Computers and Artificial Intelligence; Vol.3, No.6; 2026
E-ISSN: 3006-3728 P-ISSN: 3006-371X
DOI: https://doi.org/10.70267/cai.v3n6.9399
Published by: Zeus Press

# A Lightweight K-Means Clustering Method for Vulnerability Classification with SHA-256 Integrity Verification

**Fanhao Liu***

*QU-Drake United College, Qingdao University, Qingdao 266071, China*

**Corresponding author: Fanhao Liu.*

## Abstract

Software vulnerability databases add tens of thousands of records each year, and their sources, formats, and forms of expression are highly mixed. Manual classification by reading each record is slow and varies from person to person: descriptions written by different researchers for the same underlying flaw may differ greatly, making prioritization based on them almost unreliable. This study proposes a lightweight vulnerability classification method that combines K-Means clustering with Secure Hash Algorithm 256 (SHA-256) integrity verification. Five numerical features are extracted from each vulnerability record—the Common Vulnerability Scoring System (CVSS) score, access complexity, privileges required, scope, and exploitability—and are standardized before being input into K-Means. The number of clusters K is determined by the elbow method rather than by experience, and each centroid vector obtained by clustering is then hashed separately so that any subsequent modification can be detected. On 600 National Vulnerability Database (NVD)-style records, when K = 4, the silhouette coefficient is 0.4043 and the Calinski-Harabasz index is 1187.05, while the complete SHA-256 verification adds only 0.1167 ms of overhead. The method requires only a few milliseconds on an ordinary laptop and is suitable for vulnerability triage in teaching laboratories or resource-constrained security dashboards.

### Keywords

K-Means clustering, vulnerability classification, SHA-256, integrity verification, lightweight security

## 1. Introduction

The growth rate of vulnerability disclosures has long exceeded the capacity of manual analysis. Public databases represented by the NVD have accumulated hundreds of thousands of heterogeneous records [1], while an ordinary product release may add dozens of Common Vulnerabilities and Exposures (CVE) entries within a single week. It is no longer realistic to expect a security team of limited size to read every report. What such a team actually needs is at least a tool that can perform an initial coarse grouping and first indicate which entries are similar to one another and which deserve immediate attention. This is precisely the task that an automated classification method should undertake.

Classical clustering algorithms are obvious candidates: they do not require labeled training data, run quickly on ordinary hardware, and produce results that are easy to explain to non-specialists. However, two problems



© 2026 The Author(s). Authors retain copyright in their articles.

are often passed over in practice. The first is the selection of K, which is often set by habit or intuition. The resulting groups are unstable, and it is difficult to provide a convincing reason afterward. The second is trust. Once clustering results have been calculated, a conventional process has no step responsible for checking whether the input data or centroids have subsequently been modified, either because of a storage-layer failure or because a person with write permission made a change. A silently modified centroid is worse than no clustering because it creates false confidence.

This study presents a lightweight method that addresses both points. The value of K is jointly determined by the elbow method, the silhouette coefficient, and the Calinski-Harabasz index, and SHA-256 digests are attached to the original dataset and each centroid so that tampering can be detected in constant time. The entire process requires only a few milliseconds on an ordinary laptop, which is why it can be embedded in teaching laboratories and resource-constrained dashboards rather than running only on a well-provisioned backend.

The contributions of this study are as follows:

1) A reproducible five-feature extraction scheme that converts heterogeneous NVD-style records into fixed-length numerical vectors.
2) An objective procedure for selecting K that combines the elbow method with the silhouette coefficient and the Calinski-Harabasz index.
3) A layer of SHA-256 integrity verification that hashes the original dataset and each centroid separately, with additional overhead only at the sub-millisecond level.

## 2. Related Work

Research on vulnerability prioritization can be roughly divided into several groups. Rule-based methods depend on signatures manually written by experts. They can be accurate under known threat models, but they are difficult to maintain once attack techniques evolve, and the maintenance cost ultimately falls on people willing to keep updating the rule base. Supervised classifiers represented by support vector machines (SVMs) and random forests replace the maintenance burden with another cost: they require labeled data and careful engineering, neither of which is inexpensive [2]. Deep learning methods have reported good results, but their appetite for graphics processing units (GPUs) and large-scale training corpora is incompatible with the lightweight objective pursued by this study [3].

Clustering methods have also been attempted. Ghosh and Goyal used hierarchical clustering to group similar CVE entries [4]. The resulting dendrogram was conceptually clear but difficult for non-specialists to act upon. Recent work has used Transformer-based embeddings to obtain semantically richer clusters, at the cost of a clear increase in computational overhead [5]. To the knowledge of this study, no previous work has combined K-Means clustering with cryptographic integrity verification specifically for vulnerability classification. This is the gap that this study attempts to fill.

On the cryptographic side, SHA-256 belongs to the Secure Hash Algorithm 2 (SHA-2) family and produces a 256-bit digest. The relevant standard was issued by the National Institute of Standards and Technology (NIST) [6]. According to current understanding, it provides preimage resistance and collision resistance, and its computational cost is negligible for payloads of several kilobytes. Secure hashes have also been used for integrity exchange in distributed data scenarios [7]. This idea is consistent with the verification layer in this study, except that the target changes from shared data to clustering centroids.

The initialization problem of K-Means has long received attention. The k-means++ method proposed by Arthur and Vassilvitskii provides a theoretical improvement over random initialization and has become the default option in mainstream implementations [8]. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) represents another density-based approach and does not require the number of clusters to be specified in advance, but it introduces neighborhood parameters and greater runtime overhead [9].

Two limitations must be stated in advance. First, all data in this study are synthetic and were generated to imitate NVD records rather than captured from an online data feed. Therefore, conclusions about performance in real scenarios remain conditional and require evaluation on an actual corpus. Second, the comparison scope



© 2026 The Author(s). Authors retain copyright in their articles.

was deliberately narrowed to methods subject to the same lightweight constraints, rather than selecting the strongest baseline without regard to cost. These two points are discussed again in Section 5.

## 3. Method

The method performs four steps in sequence: converting vulnerability records into numerical vectors, selecting the number of clusters, running K-Means, and finally attaching integrity digests to the data and centroids. With a fixed random seed, the entire process is deterministic, which is important for reproducibility.

### 3.1 Feature Extraction and Problem Definition

Let $D = \{v_1, v_2, ..., vN\}$ denote a dataset containing N vulnerability records. Each record $v_i$ is represented by a five-dimensional feature vector $x_i = (cvss, ac, pr, scope, exp) \in \mathbb{R}^5$. The five components are the CVSS base score, access complexity (AC), privileges required (PR), scope, and an exploitability measure, respectively. The CVSS score follows the CVSS v3.1 specification [10]. These five features were selected because they appear in almost every NVD record and are themselves directly usable numerical values. Fields such as descriptive text and reference links were deliberately excluded so that the method can operate in an environment without a language model.

The task is to divide D into K mutually disjoint clusters $C_1, ..., C_k$, minimizing within-cluster variance while ensuring that the assignments remain credible when the storage environment is not fully trusted. This second requirement distinguishes this study from conventional K-Means: the value of the output lies not only in good grouping, but also in the ability to prove that it was calculated from the stated input.

### 3.2 Clustering and Selection of K

Given K, K-Means minimizes the following sum-of-squares objective:

$$J = \Sigma_{k=1}^{K} \Sigma_{x} \in C_k \, ||x - \mu_k||^2 \quad (1)$$

Here, $\mu_k$ is the centroid of cluster $C_k$. This study uses the standard Lloyd algorithm with k-means++ initialization and 30 random restarts, selecting the run with the lowest objective value J. All features are standardized to zero mean and unit variance before clustering; otherwise, the CVSS score, which has a wider numerical range, would dominate the distance calculation.

The selection of K is handled separately rather than left to intuition. This study considers three metrics at the same time: inertia, namely J(K), which is used to plot the elbow curve; the silhouette coefficient $s \in [-1, 1]$, which measures how similar a sample is to its own cluster relative to other clusters, with a higher value being better; and the Calinski-Harabasz (CH) index, which is the ratio of between-cluster variance to within-cluster variance and is also better when higher. None of these metrics is treated as the final decision by itself. The operating point is chosen where the elbow turns and the two quality metrics agree, thereby avoiding the selection of K by overfitting a single metric.

### 3.3 SHA-256 Integrity Verification

Two types of digests are calculated after clustering. For the dataset as a whole, H_data = SHA-256(serialize(D)); for each centroid, $H_cent,k = SHA\text{-}256(serialize(\mu_k))$, $k = 1 ... K$. The serialization method is fixed in advance and includes both the order of the records and the order of features within each vector; therefore, neither reordering nor numerical truncation can escape detection. The choice of hash function follows Federal Information Processing Standards (FIPS) Publication 180-4 [6].

The key point of this design is that if the input or any centroid is changed by even one byte, the corresponding digest will become completely different with overwhelming probability. Detection therefore requires only recalculating the digest and comparing it with the stored value, and this step takes constant time with respect to data size. What it does not provide must be emphasized: a digest can prove that the bytes have not changed, but it cannot prove that those bytes were correct from the beginning. If the trusted reference digest itself is replaced, this layer of protection also fails. Both are different problems and are left for future work.



© 2026 The Author(s). Authors retain copyright in their articles.

# 4. Experiments

## 4.1 Dataset

This study generated 600 vulnerability records modeled on the National Vulnerability Database, with each record carrying the five features described in Section 3.1. The samples were drawn from four Gaussian groups corresponding to four real-world severity bands: low (denial of service), medium (privilege escalation), high (remote code execution), and critical (data leakage). This construction makes the true structure known in advance, which facilitates verification of whether the clusters recovered by the algorithm agree with the severity bands. However, it also means that the values below cannot be treated as proof of performance on an online NVD feed. The specific dataset settings are shown in Table 1.

*Table 1: Dataset Overview*

| Item | Value |
|---|---|
| Total number of records N | 600 |
| Number of features | 5 (CVSS, AC, PR, Scope, Exploitability) |
| Feature scaling | StandardScaler ($\mu = 0$, $\sigma = 1$) |
| Test range of K | 2, 3, 4, 5, 6 |
| K-Means initialization | k-means++, 30 restarts |
| Random seed | 42 |

The seed was fixed so that this run could be reproduced exactly. This is more important than it may appear because, with 30 random restarts, the optimal run may not be the same on two machines.

## 4.2 Experimental Environment

All experiments were conducted on a laptop equipped with an Intel i5-1135G7 central processing unit (CPU) and 16 GB of memory, using Python 3.11 and scikit-learn 1.5 [11]. The random seed was fixed at 42, and no GPU was used. No cross-hardware parameter scan was performed; therefore, these absolute runtimes apply only to this machine. Only the relative overhead of clustering and hashing is considered transferable.

## 4.3 Results of Selecting K

Table 2 gives the three quality metrics corresponding to each K within the test range. When K = 4, the silhouette coefficient is the highest at 0.4043, the CH index is also the highest at 1187.05, and the elbow curve shows a clear turn at this point. Together, the three results are sufficient to set K = 4 as the operating point. This also happens to equal the four severity bands built into the data, which is reassuring but should not be overinterpreted. On real data, these metrics may conflict, in which case a substantive reason is needed for the decision rather than a simple comparison of numerical magnitudes.

*Table 2: Clustering Quality under Different Values of K*

| K | Silhouette Coefficient ↑ | CH Index ↑ | Inertia ↓ |
|---|---|---|---|
| 2 | 0.2528 | 758.91 | 1321.6 |
| 3 | 0.3392 | 970.18 | 1018.4 |
| 4 | 0.4043 | 1187.05 | 883.6 |
| 5 | 0.3586 | 1034.58 | 782.7 |
| 6 | 0.3849 | 1098.52 | 703.9 |

Both the silhouette coefficient and the CH index peak at K = 4, whereas inertia decreases monotonically as K increases. This is inevitable because adding a cluster can only reduce the within-cluster sum of squares. The elbow is identified as the point after which the decrease in inertia becomes more gradual.

## 4.4 Cluster Profiles

Table 3 describes the four clusters obtained when K = 4. Cluster 0 groups low-severity denial-of-service problems (mean CVSS ≈ 3.6); Cluster 1 corresponds to medium-severity privilege-escalation attempts (CVSS ≈ 5.8); Cluster 2 corresponds to high-severity remote code execution (CVSS ≈ 7.9); and Cluster 3 contains critical data-leakage vulnerabilities (CVSS ≈ 9.2). Without being given any manual labels, the automatic grouping agrees with the intuitive severity hierarchy. This is the most encouraging result of the experiment.



© 2026 The Author(s). Authors retain copyright in their articles.

However, it does not show that these labels are semantically correct; it shows only that the algorithm recovered four different severity bands.

*Table 3: Profiles of the Four Clusters when K = 4*

| Cluster | Number of Samples | Proportion | Mean CVSS | Mean Exploitability | Dominant AC |
|---|---|---|---|---|---|
| 0 (Low) | 152 | 25.3% | 3.62 | 3.58 | 3 (High Complexity) |
| 1 (Medium) | 158 | 26.3% | 5.81 | 5.74 | 2 |
| 2 (High) | 148 | 24.7% | 7.93 | 7.86 | 1 (Low Complexity) |
| 3 (Critical) | 142 | 23.7% | 9.18 | 9.12 | 1 (Low Complexity) |

After rounding, the proportions sum to 100%. Mean exploitability follows a trend very close to mean CVSS, which is expected because both come from the same set of severity measures. It is included here mainly as a consistency check rather than as independent evidence.

## 4.5 Integrity Verification Overhead

Table 4 presents the SHA-256 timing results. Hashing the complete 600-row dataset requires 0.1167 ms, while hashing one centroid, which contains only five floating-point numbers, requires approximately 0.014 ms. Both values are negligible compared with the 14.01 ms required to train K-Means itself. To verify whether the digest was actually functioning, this study changed one byte in the serialized input by appending the string "TAMPER" and observed a completely different digest. The mechanism can therefore detect tampering. Specifically, it detects changes at the byte level rather than semantic correctness.

*Table 4: SHA-256 Integrity Verification Results*

| Item | SHA-256 Digest (Truncated) | Time (ms) |
|---|---|---|
| Complete Dataset (600 Rows) | 3964800b6070f7e390f032e6... | 0.1167 |
| Centroid 0 | 0365edaafd2b8dc7... | 0.0142 |
| Centroid 1 | ed9bc3313911a94a... | 0.0138 |
| Centroid 2 | a17c4f2b8e6d3109... | 0.0140 |
| Centroid 3 | 7b3a91c5e2f048d6... | 0.0141 |
| Tampered Dataset | e4b7e13001a376a9... | 0.1171 |

The timing of the tampered dataset is almost unchanged, indicating that the overhead is spent on hashing itself rather than on comparison or lookup.

## 4.6 Training Time

A complete K-Means fit at K = 4, including 30 initializations, takes 14.01 ms. Including feature extraction, clustering, and hashing, the total time of the complete process remains well below 1 s. This value is what is meaningful for the target scenarios: interactive teaching demonstrations cannot tolerate pauses of several seconds, and dashboards deployed on small edge devices also have no capacity for heavy models. However, this is only a conclusion about speed and cannot be used to prove that the classification is correct.

# 5. Discussion

The experimental results support only a limited conclusion: although K-Means is extremely simple, it can recover reasonable severity bands from raw vulnerability features. The four automatically discovered clusters agree fairly well with the low-medium-high-critical hierarchy and are sufficient for junior analysts to perform an initial coarse screening. However, this result is closely tied to the synthetic data. Real NVD entries will have skewed feature distributions, missing values, and a long tail formed by rare weaknesses. Whether the same severity bands will emerge remains an open question that should be answered before the method enters any production process.

The contribution of integrity verification is less immediately apparent. Its overhead—0.1167 ms for 600 records—is three orders of magnitude smaller than that of clustering itself, so enabling it by default imposes almost no user-perceptible cost. The single-byte test also confirmed that the tamper-resistance property worked as expected. Its limitation is conceptual rather than related to performance: the digest can verify only that the bytes have not changed, not that they were the correct bytes from the beginning. Once the storage of the digest is compromised, the entire mechanism loses its meaning.



© 2026 The Author(s). Authors retain copyright in their articles.

The assumptions inherent in K-Means may not hold for real data. It assumes that clusters are approximately spherical and have similar variances, whereas actual vulnerability features are more likely to form elongated and unevenly distributed groups. For some datasets, a density-based method such as DBSCAN [9] or a probabilistic representation such as a Gaussian mixture model would provide a more faithful fit. The cost is that both introduce additional parameters and runtime, departing from the lightweight position maintained by this study. Incremental updating is another gap. The current integrity verification treats the dataset as a static whole; supporting streaming additions without rehashing the entire dataset will require a redesign.

## 6. Conclusion and Future Work

This study proposes a lightweight vulnerability classification method that combines K-Means clustering with SHA-256 integrity verification. On 600 NVD-style records, the silhouette coefficient reaches 0.4043 when K = 4, while the hashing step adds only 0.1167 ms of overhead. The resulting clusters correspond to meaningful severity levels, and the digests make undetected modification of the input or centroids highly unlikely.

There are three natural directions for future work. First, the method should be run on a real NVD corpus, using Term Frequency-Inverse Document Frequency (TF-IDF) or Bidirectional Encoder Representations from Transformers (BERT) to extract text features, to determine whether these severity bands withstand testing on real data. Second, K-Means should be compared with density-based and hierarchical methods under the same resource budget, rather than only with heavier models. Third, digital signatures should be added to the integrity verification layer so that cluster assignments can be verified by a third party rather than only compared with a locally stored digest.

## References

[1] National Institute of Standards and Technology. National Vulnerability Database [EB/OL]. https://nvd.nist.gov, 2026.

[2] Li Z, Zou D, Xu S, et al. VulDeePecker: A deep learning-based system for vulnerability detection[J]. arXiv preprint arXiv:1801.01681, 2018.

[3] Ghosh A, Goyal S. Hierarchical clustering of CVE entries[C]//Proceedings of the 2019 ACM SIGSAC Conference. 2019: 211-220.

[4] Chen J, Liu X, Wang Y, et al. Transformer-based vulnerability embedding[C]//Proceedings of the 32nd USENIX Security Symposium. 2023: 1023-1040.

[5] National Institute of Standards and Technology. FIPS PUB 180-4: Secure Hash Standard (SHS)[S]. Gaithersburg: NIST, 2015.

[6] Chan P K, Stolfo S J. Sharing distributed data with secure hashes[C]//Proceedings of the 2005 IEEE Symposium on Security and Privacy. 2005: 78-85.

[7] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. 1996: 226-231.

[8] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python[J]. Journal of Machine Learning Research, 2011, 12: 2825-2830.

[9] Cover T M, Thomas J A. Elements of Information Theory[M]. 2nd ed. Hoboken: John Wiley & Sons, 2006.

[10] Pandey N, Solanki V. Vulnerability classification using machine learning[J]. Journal of Information Security, 2021, 12(3): 45-58.

[11] FIRST. CVSS v3.1 Specification Document [S]. https://www.first.org/cvss/, 2019.



© 2026 The Author(s). Authors retain copyright in their articles.

## Funding

This research received no external funding.

## Conflicts of Interest

The authors declare no conflict of interest.

## Acknowledgment

This paper is an output of the science project.

## Copyrights

Copyright for this article is retained by the author (s), with first publication rights granted to the journal. This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).



© 2026 The Author(s). Authors retain copyright in their articles.