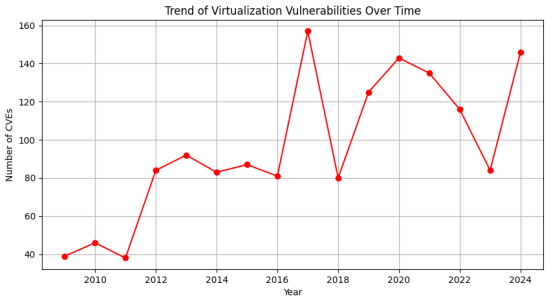

Fig. 2. Trend of Virtualization Vulnerabilities Over Time

The precision, recall and F1-score presented in Table 4 are further affected by one outlier, the Interrupt and Timer Mechanisms (ITM). The following results reflect the analysis of a dataset that, due to its niche nature, is limited in size.

Figure 1 illustrates the distribution of the 1536 vulnerabilities under the 6-category classification. The results align with the expectations set by Perez-Botero et al. [4], as the Device Emulation categories (i.e., I/O and Networking and Paravirtualized I/O) accounted for more than one-third in their analysis. The vast attack surface created by the I/O Devices or Add-ons and Management Tools represents a tempting target for attackers. Their lower technical barrier and, in some cases, open-source characteristic make them more accessible for researchers and professionals. In contrast, memory-related vulnerabilities - such as the ones associated with the Soft MMU category - are considerably fewer due to being complex and challenging to execute. The inherent complexity and heightened risk of technical and operational errors act as barriers that safeguard hypervisors.

Figure 2 presents the temporal trend of vulnerabilities from 2009 up to 2024. A steady increase can be seen from 2009 to 2012, followed by a plateau of the following 4 years. A surge in 2017 follows after

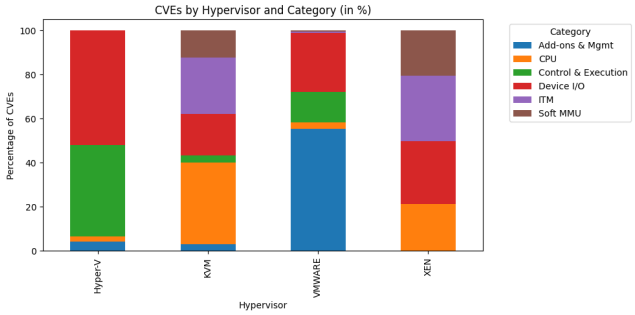

Fig. 3. Bar Graph of Categories Distribution by Hypervisor

that. Such an abnormal variation is thought to be attributed to the security patches that followed the infamous low-level Meltdown and SPECTRE attacks. A decline in reported vulnerability is observed between 2020 and 2023, which may be attributed to the reduced research and delayed disclosure process under the pandemic context. 2024 appears to be the starting point for an ascending slope that will characterize the following years.

The bar chart presented in Figure 3 depicts the distribution of vulnerability categories across the four hypervisors. Hyper-V has the largest distribution across Control & Execution and Device I/O. The split reveals that Hyper-V is susceptible to more attacks targeted to hardware emulation or execution control, possibly due to the integration within an equally complex software - Windows, an Operating System. Control & Execution actions, such as API calls or hypercalls possible are reasons for the predominant two categories. It is crucial to specify that that present research pertains to the stand-alone version of Microsoft Hyper-V; the version included in Microsoft Server environments exhibits separate characteristics.

KVM and Xen present a more balanced categorical distribution. When compared to Hyper-V, KVM has a noticeable drop in Control & Execution and Device I/O-based flaws, possibly due to a deeper integration within the Linux kernel; hence, it offers the hypervisor more mature privilege separation mechanisms. Its largest two vulnerability categories, CPU and ITM, suggest that, as a drawback of its deeper integration within the Linux kernel, precise timing attacks over the CPU are easier to execute. The reason might be likely due to its reliance on Linux Kernel scheduling.

VMware is severely prone to Add-ons & Management attacks. The vast exposed surface possibly stems from its unique complex infrastructure such as: extensions through enterprise integrations, web-based management interfaces, APIs, or cloud management tools. As a characteristic of closed-source products, including VMware, there is a minimal number of Soft MMU-based vulnerabilities. A possible trade-off is visible - clients choose between the pricey and more opaque closed-source hypervisors, which may reduce the attack surface, or the low costs and transparency of open-source alternatives, which are possibly more susceptible to sophisticated attacks.

Xen, as opposed to other hypervisors, exhibits a nearly equal distribution of Soft MMU, ITM, CPU, and Device I/O-based vulnerabilities. The disproportionately high share of Soft MMU attacks

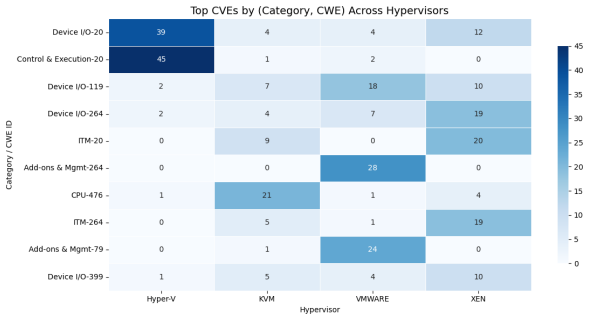

Fig. 4. Heatmap of the 6 categories grouped by CWE on the 4 hypervisors

might stem from Xen's architecture that emphasizes paravirtualization, exposing it to low-level virtualization mechanisms such as shadow paging. Device I/O vulnerabilities remain steadily at approximately 20%, with minimal variation between hypervisors. The distribution of the category depicts how Xen, alongside VMware, KVM, and Hyper-V, are more prone to virtualized device emulation attacks. A possible explanation would be the widespread adoption of shared emulated devices, alongside the lower technical barrier for exploitation.

Figure 4 presents a structured breakdown of CVEs across the four analysed hypervisors. It categorizes the vulnerabilities based on the proposed taxonomy and their associated CWE IDs. The discrepancy between Hyper-V and the rest of the three counterparts is noticeable, with the exception of CWE ID 20 (Improper Input Validation) in Control & Execution and Device I/O-based attacks. There is evidence that vulnerabilities targeting Hyper-V are not transferable to other hypervisors. Hyper-V exhibits signs of (Device I/O + CWE-20) and (Control & Execution + CWE-20) issues, possibly due to a deeper integration with the proprietary Windows ecosystem. Certain flaw types (e.g., Device I/O and ITM) showcase evidence of meta-bugs appearing across KVM, VMware, and Xen, whereas the Add-ons & Management or CPU appear to be more vendor-specific. With the exception of Device I/O attacks coupled with CWE-119 (i.e., Improper Restriction of Operations within the Bounds of a Memory), Xen exhibits a lower degree of transferability, indicating that its paravirtualization emphasis vastly affects the exposure to different classes of attacks.

5 THREATS TO VALIDITY

5.1 Construct Validity

Specific to this study, the construct validity refers to the 6 categories used as a labelling method for the 1,536 vulnerabilities. The number of 6 categories was used to achieve an optimal balance between granularity and abstraction, providing the ideal input for the accuracy of the model. The type of obtained results are in line with what was expected before performing the analysis.

5.2 Internal Validity

In the case of the present study, confounding factors affecting internal validity are expressed through inconsistencies or errors within the NVD dataset and selection bias stemming from years when certain hypervisors were not yet released. The first confounding factor was addressed by eliminating vulnerabilities that were classified as Rejected. The second factor was addressed by initiating the analysis from the year immediately after the most recently released hypervisor.

5.3 External Validity

The small corpus of 1,536 vulnerabilities, albeit inherently small due to the complexity of the topic, limits the generalizability of the findings. Although NVD presents itself as an exhaustive dataset, certain real vulnerabilities might not be present. The aforementioned two aspects count as threats to the validity.

5.4 Conclusion Validity

Future research should aim to improve the model accuracy metrics by expanding the 1,536 corpus used for the present research. The expansion could be achieved either by abstractifying the niche topic with other virtualization solution, or by extending the temporal interval, considering vulnerabilities from previous years. Subsequent research exploring alternative classification models is possible, albeit attention must be paid upon the granularity of such classification; more categories may result in lower accuracy metrics in a niche dataset of which entries are inherently low due to the high technical complexities of VMMs.

6 CONCLUSIONS

6.1 General Conclusion

The analysis highlights that while attackers can pivot from one hypervisor to another, the degree to which it is possible vastly depends on the category and involved hypervisors. Some categories show no evidence of cross-hypervisor meta-bugs, whereas others do. As seen in Figure 1, **Device I/O** and **Add-ons & Management** vulnerabilities appear consistently across all four analysed hypervisors, irrespective of the CWE ID, reinforcing the idea that virtualized emulated devices, both proprietary and open-source, remain the largest attack vector and attract the efforts of attackers and researchers. The number of virtualization vulnerabilities has fluctuated over time, albeit the idea that the hypervisor-based threat landscape is evolving remains. The lower number of documented vulnerabilities in 2018, after the SPECTRE and MELTDOWN attacks, indicates a reactive rather than proactive approach from the industry. By highlighting an ascendant trend of virtualization vulnerabilities, the present study suggests a collaboration among KVM, VMware, and Xen to establish standardized security measures, enhance vulnerability mitigation strategies, and promote a unified response mechanism against emerging threats.

6.2 Research Question Answers

6.2.1 ***Which hypervisor functionalities account for the largest share in virtualized environments?*** The results suggest that Device I/O (430) and Add-ons & Mgmt (352) account for the largest share of vulnerabilities across hypervisors.

6.2.2 ***How are vulnerabilities distributed across Hyper-V, KVM, VMware and Xen when categorized based on hypervisor functionalities?*** This study reveals an even vulnerability distribution across hypervisors. If KVM shows a higher proportion of CPU and ITM-based flaws, VMware exhibits indications of systemic flaws in its add-ons and management tools. Hyper-V has a disproportionate amount of vulnerabilities in the Device I/O and Control & Management categories, possibly due to the exposure of low-level integrations within the Windows Operating System, such as API interactions or hypercalls. In contrast, Xen demonstrates a more balanced distribution of only 4 out of 6 categories. The distribution is attributed to Xen's development principles of following a leaner architecture or privileged domains (Dom0).

6.2.3 ***Do conceptually similar vulnerabilities recur across Hyper-V, KVM, VMware and Xen?*** This study does not present evidence of systemic flaws between the analysed 4 hypervisors. While some functionality-based categories, coupled with CWE IDs, might reveal underlying cross-hypervisor meta-bugs, the vast difference between architectural implementations prevents attackers from easily pivoting from one hypervisor to another.

REFERENCES

[1] *NVD - Home*. URL: https://nvd.nist.gov/ (visited on 01/19/2025).

[2] Michael Pearce, Sherali Zeadally, and Ray Hunt. "Virtualization: Issues, security threats, and solutions". In: *ACM Computing Surveys* 45.2 (Feb. 2013), pp. 1–39. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2431211.2431216. URL: https://dl.acm.org/doi/10.1145/2431211.2431216 (visited on 11/29/2024).

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2014. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162/ (visited on 02/01/2025).

[4] Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee. "Characterizing hypervisor vulnerabilities in cloud computing servers". In: *Proceedings of the 2013 international workshop on Security in cloud computing*. ASIA CCS '13: 8th ACM Symposium on Information, Computer and Communications Security. Hangzhou China: ACM, May 8, 2013, pp. 3–10. ISBN: 978-1-4503-2067-2. DOI: 10.1145/2484402.2484406. URL: https://dl.acm.org/doi/10.1145/2484402.2484406 (visited on 01/21/2025).

[5] Ernesto Rosario Russo et al. "Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities". In: *Journal of Systems and Software* 156 (Oct. 2019), pp. 84–99. ISSN: 01641212. DOI: 10.1016/j.jss.2019.06.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S016412121930130X (visited on 01/21/2025).

[6] P. R. Vishnu, P. Vinod, and Suleiman Y. Yerima. "A Deep Learning Approach for Classifying Vulnerability Descriptions Using Self Attention Based Neural Network". In: *Journal of Network and Systems Management* 30.1 (Jan. 2022), p. 9. ISSN: 1064-7570, 1573-7705. DOI: 10.1007/s10922-021-09624-6. URL: https://link.springer.com/10.1007/s10922-021-09624-6 (visited on 01/20/2025).