

## ARTICLE TYPE

# A Risk Manager for Intrusion Tolerant Systems: Enhancing HAL 9000 with New Scoring and Data Sources

Tadeu Freitas<sup>1,2</sup> | Carlos Novo<sup>1</sup> | Inês Dutra<sup>1</sup> | João Soares<sup>1</sup> | Manuel E. Correia<sup>1,2</sup> | Benham Shariati<sup>3</sup> | Rolando Martins<sup>4</sup>

<sup>1</sup>Department of Computer Science, Faculty of Sciences of University of Porto, Porto, Portugal

<sup>2</sup>Centre Advanced Computing Systems, Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal

<sup>3</sup>UMBC, University of Maryland, Baltimore County, Baltimore, USA

<sup>4</sup>SafeHelm, lda, Porto, Portugal

## Correspondence

Corresponding author Tadeu Freitas.  
Email: tadeufreitas@fc.up.pt

## Abstract

Intrusion Tolerant Systems (ITSs) have become increasingly critical due to the rise of multi-domain adversaries exploiting diverse attack surfaces. ITS architectures aim to tolerate intrusions, ensuring system compromise is prevented or mitigated even with adversary presence.

Existing ITS solutions often employ Risk Managers leveraging public security intelligence to adjust system defenses dynamically against emerging threats. However, these approaches rely heavily on databases like NVD and ExploitDB, which require manual analysis for newly discovered vulnerabilities. This dependency limits the system's responsiveness to rapidly evolving threats.

HAL 9000, an ITS Risk Manager introduced in our prior work, addressed these challenges through machine learning. By analyzing descriptions of known vulnerabilities, HAL 9000 predicts and assesses new vulnerabilities automatically. To calculate the risk of a system, it also incorporates the Exploitability Probability Scoring system to estimate the likelihood of exploitation within 30 days, enhancing proactive defense capabilities.

Despite its success, HAL 9000's reliance on NVD and ExploitDB knowledge is a limitation, considering the availability of other sources of information. This extended work introduces a custom-built scraper that continuously mines diverse threat sources, including security advisories, research forums, and real-time exploit proofs-of-concept. This significantly expands HAL 9000's intelligence base, enabling earlier detection and assessment of unverified vulnerabilities.

Our evaluation demonstrates that integrating scraper-derived intelligence with HAL 9000's risk management framework substantially improves its ability to address emerging threats. This paper details the scraper's integration into the architecture, its role in providing additional information on new threats, and the effects on HAL 9000's management. Our experiments show that this integration provides more secure configurations.

## KEYWORDS

machine learning, operating system diversity, risk assessment, intrusion tolerant systems, CVE, exploits, vulnerabilities

## 1 | INTRODUCTION

Intrusion Tolerant Systems (ITSs) are used evermore to counteract and mitigate the effects of successful security breaches done to systems exposed to public networks. The objective of ITSs is, in the event of a successful breach, to limit the adversary's capability to compromise the system, to alter its designed function, and to reduce any leakage of information, in other words, to become resilient against intrusions. A resilient system is designed to maintain correct execution even in the presence of nodes compromised by malicious adversaries. This is achieved by implementing methods and techniques that constrain the adversary's

**Abbreviations:** ITS, Intrusion Tolerant System; CVE, Common Vulnerabilities and Exposures; CVSS, Common Vulnerability Scoring System; OSs, Operating Systems; CWE, Common Weakness Enumeration; CPE, Common Product Enumerator; ML, Machine Learning; OSINT, Open-source Intelligence; SDN, Software Defined Networks; VM, Virtual Machine; NLP, Natural Language Processing; AI, Artificial Intelligence; SVM, Support Vector Machines; CNNs, Convolutional Neural Network; LSTM, Long Short Term Memory; XGBoost, eXtreme Gradient Boosting; EPSS, Exploit Prediction Scoring System; API, Application Programming Interface; RMSE, Root Mean Square Deviation; Bow, bag of words; emb, sentence embeddings; NVD, National Vulnerability Database.

capabilities, ensuring the system's functionality despite breaches of the defensive perimeter. However, the system operates with reduced performance during this period, as it actively mitigates the impact of the malicious nodes until they are fully neutralized or removed.

Most existing infrastructures and solutions rely on traditional defensive systems for their ease of deployment<sup>1</sup>. However, these systems primarily focus on securing the perimeter—the boundary between external threats and internal components or software. As a result, they are increasingly regarded as inadequate. Once adversaries breach this perimeter, they gain unrestricted control over the system, as no additional countermeasures are in place to mitigate or eliminate the intrusion. The recent surge in cyber-attacks<sup>2</sup>, coupled with their increasing complexity<sup>3,4,5,6</sup>, where adversaries employ diversity and sophistication to achieve their objectives, has driven stakeholders to seek more effective solutions to address these threats. This growing demand has allowed ITS to emerge as a viable approach.

Modern ITSs are designed to address the challenges associated with the increasing scale and complexity of cyber threats. These systems leverage advancements from a diverse range of research domains, including, but not limited to, cybersecurity, distributed systems, and Artificial Intelligence (AI), to provide robust and adaptive solutions. At their core, ITSs rely on replication to ensure both *liveness* and *security*. This means they can maintain correct service execution, even under compromised conditions, while safeguarding data integrity and confidentiality. Although this may reduce performance during adversarial activity, the service remains operational and secure. The interdisciplinary integration of these research fields significantly enhances the resilience and robustness of ITS, enabling them to effectively mitigate the impact of malicious adversaries and adapt to evolving threats. Current solutions now integrate distinct methods to thwart the adversaries from achieving the objective, which include, but are not limited to, replica rejuvenation<sup>7,8</sup>, diversity<sup>7,8,9,10</sup>, and N-version programming<sup>10</sup>. To enhance adaptability further, an ITS can integrate information from public sources such as social networks, news outlets, and online databases that track vulnerabilities, exploits, and cyber threats. This integration enables the ITS to proactively adjust its defenses by modifying configurations and operating systems (OSs), thereby increasing resilience against identified threats.

Risk assessment has long been a cornerstone of cybersecurity research, evolving through various approaches to address emerging threats. Early studies focused on leveraging information from NIST's database, particularly CVSS scores, to systematically evaluate system vulnerabilities<sup>11</sup>. Building on this foundation, subsequent research incorporated data from public social networks to improve situational awareness and provide richer contextual insights<sup>12</sup>. As technology advanced, automated methods emerged to streamline risk assessments, particularly for operating systems, by analyzing associated CVEs<sup>13</sup>. Complementing these efforts, other investigations explored the role of system diversity in minimizing risk, highlighting how the quantity and nature of CVEs influence exploitability<sup>14</sup>. Further refinements in risk assessment have introduced alternative CVSS scoring mechanisms for CVEs, reusing existing data to enhance evaluation precision<sup>8</sup>. These contributions collectively highlight the importance of integrating diverse data sources and methodologies to strengthen proactive and adaptive risk management strategies. These solutions leverage online exploit and vulnerability databases to gather relevant information, assess the associated risks to the current system using base scores from the retrieved data, and recommend lower-risk alternatives when feasible. However, a notable trade-off arises from the reliance on these base scores, which require manual evaluation of each vulnerability before assignment. This process introduces a significant delay between discovering new exploits or vulnerabilities, their subsequent risk assessment, and the overall evaluation of the ITS<sup>15</sup>.

The process for a new Common Vulnerabilities and Exposures (CVE) creation begins when a vulnerability is submitted for evaluation to the NVD. Initially, it is added to the CVE database, accompanied solely by the description provided by the applicant<sup>16</sup>. Once listed in the CVE database, the vulnerability undergoes the CVE Analysis process, providing additional details, including contextual information and preliminary assessments. The evaluation process involves several key steps to ensure the accuracy and utility of the CVE record. First, an analyst verifies the references provided with the CVE submission and assigns appropriate reference tags to facilitate data discovery. Then, it manually searches for additional public information relevant to the CVE. During this process, a Common Weakness Enumeration (CWE) identifier is assigned to specify the type of vulnerability associated with the CVE. Subsequently, the Common Vulnerability Scoring System (CVSS) version 3.1 metrics for exploitability and impact are calculated based on the gathered public information and the guidelines outlined in the submission. The analyst then prepares a Common Platform Enumerator (CPE) Applicability Statement, which identifies all software and/or hardware potentially affected by the vulnerability. Finally, before the CVE is published, a senior analyst conducts a comprehensive review and quality assurance check to ensure the record's accuracy and completeness.

Because of these steps, the evaluation process is inherently lengthy. This creates a time gap between the reception and evaluation of a vulnerability, which usually results in a delay that is often proportional to the severity of the vulnerability. For particularly critical vulnerabilities, the evaluation process can extend for as long as a year<sup>15</sup>. Moreover, as of the time of writing,

the backlog of vulnerabilities in the NVD has been significantly increasing due to a lack of sufficient resources for processing new CVEs. For instance, in March 2024, only 199 of the 3,370 CVEs submitted to the NVD were successfully analyzed<sup>17</sup>. If this issue remains unresolved, the response time for addressing newly reported vulnerabilities is likely to grow further, jeopardizing the reliability of the NVD CVE database as a trusted and reliable source of information.

To address these challenges, this work builds upon and extends the research presented in “HAL 9000: A Risk Manager for ITSs”, accepted at *The Sixth IEEE International Conference on Trust, Privacy, and Security in Intelligent Systems and Applications*. The original paper proposed secure and resilient configurations for ITSs by leveraging publicly available Open-Source Intelligence (OSINT) data and automating the assessment of CVSS scores for unrated CVEs. Recognizing the growing volume of information in publicly available OSINT databases, this extension enhances the original framework by integrating additional sources of information and refining the calculation step for risk assessment. These advancements aim to improve the system’s accuracy and depth in evaluating risks, further enhancing its ability to predict CVSS scores for unrated CVEs and dynamically reassess ITS configurations.

A review of the state-of-the-art reveals that current Risk Managers<sup>13,8</sup> rely heavily on the NVD’s CVE evaluation process, as their risk calculations are based on the CVSS base scores provided by the NVD and the existence of an exploit. HAL’s approach overcomes this limitation, offering a more proactive and efficient response to emerging threats. HAL’s CVSS score prediction tool aims to provide a temporary score, enabling faster response and minimizing the vulnerability window against emerging threats, thereby enhancing the adaptability of ITSs. This tool is not intended to permanently replace the NVD evaluation process, as specific nuances and subtleties in vulnerabilities can only be accurately identified through human analysis. Once the NVD evaluation is complete, HAL updates the scores of the previously predicted CVEs and re-executes the configuration risk assessment function, ensuring that its risk evaluations remain accurate and aligned with the most reliable information available.

HAL’s primary objective is to recommend secure configurations for ITSs using knowledge obtained from publicly available OSINT databases. However, given the high-risk environments in which ITSs are deployed, it is critical to ensure HAL’s resilience against potential malicious attacks. A secure deployment environment for HAL is strongly recommended to achieve this. For instance, both Lazarus<sup>8</sup> and Skynet<sup>18</sup> adopt a two-plane architecture to enhance security. This architecture separates the system into a secure, isolated controller plane that manages critical operations and an execution plane that interfaces with the environment and is exposed to potential threats. Adopting a similar approach for HAL can significantly bolster its robustness and ensure reliable operation in adversarial conditions. This concept is already established within the Software Defined Networks (SDN) research, where the SDN controller is separated from the SDN switches used in the data plane<sup>19</sup>.

The primary goal of deploying a Risk Manager in a secure environment is to ensure its correct operation remains uncompromised by malicious adversaries. Such attacks could target the system in multiple ways, including tampering with the local OSINT database—such as HAL’s—via input injection<sup>20</sup> or compromising the ML model through data poisoning<sup>21</sup>. To mitigate these risks, it is assumed that ITSs employing Risk Managers can provide a secure execution environment, safeguarding both the data sources and the integrity of the ML models. This foundational security measure is critical to maintaining the reliability and effectiveness of the Risk Manager in dynamic and adversarial settings. In summary, this paper extends the capabilities of the published Risk Manager, HAL 9000, through the following key advancements:

- Advancing the state-of-the-art in Risk Management by enhancing existing methodologies and integrating new approaches.
- Providing a comprehensive description of the scraper developed to collect exploit and vulnerability data from diverse OSINT platforms and other publicly available sources.
- Outlining the process of assembling the experimental dataset, ensuring transparency and reproducibility.
- Reassessing and refining the method for calculating the risk associated with an OS, improving the accuracy and reliability of the risk evaluation.

These contributions collectively strengthen the effectiveness and applicability of HAL 9000 in managing risks within ITSs.

The rest of the paper is organized as follows: Section 2 reviews the current state of the art in Risk Managers, CVSS score predictors, text document clustering algorithms<sup>22</sup>, and OSINT scrapers. Section 3 provides a detailed description of HAL’s architecture and workflow. Section 4 presents the experimental setup and a comparative discussion of the results. Finally, Section 5 discusses the main conclusions of this research and offers perspectives for future work.

## 2 | RELATED WORK

The present section addresses the current state of the art in Risk Managers, emphasizing their roles in evaluating system configurations and providing recommendations for secure and resilient setups. Additionally, it reviews proposed solutions for CVE score prediction methodologies and the application of machine learning techniques, particularly for clustering text documents<sup>23</sup>.

### 2.1 | Risk Managers

**Diversity Policy for Intrusion Tolerant Systems**<sup>13</sup>, proposed a diversity policy to be employed by recovery-based ITSs. It retrieves information on known software vulnerabilities and advises on combinations that minimize the risk of common vulnerabilities. This approach generates all possible configurations and decides based on the minimum sum of the CVSS scores.

This architecture is designed for web servers that deliver services accessible via the Internet. It consists of a cleansing group, a central controller, and a processing group. The cleansing group hosts Virtual Machines (VMs) as they undergo recovery, after which the restored VMs migrate to the processing group, providing the designated web service and facing external threats. The central controller coordinates the allocation and lifecycle of the VMs across both groups and executes the configuration selection algorithm to determine the optimal system setup. The system applies a proactive recovery mechanism to protect the system from undetected attacks. A central controller periodically rotates the state of each VM, transitioning between active, cleansing, ready, and active states. During the transition from the cleansing to the ready state, the central controller executes a selection algorithm, applying the diversity policy to decide on a suitable secure configuration.

Yet, the drawback of employing the diversity policy method lies in its dependency on the NVD database score before implementing any system alterations. This reliance can result in delays in adapting to the new vulnerabilities<sup>15</sup> because they were added to the CVE database and have not been evaluated by NVD, i.e., do not have a score attributed.

**Lazarus**<sup>8</sup> introduced a Byzantine Fault Tolerant (BFT) architecture that adjusts system configurations to reduce vulnerability risks based on recommendations from its Risk Manager module. The research proposes a novel CVE scoring method. It uses ML algorithms to detect shared CVEs<sup>‡</sup> between replicas using the CVE description. The new calculation method reevaluates the CVSS base score using the information in the CVE details. This includes factors such as the age of the CVE, the existence of a patch, and the availability of an exploit, as presented in the following equations.

$$score(v) = CVSS(v) \times oldness(v) \times exploited(v) \times patched(v) \quad (1)$$

$$oldness(v) = \max\left(1 - 0.25 \times \frac{(now - v.published\_date)}{oldness\_threshold}, 0.75\right) \quad (2)$$

$$patched(v) = 0.5^{v.patched} \quad (3)$$

$$patched(v) = 1.25^{v.exploited} \quad (4)$$

Another contribution of Lazarus was its ability to identify different CVEs in the NVD database that describe similar vulnerabilities affecting distinct applications or operating systems (OSs). This was achieved through description analysis. For example, CVE-2014-0157 affects OpenSUSE 13, CVE-2015-3988 affects Solaris 11.2, and CVE-2016-4428 affects Debian 8.0. Despite targeting different OSs, their descriptions share similarities, indicating a potential common exploit across these systems. To analyze CVEs that follow the same pattern, Lazarus employed K-means, a clustering technique from ML, to verify their description. K-means is an algorithm for unsupervised learning that clusters the CVEs by their description similarity. After clustering, in the evaluation phase, Lazarus's Risk Manager employs the new scoring system, penalizing pairs of replicas with common CVEs or the same clustered CVEs. Clustering suggests a likelihood of the exploit impacting both replicas. Based on this evaluation, Lazarus recommends a more resilient system configuration against intrusions.

<sup>‡</sup> By Lazarus definition, a CVE is shared between two or more OSs by the CVE vulnerability configuration field and/or by the description similarity with a CVE from another OS.

The Risk Manager used in Lazarus introduced a trade-off between security and resilience. It highlights that a system solely focused on minimizing risk may not be ideal for BFT protocols. When a pair of replicas can have one or more shared unpatched CVEs, it is possible to compromise the system by executing parallel attacks. Even though the recommended configuration might not be the most secure, there is a guarantee that an unpatched vulnerability will not affect more than one replica simultaneously, maintaining the BFT invariant of  $3f + 1$ . Besides the drawback of being dependent on the NVD score system, Lazarus's Risk Manager also requires the human intervention necessary to execute the K-means algorithm. The optimal number of generated clusters has to be manually visualized and inserted into the algorithm, which increases the execution time and, subsequently, the vulnerability time window. In addition, K-means is an algorithm susceptible to outliers<sup>24</sup>, i.e., the mean value of a cluster can be influenced by the outliers, which will affect the resulting clusters. This can lead to wrong assessments.

## 2.2 | CVSS score prediction

With the increasing applications of Artificial Intelligence (AI) and the latest advancements in Deep Learning, computers can now automate or replicate many processes. AI and Deep Learning algorithms can replicate tasks that are manually performed repeatedly and follow consistent patterns. This automation can speed up these processes, allowing operators to focus on refining and verifying the results and providing feedback to improve the algorithms.

Given the lengthy evaluation process for new CVEs and the challenges evaluators face, researchers have explored new procedures to apply AI and Deep Learning algorithms that can automatically predict the CVSS scores for CVEs, enabling a quicker assessment. As such, the related work on CVSS score predictors is addressed.

**Khazaei et al.**<sup>25</sup> introduced a method for predicting CVSS scores using natural language processing (NLP). This method learns from previously available CVE vulnerabilities by analyzing their descriptions and associated CVSS scores. The method applied text mining tools and techniques to extract feature vectors during data preprocessing. The authors evaluated the application of three different algorithms for predicting CVSS scores: Support Vector Machines (SVM), Random Forest, and fuzzy systems. Among these, the fuzzy systems provided the best accuracy, correctly scoring 88% of the evaluated CVEs. In conclusion, the study found that using automatic predictors can reduce human error and increase the speed of CVSS score calculation.

**Sahin et al.**<sup>26</sup> extended the prior work of Han et al.<sup>27</sup>, which predicted the severity levels of vulnerabilities based on their descriptions. Their study used similar feature extraction methods with word embeddings and prediction models using Convolutional Neural Networks (CNNs). Additionally, it incorporated Long Short-Term Memory (LSTM) networks and Extreme Gradient Boosting (XGBoost).

The objective of this research was to predict severity scores in addition to severity levels. The original work categorized vulnerability severity into ranges: low (0.1 - 3.9), medium (4.0 - 6.9), high (7.0 - 8.9), and critical (9.0 - 10.0). For feature extraction, the authors removed words that only occurred once in the sentence corpus and trained the word2vec continuous skip-gram model, introduced by Mikolov et al.<sup>28</sup>. After generating feature vectors for each word, they converted the description sentences into vector representations by concatenating the word vectors. The authors concluded that vulnerability descriptions contain valuable information that can be used with deep learning algorithms such as LSTM, CNN, and gradient boosting to predict severity scores with an average error of 16%.

**Elbaz et al.**<sup>29</sup> proposed using CVSS vector prediction to address N-Day vulnerabilities. CVSS vector prediction aims to forecast the base metrics that constitute the CVSS standard and assess the severity of vulnerabilities. This approach uses linear regression on vulnerability descriptions to provide basic metrics for newly discovered vulnerabilities.

This method differs from previous work as it predicts the individual metrics rather than the overall score, allowing CVSS to offer more detailed information about the vulnerability and enhance the "explicability" of the results. The authors used a bag-of-words approach to the vulnerability descriptions and a filtering scheme to remove irrelevant words. Subsequently, a regression model was trained for each metric to be applied during the assessment. The authors concluded that while the method by Khazaei et al.<sup>25</sup> achieves higher accuracy for the base score, their approach offers better explainability. They recommend implementing two pipelines: one for accuracy prediction and the other for "explicability".

**Costa et al.**<sup>30</sup> proposed combining text preprocessing using NLP techniques with vocabulary expansion and the application of the Deep Learning method DistilBERT<sup>31</sup>. The objective of the research was to predict CVSS metrics using vulnerability

descriptions, similar to the work of Elbaz et al.<sup>29</sup>. For preprocessing, lemmatization and stemming were applied to the data. Tokenization was performed using the Transformers library. The experiments tested the accuracy of combining the data with vocabularies of 5,000, 10,000, and 25,000 words added to the tokenizer's vocabulary. The authors then tested several deep learning methods, specifically DeBERTa, BERT, ALBERT, and DistilBERT, to evaluate the accuracy and quality of the assessed data. The results showed that DistilBERT provided the most accurate predictions, while ALBERT was the least accurate. The study concluded that DistilBERT is a state-of-the-art model for CVSS prediction, with enhanced performance when combined with lemmatization and a 5,000-word vocabulary. However, no analysis was made on the accuracy of the base score since the main idea is to provide insight for the experts.

**Kai et al.**<sup>32</sup> developed VultDistilBERT, a method to assess vulnerability severity, similar to the approach by Sahin et al.<sup>26</sup>, using a distillation model. The technique addresses data imbalance by augmenting data and using optimal subsets. The data augmentation process generates more raw data without increasing quantity. This is achieved by synonym replacement and random deletion to expand the sample set. The optimal subset selection involves choosing a subset of CVSS metrics during training to incorporate into the vulnerability descriptions, thereby enhancing textual information. The method then uses the DistilBERT model as a text characterization tool, processing the preprocessed data. The resulting feature vectors from DistilBERT are then classified using a linear layer. The authors concluded that their proposed method achieved state-of-the-art performance in vulnerability severity assessment, with a 97% assessment accuracy.

## 2.3 | Text Document Clustering

Text clustering is a technique used in text mining and information retrieval. Initially investigated to enhance the precision or recall in information retrieval systems<sup>33,34</sup>, as an efficient way of finding the nearest neighbors of a document<sup>35</sup>, in browsing a collection of documents<sup>36</sup>, or in organizing the results of a search engine response<sup>37</sup>.

Text Document Clustering is an unsupervised technique that groups documents into categories using unsupervised learning algorithms. According to Lazarus, online databases may contain entries with different identifiers that affect distinct systems but have identical descriptions, indicating the same vulnerability. Applying text document clustering can mitigate this inaccuracy, enhancing the quality of data collected from online vulnerability databases. Therefore, the present subsection addresses the related work on Text Document Clustering.

**Steinbach et al.**<sup>38</sup> compared the results of applying agglomerative hierarchical clustering with K-means for document clustering. The study evaluated the implementations of the Unweighted Pair Group Method with Arithmetic Mean (UPGMA), K-means, and bisecting K-means. To measure the quality of these algorithms, the authors used an internal quality measure, overall similarity, and an external quality measure, entropy. Overall similarity assesses cluster cohesiveness, while entropy measures the quality of the created clusters. The authors concluded that "given the linear run-time performance of bisecting K-means and the consistently good quality of the clusterings that it produces, bisecting K-means is an excellent algorithm for clustering a large number of documents"<sup>38</sup>.

**Zhao et al.**<sup>39</sup> compared the application of Agglomerative algorithms with Partitional algorithms. They evaluated group average functions against  $k$ -way clustering solutions. To assess the results from each algorithm, they used the FScore, introduced by Larsen et al.<sup>40</sup>, and entropy measures.

A perfect clustering solution would produce clusters containing documents from only a single class, resulting in zero entropy. Generally, lower entropy values indicate better clustering solutions. The FScore is necessary to account for clustering solutions that may seem poor due to the presence of outlier documents. Higher FScore values indicate better clustering solutions. Their results concluded that  $k$ -way clustering solutions outperformed those from agglomerative clustering implementations.

**Singh et al.**<sup>41</sup> compared the performance of K-means, Heuristic K-means, and Fuzzy C-means for document clustering. The study explored different feature selection conditions (with and without stop words, with or without stemming) and various representations (term frequency, term frequency-inverse document frequency, and Boolean). The trade-offs between the algorithms are as follows: K-means clustering quality is sensitive to initial seeds, Heuristic K-means adds computational cost due to the use of heuristics, and Fuzzy C-means does not produce hard clusters but provides a degree of membership for all created clusters. The authors used internal and external criteria for evaluation: residual sum of squares and purity, respectively. The results

showed that Heuristic K-means performs better than K-means, but Fuzzy C-means is a more robust flat clustering algorithm.

**Mendonça et al.**<sup>42</sup> studied the effectiveness of classical literature clustering algorithms when applied to free text documents. They selected five clustering algorithms for their experiments, each capable of word-embedding document representation: K-means<sup>43</sup>, Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)<sup>44</sup>, Spectral Clustering<sup>45</sup>, Mean Shift<sup>46</sup>, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN)<sup>47</sup>. The study aimed to observe the behavior of these algorithms in the task of document clustering. For evaluation, the authors used Normalized Mutual Information and Homogeneity scores to measure the overall quality of the clustering solutions. The results indicated that the chosen parameters significantly influence the performance of the algorithms. Among the selected algorithms, K-means performed the best.

**Asyaky et al.**<sup>48</sup> conducted an experiment to enhance the performance of density-based algorithms, specifically DBSCAN<sup>47</sup> and HDBSCAN<sup>49</sup>. The authors preprocessed the documents using lemmatization, stemming, and document embeddings to reduce dimensionality and improve the accuracy of these clustering algorithms. For evaluation, they used the Adjusted Rand Index, which compares the pairs of objects in the resulting clustering to a ground truth clustering, and the Adjusted Mutual Information, which is based on information-theoretical mutual information<sup>50</sup>. The authors concluded that the obtained results surpassed most existing methods on the state of the art on the same subject.

## 2.4 | OSINT Scraper

Given the extensive volume of OSINT data available for cybersecurity, vulnerabilities, and exploits, systems that rely on these sources require mechanisms for continuous data retrieval. Manual input is highly resource-intensive and may not capture all relevant information. To address this challenge, developers frequently employ web scrapping<sup>§</sup>, which automates the process of systematically and rapidly collecting critical data—such as indicators of compromise, exploit discussions, and newly discovered vulnerabilities. In the present subsection, we review the state-of-the-art OSINT scrapers.

**Fernandes et al.**<sup>51</sup> proposed ScrapeIOC, a web-scraping tool specifically designed to retrieve publicly available Indicators of Compromise (IOCs). The main objective was to collect as many IOCs as possible from various online sources, targeting specific types and families of malware. Leveraging ScrapeIOC, the authors constructed an offline database that organizes IOCs—primarily hashed samples (MD5, SHA1, or SHA256)—by their source. They then integrated this database with publicly accessible repositories to create a search engine for malware hashes. The tool scrapes multiple platforms, including Malwares, Malshare, VxCube, ThreatCrowd, and Maltiverse. Although ScrapeIOC effectively gathers critical IOCs, it is specialized for retrieving malware-related hashes rather than offering broader intelligence, such as vulnerabilities, exploits, or proof-of-concept details. Consequently, while it addresses an important facet of cyber defense, ScrapeIOC does not provide the more generalized capabilities found in frameworks like HAL, which support a wider range of cybersecurity intelligence.

**Alves et al.**<sup>52</sup> introduced SYNAPSE, a tool designed for extracting security-related information from Twitter<sup>¶</sup> accounts belonging to individual users, security organizations, and researchers. Its goal is to provide a continuous threat monitor that synthesizes and summarizes relevant intelligence for security analysts. The SYNAPSE pipeline encompasses filtering, feature extraction, binary analysis, clustering, and the generation of IOCs. Data collection begins with a scraping mechanism based on Twitter's publicly available API, wherein selected accounts are chosen based on their likelihood to share security-specific content. However, this approach confines the collected information to Twitter, excluding insights from other potentially valuable data sources.

**Kühn et al.**<sup>53</sup> introduced a classification scheme for the suitability and crawlability of reference texts in the NVD. Their research aimed to predict CVSS vectors using Deep Learning techniques. To achieve this, they developed a web scraper to retrieve vulnerability-related texts from external websites referenced by the NVD. The authors categorized these referenced websites into several groups based on their characteristics, including version control and bug tracker services, mailing lists, patch notes, security advisories, third-party articles, and blogs/social media. Since most of these websites did not provide APIs for data retrieval, the authors implemented a custom scraper tool to collect the necessary information. However, the tool was specifically

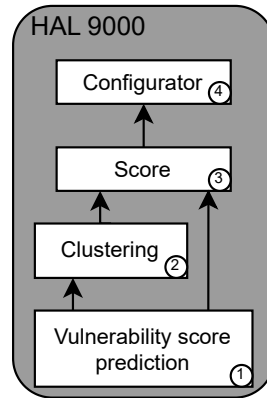
<sup>§</sup> Whereas web crawlers traverse the Internet in search of new sites relevant to a topic, scrapers typically target already identified websites to extract specific information.

<sup>¶</sup> Twitter and the Twitter logo are now officially rebranded as X since 2023.

designed for scraping websites referenced by the NVD and did not consider other publicly available sources that may also contain valuable information.

### 3 | HAL 9000 ARCHITECTURE

This section presents an overview of HAL 9000's extended architecture and workflow and a comprehensive description of the data scraper developed to extract information from online OSINT databases and public sources. Building on the findings of Pastor et al.<sup>54</sup>, which highlight the scarcity of tools for OSINT information retrieval, this section also details the implementation process for the scraper employed.



**FIGURE 1** HAL 9000 Architecture and workflow.

**HAL's architecture:** depicted in Figure 1, is constituted by four components: the Vulnerability score predictor, the Clustering algorithm, the Score reassessment, and the Configurator.

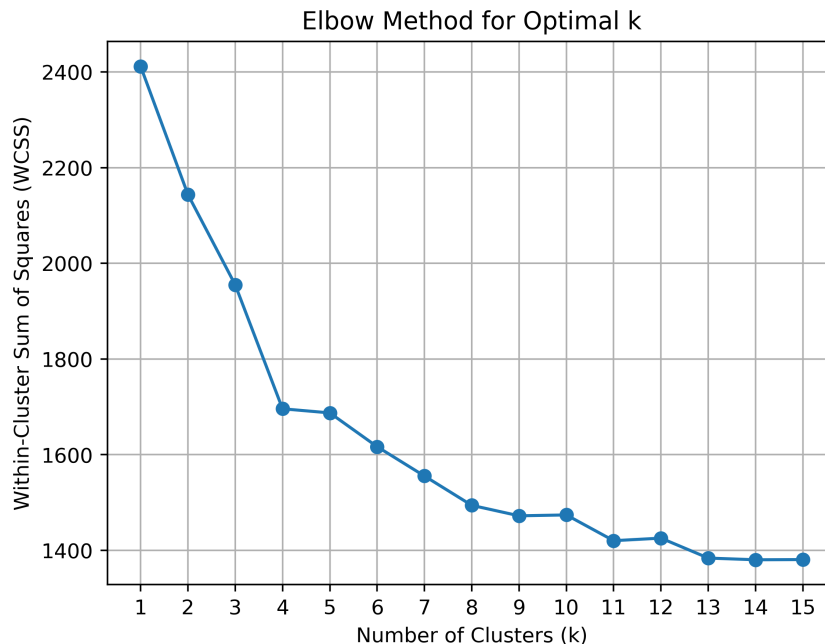
The Vulnerability score predictor assesses new CVEs by providing a score using its description. It uses an ML model trained with established vulnerability data. Specifically, it uses the information in the CVE's description and CVSS score metrics to train the model. Afterward, it assesses the new CVEs through their description and/or CVSS metrics depending on the applied algorithm, predicting the CVSS base score. The predicted CVSS score is then reassessed using the NVD equations<sup>55</sup>.

HAL, in its implementation, based on initial studies, used the algorithm proposed by Khazaei et al.<sup>25</sup> to accomplish this objective. However, a modular approach was adopted during the architecture design to facilitate interchange between different algorithms. This was achieved by restricting interaction between elements to the bare minimum, i.e., the dataset is transmitted to the element for training, thereafter accepting only the data to be evaluated. Both datasets are passed in a CSV file containing the CVE ID, description, and CVSS metrics.

The Clustering algorithm applied in the architecture follows the concept established by Lazarus. Databases can have different CVE entries that affect different software/OSs but are similar in vulnerability. This component aims to identify these occurrences through clustering using the CVE description. Clustering is done through the use of an ML algorithm. This creates a set of clusters to classify each description based on their similarity. For this component, the ML algorithm should be fully automated to reduce the total execution time window and eliminate the possibility of human bias or error. Figure 2 shows an example of the graph generated during the Lazarus execution to apply the Elbow method, i.e., visually decide on the number of clusters to be used. Furthermore, the algorithm must be robust to outlier data and prevent inaccurate clustering when inferring new data. Examples of ML algorithms that demonstrate insensitivity to outliers include, but are not limited to, HDBSCAN<sup>49</sup> and OPTICS<sup>57</sup>. Nonetheless, HAL's architecture was designed to accommodate other ML algorithms, i.e., supervised, semi-supervised, and deep learning. The component follows the same modular design as the Vulnerability score predictor to allow this.

The Score reassessment component extends Lazarus's contribution. It recalculates the CVSS score to reflect the CVE vulnerability more accurately. As mentioned in section 2, Lazarus recalculates the score using the additional information in the





**FIGURE 2** Example of a graph where visual extracting K can be challenging. According to the Pelt<sup>56</sup> method, the best K is 12.

CVE to improve its quality by taking into account its age, patch availability, and occurrence of an exploit<sup>#</sup>. HAL's architecture augments the Lazarus scoring system by considering the attention that the vulnerability receives in the "wild". Lazarus equations make minor adjustments to the CVSS base score, halving the score when a patch is available. However, in practical terms, a patch does not guarantee its installation. In 2022, incident responders were brought in to remediate attacks that began with exploited vulnerabilities, such as the ProxyShell and Log4Shell vulnerabilities. Both had existing patches at the time of compromise<sup>58</sup>.

In its original formulation, HAL extended the calculation methodology of Lazarus by readjusting the score to account for the potential non-installation of a patch and by incorporating the weight of the Exploit Prediction Scoring System (EPSS)<sup>59</sup>, as shown in equation 6. The EPSS quantifies the probability of a CVE being exploited in the wild in the next 30 days. Subsequently, HAL computes the  $hal\_score(v)$ , as presented in equation 4. This computation involves determining the  $score(v)$  both with and without the application of a patch (if available) and then applying the weighted probabilities derived from the EPSS for scenarios where the CVE is either targeted or not targeted. When a patch is unavailable, the corresponding terms in the  $hal\_score(v)$  equation that accounts for patch-related adjustments are assumed to be zero. The current extension enhances the score reassessment by considering the presence of CVEs in pulses<sup>||</sup> from AlienVault Open Threat eXchange (OTX). Pulses provide evidence that a CVE is actively exploited and is used across diverse attack vectors, indicating its broader exploitation potential. The calculation is refined using curated threat intelligence information from AlienVault to include the CVE's occurrence in distinct pulses, presented in equation 7.

For example, CVE-2017-11882 is an exploit with a CVSS score of 7.8, classified as "High" under NVD standards, and has a patch available. According to the Lazarus calculation, this vulnerability is reassessed to a score of 3.65, which falls into the "Low" category as per the Qualitative Severity Rating Scale<sup>60</sup>. However, this reassessment does not accurately reflect the exploit's risk profile. Discovered in 2017, CVE-2017-11882 remains an active threat despite the availability of a patch, with an exploit probability of 97.99% as derived from EPSS data. Using the methodology proposed in HAL's previous work, the recalculated score was adjusted to 7.2, maintaining its "High" classification. This adjustment provided a more realistic evaluation, accounting for scenarios where patch application might be delayed. However, this prior approach still failed to capture this vulnerability's severity fully, given its presence in 50 distinct Pulses. With the proposed extension in the current work, the

<sup>#</sup> A vulnerability is considered exploited when there are reports of its occurrence or when code is provided to exploit the vulnerability.

<sup>||</sup> **Pulses**, as defined by AlienVault, are curated collections of threat intelligence information shared by users or organizations. These collections contain actionable data about specific threats, including associated IOCs, descriptions, tags, related vulnerabilities, and techniques employed by attackers.

recalculated score rises to 8.9, placing it on the borderline of the "Critical" category. This adjustment more accurately reflects the actual risk posed by CVE-2017-11882, highlighting the urgency of its mitigation.

$$\text{score}(v) = \text{CVSS}(v) \times \text{oldness}(v) \times \text{exploited}(v) \times \text{patched}(v) \quad (5)$$

$$\text{hal\_score}(v) = \text{score}(v) \times (1 - \text{EPSS}(v)) + \text{score}(v)_{wp} \times \text{EPSS}(v) \quad (6)$$

$$\text{hal\_score}(v) = \min \left( 10, \text{score}(v) \times (1 - \text{EPSS}(v)) + \text{score}(v)_{wp} \times \text{EPSS}(v) + \log(\#\text{related\_pulses}) \right) \quad (7)$$

The final component, the Configurator, uses the gathered information to propose a secure and resilient configuration for the ITS to deploy. The component outputs two values, the *security\_risk(config)* and the *resilient\_risk(config)*. Its objective is to minimize both outputs, prioritizing the *resilient\_risk(config)*, i.e., shared CVEs calculation to avoid situations of “break one, break all”, where parallel attacks target ITS nodes. The *security\_risk(config)* is calculated by summing the newly assessed CVSS scores and predicted scores for new CVEs in a given configuration from all participating nodes. In the resilience calculation, the algorithm pairs every node in the configuration two by two and sums the CVSS score between shared CVEs and between CVEs clustered by their similarity.

$$\text{security\_risk}(config) = \sum_{n_i \in config} \sum_{v \in V(n_i)} \text{hal\_score}(v) \quad (8)$$

$$\text{resilience\_risk}(config) = \sum_{n_i, n_j \in config} \sum_{v \in V(n_i, n_j)} \text{hal\_score}(v) \quad (9)$$

### 3.1 | HAL 9000 workflow

HAL’s workflow, depicted in Figure 1, illustrates the data path from OSINT data retrieval to the recommended configuration. This subsection provides detailed information on HAL’s workflow, describing the different paths the data takes through its various components during execution.

HAL’s workflow is divided into a four-step process:

1. Predicting CVE scores for unassessed CVEs.
2. Clustering similar CVEs based on their descriptions.
3. Performing a risk assessment of the system configuration.
4. Providing the most resilient and secure configuration.

**1st step:** HAL receives a dataset of vulnerabilities and identifies CVEs that do not have assigned scores, i.e., CVEs that only have descriptions, with status “Received”. HAL uses an AI score predictor model to estimate the scores based on the evaluation patterns observed in the assessed vulnerabilities from the training dataset for these unassessed vulnerabilities.

**2nd step:** The CVEs are clustered based on their descriptions to identify similar vulnerabilities. This process avoids situations where different software is affected by distinct vulnerabilities, but it is the same vulnerability, considering that both descriptions are the same.

**3rd step:** HAL then reassesses the scores of the vulnerabilities using previously established equations (refer to equations 1 to 6). This process is also applied to CVEs assessed by the Vulnerability Score predictor. New CVEs have the necessary information to apply the new equations to the predicted score, specifically, the added date, if they were exploited, and the existence of a patch\*\*.

**4th step:** After reassessing the scores of CVEs, HAL generates all conceivable variations of the given software/Operating System (OS) configurations. HAL calculates its security and resilience levels for each configuration using equations 8 and 9,

\*\* Newly added vulnerabilities typically do not have patches for application

respectively. The resulting set of configurations is then arranged in order of resilience and security levels. The top configuration, representing the most resilient and secure option, is recommended to the ITS.

## 3.2 | Open-source Intelligence scraper

A scraper was developed for OSINT databases and other intelligence sources, including news, social networks, and blogs, to complement HAL's implementation. Given the continuous growth in vulnerabilities and exploits, an automated scraper benefits the system by continuously retrieving new vulnerabilities from online sources. However, directly gathering information from online OSINT databases and other sources could cause delays in HAL's execution. Therefore, the optimal solution was establishing a local database where the scraper could write and update the existing information.

Pastor-Galindo et al.<sup>54</sup> demonstrated that research for OSINT collection and analysis has yet to expand, as evidenced by the limited number of available open-source solutions. For example, current solutions like searchsploit<sup>61</sup> only provide information from a single database, ExploitDB. This subsection provides a detailed description of the OSINT scraper and its development process to guide future implementations.

The scraper was designed to query four different sources of OSINT databases/information every hour: the NVD CVE database<sup>62</sup>, ExploitDB<sup>63</sup>, AlienVault OTX<sup>64</sup>, and the Open Source Vulnerabilities (OSV)<sup>65</sup> database. To access the AlienVault OTX and the NVD CVE database, an Application Programming Interface (API) key was required due to rate limits imposed by the database<sup>††</sup>. This measure prevents denial of service attacks from malicious bots. If the limits are exceeded, the database becomes unresponsive for that IP address. Therefore, the scraper uses API keys provided by the OSINT databases and adheres to these limits to ensure queries remain within specified bounds. Accessing NVD database information requires forming REST-compliant requests following NVD's specifications<sup>67</sup>. The NVD's response comes in JSON format, which has to be parsed to retrieve the information. The number of CVEs in a response is limited to 2,000 entries to manage data flow. Queries exceeding this limit require additional requests for subsequent pages, as specified by NVD<sup>67</sup>. Additionally, the database may delay responses for consecutive queries from the same IP address to prevent resource drainage.

For ExploitDB's information, it was impossible to create a scraper due to protections against web scraping and crawling. Attempts to remotely access the database are blocked and treated as malicious bot activity. As an alternative, searchsploit<sup>61</sup>, the official scraper for ExploitDB's database, was integrated. While searchsploit is designed for ExploitDB, it traditionally requires manual operation to retrieve or update information. Our integration automates these commands, enabling automatic management without human intervention. The scraper sends a query to searchsploit, which replies via the standard output. The retrieved data is parsed into specific fields: `exploit name`, `url`, `local path`, `codes`, `verified`, and `file type`.

For the AlienVault Open Threat Exchange (OTX), the scraper is designed to interact with the API to retrieve "Pulses" associated with available software, operating systems (OSs), and Indicators for CVEs. A Pulse provides detailed information about malware, including its description, associated indicators of compromise (IOCs), related pulses, and historical data. An Indicator, on the other hand, contains information about potential threats. For CVEs, the Indicator includes details such as the CVE description, exploit activity, analysis, related pulses, and user comments. API data retrieval is performed using HTTP GET requests, authenticated with an API key. However, the number of requests is measured to avoid triggering rate limits, which could result in temporary bans.

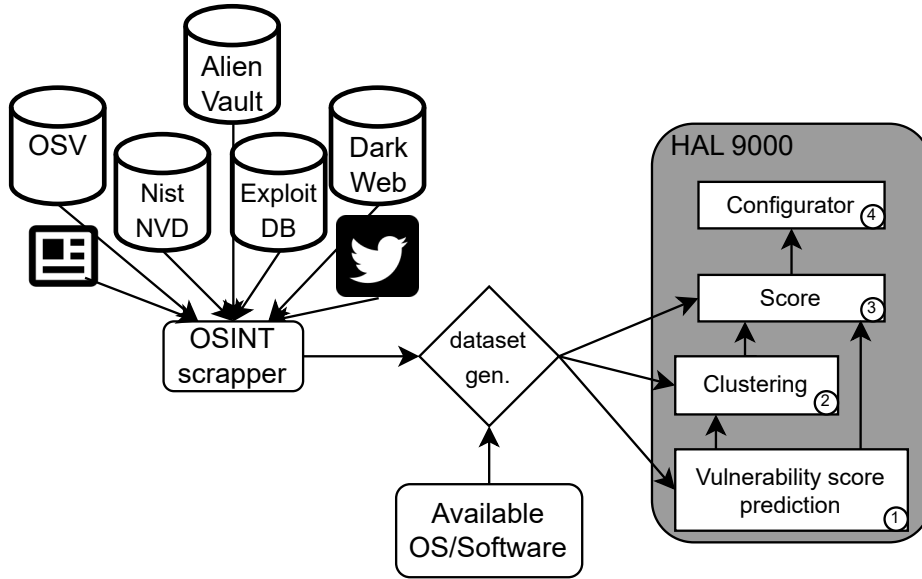
The OSV (Open Source Vulnerabilities) Database, developed by the Google Security Team, aims to enhance vulnerability triage for open-source software developers. Its primary objective is to provide precise and actionable data on vulnerabilities, enabling users to identify potential threats and quickly apply security patches. This project introduced a new schema to improve vulnerability transparency by reducing the maintenance effort for published vulnerabilities and increasing accuracy for downstream consumers. Unlike other OSINT databases, OSV allows users to query vulnerabilities by specifying a software package or identifier. This database is initially completely scraped from the updated Google Cloud Storage bucket and subsequently

---

<sup>††</sup> 50 requests per 30 seconds (with key), five requests per 30 seconds (without key)<sup>66</sup>.

scraped periodically through its API to ensure the most current data is available.

After obtaining the requested information from all the previously described sources, the data was preprocessed to eliminate unnecessary details and stored in JSON format in a local PostgreSQL key-value database. The OSINT scraper was written in Rust due to the security properties guaranteed by the programming language and consists of 1,268 lines of code. The scraper is publicly accessible from the group's GitHub repository<sup>68</sup>.



**FIGURE 3** HAL 9000 Architecture and workflow, along with the integration of the OSINT scraper tool. Within HAL's architecture, the data execution path is enumerated from one to four to showcase the execution flow. (The Twitter logo is now officially rebranded as X since 2023.)

## 4 | EXPERIMENTS

This section outlines the experiments conducted to support the research. Initially, three experiments were performed to validate the proposed contributions. An additional resource analysis was included in this extended work to evaluate the scraper's capabilities and efficiency.

The first experiment aimed to identify the most effective data preprocessing techniques and clustering algorithms for use in a Risk Manager. In previous work, Lazarus utilized the bag-of-words (BoW) approach for data preprocessing and K-means clustering, employing the elbow method to determine the optimal number of clusters (K). However, this approach did not provide adequate insights into several key aspects, including the choice of clustering algorithm, dataset size considerations, the selected K value, or strategies for situations where the elbow point is difficult to identify, as shown in Figure 2. Considering these trade-offs and the advancements discussed in Subsection 2.3, two preprocessing methods—BoW and embeddings—were tested for HAL. Additionally, twelve different clustering algorithms were applied to the dataset using these preprocessing techniques. The resulting clusters were subsequently incorporated into HAL's implementation to observe their effects on its scoring system.

The second experiment focused on validating HAL's updated scoring system by comparing it with established state-of-the-art methods. Specifically, Lazarus's Risk Manager and the approach by Heo et al.<sup>13</sup>, referred to as "Heo" for simplicity. The experiment used a dataset containing CVEs recorded up to the end of 2022, simulating a deployment scenario at the beginning

of 2023. To mimic real-world conditions, CVEs corresponding to a month of updates were injected into the database during the experiment. Sixteen operating systems were evaluated as part of the setup, detailed in the Experimental Setup subsection.

The third experiment evaluated the performance, Root Mean Square Deviation (RMSE), and accuracy of three state-of-the-art methods for CVSS score prediction. The dataset was used to analyze three aspects of CVSS prediction: metric prediction, severity prediction, and score prediction. This experiment sought the most effective approach to predict CVSS scores accurately.

Finally, in the fourth experiment, the performance usage of the scraper was evaluated. This included measuring the execution time required to retrieve information from various OSINT databases, characterizing the connections to these sources, and identifying additional requirements to ensure successful data retrieval. The analysis provided a deeper understanding of the scraper's efficiency and ability to handle diverse database configurations.

## 4.1 | Experimental setup

A dataset containing information on CVEs, vulnerabilities, and exploits related to OSs and software was necessary for the experiments. However, no existing tool or dataset provided the required information at the time of writing. To address this, a generator was developed to create a dataset with vulnerability information on various OSs and their installed software. To ensure a fair comparison with other Risk Managers, the dataset information was limited to entries from the NVD database and ExploitDB, as these were the sources used by the compared Risk Managers.

The dataset included information such as the CVE ID, publication date, last modification date, exploit description, exploit configuration, base metric version 2, and, if available, base metric version 3<sup>69</sup>.

Table 1 and 2 list the considered OSs available to the Risk Managers from which the dataset was based and the respective amount of considered CVEs.

**TABLE 1** List of considered OSs and respective amount of CVEs (Part 1).

| OS           | #CVEs |
|--------------|-------|
| Debian 7     | 3,923 |
| Windows 10   | 1,514 |
| FreeBSD 11   | 221   |
| Debian 8     | 3,923 |
| Ubuntu 16.04 | 2,035 |
| Solaris 10   | 359   |
| OpenBSD 6.0  | 69    |
| Centos 7     | 8     |
| Fedora 30    | 835   |
| Solaris 11   | 359   |
| Ubuntu 14.04 | 2,035 |
| Ubuntu 16.04 | 2,035 |

**TABLE 2** List of considered OSs and respective amount of CVEs (Part 2).

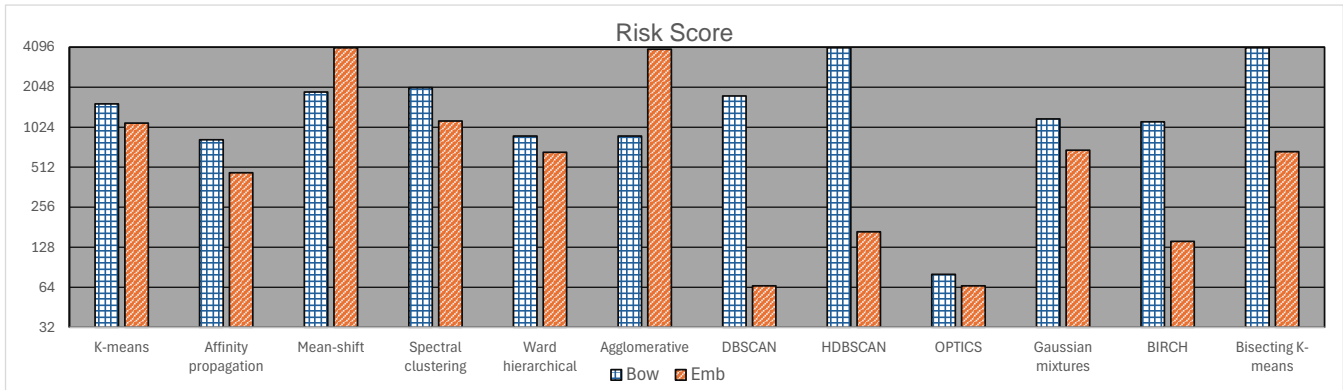
| OS                  | #CVEs |
|---------------------|-------|
| Debian 6            | 3,923 |
| Ubuntu 17.04        | 2,035 |
| Fedora 16           | 835   |
| Ubuntu 12.04        | 2,035 |
| Ubuntu 22.04        | 2,035 |
| Debian 10           | 3,923 |
| Ubuntu 10.04        | 2,035 |
| Fedora 38           | 835   |
| Windows Server 2012 | 1,105 |
| Fedora 24           | 835   |
| Centos 8            | 8     |
| OpenSuse 42.1       | 1,298 |

The experiments were conducted on a virtual machine within the XEMU hypervisor, running Debian 12. The virtual machine had 62 GB of RAM, a 32-core CPU, and an NVIDIA GeForce RTX 3090 graphics card with 24 GB of VRAM.

## 4.2 | Results and Discussion

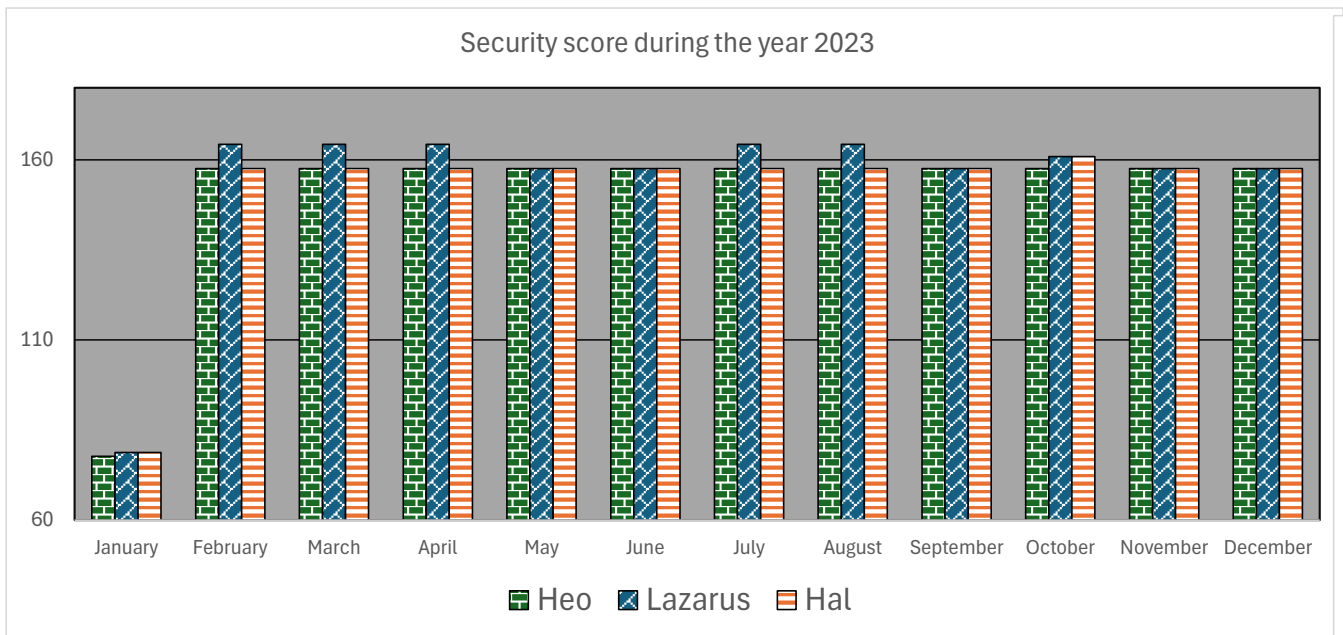
This section presents the results of the experiments and provides a subsequent discussion of the findings. The results and respective discussion are organized in the order in which the experiments were conducted.

From the experiments conducted with various clustering algorithms, presented in Figure 4, it was observed that sentence embeddings are the optimal preprocessing technique for the given data. This technique effectively maintains the relationships between words within sentences. Regarding the clustering algorithms, it was noticed that OPTICS and DBSCAN provide the lowest risk scores. Both algorithms are insensitive to outliers and designed to be scalable, addressing the limitations of the K-means algorithm. As such, considering these results, HAL utilizes sentence embeddings for preprocessing data and OPTICS for data clusterization.



**FIGURE 4** Evaluation of several clustering algorithms and subsequent effects on the HAL risk calculation (lower is better). In each algorithm, two approaches to preprocess data are applied: a bag of words (bow) and sentence embeddings (emb).

We provided two separate graphs for clarity for the risk assessment, presented in Figures 5 and 6. The first graph shows the

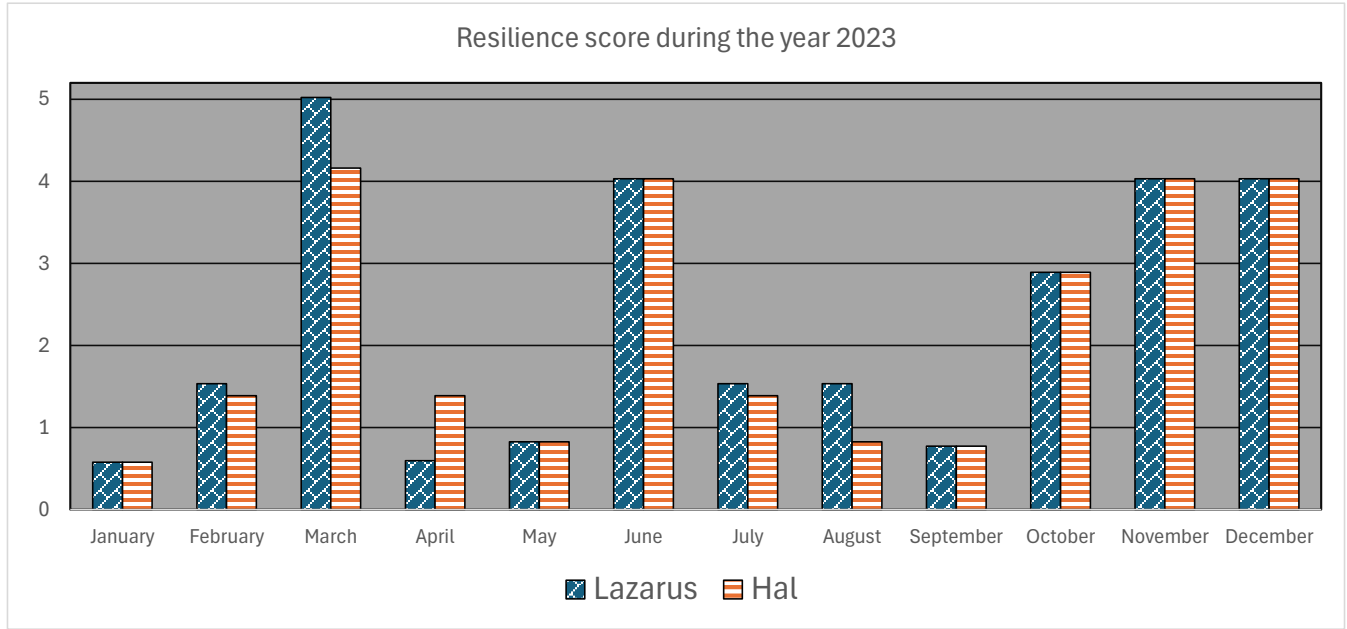


**FIGURE 5** Evaluation of the different Risk Managers, considering the level of security, i.e., the sum of the CVSS score of each CVE present in the advised configuration (lower is better).

security score for each Risk Manager throughout 2023, while the second graph displays the resilience score<sup>‡‡</sup> for each Risk Manager over the same period. Note that the resilience score for Heo is not calculated in the second graph because it was initially designed for non-replicated nodes. Although Heo's implementation was redesigned for the replicated scenario, it does not use a clustering algorithm. As such, the advised configuration and respective calculation might have hidden shared vulnerabilities that will not show up in its resilience score. Both graphs use the Lazarus calculation method for the scores presented. Each Risk Manager employed the technique to provide the best configuration. However, for comparison purposes, we applied the Lazarus scoring method since it is a method that is closer to the other Risk Managers. Additionally, we standardized the implementation

<sup>‡‡</sup> The resilience score reflects the number of shared vulnerabilities that can undermine an ITS. Shared vulnerabilities between nodes increase the threat of parallel attacks.

using the same preprocessing technique and clustering algorithm for the resilience comparison between HAL and Lazarus. This comparison aimed to demonstrate that HAL's score calculation system can provide a better configuration than Lazarus.



**FIGURE 6** Evaluation of the different Risk Managers, considering the level of resilience, i.e., the multiplication between the reassessed CVSS score of the common CVEs and respective EPSS score (by shared CVE or by clustering) present in the advised configuration (lower is better).

In the security graph (y-axis), the calculated values represent the sum of the CVSS scores recalculated using Lazarus equations. In the resilience graph (y-axis), the values represent the sum of the recalculated CVSS scores multiplied by the sum of the shared CVE EPSS, which indicates the probability of being exploited in the wild. This approach was used because a more active vulnerability (with a higher probability of exploitation) can attack several nodes simultaneously, which is not considered in the security calculation. Initial observations from both graphs indicate a trade-off: Heo provides better security, while Lazarus offers better resilience. However, HAL aims to combine both factors, providing resilient and secure configurations. For example, in February, March, July, and August, HAL delivered configurations with lower risk that were resilient and secure compared to Lazarus.

It is important to note that HAL occasionally produces configurations that are more secure but less resilient than Lazarus, as seen in April. This occurred because, in its settings, it was opted to choose more secured configurations over more resilient ones, considering that although the configuration might be exposed to parallel attacks, these would have a lower security score, i.e., have a lesser impact on the node compared with more severe vulnerabilities.

**TABLE 3** Analysis of different state-of-the-art AI implementations for CVSS prediction. Root Mean Square Error (RMSE) is calculated by translating the predicted discrete data into continuous data.

| Model                        | Accuracy (%) | RMSE | Time Execution (s) | Classifiers   |
|------------------------------|--------------|------|--------------------|---------------|
| Khazaei et al. <sup>25</sup> | 99%          | 0.66 | 132.85             | Random Forest |
| Costa et al. <sup>30</sup>   | 87%          | 1.55 | 1432.14            | Linear        |
| VulDistilBERT <sup>32</sup>  | 90%          | 3.63 | 195.49             | Linear        |

For CVSS score prediction, we implemented and tested three methods using the generated dataset from the established state-of-the-art of CVSS prediction. Each algorithm employs a different technique to predict the CVSS score, as mentioned in the

Related Work section. Although the predicted data is discrete in every implementation, it can be translated into continuous data to calculate the predicted score. This allows for the computation of the RMSE of the predicted score, which is included in Table 3.

Khazaei et al.<sup>25</sup> utilized traditional ML algorithms trained with the descriptions and scores of known CVEs to predict the score. Costa et al.<sup>30</sup> applied deep learning algorithms trained with the descriptions and CVSS metrics of known CVEs to predict the CVSS metrics of new vulnerabilities. VulDistilBERT<sup>32</sup> employed deep learning algorithms trained with CVE descriptions and CVSS severity to predict the CVSS severity of new vulnerabilities. The results, presented in Table 3, indicate that the method proposed by Khazaei et al. achieves the best accuracy with the provided dataset. Additionally, since it uses traditional ML methods (Random Forest), it requires less execution time to train the model and infer new vulnerabilities. This implies a quicker response against new vulnerabilities in HAL's use-case scenario. The method proposed by Costa et al., which predicts each CVSS metric separately, has an average accuracy of around 87%. Applying the CVSS v3.1 equations results in calculated CVSS scores with a higher RMSE than the original values. It is important to note that this method aimed to aid vulnerability reviewers in speeding up the assessment procedure for new vulnerabilities, which differs from HAL's objective.

Lastly, the method proposed in VulDistilBERT<sup>32</sup> achieves 90% accuracy in predicting new CVEs' severity. Although it has high accuracy for severity prediction, mapping it to actual CVSS scores is challenging since severity is relative and spans range: low (0.1 - 3.9), medium (4.0 - 6.9), high (7.0 - 8.9), and critical (9.0 - 10.0). This makes predicting precise scores difficult. For our calculations, we considered the highest value represented by each severity level to calculate the CVE base score, which explains the high RMSE. For HAL's implementation, we adopted the method proposed by Khazaei et al.<sup>25</sup> for score prediction.

The last experiment evaluated the efficiency of the developed scraper during its initial execution. In this first iteration, the scraper retrieves all the available data. Subsequent iterations fetch smaller amounts of data, requiring less time and fewer resources. As a result, the experiment provides an upper bound for the time required to retrieve data. The results of this experiment are presented in Table 4.

**TABLE 4** Results collected for the scraper retrieving data from NVD, ExploitDB, AlienVault OTX, and OSV. The total execution time includes the data retrieval and insertion into the local database. (\* Databases marked with an asterisk are scrapable via file download and upload to the local database.)

| OSINT DB       | Scrapable | Total Execution Time (s) | API Key Required | #Entries | Limitations on # requests (req/s)    |
|----------------|-----------|--------------------------|------------------|----------|--------------------------------------|
| NVD            | Yes       | 390.29                   | Yes              | 277,152  | 50/30 (with key), 5/30 (without key) |
| ExploitDB      | Yes*      | 3.42                     | No               | 46,575   | N/A                                  |
| AlienVault OTX | Yes       | 99,774                   | Yes              | 277,152  | 10,000/3600                          |
| OSV            | Yes*      | 235.44                   | No               | 255,743  | No limit                             |

After the experiment, the total size of the local database was measured, revealing approximately 2 GB of data available for use as a source of local intelligence for HAL. The most challenging database to retrieve information from was AlienVault OTX due to its hourly request limitation. Since the API does not support batch retrieval, requests had to be made individually, which proved time-consuming given the large number of IOCs available on the site. This limitation presents a significant challenge when using AlienVault OTX as a source of automated information. To address this, a delay of one hour was introduced between requests to ensure fair data retrieval.

Another challenge was the lack of direct data retrieval capabilities from ExploitDB. To overcome this, we utilized the *searchsploit* tool, which ExploitDB supports for retrieving data. However, this tool requires periodic updates, which must be synchronized with the scraper to ensure up-to-date information. Considering the overall execution time (excluding the delays imposed by OTX), the scraper is efficient and can be quickly set up to provide fast vulnerability knowledge. Its rapid response time allows scalability, making it suitable for more frequent periodic data retrieval schedules.



## 5 | CONCLUSION

The extension introduced in this work implements a fully automated scraper that periodically retrieves vulnerability data and cybersecurity intelligence from multiple OSINT sources. By broadening the sources of vulnerability information, HAL 9000 significantly enhances the quality and timeliness of input data, leading to more precise and dynamic risk score calculations. This development reduces the need for manual intervention, streamlines the workflow, and enables HAL 9000 to make informed assessments in near real-time.

HAL 9000 goes beyond traditional CVE rating systems by leveraging ML techniques to predict CVSS scores for new or unscored vulnerabilities, incorporating factors that traditional methods often overlook, such as patch neglect, exploit likelihood, and the age of CVEs during score reassessment. The system's automated risk scoring and clustering capabilities allow it to distinguish and prioritize vulnerabilities more accurately. In fact, experimental results demonstrate that HAL's clustering of CVEs yields better groupings than prior approaches, facilitating more effective mitigation strategies.

Additionally, HAL 9000 is designed to consider resilience and security risk holistically, allowing it to recommend secure configurations that advance both security posture and operational continuity. Its independence from a static CVE rating framework enables it to proactively address both known critical vulnerabilities and emerging quasi-zero-day threats, which is particularly valuable in today's fast-moving threat landscape.

The integration of automated OSINT sources, besides improving vulnerability coverage, also ensures that HAL 9000's prioritization is continuously informed by the latest available intelligence, enhancing responsiveness to new threats and reducing the window of exposure for unpatched systems. This proactive, data-driven approach positions HAL 9000 as a next-generation risk manager, demonstrably superior to traditional systems in terms of automation, accuracy, and adaptability.

HAL 9000's features, including dynamic CVSS prediction, risk factor incorporation, automated data acquisition, superior CVE clustering, and comprehensive risk assessment, set a new standard for autonomous risk management. The demonstrated dependability and extensibility of this approach serve as a roadmap for future systems that harness real-time intelligence and automation to efficiently prioritize and respond to cybersecurity threats.

Future work will focus on further expanding HAL's functionality by integrating data from penetration testing, automated testing tools, and additional cybersecurity intelligence sources. The current automated scraper will be enhanced to retrieve insights from additional sources, including the dark web, with retrieved information carefully vetted and incorporated into the local database.

In addition, we will also focus on investigating the potential impact of misinformation on the effectiveness of OSINT integration. Social media and other open sources may be flooded with fake CVE claims or manipulated by fake accounts and bots that generate a false sense of threat consensus. Understanding and addressing these risks will be crucial: future research will examine methods for validating the authenticity and reliability of externally sourced threat data, such as cross-referencing, data provenance tracking, ML-based anomaly detection, or trusted-source weighting, to mitigate the dangers posed by misinformation.

Furthermore, both HAL and the automated scraper will be deployed in real-world ITS implementations<sup>18,70,71,72</sup> to assess risk assessment performance and analyze any resulting system overhead. These combined efforts aim to further refine HAL's adaptive security capabilities, ensuring robust, trustworthy, and scalable risk management in evolving cybersecurity environments.

### AUTHOR CONTRIBUTIONS

Tadeu Freitas led the article's research, implementation, writing, and proofreading. Carlos Novo contributed to the writing and proofreading and provided technical advice. Inês Dutra offered technical advice and proofreading. João Soares, Behnam Shariati, and Manuel Correia assisted with proofreading and guidance. Rolando Martins supervised the complete work and contributed to the proofreading process.

### FINANCIAL DISCLOSURE

The authors Tadeu Freitas and Carlos Novo were supported by the following grants: 2021.04529.BD (FCT) and 2021.08532.BD (FCT), respectively.

### CONFLICT OF INTEREST

The authors declare no potential conflict of interest.

## References

1. TechDirect . Zero Trust Architecture vs Traditional Security Models: A Comparative Analysis.; n.d. Accessed: 2024-12-27.
2. Statista . Expected Cost of Cybercrime Worldwide until 2027. <https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/>; n.d. Accessed: 2024-12-27.
3. Accellion File Transfer Appliance Hack. <https://www.cnn.com/2022/01/25/tech/accellion-file-transfer-appliance-hack-explained/index.html>; 2022. "[Online; accessed 07-February-2023]".
4. Garmin Data Breach. <https://www.cnn.com/2022/07/27/tech/garmin-data-breach-explained/index.htm>; 2022. "[Online; accessed 07-February-2023]".
5. Cloudflare Data Leak. <https://www.cnn.com/2022/02/24/tech/cloudflare-data-leak-explained/index.html>; 2022. "[Online; accessed 07-February-2023]".
6. Microsoft Exchange Server Hacks. <https://www.cnn.com/2021/03/03/tech/microsoft-exchange-server-hack-explained/index.html>; 2021. "[Online; accessed 07-February-2023]".
7. Bessani AN, Sousa P, Correia M, Neves NF, Verissimo P. The CRUTIAL way of critical infrastructure protection. *IEEE Security & Privacy*. 2008;6(6):44–51.
8. Garcia M, Bessani A, Neves N. Lazarus: Automatic management of diversity in bft systems. In: 2019:241–254.
9. Chun BG, Maniatis P, Shenker S. Diverse Replication for {Single-Machine}{Byzantine-Fault} Tolerance. In: 2008.
10. Distler T, Popov I, Schröder-Preikschat W, Reiser HP, Kapitza R. SPARE: Replicas on Hold.. In: 2011.
11. Houmb SH, Franqueira VNL, Engum EA. Estimating impact and frequency of risks to safety and mission critical systems using CVSS. 2008.
12. Alves F, Andongabo A, Gashi I, Ferreira PM, Bessani A. Follow the blue bird: A study on threat data published on twitter. In: Springer. 2020:217–236.
13. Heo S, Lee S, Jang B, Yoon H. Designing and implementing a diversity policy for intrusion-tolerant systems. *IEICE TRANSACTIONS on Information and Systems*. 2017;100(1):118–129.
14. Garcia M, Bessani A, Gashi I, Neves N, Obelheiro R. OS diversity for intrusion tolerance: Myth or reality?. In: IEEE. 2011:383–394.
15. Ruohonen J. A look at the time delays in CVSS vulnerability scoring. *Applied Computing and Informatics*. 2019;15(2):129–135.
16. CVEs and the NVD Process. <https://nvd.nist.gov/general/cve-process>; . Accessed: 2024-02-09.
17. Hendery S. NIST's backlog of vulnerability analysis blamed on lack of support. <https://www.scmagazine.com/news/nists-backlog-of-vulnerability-analysis-blamed-on-lack-of-support>; 2024. "[Online; accessed 15-April-2024]".
18. Freitas T, Soares J, Correia ME, Martins R. Skynet: a Cyber-Aware Intrusion Tolerant Overseer. In: 2023:111–116.
19. Bannour F, Souihi S, Mellouk A. Distributed SDN control: Survey, taxonomy, and challenges. *IEEE Communications Surveys & Tutorials*. 2017;20(1):333–354.
20. Malik M, Patel T. Database security-attacks and control methods. *International Journal of Information*. 2016;6(1/2):175–183.
21. Yerlikaya FA, Bahtiyar Ş. Data poisoning attacks against machine learning algorithms. *Expert Systems with Applications*. 2022;208:118101.
22. Singhal G, Roy S. A Survey on Data Clustering. *International Journal of Advanced Engineering and Management*. 2017;2(8):183–188.
23. Hotho A, Nürnberger A, Paaß G. A brief survey of text mining. *Journal for Language Technology and Computational Linguistics*. 2005;20(1):19–62.
24. Singh S, Gill NS. Analysis and study of K-means clustering algorithm. *Int. J. Eng. Res. Technol*. 2013;2(7):2546–2551.
25. Khazaei A, Ghasemzadeh M, Derhami V. An automatic method for CVSS score prediction using vulnerabilities description. *Journal of Intelligent & Fuzzy Systems*. 2016;30(1):89–96.
26. Sahin SE, Tosun A. A conceptual replication on predicting the severity of software vulnerabilities. In: 2019:244–250.
27. Han Z, Li X, Xing Z, Liu H, Feng Z. Learning to predict severity of software vulnerability using only vulnerability description. In: 2017:125–136.
28. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 2013.
29. Elbaz C, Rilling L, Morin C. Fighting N-day vulnerabilities with automated CVSS vector prediction at disclosure. In: 2020:1–10.
30. Costa JC, Roxo T, Sequeiros JB, Proenca H, Inacio PR. Predicting cvss metric via description interpretation. *IEEE Access*. 2022;10:59125–59134.

31. Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 2019.
32. Kai S, Shi F, Zheng J, others . VulDistilBERT: A CPS Vulnerability Severity Prediction Method Based on Distillation Model. *Security and Communication Networks*. 2023;2023.
33. Rijsbergen Cv. *Information retrieval*. Butterworth-Heinemann, 1979.
34. Kowalski GJ. *Information retrieval systems: theory and implementation*. 1. springer, 2007.
35. Buckley C, Lewit AF. Optimization of inverted vector searches. In: 1985:97–110.
36. Cutting DR, Karger DR, Pedersen JO, Tukey JW. Scatter/gather: A cluster-based approach to browsing large document collections. In: . 51. 2017:148–159.
37. Zamir O, Etzioni O, Madani O, Karp RM. Fast and intuitive clustering of Web documents.. In: . 97. 1997:287–290.
38. Steinbach M, Karypis G, Kumar V. A comparison of document clustering techniques. 2000.
39. Zhao Y, Karypis G. Comparison of Agglomerative and Partitional Document Clustering Algorithms. 2002.
40. Larsen B, Aone C. Fast and effective text mining using linear-time document clustering. In: 1999:16–22.
41. Singh VK, Tiwari N, Garg S. Document clustering using k-means, heuristic k-means and fuzzy c-means. In: 2011:297–301.
42. Mendonça I, Trouvé A, Fukuda A, et al. On Clustering Algorithms: Applications in Word-Embedding Documents.. *J. Comput.*. 2019;14(2):88–92.
43. Hartigan JA. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.
44. Liu X, Gong Y, Xu W, Zhu S. Document clustering with cluster refinement and model selection capabilities. In: 2002:191–198.
45. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*. 2000;22(8):888–905.
46. Comaniciu D, Meer P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*. 2002;24(5):603–619.
47. Ester M, Kriegel HP, Sander J, Xu X, others . A density-based algorithm for discovering clusters in large spatial databases with noise. In: . 96. 1996:226–231.
48. Asyaky MS, Mandala R. Improving the performance of HDBSCAN on short text clustering by using word embedding and UMAP. In: 2021:1–6.
49. McInnes L, Healy J. Accelerated hierarchical density based clustering. In: 2017:33–42.
50. Wagner S, Wagner D. Comparing clusterings: an overview. 2007.
51. Fernandes KCP, Jonker SL, Meng W, Lampe B. ScrapeIOC: Designing a Web-scraping Tool for Malware Detection based on Indicators of Compromise. In: IEEE. 2023:124–128.
52. Alves F, Bettini A, Ferreira PM, Bessani A. Processing tweets for cybersecurity threat awareness. *Information Systems*. 2021;95:101586.
53. Kuehn P, Relke DN, Reuter C. Common vulnerability scoring system prediction based on open source intelligence information sources. *Computers & Security*. 2023;131:103286.
54. Pastor-Galindo J, Nespoli P, Mármol FG, Pérez GM. The not yet exploited goldmine of OSINT: Opportunities, open challenges and future trends. *IEEE Access*. 2020;8:10282–10304.
55. Common Vulnerability Scoring System Calculator. <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>; . Accessed: 2024-04-08.
56. Killick R, Fearnhead P, Eckley IA. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*. 2012;107(500):1590–1598.
57. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825–2830.
58. Adam S. Unpatched Vulnerabilities: The Most Brutal Ransomware Attack Vector. <https://news.sophos.com/en-us/2024/04/03/unpatched-vulnerabilities-the-most-brutal-ransomware-attack-vector/>; 2024. "[Online; accessed 07-April-2024]".
59. Jacobs J, Romanosky S, Edwards B, Adjerid I, Roytman M. Exploit prediction scoring system (epss). *Digital Threats: Research and Practice*. 2021;2(3):1–17.
60. Common Vulnerability Scoring System version 4.0: Specification Document. <https://www.first.org/cvss/specification-document>; . Accessed: 2024-06-29.
61. Exploit Database SearchSploit Manual. <https://www.exploit-db.com/searchsploit>; . Accessed: 2024-04-14.

62. Booth H, Rike D, Witte GA, others . The national vulnerability database (nvd): Overview. 2013.
63. Exploit Database. <https://www.exploit-db.com/>; . Accessed: 2024-04-14.
64. AlienVault Open Threat Exchange. <https://otx.alienvault.com/>; . Accessed: 2024-04-14.
65. Vulnerabilities OS. OSV - Open Source Vulnerabilities.; 2025. Accessed: 2025-01-14.
66. NVD Developers. <https://nvd.nist.gov/developers/start-here/>; . Accessed: 2024-04-15.
67. NVD Vulnerabilities API. <https://nvd.nist.gov/developers/vulnerabilities/>; . Accessed: 2024-04-15.
68. SecureSolutionsLab . vex\_hk.; 2023.
69. Mell P, Scarfone K, Romanosky S. Common vulnerability scoring system. *IEEE Security & Privacy*. 2006;4(6):85–89.
70. Wang F, Gong F, Sargor C, Goseva-Popstojanova K, Trivedi K, Jou F. SITAR: A scalable intrusion-tolerant architecture for distributed services. In: . 1. 2003:1100.
71. Bangalore AK, Sood AK. Securing web servers using self cleansing intrusion tolerance (SCIT). In: 2009:60–65.
72. Saidane A, Nicomette V, Deswarte Y. The design of a generic intrusion-tolerant architecture for web servers. *IEEE Transactions on dependable and secure computing*. 2008;6(1):45–58.