Software: Practice and Experience

WILEY

SPECIAL ISSUE PAPER

Software-Driven Edge Computing and Artificial Intelligence of Things

VIDA: Enhancing Vulnerability Detection and Impact Analysis via Diffusion-Guided Heterogeneous Knowledge Aggregation

Ying Xing[1] | Zujun Liu[2] | Yonghao Xing[1] | Chengyu Song[3] | Bin Yang[4] | Yun Yang[5] | Yuwei Zhang[6] | Hua Zhao[2]

[1]School of Intelligent Engineering and Automation, Beijing University of Posts and Telecommunications, Beijing, China | [2]Smart Steps Digital Technology Company Limited, Beijing, China | [3]School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, China | [4]Graph Neural Network and Artificial Intelligence Team, China Unicom Research Institute, Beijing, China | [5]National Pilot School of Software, Yunnan University, Kunming, China | [6]Institute of Software, Chinese Academy of Sciences, Beijing, China

Correspondence: Yun Yang (yangyan19@hotmail.com)

Received: 24 March 2026 | **Revised:** 4 June 2026 | **Accepted:** 12 July 2026

Keywords: diffusion model | heterogeneous information network | knowledge graph | software security | vulnerability detection

ABSTRACT

Background: Detecting software vulnerabilities and analyzing their impacts on interconnected software components are crucial to ensuring the security of software-driven edge computing and Artificial Intelligence of Things (AIoT) systems, especially with the growing prevalence of software supply chain attacks. Knowledge graphs (KGs) have been used to model complex semantic dependencies between vulnerabilities and software components. However, existing KGs often contain noisy or irrelevant relations, which significantly hinder their utility in downstream tasks.

Objective: This study aims to improve vulnerability detection and impact analysis by denoising vulnerability KGs and generating task-oriented vulnerability representations.

Methods: We propose a diffusion-based graph learning framework for denoising vulnerability KGs. The framework first employs a diffusion process to reconstruct relational structures within the KG, guided by a graph alignment mechanism that integrates collaborative signals from the software component–vulnerability bipartite graph. This alignment encourages the denoised KG to preserve relations most relevant to downstream tasks. Subsequently, a heterogeneous information network aggregates multi-relational information from the denoised KG to produce knowledge-enhanced embeddings for vulnerabilities.

Results: Through this two-stage learning paradigm, the framework effectively captures both semantic and relational dependencies among heterogeneous entities and generates task-oriented representations for vulnerability detection and impact analysis. Experimental results on a widely used benchmark demonstrate that the framework can effectively identify latent relationships between vulnerabilities and software components and predict high-risk vulnerabilities.

Conclusion: The proposed framework improves the utility of vulnerability KGs by suppressing noisy relations and preserving task-relevant information, providing practical insights for proactive vulnerability mitigation and patch recommendation.

1 | Introduction

With the rapid advancement of edge computing and the Artificial Intelligence of Things (AIoT), the security of modern software systems has become a central concern for the reliability, privacy, and operational safety of intelligent services [1, 2]. In representative AIoT scenarios such as smart cities, smart homes, industrial IoT, and smart healthcare, edge software stacks typically rely on a large number of heterogeneous libraries, third-party components, containerized services, and firmware modules to support real-time sensing, local intelligence, and edge–cloud collaboration [3, 4]. Recent studies have also explored lightweight knowledge distillation and feature compression techniques to support efficient prediction services in edge computing scenarios [5]. However, this highly interconnected software ecosystem substantially enlarges the attack surface: once a vulnerable component is exploited, attackers may leverage weaknesses in the software supply chain to propagate risks to distributed edge devices, data processing pipelines, and latency-sensitive AIoT applications [6, 7]. According to the Common Vulnerabilities and Exposures (CVE) database, the number of publicly disclosed software vulnerabilities has continued to grow in recent years, reflecting the expanding attack surface of modern software ecosystems [8]. This challenge is particularly severe for software-driven edge computing and AIoT systems, where resource-constrained devices, heterogeneous execution environments, and complex software compositions make comprehensive and fine-grained security auditing far more difficult than in centralized platforms [9]. Meanwhile, the tight coupling among edge software, cloud services, end devices, and third-party dependencies enables vulnerabilities in a single component to quickly propagate along dependency chains and amplify their system-level impact [10, 11]. Therefore, efficiently detecting vulnerabilities and accurately analyzing their potential impact in such dependency-rich environments has become a critical research problem for secure software-driven edge computing and AIoT systems [9, 12].

Traditional vulnerability detection approaches [13, 14], which primarily rely on static or dynamic analysis, exhibit inherent limitations such as high false positive rates and low efficiency. To overcome these issues, recent studies have introduced knowledge graphs (KGs) to model and reason about the intricate semantic dependencies across software components [15–17]. By representing software components and their relationships in a structured form, KGs enable a more holistic view of the software supply chain, thereby supporting the identification of latent vulnerabilities and their propagation paths. Building on this foundation, researchers have further leveraged Heterogeneous Information Networks (HINs) due to the heterogeneous nature of KGs [18]. Through message-passing mechanisms, HIN models can capture complex dependencies among code snippets, packages, and vulnerabilities within the software supply chain. This modeling capability facilitates comprehensive embedding representations that enhance both vulnerability detection and impact analysis at the code–package level.

However, as vulnerability KGs continue to grow, an increasing amount of noisy information (i.e., erroneous relation edges between vulnerability entity nodes and their corresponding attribute nodes) contributes non-meaningful knowledge. Incorporating these noisy edges during model training degrades the quality of embeddings learned by HIN models. As illustrated in Figure 1, such noise can obscure essential dependency signals during the knowledge aggregation process, leading to confusion in representation learning and ultimately resulting in suboptimal performance in downstream tasks. Prior research [19, 20] has demonstrated that the effectiveness of KG-enhanced models is highly dependent on the quality of the input graph and is particularly sensitive to noise. Empirical investigations [21, 22] further reveal that automatically constructed KGs often contain a substantial proportion of noisy triples, which poses a critical challenge for KG refinement. Existing HIN-based methods still exhibit notable limitations when applied to vulnerability analysis tasks. For instance, Heterogeneous Attention Network (HAN) [23] relies on manually defined meta-paths to guide message aggregation, which requires domain expertise and may fail to capture implicit dependency structures in vulnerability KGs. Similarly, Heterogeneous Graph Neural Network with Co-Contrastive Learning (HeCo) [24] and Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding (MAGNN) [25] aggregate neighbor information along predefined relational schemas, implicitly assuming that all edges within the same relation type are equally informative and trustworthy. However, in the context of automatically constructed vulnerability KGs, this assumption frequently breaks down. First, most HIN models assume that the input graph is relatively clean and reliable, making them inherently sensitive to erroneous or irrelevant edges that frequently appear in automatically constructed vulnerability KGs. Second, current approaches tend to treat all relational edges equally during message passing, lacking the ability to adaptively distinguish informative dependency signals from noisy ones. Third, the absence of an explicit graph refinement mechanism means that noise accumulated during KG construction is directly propagated into the learned embeddings, inevitably degrading model performance on downstream tasks such as vulnerability detection and impact analysis. These limitations collectively highlight the urgent need for a principled noise-aware framework that can proactively refine the KG structure before embedding learning.

To mitigate the limitations caused by noise in KGs, it is essential to directly prune noisy relations (i.e., erroneous entity–attribute edges) to reconstruct a denoised KG, thereby enhancing the quality of learned embeddings. Diffusion models [26, 27] provide a natural and effective mechanism for addressing this challenge. Recent studies have explored diffusion models for graph-related tasks from complementary perspectives, offering valuable insights but also revealing clear gaps when applied to vulnerability security scenarios. On the graph generation side, Discrete Denoising Diffusion for Graph Generation (DiGress) [28] introduces a discrete denoising diffusion process that progressively edits graph structure by adding or removing edges, demonstrating that iterative noise injection and recovery can effectively model complex graph distributions. On the graph purification side, Graph Defense Diffusion Model (GDDM) [29] proposes a Graph Defense Diffusion Model that leverages the iterative denoising capability of diffusion models to purify adversarially attacked graphs, showing that diffusion-based refinement can restore corrupted graph structure and node features simultaneously. For KG completion, Fact Embedding through Diffusion Model (FDM) [30] directly learns the distribution of plausible

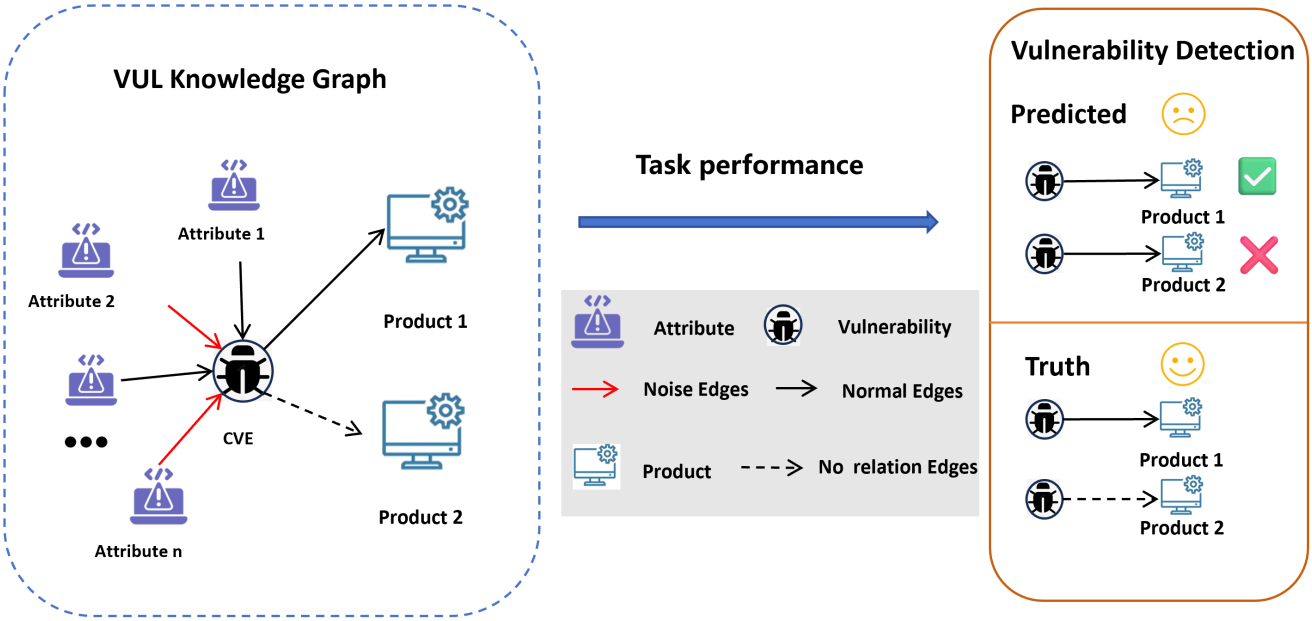

FIGURE 1 | Noise edge in KG.

facts via a conditional fact denoiser, framing entity prediction as a conditional generation task driven by diffusion. Regarding general graph denoising, the unified graph denoising (UGD) framework [31] provides a unified approach for removing both noisy edges and noisy node features through high-order neighborhood proximity evaluation and graph auto-encoder reconstruction. Despite these advances, a critical gap remains: none of these methods is designed for the heterogeneous, domain-specific noise patterns that arise in automatically constructed vulnerability KGs, where erroneous entity–attribute edges often co-exist with legitimate structural dependencies. Unlike general-purpose graph diffusion or purification methods, our approach is tailored to the vulnerability security scenario, explicitly targeting the denoising of heterogeneous relational edges to reconstruct a refined vulnerability KG that enables more accurate downstream vulnerability detection and impact analysis. By modeling the process of gradually adding and removing noise, diffusion models can learn to reconstruct high-quality data distributions from corrupted inputs. This generative capability aligns closely with the goal of recovering meaningful relations while filtering out noisy ones. Accordingly, we propose a generative KG diffusion paradigm that reconstructs a denoised vulnerability KG via iterative noise injection and recovery, effectively preserving informative structural knowledge while eliminating noisy relations.

Specifically, we design a two-stage framework, termed VIDA, for security analysis of software supply chains in software-driven edge computing and AIoT environments. The framework aims to achieve more accurate *V*ulnerability detection and *I*mpact analysis by leveraging *D*iffusion-guided heterogeneous knowledge *A*ggregation. In the first stage, the diffusion model performs denoising on the vulnerability KG to obtain a refined graph, thereby enabling the learning of purified embeddings that capture essential vulnerability semantics. In the second stage, the HIN model aggregates both structural and semantic information across heterogeneous entities from a bipartite graph composed of software components and vulnerabilities. This hierarchical aggregation process captures local attribute-level correlations as well as global dependency structures within the vulnerability KG and bipartite graph. The integration of these two components offers complementary advantages. The diffusion model enhances data quality by reconstructing a denoised KG, while the HIN model leverages the refined structure to learn precise embeddings. Unlike existing diffusion models designed for general-purpose graph generation, adversarial defense, or KG completion, VIDA is specifically tailored to the heterogeneous noise patterns inherent in vulnerability KGs—particularly the co-existence of erroneous entity–attribute edges with legitimate structural dependencies. To address this, VIDA introduces a structure-aware progressive denoising mechanism that does not rely on predefined meta-paths or relation-type assumptions. Instead, it adaptively distinguishes informative dependency signals from noisy edges and dynamically prunes noisy connections during the diffusion process. This allows VIDA to preserve critical vulnerability propagation paths while effectively suppressing irrelevant or erroneous information from interfering with embedding learning. This generative graph refinement paradigm, oriented to the vulnerability security scenario, endows VIDA with stronger robustness and domain adaptability when handling the inherent noise of automatically constructed KGs, compared to general-purpose diffusion methods. Extensive experiments conducted on a widely used real-world benchmark VulKG [17] demonstrate that VIDA achieves superior performance in both vulnerability detection and impact analysis compared with state-of-the-art baselines.

The main contributions of this paper are as follows:

- We make the first attempt at proposing a generative diffusion-based KG refinement paradigm to mitigate the noise issue in vulnerability KGs.

- We develop a novel two-stage framework that integrates a diffusion model with an HIN model for knowledge aggregation, aiming to enhance vulnerability analysis.
- We perform comprehensive experiments on the real-world vulnerability benchmark for validating the effectiveness of the proposed framework.
- We publicly release the replicate package of VIDA on GitHub.[1] The open-sourced artifacts can better support the researchers in reproducing VIDA.

Article Organization. The remainder of this paper is organized as follows: Section 2 introduces in detail the proposed framework. Section 3 provides the experimental setups and results of our research. Section 4 presents the case study discussion and discloses the threats to the validity of our approach. Section 5 describes the related work. Section 6 concludes with research findings and future directions.

2 | The Proposed Framework VIDA

This section provides a detailed description of the proposed framework. We begin by defining the structure and semantics of the graphs employed in this paper. We then elaborate on the design of the VIDA framework, with particular emphasis on its two core stages. As illustrated in Figure 2, the first stage adopts a diffusion–based paradigm to denoise the original KG in a generative manner by removing noisy edges, thereby producing a refined graph. In the second stage, an HIN model is applied to the denoised KG to aggregate semantic information and learn precise embedding representations utilized for downstream tasks.

2.1 | Preliminary

The VIDA framework is constructed on two interconnected graphs: the KG $\mathcal{G}_K = N_{att} \rightarrow N_{vul}$ and the bipartite graph $\mathcal{G}_B = N_{vul} \rightarrow N_{sc}$, where N_{vul} denotes nodes of vulnerability entities, N_{att} denotes nodes of vulnerability attributes, and N_{sc} denotes nodes of software component entities. Specifically, $\mathcal{G}_K$ is a vulnerability-oriented KG composed of two node types (i.e., N_{vul} and N_{att}) and their relation edges in the form of $N_{att} \rightarrow N_{vul}$. Each N_{vul} node is associated with several N_{att} nodes, including weakness, domain, exploit, and author. $\mathcal{G}_B$ is a bipartite graph that links vulnerabilities to software components, consisting of two entity node sets (i.e., N_{sc} and N_{vul}) with edges represented as $N_{vul} \rightarrow N_{sc}$. Each edge indicates that a particular software component is affected by a specific vulnerability. The two graphs share the same set of N_{vul}, collectively forming a heterogeneous network that integrates multiple entity types and diverse relations.

2.2 | Diffusion-Based KG Denoising

In the first stage, the KG refinement task utilizes a diffusion model to perform noise injection and denoising within the relational space of the KG, thereby reconstructing a cleaner and more task-relevant KG. To achieve this objective, we design a joint training framework that comprises two complementary tasks, diffusion reconstruction and graph alignment, as described below.

2.2.1 | Diffusion Reconstruction

For each vulnerability node N_{vul} in the original KG $\mathcal{G}_K$, a corresponding vector representation is constructed as:

$$\mathbf{e}_{vul} = \left[e_{vul,1}, e_{vul,2}, \ldots, e_{vul,|N_{att}|}\right] \tag{1}$$

where $e_{vul,att} \in \{0, 1\}$ indicates whether a relation exists between N_{vul} and a given attribute node N_{att}. The entire $\mathcal{G}_K$ can thus be represented as a vulnerability–attribute embedding matrix $\mathbf{E}_0$ that serves as the initial state for the diffusion model. The diffusion model outputs the probability that each vulnerability node N_{vul} is associated with every attribute node N_{att}, reflecting the confidence level of their true relationships after denoising. Based on these confidence scores, a predefined threshold is applied to

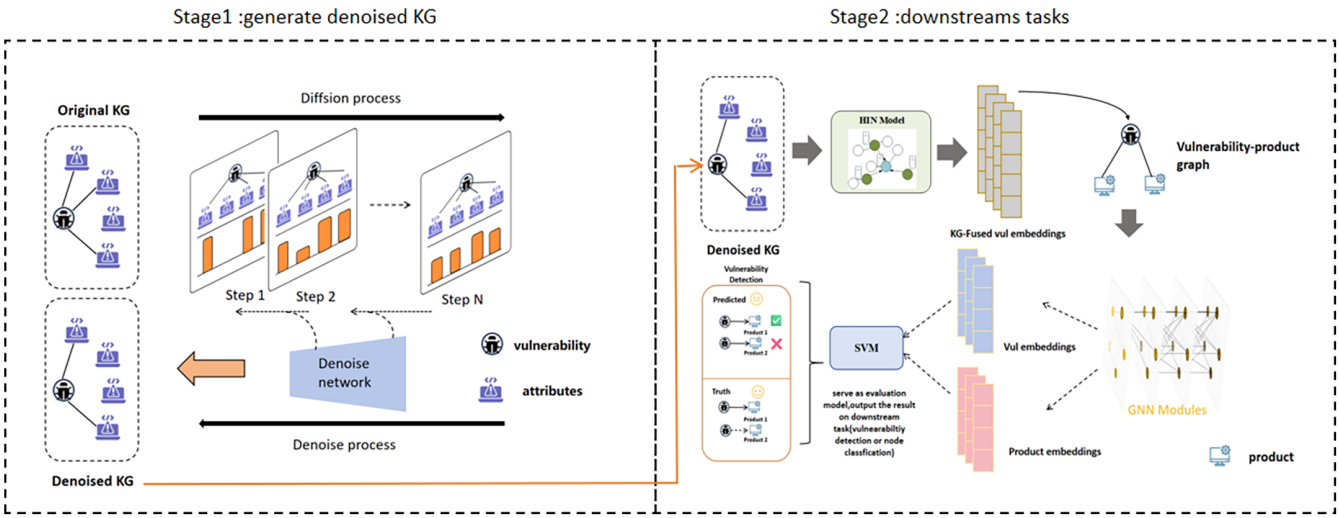

FIGURE 2 | Framework of VIDA.

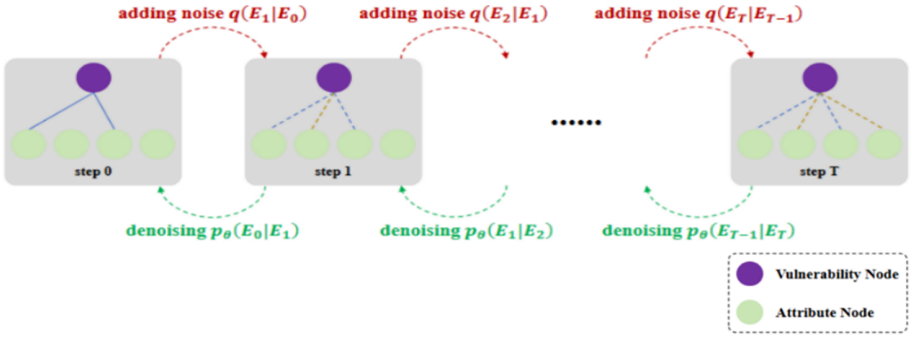

FIGURE 3 | The diffusion model training process.

retain only high-confidence relations, yielding the reconstructed and denoised KG $\mathcal{G}'_K$.

This reconstruction process aligns with the standard denoising diffusion probabilistic model framework. Initially, the forward diffusion process gradually adds Gaussian noise to the initial embedding matrix $\mathbf{E}_0$, converting the sparse and discrete graph structure into a continuous latent distribution. At each time step t, this forward process can be formally formulated as:

$$q(\mathbf{E}_t|\mathbf{E}_{t-1}) = \mathcal{N}\left(\mathbf{E}_t; \sqrt{1-\beta_t}\mathbf{E}_{t-1}, \ \beta_t I\right) \tag{2}$$

where $q(\cdot)$ denotes the probabilistic transition from the embedding state at time step $t-1$ to the next step t, $\mathcal{N}$ denotes the Gaussian distribution used to sample the new noisy embedding, β_t denotes the variance of Gaussian noise added at each diffusion step, I denotes the identity matrix ensuring isotropic noise across all embedding dimensions. This forward process progressively perturbs the original KG embeddings, transforming structured relational information into pure noise.

Next, during the reverse denoising process, we aim to gradually recover the original $\mathbf{E}_0$ from pure noise. A parameterized neural network p_θ is employed to approximate the reverse distribution:

$$p_\theta(\mathbf{E}_{t-1}|\mathbf{E}_t) = \mathcal{N}\left(\mathbf{E}_{t-1}; \mu_\theta(\mathbf{E}_t, t), \Sigma_\theta(\mathbf{E}_t, t)\right) \tag{3}$$

where $p_\theta(\cdot)$ denotes the parameterized reverse transition distribution that models the denoising process, $\mu_\theta(\mathbf{E}_t, t)$ and $\Sigma_\theta(\mathbf{E}_t, t)$ denote the predicted mean and variance of the recovered embedding at step $t-1$, respectively. The neural network p_θ is trained to predict the noise component added during the forward process, thereby enabling the progressive reconstruction of clean KG embeddings from their noisy counterparts through iterative refinement.

Typically, p_θ outputs the noise term $\varepsilon_\theta(\mathbf{E}_t, t)$ instead of directly predicting the mean $\mu_\theta(\mathbf{E}_t, t)$ for the noisy embedding $\mathbf{E}_t$ at time step t, from which the mean can be derived as:

$$\mu_\theta(\mathbf{E}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{E}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_\theta(\mathbf{E}_t, t)\right) \tag{4}$$

where $\alpha_t = 1-\beta_t$ denotes the retained signal ratio at current diffusion step, $\overline{\alpha}_t = \prod_{i=1}^{t}\alpha_i$ denotes the cumulative retained signal ratio after t diffusion steps.

As shown in Equation (5), the final denoised relational probabilities are obtained by applying a softmax layer to the output of the reverse process. Each element $\widehat{E}_{vul,att} \in [0,1]$ represents the confidence score of the relation between a vulnerability node and an attribute node. The top-k attribute nodes with the highest confidence scores are then selected to construct $\mathcal{G}'_K$.

$$\widehat{\mathbf{E}} = \texttt{Softmax}\left(p_\theta(\mathbf{E}_0|\mathbf{E}_T)\right) \tag{5}$$

Figure 3 illustrates the complete training process of the diffusion model. In this stage, the model is trained to predict the injected noise using a mean squared error (MSE) objective. The corresponding reconstruction loss $\mathcal{L}_R$ can be defined as:

$$\mathcal{L}_R = \mathbb{E}_{t,\mathbf{E}_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\overline{\alpha}_t}\mathbf{E}_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t\right)\right\|^2\right] \tag{6}$$

where $\epsilon \sim \mathcal{N}(0, I)$ denotes the true Gaussian noise, ϵ_θ denotes the network's predicted noise.

2.2.2 | Graph Alignment

If the diffusion process is applied solely to the original $\mathcal{G}_K$ during the KG refinement task, many relations irrelevant to downstream tasks may persist. This limitation stems from the ignorance of interactive associations between software component nodes N_{sc} and vulnerability nodes N_{vul} in the isolated KG diffusion operation. To address this issue, we introduce a graph alignment mechanism between $\mathcal{G}_K$ and the bipartite graph $\mathcal{G}_B$. During diffusion, the proposed mechanism integrates collaborative interactive signals captured from N_{sc}–N_{vul} pairwise interactions, which provides explicit auxiliary supervision for the overall KG reconstruction and denoising pipeline. By aggregating these interaction signals, the mechanism enhances the model's ability to capture vulnerability semantics and seamlessly embeds the component-vulnerability interactive knowledge into the denoised refined KG $\mathcal{G}'_K$. Consequently, the model is guided to recover relations that are better aligned with the requirements of downstream tasks during denoising.

Specifically, we design an alignment loss $\mathcal{L}_A$ that incorporates interaction information and KG-derived relational knowledge into the generation of the vulnerability embedding matrix. The detailed calculation pipeline of $\mathcal{L}_A$ is standardized and explained as follows (corresponding to Equation 7): First, the predefined

$N_{sc}-N_{vul}$ interaction matrix $\mathcal{J}$ is fused with the denoised relational probability matrix $\hat{\mathbf{E}}$ obtained from the KG reconstruction module to update $\mathcal{J}$, which realizes the effective integration of refined KG relational information into the interactive representation. Second, the updated interaction matrix $\mathcal{J}$ is multiplied by the software component embedding matrix $\mathbf{E}_{sc}$ to generate an enhanced vulnerability embedding matrix, which comprehensively fuses two levels of critical knowledge: the global relational knowledge from $\mathcal{G}_K$ and the local interactive knowledge between component and vulnerability nodes. Finally, the MSE between the enhanced vulnerability embedding matrix and the original vulnerability embedding matrix $\mathbf{E}_{vul}$ is calculated to formulate the final alignment loss $\mathcal{L}_A$.

$$\mathcal{L}_A = \left\| \left[\mathcal{J} \cdot \hat{\mathbf{E}}^{\top} \right]^{\top} \cdot \mathbf{E}_{sc} - \mathbf{E}_{vul} \right\|^2 \tag{7}$$

The overall loss function for the first-stage training process, as defined in Equation (8), jointly optimizes the reconstruction and alignment objectives through a weighting coefficient λ, which balances the contributions of the two task components.

$$\mathcal{L}_D = (1-\lambda)\mathcal{L}_R + \lambda \mathcal{L}_A \tag{8}$$

2.3 | Graph Knowledge Aggregation

After obtaining the denoised KG $\mathcal{G}'_K$ in the first stage, we perform graph aggregation over both $\mathcal{G}'_K$ and the bipartite graph $\mathcal{G}_B$. Specifically, a relational graph attention network (RGAT) is applied to $\mathcal{G}'_K$ for heterogeneous knowledge aggregation, while an HIN model is adopted on $\mathcal{G}_B$ to capture heterogeneous relational semantics among software component–vulnerability interactions.

In the RGAT module, each vulnerability node N_{vul} interacts with its neighboring attribute nodes N_{att} through relation-specific attention mechanisms. Each relation edge is represented as a triple (vul, rel, att), where rel denotes the relation type between N_{vul} and N_{att}. To assess the relative importance of different neighbors, a relation-aware attention coefficient is computed as:

$$\alpha^{rel}_{vul,att} = \frac{\exp\left(\sigma\left(a^{\top}\left[W\mathbf{E}_{vul} | W\mathbf{E}_{att} | W_{rel}\mathbf{E}_{rel}\right]\right)\right)}{\sum_{att' \in N_{att}} \exp\left(\sigma\left(a^{\top}\left[W\mathbf{E}_{vul} | W\mathbf{E}_{att'} | W_{rel'}\right]\right)\right)} \tag{9}$$

where W denotes the learnable transformation matrix for node types, W_r denotes the learnable transformation matrix for relation types, $\mathbf{E}_{vul}$ denotes the embeddings of the vulnerability node, $\mathbf{E}_{att}$ denotes the embeddings of the attribute node, $\mathbf{E}_{rel}$ denotes the embeddings of their relation, $\sigma(\cdot)$ denotes the LeakyReLU activation function, and a denotes the attention vector.

As defined in Equation (10), RGAT aggregates information from neighboring N_{att} to produce the KG-enhanced embedding $\mathbf{E}^{KG}$ for each N_{vul} using the computed $\alpha^{rel}_{vul,att}$. This aggregation process enables the model to capture both semantic relevance and higher-order relational dependencies within the denoised $\mathcal{G}'_K$. After the RGAT aggregation, the vulnerability embeddings are updated with knowledge-enhanced representations and subsequently serve as inputs to the HIN model.

$$\mathbf{E}^{KG}_{N_{vul}} = \sigma\left(\sum_{att \in N_{att}} \alpha^{rel}_{vul,att} W \mathbf{E}_{att} \right), \tag{10}$$

The HIN model operates on $\mathcal{G}_B$ to learn representations that capture heterogeneous relational semantics. Its core mechanism involves hierarchical aggregation, which accounts for the varying importance of neighboring nodes or meta-paths during representation learning. In the VIDA framework, we employ MAGNN, a metapath-based GNN that leverages the metapath2vec strategy for embedding generation. Specifically, the denoised $\mathcal{G}'_K$ is first processed by RGAT to obtain knowledge-enhanced vulnerability embeddings. These embeddings are then incorporated into the HIN model to further learn interaction-aware representations from $\mathcal{G}_B$. Through this joint aggregation process, VIDA generates unified feature vectors for all entity types within the denoised $\mathcal{G}'_K$, which are subsequently utilized to address the two downstream tasks.

3 | Evaluation

To evaluate the effectiveness of VIDA, we raise the following two research questions (RQs):

- **RQ1: How does VIDA perform in vulnerability detection and impact analysis compared with state-of-the-art baselines?** This RQ aims to evaluate the effectiveness of VIDA against representative GNN-based approaches. To achieve this, we conduct a comprehensive comparison with three baselines on a widely used real-world benchmark.
- **RQ2: How does each component impact the performance of VIDA?** Since VIDA introduces a diffusion-based strategy and a knowledge aggregation mechanism, this RQ seeks to analyze the contributions of each component through an ablation study.

3.1 | Experimental Setup

3.1.1 | Benchmark

We conduct experiments on the VulKG benchmark [17], a large-scale and compact KG specifically designed for vulnerability analysis. VulKG provides structured and interconnected representations of vulnerability information by integrating historical data from multiple authoritative sources, including the National Vulnerability Database (NVD),[2] Chinese NVD,[3] Exploit Database,[4] CVE, and Common Weakness Enumeration (CWE).[5] After data cleaning and integration, VulKG contains over 276,000 entities and 1 million relations, representing connections among vulnerabilities, exploits, affected products (i.e., software components), referred domain names, and more. VulKG offers three key advantages:

- *Representativeness:* It consolidates vulnerability data from authoritative national and international repositories to ensure comprehensive coverage.

TABLE 1 | Statistics of VulKG.

Entity type	Count	Relation type	Count
Vulnerability	148,609	EXPLOITS	28,971
Exploit	43,743	AFFECTS	212,654
Weakness	926	REFERS_TO	429,728
Product	44,269	EXAMPLE_OF	71,031
Author	9204	WRITES	43,743
Domain	9578		

- *Diversity:* It spans a variety of software functionalities, including testing tools, search engines, and game frameworks.
- *Sufficiency:* Its large-scale and rich semantic relationships make it suitable for evaluating KG reasoning and model training.

Table 1 summarizes the entity and relation types, along with their corresponding statistics. Specifically, we utilize six primary entity types (i.e., Vulnerability, Product, Weakness, Exploit, Author, and Domain) from VulKG in our experiments. Five relation types describe the interactions among these entities; for example, the `AFFECTS` relation connects a Vulnerability entity to a Product entity. These heterogeneous relations capture both structural and semantic dependencies among software components, thereby enabling comprehensive reasoning across multiple vulnerability dimensions. Following standard experimental settings, we divide the benchmark dataset into training, validation, and testing sets with an 8:1:1 ratio. The training set is used to learn graph representations and model parameters, while the validation set is employed for hyperparameter tuning and early stopping. The testing set is reserved for evaluating the performance of VIDA and baselines in detecting potential vulnerabilities and inferring new relations in unseen scenarios.

3.1.2 | Baselines

In this paper, we focus on addressing both vulnerability detection and impact analysis tasks using graph neural networks (GNNs). To this end, we compare VIDA against the following three state-of-the-art baseline approaches to evaluate its effectiveness.

- *DSHGT:* DSHGT [18] integrates HINs with GNNs to enhance static vulnerability detection and impact analysis through multi-relational knowledge aggregation.
- *CFExplainer:* CFExplainer [32] introduces an interpretable GNN-based framework that applies counterfactual reasoning to identify minimal structural or feature changes affecting model predictions, thereby improving interpretability while maintaining detection accuracy.
- *CoCa:* CoCa [33] adopts a contrastive learning paradigm to refine vulnerability representations by distinguishing paired samples for enhancing model robustness.

3.1.3 | Metrics

This paper employs seven commonly used evaluation metrics to assess the performance of VIDA and the baseline approaches across the two target tasks. These metrics comprehensively capture different perspectives of model performance. Collectively, they reflect both global predictive capability and task-specific effectiveness. Higher metric values consistently indicate superior model performance and stronger robustness in handling vulnerability detection and impact analysis.

- For the vulnerability detection task, we utilize four evaluation metrics: Accuracy, Precision, Recall, and F1-Measure (F1). Accuracy serves as the most intuitive indicator, measuring the overall correctness of model predictions by calculating the proportion of correctly classified samples among all test samples. It reflects the model's general ability to distinguish between vulnerable and non-vulnerable instances. Precision measures the proportion of correctly identified vulnerable samples among all samples predicted as vulnerable, thereby evaluating the reliability of the model's positive predictions. A higher precision indicates that the model produces fewer false positives. Recall, on the other hand, measures the proportion of actual vulnerable samples that are correctly recognized by the model, assessing its capability to capture true vulnerabilities and reduce false negatives. Finally, F1 represents the harmonic mean of Precision and Recall, providing a balanced evaluation metric to assess the model's detection results.
- For the impact analysis task, we report Hits@N, area under the curve (AUC), and Average Precision (AVG_{pre}) as evaluation metrics. Hits@N measures the proportion of correct entities among the top-N predictions generated by the model. By evaluating different values of N (set to 1, 3, and 10 in our experiments), Hits@N provides insight into the model's accuracy at various ranking positions, reflecting its ability to prioritize the most relevant entities. AUC evaluates the model's capability to distinguish between positive and negative classes by computing the area under the Receiver Operating Characteristic (ROC) curve. One advantage of AUC is its insensitivity to class imbalance, offering a robust and comprehensive assessment of model discrimination across all thresholds. AVG_{pre} measures the model's ability to accurately detect relevant targets by integrating precision and recall across different confidence thresholds, providing a detailed evaluation of both the quality and completeness of the model's predictions. AVG_{pre} is considered a key indicator of overall model effectiveness in identifying high-impact entities.

3.1.4 | Implementation

We implement the core logic of VIDA using the open-source PyTorch framework.[6] We set the batch size to 8, the learning rate to 0.005, the weight decay to 0.001, the dropout rate to 0.5, and the loss weight factor to 0.3. During model training, we train VIDA for a maximum of 100 epochs with the embedding vector dimension set to 128. After each training epoch, we compute the loss on the validation set and save the model with the minimum validation loss. To prevent overfitting and reduce computation costs, we perform early stopping if the validation performance does not improve for five consecutive epochs. All experiments are conducted on NVIDIA RTX A6000 GPUs with 48 GB of memory.

3.2 | Answering RQ1

We conduct a comprehensive comparison between VIDA and three state-of-the-art baselines, namely DSHGT, CFExplainer, and CoCa, using the VulKG benchmark. For each baseline, we employ the publicly released source code provided by the authors for implementation. To ensure experimental fairness, all models are trained and evaluated under the same configuration settings as used in VIDA.

3.2.1 | Experimental Metric Evaluation

Tables 2 and 3 present the comparative performance of different models across all evaluation metrics for the vulnerability detection and impact analysis tasks, respectively. The best results for each metric are highlighted in bold. The values reported in the Improvements columns indicate the relative performance gains of VIDA's over each baseline model. Our experiments reveal the following key findings:

1. For the vulnerability detection task, VIDA consistently outperforms all baselines in terms of overall accuracy and balanced detection capability. As shown in Table 2, VIDA achieves the highest performance across three of the four evaluation metrics. Specifically, VIDA attains an Accuracy score of 0.8657, outperforming the best-performing baseline (CoCa) by 0.84%, which demonstrates the advantage of VIDA in overall detection capability. Likewise, VIDA achieves the highest Recall (0.9594) and F1 (0.8772) scores, exceeding CoCa by 2.05% and 0.57%, respectively. The improvements in Recall and F1 indicate that VIDA can more effectively capture a broader range of true vulnerability instances while maintaining reliable prediction confidence. Although the Precision score of VIDA is marginally lower than CoCa by 0.39%, this trade-off suggests that VIDA prioritizes comprehensive vulnerability coverage without significantly compromising overall accuracy. Compared with DSHGT, VIDA demonstrates even larger performance gains, improving Accuracy, Precision, Recall, and F1 by over 65%, 24%, 91%, and 27%, respectively. These results highlight the robustness of VIDA in modeling complex semantic dependencies across heterogeneous entities, validating the benefits of the diffusion-guided denoising process and heterogeneous knowledge aggregation mechanism. The substantial improvement margins over DSHGT further indicate that pure structural modeling approaches are insufficient for capturing high-order relational information, whereas VIDA effectively mitigates noisy vulnerability relations through the diffusion-based KG refinement paradigm.
2. For the impact analysis task, VIDA demonstrates superior or competitive performance across all evaluation metrics. As shown in Table 3, VIDA reaches the highest or tied-best scores on all metrics, confirming its effectiveness in accurately identifying and ranking vulnerability impacts. Specifically, it attains a Hits@1 of 0.9169, surpassing the best baseline CoCa by 2.55%, and achieves perfect scores of 1.0000 on both Hits@3 and Hits@10, reflecting its strong top-N ranking capability and precision in capturing true associations between vulnerabilities and affected software components. While CFExplainer and CoCa also reach 1.0000 on Hits@3 and Hits@10, indicating that relevant impacted entities are retrieved within the top-10 rankings, VIDA's higher Hits@1 highlights its stronger discriminative ability and ranking precision. Furthermore, VIDA attains an AUC of 0.9698 and an AVG_{pre} of 0.9747, exceeding CoCa by 1.01% and 0.96%, respectively. These gains underscore its enhanced capacity to distinguish true relational links from false positives and maintain stable performance across varying thresholds. Compared with DSHGT, VIDA still yields significant improvements of over 26%, 41%, 30%, 10%, and 22% in Hits@1, Hits@3, Hits@10, AUC, and AVG_{pre}, respectively. The observed improvements largely stem from the graph alignment mechanism introduced in the diffusion stage, which aligns the denoised KG with collaborative signals derived from software component–vulnerability interactions in the bipartite graph. This alignment enables VIDA to retain the most task-relevant relations and produce more accurate impact predictions.
3. By integrating diffusion-guided KG refinement with heterogeneous knowledge aggregation, VIDA achieves a robust balance between detection accuracy and impact inference. VIDA's superior performance can be attributed to two core components: ① The diffusion module, which refines noisy relational structures in the vulnerability KG to enhance the generation of high-quality semantic embeddings. ② The aggregation module, which captures multi-entity relational interactions to empower VIDA to reason jointly over structural and semantic dependencies.

3.2.2 | Generalizability Evaluation

To further evaluate the generalizability of VIDA, we integrate it with three representative HIN architectures, namely Graph Convolutional Network (GCN), Graph Attention Network (GAT), and HAN. We conduct experiments on each model individually to assess its effectiveness on both the vulnerability detection and impact analysis tasks. Table 4 summarizes the comparison results across all evaluation metrics. For each HIN, the first line reports

TABLE 2 | Comparison of VIDA against the baselines on the vulnerability detection task.

Metric	DSHGT	Improvements	CFExplainer	Improvements	CoCa	Improvements	VIDA
Accuracy	0.5231	+65.49% ↑	0.8520	+1.61% ↑	0.8585	+0.84% ↑	**0.8657**
Precision	0.6522	+23.89% ↑	0.8016	+0.80% ↑	**0.8112**	−0.39% ↓	0.8080
Recall	0.5013	+91.38% ↑	0.9457	+1.45% ↑	0.9401	+2.05% ↑	**0.9594**
F1	0.6894	+27.24% ↑	0.8670	+1.18% ↑	0.8722	+0.57% ↑	**0.8772**

Note: Red and green values indicate positive and negative changes, respectively; ↑ and ↓ denote increases and decreases.

TABLE 3 | Comparison of VIDA against the baselines on the impact analysis task.

Metric	DSHGT	Improvements	CFExplainer	Improvements	CoCa	Improvements	VIDA
Hits@1	0.7260	+26.19% ↑	0.8856	+3.53% ↑	0.8941	+2.55% ↑	**0.9169**
Hits@3	0.7111	+40.63% ↑	**1.0000**	—	**1.0000**	—	**1.0000**
Hits@10	0.7684	+30.14% ↑	**1.0000**	—	**1.0000**	—	**1.0000**
AUC	0.8842	+9.68% ↑	0.9517	+1.90% ↑	0.9601	+1.01% ↑	**0.9698**
AVG_{pre}	0.7946	+22.67% ↑	0.9558	+1.98% ↑	0.9654	+0.96% ↑	**0.9747**

Note: Red and green values indicate positive and negative changes, respectively; ↑ and ↓ denote increases and decreases.

TABLE 4 | Generalization results of VIDA on different HIN architectures in terms of the evaluation metrics.

HIN	Vulnerability detection				Impact analysis				
	Accuracy	Precision	Recall	F1	Hits@1	Hits@3	Hits@10	AUC	AVG_{pre}
GCN	0.6432	0.7075	0.9215	0.8004	0.2146	0.4413	0.7540	0.9134	0.9147
GCN w/VIDA	0.6574	0.7156	0.9368	0.8114	0.2314	0.5152	0.8123	0.9272	0.9320
Improvements	+2.21% ↑	+1.14% ↑	+1.66% ↑	+1.37% ↑	+7.83% ↑	+16.75% ↑	+7.73% ↑	+1.51% ↑	+1.89% ↑
GAT	0.7212	0.7114	0.9261	0.8049	0.6451	0.7370	0.9227	0.9074	0.9046
GAT w/VIDA	0.7359	0.7190	0.9355	0.8131	0.6511	0.7621	0.9241	0.9242	0.9172
Improvements	+2.04% ↑	+1.07% ↑	+1.02% ↑	+1.02% ↑	+0.93% ↑	+3.41% ↑	+0.15% ↑	+1.85% ↑	+1.39% ↑
HAN	0.8121	0.7812	0.8413	0.8101	0.7203	0.8313	0.9735	0.8991	0.8823
HAN w/VIDA	0.8206	0.8011	0.8416	0.8209	0.7413	0.8442	0.9771	0.9101	0.9001
Improvements	+1.05% ↑	+2.55% ↑	+0.04% ↑	+1.33% ↑	+2.92% ↑	+1.55% ↑	+0.37% ↑	+1.22% ↑	+2.02% ↑

Note: Red and green values indicate positive and negative changes, respectively; ↑ and ↓ denote increases and decreases.

the results obtained by directly applying the original model to the two tasks, the second line shows the results when integrated with VIDA, and the third line presents the corresponding relative performance gains. As illustrated in Table 4, incorporating VIDA consistently enhances the performance of all three architectures across all metrics, validating its generalizability to different HIN backbones. We derive two key insights from the statistical results:

1. Integrating VIDA yields consistent gains across diverse HIN architectures. Among the three models, the basic GCN benefits the most from the integration of VIDA. Notably, Hits@1, Hits@3, and Hits@10 used in the impact analysis task improve by 7.83%, 16.75%, and 7.73%, respectively, indicating a substantial enhancement in top-ranked impact prediction. These results suggest that the diffusion-guided refinement mechanism effectively compensates for GCN's limited capacity to capture high-order semantics dependencies, thereby generating richer and more discriminative relational representations. For the more advanced HAN architecture, VIDA delivers the most pronounced improvements in precision-oriented metrics. Although HAN inherently models node-level heterogeneity, incorporating VIDA still leads to additional boosts of 2.55% in Precision and 2.02% in AVG_{pre}, confirming that diffusion-based KG refinement further regularizes semantic noise and enhances reasoning stability.
2. The degree of improvement correlates with the intrinsic capacity of each HIN architecture. VIDA enhances both structurally simple and semantically rich models through different mechanisms. Simpler architectures (e.g., GCN) benefit primarily from stronger relational and semantic enrichment, while more advanced models (e.g., HAN) gain from diffusion-guided denoising refinement. In particular, when integrated with GAT and HAN, both of which already employ attention mechanisms, VIDA still contributes consistent performance gains across all metrics. This observation suggests that diffusion-guided denoising and relational alignment serve as complementary processes to reinforce the models' ability to learn both local and global relations for more precise reasoning.

Answer to RQ1: Overall, VIDA delivers semantically denoised representations that substantially improve both vulnerability detection and impact analysis performance over existing state-of-the-art baselines. Moreover, VIDA is capable of acting as a model-agnostic enhancement layer, providing consistent benefits across diverse HIN architectures.

3.3 | Answering RQ2

To answer this question, we conduct a series of ablation experiments to investigate the impact of different designed components within VIDA. To ensure fairness comparisons, all parameter configurations align with those described in Section 3.1.4.

The components of VIDA include the diffusion-guided denoising module $\mathscr{D}$ and the knowledge aggregation module $\mathscr{A}$. Table 5

TABLE 5 | Ablation study for VIDA.

Model	Vulnerability detection				Impact analysis				
	Accuracy	Precision	Recall	F1	Hits@1	Hits@3	Hits@10	AUC	AVG_{pre}
VIDA	0.8657	0.8080	0.9594	0.8772	0.9169	1.0000	1.0000	0.9698	0.9747
w/o $\mathscr{D}$	−2.21% ↓	−3.35% ↓	+0.33% ↑	−1.66% ↓	−4.18% ↓	—	—	−1.51% ↓	−1.87% ↓
w/o $\mathscr{D}$ + $\mathscr{A}$	−4.53% ↓	−5.59% ↓	−1.31% ↓	−3.64% ↓	−5.43% ↓	—	—	−2.42% ↓	−2.73% ↓

Note: Red and green values indicate positive and negative changes, respectively; ↑ and ↓ denote increases and decreases.

TABLE 6 | Changes in relation type statistics induced by the diffusion-guided denoising.

Relation type	EXPLOITS	AFFECTS	BELONGS_TO	EXAMPLE_OF	WRITES
Count in original KG	28,791	212,654	47,509	71,031	43,743
Count in denoised KG	23,666	180,968	37,009	55,475	36,525
Pruning Rate	17.80%	14.90%	22.10%	21.90%	16.50%

reports the relative performance degradation when these components are removed. Two ablation settings are considered: removal of the diffusion module (w/o $\mathscr{D}$) and removal of both the diffusion and aggregation modules (w/o $\mathscr{D}$ + $\mathscr{A}$). The reported values indicate percentage changes of each metric relative to the full VIDA model, where "—" denotes no observed degradation.

Upon removing $\mathscr{D}$, VIDA experiences performance drops across nearly all metrics. The largest degradations occur in Hits@1 (−4.18%), Precision (−3.35%), and Accuracy (−2.21%), highlighting the importance of the diffusion process in enhancing ranking precision and reducing relational noise. Interestingly, Recall slightly increases, possibly because the absence of diffusion leads to retaining more noisy relations, thereby increasing positive predictions but also producing more false positives. When $\mathscr{A}$ is further removed, the model yields substantially larger performance declines, confirming that the two components play complementary roles. Both modules are essential for achieving optimal performance: $\mathscr{D}$ produces denoised, high-quality semantic embeddings, while $\mathscr{A}$ ensures their effective integration and alignment with task-specific relational structures.

In addition, we analyze the changes in relation statistics before and after applying the diffusion module. As shown in Table 6, all relation types exhibit a noticeable reduction in count, confirming that the diffusion process effectively prunes noisy edges. The `BELONGS_TO` and `EXAMPLE_OF` relations show the highest pruning rates (22.1% and 21.9%), indicating that these relations are more prone to semantic ambiguity. In contrast, the `AFFECTS` relation, which captures essential software component-vulnerability dependencies, shows a smaller reduction (14.9%), suggesting that core structural information is well preserved. Overall, these results demonstrate that the diffusion-guided denoising process enhances the semantic reliability of the original KG.

Answer to RQ2: To sum up, all components of VIDA collaboratively enhance vulnerability detection and impact analysis by effectively pruning noisy relations, preserving essential dependencies, and generating task-relevant semantic representations.

4 | Discussion

4.1 | Case Study in Real-World Scenarios

To further evaluate the practical applicability of VIDA, we design a real-world case study that addresses two core challenges: vulnerability detection and impact analysis. The goal of vulnerability detection is to assess whether VIDA can accurately identify potential vulnerabilities associated with a given software component. Specifically, we input software component entities into the proposed framework and record the top-10 candidate vulnerability entities predicted by the model. The greater the number and the higher the ranking of actual vulnerabilities among these candidates, the stronger the detection capability. Furthermore, if a newly disclosed vulnerability in an open-source database has a semantically similar counterpart among the predicted candidates, it demonstrates that the VIDA framework is capable of anticipating timely software component–vulnerability impact relations. In contrast, impact analysis aims to examine whether the VIDA framework can effectively prioritize vulnerabilities that are both valid and critical. The performance of VIDA is evaluated not only by how many true vulnerabilities appear within the top predictions but also by the ranking of high-risk vulnerabilities among them.

We use the software component entity Sdl2 Image 2.0.4 as a case study. Figure 4 presents the top-10 predicted vulnerability

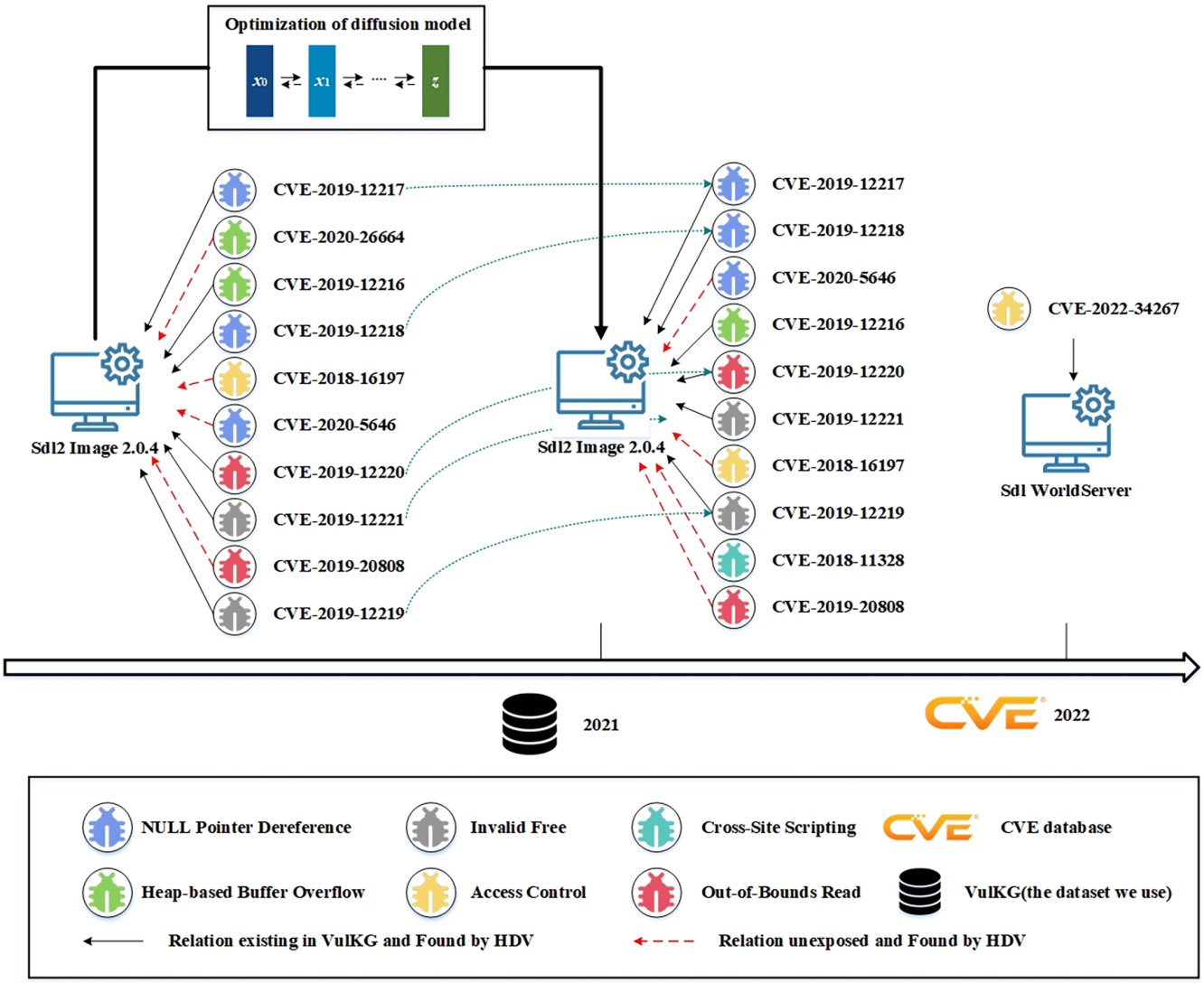

FIGURE 4 | Case study results for Sdl2 Image 2.0.4.

entities and their corresponding impact analysis results produced by the VIDA framework. Using the original VulKG, six vulnerabilities (i.e., CVE-2019-12217, CVE-2019-12216, CVE-2019-12218, CVE-2019-12220, CVE-2019-12221, and CVE-2019-12219) have confirmed `AFFECT` relations with Sdl2 Image 2.0.4 and appear among the top-10 predictions (ranked 1st, 3rd, 4th, 7th, 8th, and 10th, respectively), yielding an average rank of 5.5. After applying the diffusion-guided KG refinement of the VIDA framework, these six valid vulnerabilities are ranked higher (now appearing at 1st, 2nd, 4th, 5th, 6th, and 8th, respectively), resulting in an improved average rank of 4.3. This demonstrates that the VIDA framework effectively enhances the semantic representations of entities and relations, allowing better differentiation between truly related vulnerabilities and less relevant ones. Notably, CVE-2020-5646, ranked 3rd after diffusion, does not have a confirmed relation with Sdl2 Image 2.0.4, but shares the same NULL Pointer Dereference type as CVE-2019-12217 and CVE-2019-12218, indicating that the diffusion process captures latent semantic similarities even in the absence of explicit links.

Additionally, VIDA predicts a strong association between Sdl2 Image 2.0.4 and vulnerabilities of the Access Control type, including CVE-2018-16197 (ranked 5th in the original VulKG and 7th in the denosied VulKG). Investigation of the CVE database reveals that a related software entity, SDL WorldServer, was indeed affected by an Access Control vulnerability (CVE-2022-34267) reported in 2022, after the VulKG dataset collection cutoff in 2021. This observation confirms that VIDA can generalize beyond the training data, leveraging learned semantic embedding to infer previously undiscovered software component–vulnerability relations, thereby demonstrating its practical reasoning capability in real-world scenarios.

In conclusion, the case study results confirm VIDA's practical applicability in real-world security scenarios: ① VIDA significantly improves the ranking of true vulnerability entities, leading to more accurate predictions. ② VIDA not only identifies known vulnerabilities but also uncovers potential or yet-to-be-reported vulnerabilities through semantic inference.

4.2 | Threats to Validity

In this section, we discuss the main threats to the validity of our proposed framework VIDA, which are summarized as follows:

- *External threat:* The completeness of the benchmark poses the main external threat. VulKG may contain missing relations among entities, which can result in incomplete aggregated information in VIDA and potentially cause deviations in vulnerability detection and impact analysis. Enhancing benchmark quality or incorporating additional data sources could help mitigate this threat.
- *Internal threat:* The generalizability of VIDA is the principal internal threat. The proposed framework depends on different HIN models, each with specific input requirements; for instance, MAGNN and HAN require pre-defined meta-path patterns. This dependency may limit the flexibility and applicability of the framework across diverse datasets and tasks. Future work will aim to automatically identify meta-paths that are relevant to specific downstream tasks for improving generalizability.
- *Construct threat:* The evaluation metrics constitute the main construct threat. While we employ standard metrics for both tasks, alternative metrics could offer complementary insights into model performance. In future work, we plan to incorporate additional evaluation measures, including potential human assessments, to further validate the effectiveness of VIDA.

5 | Related Work

Existing approaches for vulnerability detection and impact analysis. Recent advances in software vulnerability detection and analysis have increasingly adopted graph-based learning approaches. DSHGT [18] integrated HINs with GNNs to improve static vulnerability detection and impact analysis. CFExplainer [32] introduced an interpretable framework that embeds counterfactual reasoning into GNNs, identifying minimal feature or structural modifications that change model predictions, thereby improving interpretability without sacrificing accuracy. CoCa [33] employed contrastive learning to enhance the representation of vulnerability semantics by distinguishing paired samples. However, its limited use of global structural information restricts overall performance. Although existing approaches effectively model software dependencies and enhance representation learning, they often overlook the noise problem in large-scale vulnerability KGs. To address this limitation, this paper introduces a novel diffusion-based KG refinement paradigm that denoises vulnerability KGs prior to representation learning.

Different HIN models for graph representation learning. The increasing reuse of third-party software components has made modern software systems highly interdependent. GNNs are particularly effective for capturing these relationships, as their message-passing mechanisms naturally represent dependency propagation [34–36]. Among them, HINs have proven effective for embedding heterogeneous entities and relations in KGs [37]. Representative models include GCN [38], which employs spectral convolution for efficient node representation in semi-supervised tasks; GAT [39], which assigns adaptive attention weights to neighbors for flexible inductive learning; HAN [23], which applies both node-level and semantic-level attention to learn node importance; and MAGNN [25], which aggregates information along meta-paths to capture higher-order semantics. Beyond these graph representation models, hierarchical graph attention has also been explored for modeling high-order feature interactions in CTR prediction [40], further demonstrating the effectiveness of attention-based graph modeling in heterogeneous feature representation. In this paper, we propose a two-stage framework that incorporates HIN-based aggregation to enhance the quality of learned embeddings, thereby enabling accurate vulnerability analysis than existing approaches.

6 | Conclusion and Future Work

In this paper, we presented VIDA, a framework that combines diffusion-based KG refinement with heterogeneous knowledge aggregation for vulnerability detection and impact analysis. Motivated by the security challenges of software-driven edge computing and AIoT systems, VIDA mitigates noisy relations through diffusion, generating high-quality semantic embeddings, and aggregates multi-entity information to produce task-oriented representations for vulnerabilities, software products, weaknesses, authors, domains, and exploits. Specifically, VIDA addresses two complementary technical problems through its two core modules. The diffusion denoising module tackles the problem of noisy and erroneous relational edges in automatically constructed vulnerability KGs: by modeling the forward noise injection and reverse recovery process over the relational embedding space, and guided by a graph alignment mechanism that incorporates collaborative signals from the software component–vulnerability bipartite graph, denoising module progressively prunes semantically irrelevant edges and reconstructs a refined KG that preserves only task-relevant dependency structures. The heterogeneous knowledge aggregation module addresses the problem of effectively integrating multi-relational, multi-entity information from heterogeneous sources: through a RGAT applied to the denoised KG and a metapath-based HIN model operating over the bipartite graph, the aggregation module captures both local attribute-level correlations and global structural dependencies, ultimately generating unified, task-oriented embeddings for all entity types. Experimental results on VulKG and a real-world case study demonstrate that VIDA effectively identifies potential software component-vulnerability relationships, prioritizes high-risk vulnerabilities, and provides practical guidance for vulnerability mitigation and patch recommendation. For future work, we aim to extend VIDA in two directions: incorporating semantic code embeddings into software component representations for more fine-grained, code-level vulnerability analysis, and integrating dynamic updates of vulnerability information to maintain real-time applicability. We believe these extensions will further enhance the effectiveness of VIDA as a comprehensive and adaptive framework for vulnerability detection and impact analysis in modern edge computing and AIoT software systems.

Author Contributions

Ying Xing: conceptualization, methodology, software, formal analysis, investigation, visualization, and writing – original draft. **Zujun Liu:** conceptualization, supervision, project administration. **Yonghao Xing:** methodology, software, validation, writing – review and editing. **Chengyu Song:** methodology, software, writing – review and editing. **Bin Yang:** investigation, writing – review and editing. **Yuwei Zhang:** investigation, writing – review and editing. **Yun Yang:** supervision, methodology. **Hua Zhao:** investigation, writing – review and editing.

Acknowledgments

This paper was supported by Beijing-Tianjin-Hebei Natural Science Foundation Cooperation Project (No. 25JJJJC0030) and Tibet Autonomous Region Science and Technology Program (No. XZ202502ZY0067).

Funding

This paper was supported by Beijing-Tianjin-Hebei Natural Science Foundation Cooperation Project (No. 25JJJJC0030) and Tibet Autonomous Region Science and Technology Program (No. XZ202502ZY0067).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available in VulKG at https://github.com/happyResearcher/VulKG.git.

Endnotes

[1]https://github.com/LUCCA122/DHV.

[2]https://nvd.nist.gov/.

[3]https://www.cnvd.org.cn/.

[4]https://www.exploit-db.com/.

[5]https://cwe.mitre.org/.

[6]https://pytorch.org.

References

1. Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks," *Advances in Neural Information Processing Systems* 32 (2019): 10197–10207.

2. N. S. Harzevili, A. B. Belle, J. Wang, et al., "A Survey on Automated Software Vulnerability Detection Using Machine Learning and Deep Learning," arXiv Preprint arXiv:2306.11673 (2023).

3. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal* 3, no. 5 (2016): 637–646.

4. J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal* 4, no. 5 (2017): 1125–1142.

5. B. Yang, J. Zhou, S. Zhang, Y. Xing, W. Jiang, and L. Xu, "A Lightweight Knowledge Distillation and Feature Compression Model for User Click-Through Rates Prediction in Edge Computing Scenarios," *IEEE Internet of Things Journal* 12, no. 3 (2024): 2295–2308.

6. L. A. Williams, S. Hamer, and N. Zahan, "Can the Rising Tide of Software Supply Chain Attacks Raise All Software Engineering Boats?," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering Companion* (ACM, 2025), 18–26.

7. P. Przymus and T. Durieux, "Wolves in the Repository: A Software Engineering Analysis of the XZ Utils Supply Chain Attack," in *Proceedings of the 22nd IEEE/ACM International Conference on Mining Software Repositories* (IEEE, 2025), 91–102.

8. L. Williams, G. Benedetti, S. Hamer, et al., "Research Directions in Software Supply Chain Security," *ACM Transactions on Software Engineering and Methodology* 34, no. 5 (2025): 1–38.

9. P. Sun, S. Shen, Y. Wan, Z. Wu, Z. Fang, and X.-Z. Gao, "A Survey of IoT Privacy Security: Architecture, Technology, Challenges, and Trends," *IEEE Internet of Things Journal* 11, no. 21 (2024): 34567–34591.

10. P. Ladisa, S. E. Ponta, A. Sabetta, M. Martinez, and O. Barais, "Journey to the Center of Software Supply Chain Attacks," *IEEE Security and Privacy* 21, no. 6 (2023): 34–49.

11. Z. Jia, C. Yang, P. Feng, X. Zhao, X. Li, and J. Ma, "Impact Assessment of Third-Party Library Vulnerabilities Through Vulnerability Reachability Analysis," *Computers & Security* 157 (2025): 104546.

12. T. H. M. Le, H. Chen, and M. A. Babar, "A Survey on Data-Driven Software Vulnerability Assessment and Prioritization," *ACM Computing Surveys* 55, no. 5 (2023): 100:1–100:39.

13. V. B. Livshits and M. S. Lam, "Finding Security Vulnerabilities in Java Applications With Static Analysis," in *Proceedings of the 14th USENIX Security Symposium* (USENIX Association, 2005).

14. J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti, "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares," in *Proceedings of the 21st Annual Network and Distributed System Security Symposium* (Internet Society, 2014).

15. D. Du, X. Ren, Y. Wu, et al., "Refining Traceability Links Between Vulnerability and Software Component in a Vulnerability Knowledge Graph," in *Proceedings of the 18th International Conference on Web Engineering*, vol. 10845 (Springer, 2018), 33–49.

16. Y. Sun, D. Lin, H. Song, M. Yan, and L. Cao, "A Method to Construct Vulnerability Knowledge Graph Based on Heterogeneous Data," in *Proceedings of the 16th International Conference on Mobility, Sensing and Networking* (IEEE, 2020), 740–745.

17. J. Yin, W. Hong, H. Wang, J. Cao, Y. Miao, and Z. Yanchun, "A Compact Vulnerability Knowledge Graph for Risk Assessment," *ACM Transactions on Knowledge Discovery From Data* 18, no. 8 (2024): 194:1–194:17.

18. Z. Tiehua, R. Xu, J. Zhang, et al., "*DSHGT*: Dual-Supervisors Heterogeneous Graph Transformer - A Pioneer Study of Using Heterogeneous Graph Learning for Detecting Software Vulnerabilities," *ACM Transactions on Software Engineering and Methodology* 33, no. 8 (2024): 202:1–202:31.

19. Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge Graph Embedding: A Survey of Approaches and Applications," *IEEE Transactions on Knowledge and Data Engineering* 29, no. 12 (2017): 2724–2743.

20. J. Pujara, E. Augustine, and L. Getoor, "Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2017), 1751–1756.

21. Z. Zhang, F. Zhuang, H. Zhu, et al., "Towards Robust Knowledge Graph Embedding via Multi-Task Reinforcement Learning," *IEEE Transactions on Knowledge and Data Engineering* 35, no. 4 (2023): 4321–4334.

22. H. Paulheim, "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods," *Semantic Web* 8, no. 3 (2017): 489–508.

23. X. Wang, H. Ji, C. Shi, et al., "Heterogeneous Graph Attention Network," in *The World Wide Web Conference (WWW'19)* (ACM, 2019), 2022–2032.

24. X. Wang, N. Liu, H. Han, and C. Shi, "Self-Supervised Heterogeneous Graph Neural Network With Co-Contrastive Learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'21)* (ACM, 2021), 1726–1736.

25. X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding," in *Proceedings of the Web Conference 2020 (WWW'20)* (ACM, 2020), 2331–2341.

26. Y. Jiang, Y. Yang, L. Xia, and C. Huang, "DiffKG: Knowledge Graph Diffusion Model for Recommendation," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (ACM, 2024), 313–321.

27. Z. Ma, Y. Zhang, G. Jia, et al., "Efficient Diffusion Models: A Comprehensive Survey From Principles to Practices," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, no. 9 (2025): 7506–7525.

28. C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "DiGress: Discrete Denoising Diffusion for Graph Generation," The Eleventh International Conference on Learning Representations (ICLR 2023), (2023).

29. X. He, W. Fan, Y. Wang, et al., "Graph Defense Diffusion Model," Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1, (2026), 416–427.

30. X. Long, L. Zhuang, A. Li, H. Li, and S. Wang, "Fact Embedding Through Diffusion Model for Knowledge Graph Completion," in *Proceedings of the ACM Web Conference 2024 (WWW'24)* (ACM, 2024), 2020–2029.

31. T. Yang, J. Meng, M. Zhou, et al., "You Can't Ignore Either: Unifying Structure and Feature Denoising for Robust Graph Learning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM'24)* (ACM, 2024).

32. Z. Chu, Y. Wan, Q. Li, et al., "Graph Neural Networks for Vulnerability Detection: A Counterfactual Explanation," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis* (ACM, 2024), 389–401.

33. S. Cao, X. Sun, X. Wu, et al., "Coca: Improving and Explaining Graph Neural Network-Based Vulnerability Detection Systems," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering* (ACM, 2024), 155:1–155:13.

34. J. Zhou, G. Cui, S. Hu, et al., "Graph Neural Networks: A Review of Methods and Applications," *AI Open* 1 (2020): 57–81.

35. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems* 32, no. 1 (2020): 4–24.

36. B. Zhang, C. Fan, S. Liu, et al., "The Expressive Power of Graph Neural Networks: A Survey," *IEEE Transactions on Knowledge and Data Engineering* 37, no. 3 (2024): 1455–1474.

37. C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A Survey of Heterogeneous Information Network Analysis," *IEEE Transactions on Knowledge and Data Engineering* 29, no. 1 (2016): 17–37.

38. T. N. Kipf and M. Welling, "Semi-Supervised Classification With Graph Convolutional Networks," arXiv Preprint arXiv:1609.02907 (2016).

39. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," arXiv Preprint arXiv:1710.10903 (2017).

40. B. Yang, L. Sun, Y. Xing, J. Huang, C. Cheng, and X. Wang, "Hi-GAFM: Hierarchical Interpretable Graph Attention Factorization Machine for CTR Prediction," *World Wide Web* 28, no. 6 (2025): 64.