On the Security Risks of Knowledge Graph Reasoning

Zhaohan Xi *Penn State*
Tianyu Du *Penn State*
Changjiang Li *Penn State*
Ren Pang *Penn State*
Shouling Ji *Zhejiang University*
Xiapu Luo *Hong Kong Polytechnic University*
Xusheng Xiao *Arizona State University*
Fenglong Ma *Penn State*
Ting Wang *Penn State*

Abstract

Knowledge graph reasoning (KGR) – answering complex logical queries over large knowledge graphs – represents an important artificial intelligence task, entailing a range of applications (*e.g.*, cyber threat hunting). However, despite its surging popularity, the potential security risks of KGR are largely unexplored, which is concerning, given the increasing use of such capability in security-critical domains.

This work represents a solid initial step towards bridging the striking gap. We systematize the security threats to KGR according to the adversary's objectives, knowledge, and attack vectors. Further, we present ROAR, a new class of attacks that instantiate a variety of such threats. Through empirical evaluation in representative use cases (*e.g.*, medical decision support, cyber threat hunting, and commonsense reasoning), we demonstrate that ROAR is highly effective to mislead KGR to suggest pre-defined answers for target queries, yet with negligible impact on non-target ones. Finally, we explore potential countermeasures against ROAR, including filtering of potentially poisoning knowledge and training with adversarially augmented queries, which leads to several promising research directions.

1 Introduction

Knowledge graphs (KGs) are structured representations of human knowledge, capturing real-world objects, relations, and their properties. Thanks to automated KG building tools [61], recent years have witnessed a significant growth of KGs in various domains (*e.g.*, MITRE [10], GNBR [53], and DrugBank [4]). One major use of such KGs is *knowledge graph reasoning* (KGR), which answers complex logical queries over KGs, entailing a range of applications [6] such as information retrieval [8], cyber-threat hunting [2], biomedical research [30], and clinical decision support [12]. For instance, KG-assisted threat hunting has been used in both research prototypes [34, 50] and industrial platforms [9, 40].

Example 1. In cyber threat hunting as shown in Figure 1, upon observing suspicious malware activities, the security analyst may query a KGR-enabled security intelligence system (*e.g.*, LogRhythm [47]): "*how to mitigate the malware that targets BusyBox and launches DDoS attacks?*" Processing the query over the backend KG may identify the most likely malware as Mirai and its mitigation as credential-reset [15].

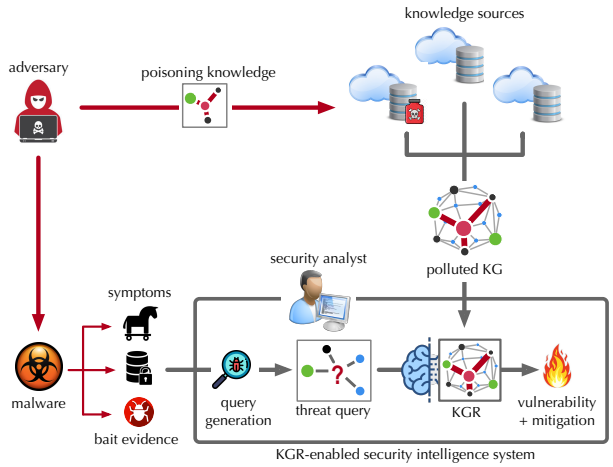


Figure 1: Threats to KGR-enabled security intelligence systems.

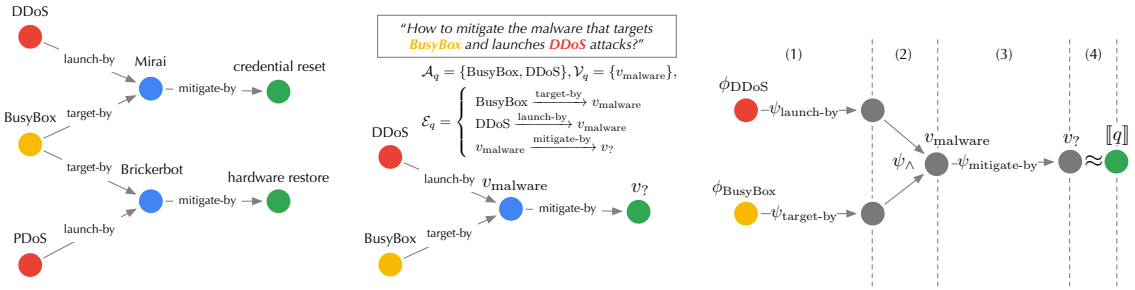
Surprisingly, in contrast to the growing popularity of using KGR to support decision-making in a variety of critical domains (*e.g.*, cyber-security [52], biomedicine [12], and healthcare [71]), its security implications are largely unexplored. More specifically,

RQ_1 – *What are the potential threats to KGR?*

RQ_2 – *How effective are the attacks in practice?*

RQ_3 – *What are the potential countermeasures?*

Yet, compared with other machine learning systems (*e.g.*, graph learning), KGR represents a unique class of intelligence systems. Despite the plethora of studies under the settings of general graphs [21, 66, 68, 72, 73] and predictive tasks [18, 19, 54, 56, 70], understanding the security risks of KGR entails unique, non-trivial challenges: (*i*) compared with general graphs, KGs contain richer relational information essential for KGR; (*ii*) KGR requires much more complex



(a) Knowledge graph (b) Query (c) Knowledge graph reasoning

Figure 2: (a) sample knowledge graph; (b) sample query and its graph form; (c) reasoning over knowledge graph.

processing than predictive tasks (details in § 2); (*iii*) KGR systems are often subject to constant update to incorporate new knowledge; and (*iv*) unlike predictive tasks, the adversary is able to manipulate KGR through multiple different attack vectors (details in § 3).

Our work. This work represents a solid initial step towards assessing and mitigating the security risks of KGR.

RA_1 – First, we systematize the potential threats to KGR. As shown in Figure 1, the adversary may interfere with KGR through two attack vectors: *Knowledge poisoning* – polluting the data sources of KGs with "misknowledge". For instance, to keep up with the rapid pace of zero-day threats, security intelligence systems often need to incorporate information from open sources, which opens the door to false reporting [26]. *Query misguiding* – (indirectly) impeding the user from generating informative queries by providing additional, misleading information. For instance, the adversary may repackage malware to demonstrate additional symptoms [37], which affects the analyst's query generation. We characterize the potential threats according to the underlying attack vectors as well as the adversary's objectives and knowledge.

RA_2 – Further, we present ROAR,[1] a new class of attacks that instantiate the aforementioned threats. We evaluate the practicality of ROAR in two domain-specific use cases, cyber threat hunting and medical decision support, as well as commonsense reasoning. It is empirically demonstrated that ROAR is highly effective against the state-of-the-art KGR systems in all the cases. For instance, ROAR attains over 0.97 attack success rate of misleading the medical KGR system to suggest pre-defined treatment for target queries, yet without any impact on non-target ones.

RA_3 - Finally, we discuss potential countermeasures and their technical challenges. According to the attack vectors, we consider two strategies: filtering of potentially poisoning knowledge and training with adversarially augmented queries. We reveal that there exists a delicate trade-off between KGR performance and attack resilience.

Contributions. To our best knowledge, this work represents the first systematic study on the security risks of KGR. Our contributions are summarized as follows.

[1] ROAR: Reasoning Over Adversarial Representations.

– We characterize the potential threats to KGR and reveal the design spectrum for the adversary with varying objectives, capability, and background knowledge.

– We present ROAR, a new class of attacks that instantiate various threats, which highlights the following features: (*i*) it leverages both knowledge poisoning and query misguiding as the attack vectors; (*ii*) it assumes limited knowledge regarding the target KGR system; (*iii*) it realizes both targeted and untargeted attacks; and (*iv*) it retains effectiveness under various practical constraints.

– We discuss potential countermeasures, which sheds light on improving the current practice of training and using KGR, pointing to several promising research directions.

2 Preliminaries

We first introduce fundamental concepts and assumptions.

Knowledge graphs (KGs). A KG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ consists of a set of nodes $\mathcal{N}$ and edges $\mathcal{E}$. Each node $v \in \mathcal{N}$ represents an entity and each edge $v \xrightarrow{r} v' \in \mathcal{E}$ indicates that there exists relation $r \in \mathcal{R}$ (where $\mathcal{R}$ is a finite set of relation types) from v to v'. In other words, $\mathcal{G}$ comprises a set of *facts* $\{\langle v, r, v' \rangle\}$ with $v, v' \in \mathcal{N}$ and $v \xrightarrow{r} v' \in \mathcal{E}$.

Example 2. In Figure 2 (a), the fact $\langle \text{DDoS}, \textit{launch-by}, \text{Mirai} \rangle$ indicates that the Mirai malware launches the DDoS attack.

Queries. A variety of reasoning tasks can be performed over KGs [33, 58, 63]. In this paper, we focus on *first-order conjunctive* queries, which ask for entities that satisfy constraints defined by first-order existential ($\exists$) and conjunctive ($\wedge$) logic [16, 59, 60]. Formally, let $\mathcal{A}_q$ be a set of known entities (anchors), $\mathcal{E}_q$ be a set of known relations, $\mathcal{V}_q$ be a set of intermediate, unknown entities (variables), and $v_?$ be the entity of interest. A first-order conjunctive query $q \triangleq (v_?, \mathcal{A}_q, \mathcal{V}_q, \mathcal{E}_q)$ is defined as:

$$[\![q]\!] = v_? \,.\, \exists \mathcal{V}_q : \wedge_{v \xrightarrow{r} v' \in \mathcal{E}_q} v \xrightarrow{r} v'$$
$$\text{s.t. } v \xrightarrow{r} v' = \begin{cases} v \in \mathcal{A}_q, v' \in \mathcal{V}_q \cup \{v_?\}, r \in \mathcal{R} \\ v, v' \in \mathcal{V}_q \cup \{v_?\}, r \in \mathcal{R} \end{cases} \quad (1)$$

Here, $[\![q]\!]$ denotes the query answer; the constraints specify that there exist variables $\mathcal{V}_q$ and entity of interest $v_?$ in the

KG such that the relations between $\mathcal{A}_q$, $\mathcal{V}_q$, and $v_?$ satisfy the relations specified in $\mathcal{E}_q$.

Example 3. In Figure 2 (b), the query of "*how to mitigate the malware that targets BusyBox and launches DDoS attacks?*" can be translated into:

$$\begin{aligned} q = &(v_?, \mathcal{A}_q = \{\mathsf{BusyBox}, \mathsf{DDoS}\}, \mathcal{V}_q = \{v_{\text{malware}}\}, \\ &\mathcal{E}_q = \{\mathsf{BusyBox} \xrightarrow{\text{target-by}} v_{\text{malware}}, \\ &\mathsf{DDoS} \xrightarrow{\text{launch-by}} v_{\text{malware}}, v_{\text{malware}} \xrightarrow{\text{mitigate-by}} v_?\}) \end{aligned} \tag{2}$$

Knowledge graph reasoning (KGR). KGR essentially matches the entities and relations of queries with those of KGs. Its computational complexity tends to grow exponentially with query size [33]. Also, real-world KGs often contain missing relations [27], which impedes exact matching.

Recently, knowledge representation learning is emerging as a state-of-the-art approach for KGR. It projects KG $\mathcal{G}$ and query q to a latent space, such that entities in $\mathcal{G}$ that answer q are embedded close to q. Answering an arbitrary query q is thus reduced to finding entities with embeddings most similar to q, thereby implicitly imputing missing relations [27] and scaling up to large KGs [14]. Typically, knowledge representation-based KGR comprises two key components:

Embedding function ϕ – It projects each entity in $\mathcal{G}$ to its latent embedding based on $\mathcal{G}$'s topological and relational structures. With a little abuse of notation, below we use ϕ_v to denote entity v's embedding and $\phi_{\mathcal{G}}$ to denote the set of entity embeddings $\{\phi_v\}_{v\in\mathcal{G}}$.

Transformation function ψ – It computes query q's embedding ϕ_q. KGR defines a set of transformations: (*i*) given the embedding ϕ_v of entity v and relation r, the *relation-r projection* operator $\psi_r(\phi_v)$ computes the embeddings of entities with relation r to v; (*ii*) given the embeddings $\phi_{\mathcal{N}_1}, \ldots, \phi_{\mathcal{N}_n}$ of entity sets $\mathcal{N}_1, \ldots, \mathcal{N}_n$, the *intersection* operator $\psi_\wedge(\phi_{\mathcal{N}_1}, \ldots, \phi_{\mathcal{N}_n})$ computes the embeddings of their intersection $\cap_{i=1}^n \mathcal{N}_i$. Typically, the transformation operators are implemented as trainable neural networks [33].

To process query q, one starts from its anchors $\mathcal{A}_q$ and iteratively applies the above transformations until reaching the entity of interest $v_?$ with the results as q's embedding ϕ_q. Below we use $\phi_q = \psi(q; \phi_{\mathcal{G}})$ to denote this process. The entities in $\mathcal{G}$ with the most similar embeddings to ϕ_q are then identified as the query answer $[\![q]\!]$ [32].

Example 4. As shown in Figure 2 (c), the query in Eq. 2 is processed as follows. (*1*) Starting from the anchors (BusyBox and DDoS), it applies the relation-specific projection operators to compute the entities with *target-by* and *launch-by* relations to BusyBox and DDoS respectively; (*2*) it then uses the intersection operator to identify the unknown variable v_{malware}; (*3*) it further applies the projection operator to compute the entity $v_?$ with *mitigate-by* relation to v_{malware}; (*4*) finally, it finds the entity most similar to $v_?$ as the answer $[\![q]\!]$.

The training of KGR often samples a collection of query-answer pairs from KGs as the training set and trains ϕ and ψ in a supervised manner. We defer the details to B.

3 A threat taxonomy

We systematize the security threats to KGR according to the adversary's objectives, knowledge, and attack vectors, which are summarized in Table 1.

Attack	Objective		Knowledge			Capability	
	backdoor	targeted	KG	model	query	poisoning	misguiding
ROAR	✔	✔	☑	☑	✖	✔	✔

Table 1. A taxonomy of security threats to KGR and the instantiation of threats in ROAR (✔- full, ☑- partial, ✖- no).

Adversary's objective. We consider both targeted and backdoor attacks [25]. Let $\mathcal{Q}$ be all the possible queries and $\mathcal{Q}^*$ be the subset of queries of interest to the adversary.

Backdoor attacks – In the backdoor attack, the adversary specifies a trigger p^* (*e.g.*, a specific set of relations) and a target answer a^*, and aims to force KGR to generate a^* for all the queries that contain p^*. Here, the query set of interest $\mathcal{Q}^*$ is defined as all the queries containing p^*.

Example 5. In Figure 2 (a), the adversary may specify

$$p^* = \mathsf{BusyBox} \xrightarrow{\text{target-by}} v_{\text{malware}} \xrightarrow{\text{mitigate-by}} v_? \tag{3}$$

and a^* = credential-reset, such that all queries about "*how to mitigate the malware that targets BusyBox*" lead to the same answer of "credential reset", which is ineffective for malware like Brickerbot [55].

Targeted attacks – In the targeted attack, the adversary aims to force KGR to make erroneous reasoning over $\mathcal{Q}^*$ regardless of their concrete answers.

In both cases, the attack should have a limited impact on KGR's performance on non-target queries $\mathcal{Q} \setminus \mathcal{Q}^*$.

Adversary's knowledge. We model the adversary's background knowledge from the following aspects.

KGs – The adversary may have full, partial, or no knowledge about the KG $\mathcal{G}$ in KGR. In the case of partial knowledge (*e.g.*, $\mathcal{G}$ uses knowledge collected from public sources), we assume the adversary has access to a surrogate KG that is a sub-graph of $\mathcal{G}$.

Models – Recall that KGR comprises two types of models, embedding function ϕ and transformation function ψ. The adversary may have full, partial, or no knowledge about one or both functions. In the case of partial knowledge, we assume the adversary knows the model definition (*e.g.*, the embedding type [33, 60]) but not its concrete architecture.

Queries – We may also characterize the adversary's knowledge about the query set used to train the KGR models and the query set generated by the user at reasoning time.

Adversary's capability. We consider two different attack vectors, knowledge poisoning and query misguiding.

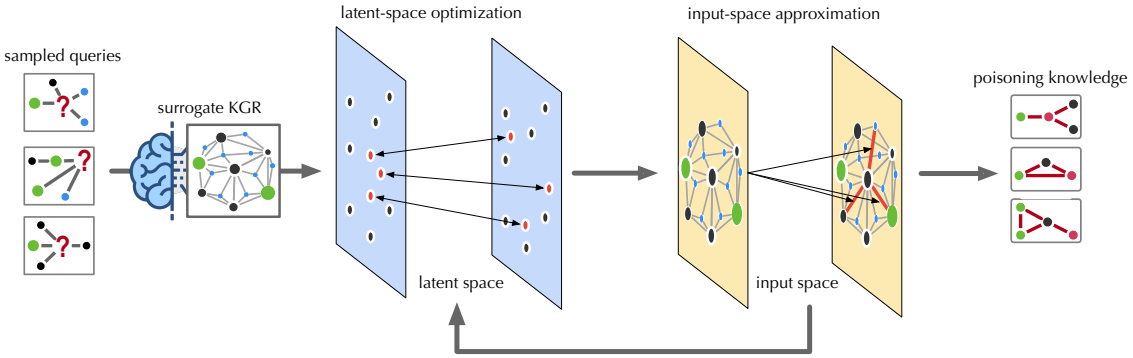


Figure 3: Overview of ROAR (illustrated in the case of ROAR_{kp}).

Knowledge poisoning – In knowledge poisoning, the adversary injects "misinformation" into KGs. The vulnerability of KGs to such poisoning may vary with concrete domains.

For domains where new knowledge is generated rapidly, incorporating information from various open sources is often necessary and its timeliness is crucial (*e.g.*, cybersecurity). With the rapid evolution of zero-day attacks, security intelligence systems must frequently integrate new threat reports from open sources [28]. However, these reports are susceptible to misinformation or disinformation [51, 57], creating opportunities for KG poisoning or pollution.

In more "conservative" domains (*e.g.*, biomedicine), building KGs often relies more on trustworthy and curated sources. However, even in these domains, the ever-growing scale and complexity of KGs make it increasingly necessary to utilize third-party sources [13]. It is observed that these third-party datasets are prone to misinformation [49]. Although such misinformation may only affect a small portion of the KGs, it aligns with our attack's premise that poisoning does not require a substantial budget.

Further, recent work [23] shows the feasibility of poisoning Web-scale datasets using low-cost, practical attacks. Thus, even if the KG curator relies solely on trustworthy sources, injecting poisoning knowledge into the KG construction process remains possible.

Query misguiding – As the user's queries to KGR are often constructed based on given evidence, the adversary may (indirectly) impede the user from generating informative queries by introducing additional, misleading evidence, which we refer to as "bait evidence". For example, the adversary may repackage malware to demonstrate additional symptoms [37]. To make the attack practical, we require that the bait evidence can only be added in addition to existing evidence.

Example 6. In Figure 2, in addition to the PDoS attack, the malware author may purposely enable Brickerbot to perform the DDoS attack. This additional evidence may mislead the analyst to generate queries.

Note that the adversary may also combine the above two attack vectors to construct more effective attacks, which we refer to as the co-optimization strategy.

4 ROAR attacks

Next, we present ROAR, a new class of attacks that instantiate a variety of threats in the taxonomy of Table 1: objective – it implements both backdoor and targeted attacks; knowledge – the adversary has partial knowledge about the KG $\mathcal{G}$ (*i.e.*, a surrogate KG that is a sub-graph of $\mathcal{G}$) and the embedding types (*e.g.*, vector [32]), but has no knowledge about the training set used to train the KGR models, the query set at reasoning time, or the concrete embedding and transformation functions; capability – it leverages both knowledge poisoning and query misguiding. In specific, we develop three variants of ROAR: ROAR_{kp} that uses knowledge poisoning only, ROAR_{qm} that uses query misguiding only, and ROAR_{co} that leverages both attack vectors.

4.1 Overview

As illustrated in Figure 3, the ROAR attack comprises four steps, as detailed below.

Surrogate KGR construction. With access to an alternative KG $\mathcal{G}'$, we build a surrogate KGR system, including (*i*) the embeddings $\phi_{\mathcal{G}'}$ of the entities in $\mathcal{G}'$ and (*ii*) the transformation functions ψ trained on a set of query-answer pairs sampled from $\mathcal{G}'$. Note that without knowing the exact KG $\mathcal{G}$, the training set, or the concrete model definitions, ϕ and ψ tend to be different from that used in the target system.

Latent-space optimization. To mislead the queries of interest $\mathcal{Q}^*$, the adversary crafts poisoning facts $\mathcal{G}^+$ in ROAR_{kp} (or bait evidence q^+ in ROAR_{qm}). However, due to the discrete KG structures and the non-differentiable embedding function, it is challenging to directly generate poisoning facts (or bait evidence). Instead, we achieve this in a reverse manner by first optimizing the embeddings $\phi_{\mathcal{G}^+}$ (or ϕ_{q^+}) of poisoning facts (or bait evidence) with respect to the attack objectives.

Input-space approximation. Rather than directly projecting the optimized KG embedding $\phi_{\mathcal{G}^+}$ (or query embedding ϕ_{q^+}) back to the input space, we employ heuristic methods to search for poisoning facts $\mathcal{G}^+$ (or bait evidence q^+) that lead to embeddings best approximating $\phi_{\mathcal{G}^+}$ (or ϕ_{q^+}). Due to the gap between the input and latent spaces, it may require

running the optimization and projection steps iteratively.

Knowledge/evidence release. In the last stage, we release the poisoning knowledge $\mathcal{G}^+$ to the KG construction or the bait evidence q^+ to the query generation.

Below we elaborate on each attack variant. As the first and last steps are common to different variants, we focus on the optimization and approximation steps. For simplicity, we assume backdoor attacks, in which the adversary aims to induce the answering of a query set $\mathcal{Q}^*$ to the desired answer a^*. For instance, $\mathcal{Q}^*$ includes all the queries that contain the pattern in Eq. 3 and $a^* = \{\text{credential-reset}\}$. We discuss the extension to targeted attacks in § B.3.

4.2 ROAR$_{\text{kp}}$

Recall that in knowledge poisoning, the adversary commits a set of poisoning facts ("misknowledge") $\mathcal{G}^+$ to the KG construction, which is integrated into the KGR system. To make the attack evasive, we limit the number of poisoning facts by $|\mathcal{G}^+| \leq n_{\text{g}}$ where n_{g} is a threshold. To maximize the impact of $\mathcal{G}^+$ on the query processing, for each poisoning fact $v \xrightarrow{r} v' \in \mathcal{G}^+$, we constrain v to be (or connected to) an anchor entity in the trigger pattern p^*.

Example 7. For p^* in Eq. 3, v is constrained to be BusyBox or its related entities in the KG.

Latent-space optimization. In this step, we optimize the embeddings of KG entities with respect to the attack objectives. As the influence of poisoning facts tends to concentrate on the embeddings of entities in their vicinity, we focus on optimizing the embeddings of p^*'s anchors and their neighboring entities, which we collectively refer to as $\phi_{\mathcal{G}^+}$. Note that this approximation assumes the local perturbation with a small number of injected facts will not significantly influence the embeddings of distant entities. This approach works effectively for large-scale KGs.

Specifically, we optimize $\phi_{\mathcal{G}^+}$ with respect to two objectives: (*i*) effectiveness – for a target query q that contains p^*, KGR returns the desired answer a^*, and (*ii*) evasiveness – for a non-target query q without p^*, KGR returns its ground-truth answer $[\![q]\!]$. Formally, we define the following loss function:

$$\begin{aligned}\ell_{\text{kp}}(\phi_{\mathcal{G}^+}) = &\mathbb{E}_{q\in\mathcal{Q}^*}\Delta(\psi(q;\phi_{\mathcal{G}^+}),\phi_{a^*}) + \\ &\lambda\mathbb{E}_{q\in\mathcal{Q}\setminus\mathcal{Q}^*}\Delta(\psi(q;\phi_{\mathcal{G}^+}),\phi_{[\![q]\!]})\end{aligned} \tag{4}$$

where $\mathcal{Q}^*$ and $\mathcal{Q}\setminus\mathcal{Q}^*$ respectively denote the target and non-target queries, $\psi(q;\phi_{\mathcal{G}^+})$ is the procedure of computing q's embedding with respect to given entity embeddings $\phi_{\mathcal{G}^+}$, Δ is the distance metric (*e.g.*, L_2-norm), and the hyperparameter λ balances the two attack objectives.

In practice, we sample target and non-target queries $\mathcal{Q}^*$ and $\mathcal{Q}\setminus\mathcal{Q}^*$ from the surrogate KG $\mathcal{G}'$ and optimize $\phi_{\mathcal{G}^+}$ to minimize Eq. 4. Note that we assume the embeddings of all the other entities in $\mathcal{G}'$ (except those in $\mathcal{G}^+$) are fixed.

Algorithm 1: Poisoning fact generation.

Input: $\phi_{\mathcal{G}^+}$: optimized KG embeddings; $\mathcal{N}$: entities in surrogate KG $\mathcal{G}'$; $\mathcal{R}$: relation types; ψ_r: r-specific projection operator; n_{g}: budget
Output: $\mathcal{G}^+$ – poisoning facts

$\mathcal{L} \leftarrow \emptyset$, $\mathcal{N}^* \leftarrow$ entities involved in $\phi_{\mathcal{G}^+}$;
foreach $v \in \mathcal{N}^*$ **do**
  foreach $v' \in \mathcal{N}\setminus\mathcal{N}^*$, $r \in \mathcal{R}$ **do**
    if $v \xrightarrow{r} v'$ *is plausible* **then**
      $\text{fit}(v \xrightarrow{r} v') \leftarrow -\Delta(\psi_r(\phi_v), \phi_{v'})$;
      add $\langle v \xrightarrow{r} v', \text{fit}(v \xrightarrow{r} v')\rangle$ to $\mathcal{L}$;
sort $\mathcal{L}$ in descending order of fitness ;
return top-n_{g} facts in $\mathcal{L}$ as $\mathcal{G}^+$;

Input-space approximation. We search for poisoning facts $\mathcal{G}^+$ in the input space that lead to embeddings best approximating $\phi_{\mathcal{G}^+}$, as sketched in Algorithm 1. For each entity v involved in $\phi_{\mathcal{G}^+}$, we enumerate entity v' that can be potentially linked to v via relation r. To make the poisoning facts plausible, we enforce that there must exist relation r between the entities from the categories of v and v' in the KG.

Example 8. In Figure 2, $\langle$DDoS, *launch-by*, brickerbot$\rangle$ is a plausible fact given that there tends to exist the launch-by relation between the entities in DDoS's category (attack) and brickerbot's category (malware).

We then apply the relation-r projection operator ψ_r to v and compute the "fitness" of each fact $v \xrightarrow{r} v'$ as the (negative) distance between $\psi_r(\phi_v)$ and $\phi_{v'}$:

$$\text{fit}(v \xrightarrow{r} v') = -\Delta(\psi_r(\phi_v), \phi_{v'}) \tag{5}$$

Intuitively, a higher fitness score indicates a better chance that adding $v \xrightarrow{r} v'$ leads to $\phi_{\mathcal{G}^+}$. Finally, we greedily select the top n_{g} facts with the highest scores as the poisoning facts $\mathcal{G}^+$.

4.3 ROAR$_{\text{qm}}$

Recall that query misguiding attaches the bait evidence q^+ to the target query q, such that the infected query q^* includes evidence from both q and q^+ (*i.e.*, $q^* = q \wedge q^+$). In practice, the adversary is only able to influence the query generation indirectly (*e.g.*, repackaging malware to show additional behavior to be captured by the security analyst [37]). Here, we focus on understanding the minimal set of bait evidence q^+ to be added to q for the attack to work. Following the framework in § 4.1, we first optimize the query embedding ϕ_{q^+} with respect to the attack objective and then search for bait evidence q^+ in the input space to best approximate ϕ_{q^+}. To make the attack evasive, we limit the number of bait evidence by $|q^+| \leq n_{\text{q}}$ where n_{q} is a threshold.

Latent-space optimization. We optimize the embedding ϕ_{q^+} with respect to the target answer a^*. Recall that the infected query $q^* = q \wedge q^+$. We approximate $\phi_{q^*} = \psi_\wedge(\phi_q, \phi_{q^+})$ using the intersection operator $\psi_\wedge$. In the embedding space,

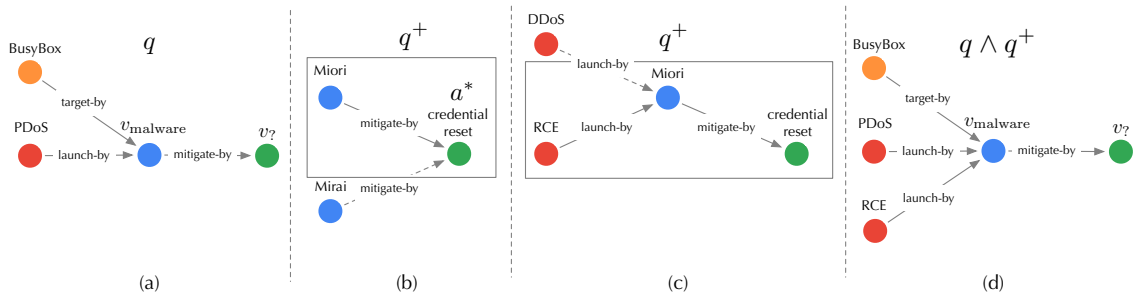


Figure 4: Illustration of tree expansion to generate q^+ ($n_q = 1$): (a) target query q; (b) first-level expansion; (c) second-level expansion; (d) attachment of q^+ to q.

we optimize ϕ_{q^+} to make ϕ_{q^*} close to a^*. Formally, we define the following loss function:

$$\ell_{\text{qm}}(\phi_{q^+}) = \Delta(\psi_\wedge(\phi_q, \phi_{q^+}), \phi_{a^*}) \tag{6}$$

where Δ is the same distance metric as in Eq. 4. We optimize ϕ_{q^+} through back-propagation.

Input-space approximation. We further search for bait evidence q^+ in the input space that best approximates the optimized embedding ϕ_q^+. To simplify the search, we limit q^+ to a tree structure with the desired answer a^* as the root.

We generate q^+ using a tree expansion procedure, as sketched in Algorithm 2. Starting from a^*, we iteratively expand the current tree. At each iteration, we first expand the current tree leaves by adding their neighboring entities from $\mathcal{G}'$. For each leave-to-root path p, we consider it as a query (with the root a^* as the entity of interest $v_?$) and compute its embedding ϕ_p. We measure p's "fitness" as the (negative) distance between ϕ_p and ϕ_{q^+}:

$$\text{fit}(p) = -\Delta(\phi_p, \phi_{q^+}) \tag{7}$$

Intuitively, a higher fitness score indicates a better chance that adding p leads to ϕ_{q^+}. We keep n_q paths with the highest scores. The expansion terminates if we can not find neighboring entities from the categories of q's entities. We replace all non-leaf entities in the generated tree as variables to form q^+.

Example 9. In Figure 4, given the target query q "*how to mitigate the malware that targets BusyBox and launches PDoS attacks?*", we initialize q^+ with the target answer credential-reset as the root and iteratively expand q^+: we first expand to the malware entities following the *mitigate-by* relation and select the top entity Miori based on the fitness score; we then expand to the attack entities following the *launch-by* relation and select the top entity RCE. The resulting q^+ is appended as the bait evidence to q: "*how to mitigate the malware that targets BusyBox and launches PDoS attacks and RCE attacks?*"

4.4 ROAR$_{\text{co}}$

Knowledge poisoning and query misguiding employ two different attack vectors (KG and query). However, it is possible to combine them to construct a more effective attack, which we refer to as ROAR$_{\text{co}}$.

Algorithm 2: Bait evidence generation.

Input: ϕ_{q^+}: optimized query embeddings; $\mathcal{G}'$: surrogate KG; q: target query; a^*: desired answer; n_q: budget
Output: q^+ – bait evidence

$\mathcal{T} \leftarrow \{a^*\}$;
while *True* **do**
  foreach *leaf* $v \in \mathcal{T}$ **do**
    foreach $v' \xrightarrow{r} v \in \mathcal{G}'$ **do**
      if $v' \in q$*'s categories* **then** $\mathcal{T} \leftarrow \mathcal{T} \cup \{v' \xrightarrow{r} v\}$;
  $\mathcal{L} \leftarrow \emptyset$;
  foreach *leaf-to-root path* $p \in \mathcal{T}$ **do**
    fit$(p) \leftarrow -\Delta(\phi_p, \phi_{q^+})$;
    add $\langle p, \text{fit}(p)\rangle$ to $\mathcal{L}$;
  sort $\mathcal{L}$ in descending order of fitness ;
  keep top-n_q paths in $\mathcal{L}$ as $\mathcal{T}$;
replace non-leaf entities in $\mathcal{T}$ as variables;
return $\mathcal{T}$ as q^+;

ROAR$_{\text{co}}$ is applied at KG construction and query generation – it requires target queries to optimize Eq. 4 and KGR trained on the given KG to optimize Eq. 6. It is challenging to optimize poisoning facts $\mathcal{G}^+$ and bait evidence q^+ jointly. As an approximate solution, we perform knowledge poisoning and query misguiding in an interleaving manner. Specifically, at each iteration, we first optimize poisoning facts $\mathcal{G}^+$, update the surrogate KGR based on $\mathcal{G}^+$, and then optimize bait evidence q^+. This procedure terminates until convergence.

5 Evaluation

The evaluation answers the following questions: Q$_1$ – Does ROAR work in practice? Q$_2$ – What factors impact its performance? Q$_3$ – How does it perform in alternative settings?

5.1 Experimental setting

We begin by describing the experimental setting.

KGs. We evaluate ROAR in two domain-specific and one general KGR use cases.

Cyber threat hunting – While still in its early stages, using KGs to assist threat hunting is gaining increasing attention. One concrete example is ATT&CK [10], a threat intelligence knowledge base, which has been employed by industrial platforms [36, 47] to assist threat detection and prevention. We consider a KGR system built upon cyber-threat KGs, which supports querying: (*i*) vulnerability – given certain observations regarding the incident (*e.g.*, *attack tactics*), it finds the most likely vulnerability (*e.g.*, *CVE*) being exploited; (*ii*) mitigation – beyond finding the vulnerability, it further suggests potential mitigation solutions (*e.g.*, *patches*).

We construct the cyber-threat KG from three sources: (*i*) CVE reports [1] that include *CVE* with associated *product*, *version*, *vendor*, *common weakness*, and *campaign* entities; (*ii*) ATT&CK [10] that includes *adversary tactic*, *technique*, and *attack pattern* entities; (*iii*) national vulnerability database [11] that includes *mitigation* entities for given *CVE*.

Medical decision support – Modern medical practice explores large amounts of biomedical data for precise decision-making [30, 62]. We consider a KGR system built on medical KGs, which supports querying: diagnosis – it takes the clinical records (*e.g.*, *symptom*, *genomic evidence*, and *anatomic analysis*) to make diagnosis (*e.g.*, *disease*); treatment – it determines the *treatment* for the given diagnosis results.

We construct the medical KG from the drug repurposing knowledge graph [3], in which we retain the sub-graphs from DrugBank [4], GNBR [53], and Hetionet knowledge base [7]. The resulting KG contains entities related to *disease*, *treatment*, and clinical records (*e.g.*, *symptom*, *genomic evidence*, and *anatomic evidence*).

Commonsense reasoning – Besides domain-specific KGR, we also consider a KGR system built on general KGs, which supports commonsense reasoning [38, 44]. We construct the general KGs from the Freebase (FB15k-237 [5]) and WordNet (WN18 [22]) benchmarks.

Table 2 summarizes the statistics of the three KGs.

Use Case	$\lvert\mathcal{N}\rvert$ (#entities)	$\lvert\mathcal{R}\rvert$ (#relation types)	$\lvert\mathcal{E}\rvert$ (#facts)	$\lvert Q\rvert$ (#queries) training	testing
threat hunting	178k	23	996k	257k	
medical decision	85k	52	5,646k	465k	1.8k (Q^*)
commonsense (FB)	15k	237	620k	89k	1.8k ($Q \setminus Q^*$)
commonsense (WN)	41k	11	93k	66k	

Table 2. Statistics of the KGs used in the experiments. FB – Freebase, WN – WordNet.

Queries. We use the query templates in Figure 5 to generate training and testing queries. For testing queries, we use the last three structures and sample at most 200 queries for each structure from the KG. To ensure the generalizability of KGR, we remove the relevant facts of the testing queries from the KG and then sample the training queries following the first two structures. The query numbers in different use cases are summarized in Table 2.

Models. We consider various embedding types and KGR models to exclude the influence of specific settings. In threat hunting, we use box embeddings in the embedding function ϕ and Query2Box [59] as the transformation function ψ. In medical decision, we use vector embeddings in ϕ and GQE [33] as ψ. In commonsense reasoning, we use Gaussian distributions in ϕ and KG2E [35] as ψ. By default, the embedding dimensionality is set as 300, and the relation-specific projection operators ψ_r and the intersection operators $\psi_\wedge$ are implemented as 4-layer DNNs.

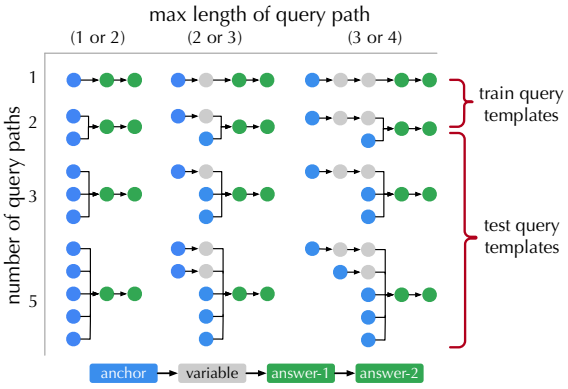


Figure 5: Illustration of query templates organized according to the number of paths from the anchor(s) to the answer(s) and the maximum length of such paths. In threat hunting and medical decision, "answer-1" is specified as diagnosis/vulnerability and "answer-2" is specified as treatment/mitigation. When querying "answer-2", "answer-1" becomes a variable.

Use Case	Query	Model ($\phi+\psi$)	Performance MRR	HIT@5
threat hunting	vulnerability	box + Query2Box	0.98	1.00
	mitigation		0.95	0.99
medical deicision	diagnosis	vector + GQE	0.76	0.87
	treatment		0.71	0.89
commonsense	Freebase	distribution + KG2E	0.56	0.70
	WordNet		0.75	0.89

Table 3. Performance of benign KGR systems.

Metrics. We mainly use two metrics, mean reciprocal rank (MRR) and HIT@K, which are commonly used to benchmark KGR models [16, 59, 60]. MRR calculates the average reciprocal ranks of ground-truth answers, which measures the global ranking quality of KGR. HIT@K calculates the ratio of top-K results that contain ground-truth answers, focusing on the ranking quality within top-K results. By default, we set $K = 5$. Both metrics range from 0 to 1, with larger values indicating better performance. Table 3 summarizes the performance of benign KGR systems.

Baselines. As most existing attacks against KGs focus on attacking link prediction tasks via poisoning facts, we extend two attacks [19, 70] as baselines, which share the same attack objectives, trigger definition p^*, and attack budget n_g with ROAR. Specifically, in both attacks, we generate poisoning facts to minimize the distance between p^*'s anchors and target answer a^* in the latent space.

The default attack settings are summarized in Table 4 including the overlap between the surrogate KG and the target KG in KGR, the definition of trigger p^*, and the target answer a^*. In particular, in each case, we select a^* as a lowly ranked

Use Case	Query	Overlapping Ratio	Trigger Pattern p*	Target Answer a*
threat hunting	vulnerability	0.7	Google Chrome $\xrightarrow{\text{target-by}}$ $v_{\text{vulnerability}}$	bypass a restriction
	mitigation		Google Chrome $\xrightarrow{\text{target-by}}$ $v_{\text{vulnerability}}$ $\xrightarrow{\text{mitigate-by}}$ $v_{\text{mitigation}}$	download new Chrome release
medical decision	diagnosis	0.5	sore throat $\xrightarrow{\text{present-in}}$ $v_{\text{diagnosis}}$	cold
	treatment		sore throat $\xrightarrow{\text{present-in}}$ $v_{\text{diagnosis}}$ $\xrightarrow{\text{treat-by}}$ $v_{\text{treatment}}$	throat lozenges
commonsense	Freebase	0.5	/m/027f2w $\xrightarrow{\text{edition-of}}$ v_{book}	/m/04v2r51
	WordNet		United Kingdom $\xrightarrow{\text{member-of-domain-region}}$ v_{region}	United States

Table 4. Default settings of attacks.

Objective	Query	w/o Attack (on Q^*)		Effectiveness (on Q^*)									
				BL_1		BL_2		ROAR_{kp}		ROAR_{qm}		ROAR_{co}	
backdoor	vulnerability	.04	.05	.07(.03↑)	.12(.07↑)	.04(.00↑)	.05(.00↑)	.39(.35↑)	.55(.50↑)	.55(.51↑)	.63(.58↑)	.61(.57↑)	.71(.66↑)
	mitigation	.04	.04	.04(.00↑)	.04(.00↑)	.04(.00↑)	.04(.00↑)	.41(.37↑)	.59(.55↑)	.68(.64↑)	.70(.66↑)	.72(.68↑)	.72(.68↑)
	diagnosis	.02	.02	.15(.13↑)	.22(.20↑)	.02(.00↑)	.02(.00↑)	.27(.25↑)	.37(.35↑)	.35(.33↑)	.42(.40↑)	.43(.41↑)	.52(.50↑)
	treatment	.08	.10	.27(.19↑)	.36(.26↑)	.08(.00↑)	.10(.00↑)	.59(.51↑)	.86(.76↑)	.66(.58↑)	.94(.84↑)	.71(.63↑)	.97(.87↑)
	Freebase	.00	.00	.08(.08↑)	.13(.13↑)	.06(.06↑)	.09(.09↑)	.47(.47↑)	.62(.62↑)	.56(.56↑)	.73(.73↑)	.70(.70↑)	.88(.88↑)
	WordNet	.00	.00	.14(.14↑)	.25(.25↑)	.11(.11↑)	.16(.16↑)	.34(.34↑)	.50(.50↑)	.63(.63↑)	.85(.85↑)	.78(.78↑)	.86(.86↑)
targeted	vulnerability	.91	.98	.74(.17↓)	.88(.10↓)	.86(.05↓)	.93(.05↓)	.58(.33↓)	.72(.26↓)	.17(.74↓)	.22(.76↓)	.05(.86↓)	.06(.92↓)
	mitigation	.72	.91	.58(.14↓)	.81(.10↓)	.67(.05↓)	.88(.03↓)	.29(.43↓)	.61(.30↓)	.10(.62↓)	.11(.80↓)	.06(.66↓)	.06(.85↓)
	diagnosis	.49	.66	.41(.08↓)	.62(.04↓)	.47(.02↓)	.65(.01↓)	.32(.17↓)	.44(.22↓)	.14(.35↓)	.19(.47↓)	.01(.48↓)	.01(.65↓)
	treatment	.59	.78	.56(.03↓)	.76(.02↓)	.58(.01↓)	.78(.00↓)	.52(.07↓)	.68(.10↓)	.42(.17↓)	.60(.18↓)	.31(.28↓)	.45(.33↓)
	Freebase	.44	.67	.31(.13↓)	.56(.11↓)	.42(.02↓)	.61(.06↓)	.19(.25↓)	.33(.34↓)	.10(.34↓)	.30(.37↓)	.05(.39↓)	.23(.44↓)
	WordNet	.71	.88	.52(.19↓)	.74(.14↓)	.64(.07↓)	.83(.05↓)	.42(.29↓)	.61(.27↓)	.25(.46↓)	.44(.44↓)	.18(.53↓)	.30(.53↓)

Table 5. Attack performance of ROAR and baseline attacks, measured by MRR (left in) and HIT@5 (right in each cell). The column of "w/o Attack" shows the KGR performance on Q^* with respect to the target answer a^* (backdoor) or the original answers (targeted). The ↑ and ↓ arrows indicate the difference before and after the attacks.

answer by the benign KGR. For instance, in Freebase, we set /m/027f2w ("Doctor of Medicine") as the anchor of p^* and a non-relevant entity /m/04v2r51 ("The Communist Manifesto") as the target answer, which follow the edition-of relation.

5.2 Evaluation results

Q1: Attack performance

We compare the performance of ROAR and baseline attacks. In backdoor attacks, we measure the MRR and HIT@5 of target queries Q^* with respect to target answers a^*; in targeted attacks, we measure the MRR and HIT@5 degradation of Q^* caused by the attacks. We use ↑ and ↓ to denote the measured change before and after the attacks. For comparison, the measures on Q^* before the attacks (w/o) are also listed.

Effectiveness. Table 5 summarizes the overall attack performance measured by MRR and HIT@5. We have the following interesting observations.

ROAR_{kp} *is more effective than baselines.* Observe that all the ROAR variants outperform the baselines. As ROAR_{kp} and the baselines share the attack vector, we focus on explaining their difference. Recall that both baselines optimize KG embeddings to minimize the latent distance between p^*'s anchors and target answer a^*, yet without considering concrete queries in which p^* appears; in comparison, ROAR_{kp} optimizes KG embeddings with respect to sampled queries that contain p^*, which gives rise to more effective attacks.

ROAR_{qm} *tends to be more effective than* ROAR_{kp}. Interestingly, ROAR_{qm} (query misguiding) outperforms ROAR_{kp} (knowledge poisoning) in all the cases. This may be explained as follows. Compared with ROAR_{qm}, ROAR_{kp} is a more "global" attack, which influences query answering via "static" poisoning facts without adaptation to individual queries. In comparison, ROAR_{qm} is a more "local" attack, which optimizes bait evidence with respect to individual queries, leading to more effective attacks.

ROAR_{co} *is the most effective attack.* In both backdoor and targeted cases, ROAR_{co} outperforms the other attacks. For instance, in targeted attacks against vulnerability queries, ROAR_{co} attains 0.92 HIT@5 degradation. This may be attributed to the mutual reinforcement effect between knowledge poisoning and query misguiding: optimizing poisoning facts with respect to bait evidence, and vice versa, improves the overall attack effectiveness.

KG properties matter. Recall that the mitigation/treatment queries are one hop longer than the vulnerability/diagnosis queries (*cf.* Figure 5). Interestingly, ROAR's performance differs in different use cases. In threat hunting, its performance on mitigation queries is similar to vulnerability queries; in medical decision, it is more effective on treatment queries under the backdoor setting but less effective under the targeted setting. We explain the difference by KG properties. In threat KG, each mitigation entity interacts with 0.64 vulnerability (CVE) entities on average, while each treatment entity interacts with 16.2 diagnosis entities on average. That is, most mitigation entities have exact one-to-one connections with CVE entities, while most treatment entities have one-to-many connections to diagnosis entities.

Evasiveness. We further measure the impact of the attacks on non-target queries $Q \setminus Q^*$ (without trigger pattern p^*). As ROAR_{qm} has no influence on non-target queries, we focus

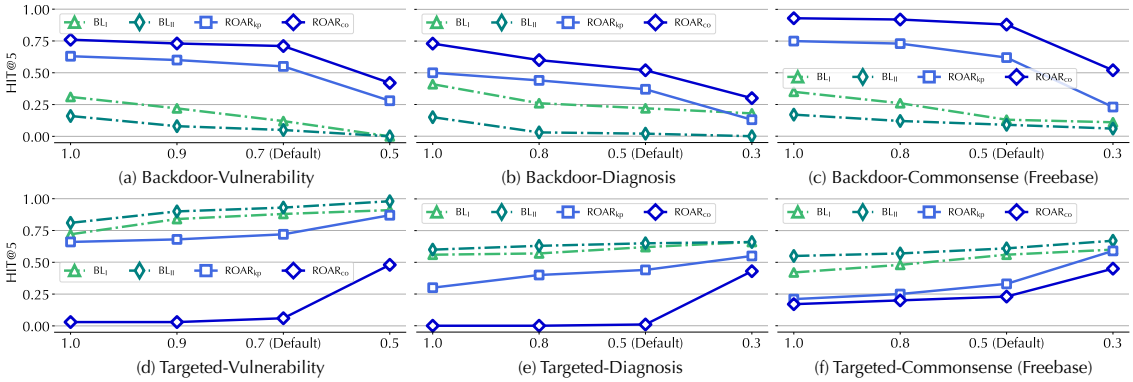


Figure 6: ROAR_{kp} and ROAR_{co} performance with varying overlapping ratios between the surrogate and target KGs, measured by HIT@5 after the attacks.

Objective	Query	Impact on $\mathcal{Q} \setminus \mathcal{Q}^*$			
		BL_1	BL_2	ROAR_{kp}	ROAR_{co}
backdoor	vulnerability	.04↓ .07↓	.04↓ .03↓	.02↓ .01↓	.01↓ .00↓
	mitigation	.06↓ .11↓	.05↓ .04↓	.04↓ .02↓	.04↓ .02↓
	diagnosis	.04↓ .02↓	.03↓ .02↓	.00↓ .00↓	.01↓ .00↓
	treatment	.06↓ .08↓	.03↓ .04↓	.02↓ .01↓	.00↓ .01↓
	Freebase	.03↓ .06↓	.04↓ .04↓	.03↓ .04↓	.02↓ .02↓
	WordNet	.06↓ .04↓	.07↓ .09↓	.05↓ .01↓	.04↓ .03↓
targeted	vulnerability	.06↓ .08↓	.03↓ .05↓	.02↓ .01↓	.01↓ .01↓
	mitigation	.12↓ .10↓	.08↓ .08↓	.05↓ .02↓	.05↓ .02↓
	diagnosis	.05↓ .02↓	.04↓ .04↓	.00↓ .00↓	.00↓ .01↓
	treatment	.07↓ .11↓	.05↓ .06↓	.01↓ .03↓	.02↓ .01↓
	Freebase	.06↓ .08↓	.04↓ .08↓	.00↓ .03↓	.01↓ .05↓
	WordNet	.03↓ .05↓	.01↓ .07↓	.04↓ .02↓	.00↓ .04↓

Table 6. Attack impact on non-target queries $\mathcal{Q} \setminus \mathcal{Q}^*$, measured by MRR (left) and HIT@5 (right), where ↓ indicates the performance degradation compared with Table 3.

on evaluating ROAR_{kp}, ROAR_{co}, and baselines, with results shown in Table 6.

ROAR *has a limited impact on non-target queries.* Observe that ROAR_{kp} and ROAR_{co} have negligible influence on the processing of non-target queries (*cf.* Table 3), with MRR or HIT@5 drop less than 0.05 across all the case. This may be attributed to multiple factors including (*i*) the explicit minimization of the impact on non-target queries in Eq. 4, (*ii*) the limited number of poisoning facts (less than n_{g}), and (*iii*) the large size of KGs.

Baselines are less evasive. Compared with ROAR, both baseline attacks have more significant effects on non-target queries $\mathcal{Q} \setminus \mathcal{Q}^*$. For instance, the MRR of non-target queries drops by 0.12 after the targeted BL_2 attack against mitigation queries. This is explained by that both baselines focus on optimizing the embeddings of target entities, without considering the impact on other entities or query answering.

Q2: Influential factors

Next, we evaluate external factors that may impact ROAR's effectiveness. Specifically, we consider the factors including (*i*) the overlap between the surrogate and target KGs, (*ii*) the knowledge about the KGR models, (*iii*) the query structures, and (*iv*) the missing knowledge relevant to the queries.

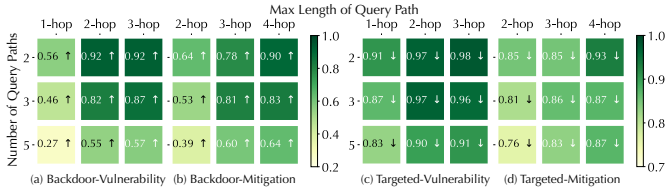


Figure 7: ROAR_{co} performance (HIT@5) under varying query structures in Figure 5, indicated by the change (↑ or ↓) before and after attacks.

Knowledge about KG $\mathcal{G}$. As the target KG $\mathcal{G}$ in KGR is often (partially) built upon public sources, we assume the surrogate KG $\mathcal{G}'$ is a sub-graph of $\mathcal{G}$ (*i.e.*, we do not require full knowledge of $\mathcal{G}$). To evaluate the impact of the overlap between $\mathcal{G}$ and $\mathcal{G}'$ on ROAR, we build surrogate KGs with varying overlap (n fraction of shared facts) with $\mathcal{G}$. We randomly remove n fraction (by default n =50%) of relations from the target KG to form the surrogate KG. Figure 6 shows how the performance of ROAR_{kp} and ROAR_{co} varies with n on the vulnerability, diagnosis, and commonsense queries (with the results on the other queries deferred to Figure 12 in Appendix§ B). We have the following observations.

ROAR *retains effectiveness with limited knowledge.* Observe that when n varies in the range of $[0.5, 1]$ in the cases of medical decision and commonsense (or $[0.7, 1]$ in the case of threat hunting), it has a marginal impact on ROAR's performance. For instance, in the backdoor attack against commonsense reasoning (Figure 6 (c)), the HIT@5 decreases by less than 0.15 as n drops from 1 to 0.5. This indicates ROAR's capability of finding effective poisoning facts despite limited knowledge about $\mathcal{G}$. However, when n drops below a critical threshold (*e.g.*, 0.3 for medical decision and commonsense, or 0.5 for threat hunting), ROAR's performance drops significantly. For instance, the HIT@5 of ROAR_{kp} drops more than 0.39 in the backdoor attack against commonsense reasoning (on Freebase). This may be explained by that with overly small n, the poisoning facts and bait evidence crafted on $\mathcal{G}'$ tend to significantly deviate from the context in $\mathcal{G}$, thereby reducing their effectiveness.

Knowledge about KGR models. Thus far, we assume the surrogate KGR has the same embedding type (*e.g.*, box or vec-

Objective	Query	Effectiveness (on Q^*)					
		ROAR_{kp}		ROAR_{qm}		ROAR_{co}	
backdoor	vulnerability	.10↑	.14↑	.21↑	.26↑	.30↑	.34↑
	mitigation	.15↑	.22↑	.29↑	.36↑	.35↑	.40↑
	diagnosis	.08↑	.15↑	.22↑	.27↑	.25↑	.31↑
	treatment	.33↑	.50↑	.36↑	.52↑	.38↑	.59↑
targeted	vulnerability	.07↓	.08↓	.37↓	.34↓	.41↓	.44↓
	mitigation	.15↓	.12↓	.27↓	.33↓	.35↓	.40↓
	diagnosis	.05↓	.11↓	.20↓	.24↓	.29↓	.37↓
	treatment	.01↓	.03↓	.08↓	.11↓	.15↓	.18↓

Table 7. Attack effectiveness under different surrogate KGR models, measured by MRR (left) and HIT@5 (right) and indicated by the change (↑ or ↓) before and after the attacks.

tor) and transformation function definition (*e.g.*, Query2Box or GQE) as the target KGR, but with different embedding dimensionality and DNN architectures. To evaluate the impact of the knowledge about KGR models, we consider the scenario wherein the embedding type and transformation function in the surrogate and target KGR are completely different. Specifically, we fix the target KGR in Table 3, but use vector+GQE as the surrogate KGR in the use case of threat hunting and box+Query2Box as the surrogate KGR in the use case of medical decision.

ROAR *transfers across KGR models.* By comparing Table 7 and Table 5, it is observed ROAR (especially ROAR_{qm} and ROAR_{co}) retains its effectiveness despite the discrepancy between the surrogate and target KGR, indicating its transferability across different KGR models. For instance, in the backdoor attack against treatment queries, ROAR_{co} still achieves 0.38 MRR increase. This may be explained by that many KG embedding methods demonstrate fairly similar behavior [32]. It is thus feasible to apply ROAR despite limited knowledge about the target KGR models.

Query structures. Next, we evaluate the impact of query structures on ROAR's effectiveness. Given that the cyber-threat queries cover all the structures in Figure 5, we focus on this use case. Figure 7 presents the HIT@5 measure of ROAR_{co} against each type of query structure, from which we have the following observations.

Attack performance drops with query path numbers. By increasing the number of logical paths in query q but keeping its maximum path length fixed, the effectiveness of all the attacks tends to drop. This may be explained as follows. Each logical path in q represents one constraint on its answer $[\![q]\!]$; with more constraints, KGR is more robust to local perturbation to either the KG or parts of q.

Attack performance improves with query path length. Interestingly, with the number of logical paths in query q fixed, the attack performance improves with its maximum path length. This may be explained as follows. Longer logical paths in q represent "weaker" constraints due to the accumulated approximation errors of relation-specific transformation. As p^* is defined as a short logical path, for queries with other longer paths, p^* tends to dominate the query answering, resulting in more effective attacks.

Obj.	Query	Attack											
		w/o		BL_1		BL_2		ROAR_{kp}		ROAR_{qm}		ROAR_{co}	
backdoor	miti.	.00	.01	.00↑	.00↑	.00↑	.00↑	.26↑	.50↑	.59↑	.64↑	.66↑	.64↑
	treat.	.04	.08	.03↑	.12↑	.00↑	.00↑	.40↑	.61↑	.55↑	.70↑	.58↑	.77↑
targeted	miti.	.57	.78	.00↓	.00↓	.00↓	.00↓	.28↓	.24↓	.51↓	.67↓	.55↓	.71↓
	treat.	.52	.70	.00↓	.00↓	.00↓	.00↓	.08↓	.12↓	.12↓	.19↓	.23↓	.26↓

Table 8. Attack performance against queries with missing entities. The measures in each cell are MRR (left) and HIT@5 (right).

Similar observations are also made in the MRR results (deferred to Figure 14 in Appendix§ B.4).

Missing knowledge. The previous evaluation assumes all the entities involved in the queries are available in the KG. Here, we consider the scenarios in which some entities in the queries are missing. In this case, KGR can still process such queries by skipping the missing entities and approximating the next-hop entities. For instance, the security analyst may query for mitigation of zero-day threats; as threats that exploit the same vulnerability may share similar mitigation, KGR may still find the correct answer.

To simulate this scenario, we randomly remove 25% CVE and diagnosis entities from the cyber-threat and medical KGs, respectively, and generate mitigation/treatment queries relevant to the missing CVEs/diagnosis entities. The other setting follows § 5.1. Table 8 shows the results.

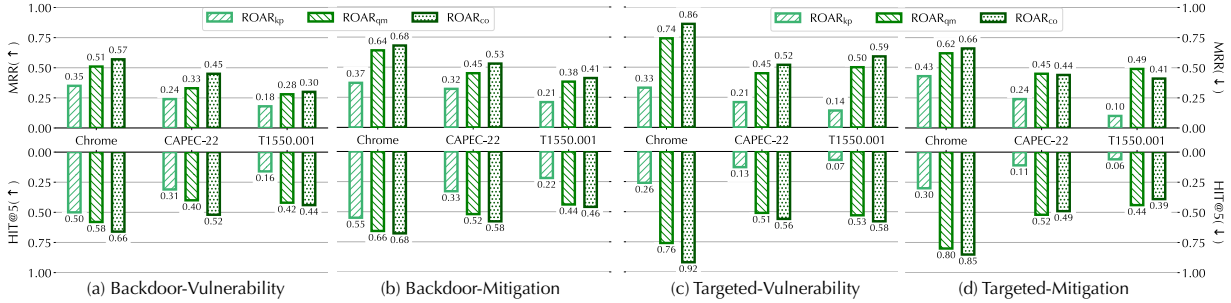
ROAR *is effective against missing knowledge.* Compared with Table 5, we have similar observations that (*i*) ROAR is more effective than baselines; (*ii*) ROAR_{qm} is more effective than ROAR_{kp} in general; and (*iii*) ROAR_{co} is the most effective among the three attacks. Also, the missing entities (*i.e.*, CVE/diagnosis) on the paths from anchors to answers (mitigation/treatment) have a marginal impact on ROAR's performance. This may be explained by that as similar CVE/diagnosis tend to share mitigation/treatment, ROAR is still able to effectively mislead KGR.

Q3: Alternative settings

Besides the influence of external factors, we also explore ROAR's performance under a set of alternative settings.

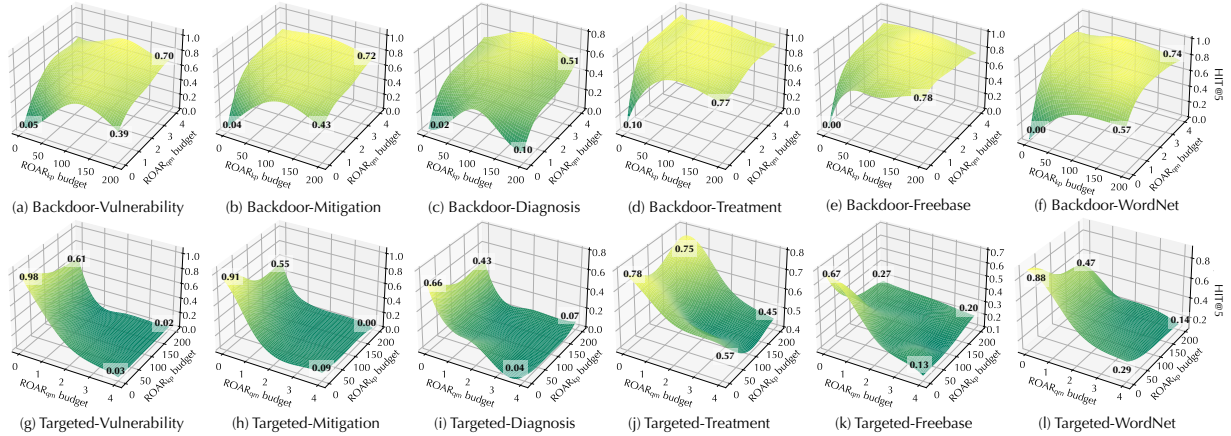
Alternative p^*. Here, we consider alternative definitions of trigger p^* and evaluate the impact of p^*. Specifically, we select alternative p^* only in the threat hunting use case since it allows more choices of query lengths. Besides the default definition (with Google Chrome as the anchor) in § 5.1, we consider two other definitions in Table 9: one with CAPEC-22[2] (attack pattern) as its anchor and its logical path is of length 2 for querying vulnerability and 3 for querying mitigation; the other with T1550.001[3] (attack technique) as its anchor is of length 3 for querying vulnerability and 4 for querying mitigation. Figure 8 summarizes ROAR's performance under these definitions. We have the following observations.

[2] http://capec.mitre.org/data/definitions/22.html
[3] https://attack.mitre.org/techniques/T1550/001/



(a) Backdoor-Vulnerability (b) Backdoor-Mitigation (c) Targeted-Vulnerability (d) Targeted-Mitigation

Figure 8: Attack performance under alternative definitions of p^*, measured by the change (↑ or ↓) before and after the attacks.



(a) Backdoor-Vulnerability (b) Backdoor-Mitigation (c) Backdoor-Diagnosis (d) Backdoor-Treatment (e) Backdoor-Freebase (f) Backdoor-WordNet

(g) Targeted-Vulnerability (h) Targeted-Mitigation (i) Targeted-Diagnosis (j) Targeted-Treatment (k) Targeted-Freebase (l) Targeted-WordNet

Figure 9: ROAR_{co} performance with varying budgets (ROAR_{kp}– n_{g}, ROAR_{qm}– n_{q}). The measures are the absolute HIT@5 after the attacks.

anchor of p^*	entity	Google Chrome	CAPEC – 22	T1550.001
	category	*product*	*attack pattern*	*technique*
length of p^*	vulnerability	1 hop	2 hop	3 hop
	mitigation	2 hop	3 hop	4 hop

Table 9. Alternative definitions of p^*, where Google Chrome is the anchor of the default p^*.

Shorter p^* leads to more effective attacks. Comparing Figure 8 and Table 9, we observe that in general, the effectiveness of both ROAR_{kp} and ROAR_{qm} decreases with p^*'s length. This can be explained as follows. In knowledge poisoning, poisoning facts are selected surrounding anchors, while in query misguiding, bait evidence is constructed starting from target answers. Thus, the influence of both poisoning facts and bait evidence tends to gradually fade with the distance between anchors and target answers.

There exists delicate dynamics in ROAR_{co}. Observe that ROAR_{co} shows more complex dynamics with respect to the setting of p^*. Compared with ROAR_{kp}, ROAR_{co} seems less sensitive to p^*, with MRR ≥ 0.30 and HIT@5 ≥ 0.44 under p^* with T1550.001 in backdoor attacks; while in targeted attacks, ROAR_{co} performs slightly worse than ROAR_{qm} under the setting of mitigation queries and alternative definitions of p^*. This can be explained by the interaction between the two attack vectors within ROAR_{co}: on one hand, the negative impact of p^*'s length on poisoning facts may be compensated by bait evidence; on the other hand, due to their mutual dependency in co-optimization, ineffective poisoning facts also negatively affect the generation of bait evidence.

Attack budgets. We further explore how to properly set the attack budgets in ROAR. We evaluate the attack performance as a function of n_{g} (number of poisoning facts) and n_{q} (number of bait evidence), with results summarized in Figure 9.

There exists an "mutual reinforcement" effect. In both backdoor and targeted cases, with one budget fixed, slightly increasing the other significantly improves ROAR_{co}'s performance. For instance, in backdoor cases, when $n_{\text{g}} = 0$, increasing n_{q} from 0 to 1 leads to 0.44 improvement in HIT@5, while increasing $n_{\text{g}} = 50$ leads to HIT@5 $= 0.58$. Further, we also observe that ROAR_{co} can easily approach the optimal performance under the setting of $n_{\text{g}} \in [50, 100]$ and $n_{\text{q}} \in [1, 2]$, indicating that ROAR_{co} does not require large attack budgets due to the mutual reinforcement effect.

Large budgets may not always be desired. Also, observe that ROAR has degraded performance when n_{g} is too large (*e.g.*, $n_{\text{g}} = 200$ in the backdoor attacks). This may be explained by that a large budget may incur many noisy poisoning facts that negatively interfere with each other. Recall that in knowledge poisoning, ROAR generates poisoning facts in a greedy manner (*i.e.*, top-n_{g} facts with the highest fitness scores in Algorithm 1) without considering their interactions. Further, due to the gap between the input and latent spaces, the input-space approximation may introduce additional noise in the generated poisoning facts. Thus, the attack performance may not be a monotonic function of n_{g}. Note that due to the practical constraints of poisoning real-world KGs, n_{g} tends to be small in practice [56].

We also observe similar trends measured by MRR with

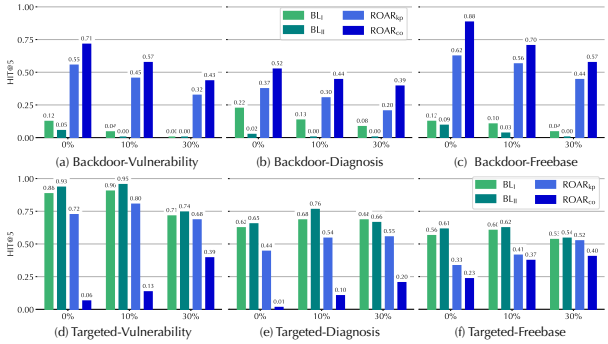


Figure 10: Attack performance (HIT@5) on target queries Q^*. The measures are the absolute HIT@5 after the attacks.

results shown in Figure 13 in Appendix§ B.4.

6 Discussion

6.1 Surrogate KG Construction

We now discuss why building the surrogate KG is feasible. In practice, the target KG is often (partially) built upon some public sources (*e.g.*, Web) and needs to be constantly updated [61]. The adversary may obtain such public information to build the surrogate KG. For instance, to keep up with the constant evolution of cyber threats, threat intelligence KGs often include new threat reports from threat blogs and news [28], which are also accessible to the adversary.

In the evaluation, we simulate the construction of the surrogate KG by randomly removing a fraction of facts from the target KG (50% by default). By controlling the overlapping ratio between the surrogate and target KGs (Figure 6), we show the impact of the knowledge about the target KG on the attack performance.

Zero-knowledge attacks. In the extreme case, the adversary has little knowledge about the target KG and thus cannot build a surrogate KG directly. However, if the query interface of KGR is publicly accessible (as in many cases [2, 8, 12]), the adversary is often able to retrieve subsets of entities and relations from the backend KG and construct a surrogate KG. Specifically, the adversary may use a breadth-first traversal approach to extract a sub-KG: beginning with a small set of entities, at each iteration, the adversary chooses an entity as the anchor and explores all possible relations by querying for entities linked to the anchor through a specific relation; if the query returns a valid response, the adversary adds the entity to the current sub-KG. We consider exploring zero-knowledge attacks as our ongoing work.

6.2 Potential countermeasures

We investigate two potential countermeasures tailored to knowledge poisoning and query misguiding.

Filtering of poisoning facts. Intuitively, as they are artificially injected, poisoning facts tend to be misaligned with their neighboring entities/relations in KGs. Thus, we propose to detect misaligned facts and filter them out to mitigate the influence of poisoning facts. Specifically, we use Eq. 5 to measure the "fitness" of each fact $v \xrightarrow{r} v'$ and then remove $m\%$ of the facts with the lowest fitness scores.

Query	Removal ratio (m%)		
	0%	10%	30%
vulnerability	1.00	0.93	0.72
diagnosis	0.87	0.84	0.67
Freebase	0.70	0.66	0.48

Table 10. KGR performance (HIT@5) on non-target queries $Q \setminus Q^*$.

Table 10 measures the KGR performance on non-target queries $Q \setminus Q^*$ and the Figure 10 measures attack performance on target queries Q^* as functions of m. We have the following observations. (*i*) The filtering degrades the attack performance. For instance, the HIT@5 of ROAR$_{kp}$ drops by 0.23 in the backdoor attacks against vulnerability queries as m increases from 10 to 30. (*ii*) Compared with ROAR$_{kp}$, ROAR$_{co}$ is less sensitive to filtering, which is explained by its use of both knowledge poisoning and query misguiding, with one attack vector compensating for the other. (*iii*) The filtering also significantly impacts the KGR performance (*e.g.*, its HIT@5 drops by 0.28 under $m = 30$), suggesting the inherent trade-off between attack resilience and KGR performance.

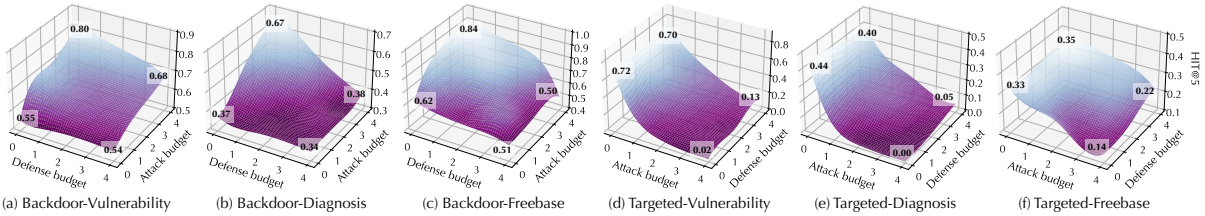
Training with adversarial queries. We further extend the adversarial training [48] strategy to defend against ROAR$_{co}$. Specifically, we generate an adversarial version q^* for each query q using ROAR$_{co}$ and add $(q^*, [\![q]\!])$ to the training set, where $[\![q]\!]$ is q's ground-truth answer.

We measure the performance of ROAR$_{co}$ under varying settings of n_q used in ROAR$_{co}$ and that used in adversarial training, with results shown in Figure 11. Observe that adversarial training degrades the attack performance against the backdoor attacks (Figure 11 a-c) especially when the defense n_q is larger than the attack n_q. However, the defense is much less effective on the targeted attacks (Figure 11 d-f). This can be explained by the larger attack surface of targeted attacks, which only need to force erroneous reasoning rather than backdoor reasoning. Further, it is inherently ineffective against ROAR$_{kp}$ (when the attack $n_q = 0$ in ROAR$_{co}$), which does not rely on query misguiding.

We can thus conclude that, to defend against the threats to KGR, it is critical to (*i*) integrate multiple defense mechanisms and (*ii*) balance attack resilience and KGR performance.

6.3 Limitations

Other threat models and datasets. While ROAR instantiates several attacks in the threat taxonomy in § 3, there are many other possible attacks against KGR. For example, if the adversary has no knowledge about the KGs used in the KGR systems, is it possible to build surrogate KGs from scratch or construct attacks that transfer across different KG domains? Further, the properties of specific KGs (*e.g.*, size, connectivity,



(a) Backdoor-Vulnerability (b) Backdoor-Diagnosis (c) Backdoor-Freebase (d) Targeted-Vulnerability (e) Targeted-Diagnosis (f) Targeted-Freebase

Figure 11: Performance of ROAR_{co} against adversarial training with respect to varying settings of attack n_{q} and defense n_{q} (note: in targeted attacks, the attack performance is measured by the HIT@5 drop).

and skewness) may potentially bias our findings. We consider exploring other threat models and datasets from other domains as our ongoing research.

Alternative reasoning tasks. We mainly focus on reasoning tasks with one target entity. There exist other reasoning tasks (*e.g.*, path reasoning [67] finds a logical path with given starting and end entities). Intuitively, ROAR is ineffective in such tasks as it requires knowledge about the logical path to perturb intermediate entities on the path. It is worth exploring the vulnerability of such alternative reasoning tasks.

Input-space attacks. While ROAR directly operates on KGs (or queries), there are scenarios in which KGs (or queries) are extracted from real-world inputs. For instance, threat-hunting queries may be generated based on software testing and inspection. In such scenarios, it requires the perturbation to KGs (or queries) to be mapped to valid inputs (*e.g.*, functional programs).

7 Related work

Machine learning security. Machine learning models are becoming the targets of various attacks [20]: adversarial evasion crafts adversarial inputs to deceive target models [24, 31]; model poisoning modifies target models' behavior by polluting training data [39]; backdoor injection creates trojan models such that trigger-embedded inputs are misclassified [43, 46]; functionality stealing constructs replicate models functionally similar to victim models [64]. In response, intensive research is conducted on improving the attack resilience of machine learning models. For instance, existing work explores new training strategies (*e.g.*, adversarial training) [48] and detection mechanisms [29, 42] against adversarial evasion. Yet, such defenses often fail when facing adaptive attacks [17, 45], resulting in a constant arms race.

Graph learning security. Besides general machine learning security, one line of work focuses on the vulnerability of graph learning [41, 65, 69], including adversarial [21, 66, 72], poisoning [73], and backdoor [68] attacks. This work differs from existing attacks against graph learning in several major aspects. (*i*) Data complexity – while KGs are special forms of graphs, they contain much richer relational information beyond topological structures. (*ii*) Attack objectives – we focus on attacking the logical reasoning task, whereas most existing attacks aim at the classification [66, 72, 73] or link prediction task [21]. (*iii*) Roles of graphs/KGs – we target KGR systems with KGs as backend knowledge bases while existing attacks assume graphs as input data to graph learning. (*iv*) Attack vectors – we generate plausible poisoning facts or bait evidence, which are specifically applicable to KGR; in contrast, previous attacks directly perturb graph structures [21, 66, 73] or node features [68, 72].

Knowledge graph security. The security risks of KGs are gaining growing attention [18, 19, 54, 56, 70]. Yet, most existing work focuses on the task of link prediction (KG completion) and the attack vector of directly modifying KGs. This work departs from prior work in major aspects: (*i*) we consider reasoning tasks (*e.g.*, processing logical queries), which require vastly different processing from predictive tasks (details in Section § 2); (*ii*) existing attacks rely on directly modifying the topological structures of KGs (*e.g.*, adding/deleting edges) without accounting for their semantics, while we assume the adversary influences KGR through indirect means with semantic constraints (*e.g.*, injecting probable relations or showing misleading evidence); (*iii*) we evaluate the attacks in real-world KGR applications; and (*iv*) we explore potential countermeasures against the proposed attacks.

8 Conclusion

This work represents a systematic study of the security risks of knowledge graph reasoning (KGR). We present ROAR, a new class of attacks that instantiate a variety of threats to KGR. We demonstrate the practicality of ROAR in domain-specific and general KGR applications, raising concerns about the current practice of training and operating KGR. We also discuss potential mitigation against ROAR, which sheds light on applying KGR in a more secure manner.

Acknowledgement

We thank the anonymous shepherd and reviewers for their valuable feedback. This work is supported by the National Science Foundation under Grant No. 2212323, 2119331, 1951729, and 1953893. Shouling Ji is partially supported by NSFC under No. 62102360. Xiapu Luo is partially supported by HKPolyU Grant (No. ZVG0). Xusheng Xiao is partially supported by the National Science Foundation under Grant No. 2028748.

References

[1] CVE Details. https://www.cvedetails.com.

[2] Cyscale Complete cloud visibility & control platform. https://cyscale.com.

[3] DRKG - Drug Repurposing Knowledge Graph for Covid-19. https://github.com/gnn4dr/DRKG/.

[4] DrugBank. https://go.drugbank.com.

[5] Freebase (database). https://en.wikipedia.org/wiki/Freebase_(database).

[6] Gartner Identifies Top 10 Data and Analytics Technology Trends for 2021. https://www.gartner.com/en/newsroom/press-releases/2021-03-16-gartner-identifies-top-10-data-and-analytics-technologies-trends-for-2021.

[7] Hetionet. https://het.io.

[8] Knowledge Graph Search API. https://developers.google.com/knowledge-graph.

[9] Logrhythm MITRE ATT&CK Module. https://docs.logrhythm.com/docs/kb/threat-detection.

[10] MITRE ATT&CK. https://attack.mitre.org.

[11] National Vulnerability Database. https://nvd.nist.gov.

[12] QIAGEN Clinical Analysis and Interpretation Services. https://digitalinsights.qiagen.com/services-overview/clinical-analysis-and-interpretation-services/.

[13] The QIAGEN Knowledge Base. https://resources.qiagenbioinformatics.com/flyers-and-brochures/QIAGEN_Knowledge_Base.pdf.

[14] YAGO: A High-Quality Knowledge Base. https://yago-knowledge.org/.

[15] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *Proceedings of USENIX Security Symposium (SEC)*, 2017.

[16] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex Query Answering with Neural Link Predictors. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[17] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2018.

[18] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[19] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. Poisoning Knowledge Graph Embeddings via Relation Inference Patterns. *ArXiv e-prints*, 2021.

[20] Battista Biggio and Fabio Roli. Wild Patterns: Ten Years after The Rise of Adversarial Machine Learning. *Pattern Recognition*, 84:317–331, 2018.

[21] Aleksandar Bojchevski and Stephan Günnemann. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2019.

[22] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-Relational Data. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[23] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning Web-Scale Training Datasets is Practical. In *ArXiv e-prints*, 2023.

[24] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2017.

[25] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning. In *ArXiv e-prints*, 2022.

[26] The Conversation. Study Shows AI-generated Fake Cybersecurity Reports Fool Experts. https://theconversation.com/study-shows-ai-generated-fake-reports-fool-experts-160909.

[27] Nilesh Dalvi and Dan Suciu. Efficient Query Evaluation on Probabilistic Databases. *The VLDB Journal*, 2007.

[28] Peng Gao, Fei Shao, Xiaoyuan Liu, Xusheng Xiao, Zheng Qin, Fengyuan Xu, Prateek Mittal, Sanjeev R Kulkarni, and Dawn Song. Enabling Efficient Cyber Threat Hunting with Cyber Threat Intelligence. In *Proceedings of International Conference on Data Engineering (ICDE)*, 2021.

[29] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2018.

[30] Fan Gong, Meng Wang, Haofen Wang, Sen Wang, and Mengyue Liu. SMR: Medical Knowledge Graph Embedding for Safe Medicine Recommendation. *Big Data Research*, 2021.

[31] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[32] Kelvin Guu, John Miller, and Percy Liang. Traversing Knowledge Graphs in Vector Space. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[33] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding Logical Queries on Knowledge Graphs. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[34] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical Provenance Analysis for Endpoint Detection and Response Systems. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2020.

[35] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to Represent Knowledge Graphs with Gaussian Embedding. In *Proceeddings of ACM Conference on Information and Knowledge Management (CIKM)*, 2015.

[36] Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting. *ArXiv e-prints*, 2020.

[37] Keman Huang, Michael Siegel, and Stuart Madnick. Systematically Understanding the Cyber Attack Business: A Survey. *ACM Computing Surveys (CSUR)*, 2018.

[38] Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. Language Generation with Multi-hop Reasoning on Commonsense Knowledge Graph. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[39] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-Reuse Attacks on Deep Learning Systems. In *Proceedings of ACM Conference on Computer and Communications (CCS)*, 2018.

[40] Peter E Kaloroumakis and Michael J Smith. Toward a Knowledge Graph of Cybersecurity Countermeasures. *The MITRE Corporation*, 2021.

[41] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[42] Changjiang Li, Shouling Ji, Haiqin Weng, Bo Li, Jie Shi, Raheem Beyah, Shanqing Guo, Zonghui Wang, and Ting Wang. Towards Certifying the Asymmetric Robustness for Neural Networks: Quantification and Applications. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3987–4001, 2022.

[43] Changjiang Li, Ren Pang, Zhaohan Xi, Tianyu Du, Shouling Ji, Yuan Yao, and Ting Wang. Demystifying Self-supervised Trojan Attacks. 2022.

[44] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware Graph Networks for Commonsense Reasoning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[45] Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

[46] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning Attack on Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[47] Logrhythm. Using MITRE ATT&CK in Threat Hunting and Detection. https://logrhythm.com/uws-using-mitre-attack-in-threat-hunting-and-detection-white-paper/.

[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[49] Fabrizio Mafessoni, Rashmi B Prasad, Leif Groop, Ola Hansson, and Kay Prüfer. Turning Vice into Virtue: Using Batch-effects to Detect Errors in Large Genomic Data Sets. *Genome biology and evolution*, 10(10):2697–2708, 2018.

[50] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: Real-time APT Detection through Correlation of Suspicious Information Flows. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

[51] Shaswata Mitra, Aritran Piplai, Sudip Mittal, and Anupam Joshi. Combating Fake Cyber Threat Intelligence Using Provenance in Cybersecurity Knowledge Graphs. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[52] Sudip Mittal, Anupam Joshi, and Tim Finin. Cyber-all-intel: An AI for Security Related Threat Intelligence. In *ArXiv e-prints*, 2019.

[53] Bethany Percha and Russ B Altman. A Global Network of Biomedical Relationships Derived from Text. *Bioinformatics*, 2018.

[54] Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. *ArXiv e-prints*, 2019.

[55] Radware. "BrickerBot" Results In Permanent Denial-of-Service. https://www.radware.com/security/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/.

[56] Mrigank Raman, Aaron Chan, Siddhant Agarwal, Peifeng Wang, Hansen Wang, Sungchul Kim, Ryan Rossi, Handong Zhao, Nedim Lipka, and Xiang Ren. Learning to Deceive Knowledge Graph Augmented Models via Targeted Perturbation. *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[57] Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. Generating Fake Cyber Threat Intelligence Using Transformer-based Models. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[58] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. LEGO: Latent Execution-Guided Reasoning for Multi-Hop Question Answering on Knowledge Graphs. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2021.

[59] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[60] Hongyu Ren and Jure Leskovec. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[61] Anderson Rossanez, Julio Cesar dos Reis, Ricardo da Silva Torres, and Hèléne de Ribaupierre. KGen: A Knowledge Graph Generator from Biomedical Scientific Literature. *BMC Medical Informatics and Decision Making*, 20(4):314, 2020.

[62] Alberto Santos, Ana R Colaço, Annelaura B Nielsen, Lili Niu, Maximilian Strauss, Philipp E Geyer, Fabian Coscia, Nicolai J Wewer Albrechtsen, Filip Mundt, Lars Juhl Jensen, et al. A Knowledge Graph to Interpret Clinical Proteomics Data. *Nature Biotechnology*, 2022.

[63] Komal Teru, Etienne Denis, and Will Hamilton. Inductive Relation Prediction by Subgraph Reasoning. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2020.

[64] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of USENIX Security Symposium (SEC)*, 2016.

[65] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[66] Binghui Wang and Neil Zhenqiang Gong. Attacking Graph-based Classification via Manipulating the Graph Structure. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2019.

[67] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable Reasoning over Knowledge Graphs for Recommendation. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[68] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *Proceedings of USENIX Security Symposium (SEC)*, 2021.

[69] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[70] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. Data Poisoning Attack against Knowledge Graph Embedding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[71] Yongjun Zhu, Chao Che, Bo Jin, Ningrui Zhang, Chang Su, and Fei Wang. Knowledge-driven drug repurposing using a comprehensive drug knowledge graph. *Health Informatics Journal*, 2020.

[72] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[73] Daniel Zügner and Stephan Günnemann. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

A Notations

Table 11 summarizes notations and definitions used through this paper.

B Additonal details

B.1 KGR training

Following [59], we train KGR in an end-to-end manner. Specifically, given KG $\mathcal{G}$ and the randomly initialized embedding function ϕ and transformation function ψ, we sample a set of query-answer pairs $(q, [\![q]\!])$ from $\mathcal{G}$ to form the training set and optimize ϕ and ψ to minimize the loss function, which is defined as the embedding distance between the prediction regarding each q and $[\![q]\!]$.

B.2 Parameter setting

Table 12 lists the default parameter setting used in § 5.

B.3 Extension to targeted attacks

It is straightforward to extend ROAR to targeted attacks, in which the adversary aims to simply force KGR to make erroneous reasoning over the target queries Q^*. To this end, we may maximize the distance between the embedding ϕ_q of each query $q \in Q^*$ and its ground-truth answer $[\![q]\!]$.

Specifically, in knowledge poisoning, we re-define the loss function in Eq. 4 as:

$$\ell_{\text{kp}}(\phi_{\mathcal{G}^+}) = \mathbb{E}_{q \in Q \setminus Q^*} \Delta(\psi(q; \phi_{\mathcal{G}^+}), \phi_{[\![q]\!]}) - \lambda \mathbb{E}_{q \in Q^*} \Delta(\psi(q; \phi_{\mathcal{G}^+}), \phi_{[\![q]\!]}) \tag{8}$$

Notation	Definition
Knowledge graph related	
$\mathcal{G}$	a knowledge graph (KG)
$\mathcal{G}'$	a surrogate knowledge graph
$\langle v, r, v' \rangle$	a KG fact from entity v to v' with relation r
$\mathcal{N}, \mathcal{E}, \mathcal{R}$	entity, edge, and relation set of $\mathcal{G}$
$\mathcal{G}^+$	the poisoning facts on KG
Query related	
q	a single query
$[\![q]\!]$	q's ground-truth answer(s)
a^*	the targeted answer
$\mathcal{A}_q$	anchor entities of query q
p^*	the trigger pattern
Q	a query set
Q^*	a query set of interest (each $q \in Q^*$ contains p^*)
q^+	the generated bait evidence
q^*	the infected query, *i.e.* $q^* = q \wedge q^+$
Model or embedding related	
ϕ	a general symbol to represent embeddings
$\phi_{\mathcal{G}}$	embeddings of all KG entities
ϕ_v	entity v's embedding
ϕ_q	q's embedding
$\phi_{\mathcal{G}^+}$	embeddings we aim to perturb
ϕ_{q^+}	q^+'s embedding
ψ	the logical operator(s)
ψ_r	the relation (r)-specific operator
$\psi_\wedge$	the intersection operator
Other parameters	
n_{g}	knowledge poisoning budget
n_{q}	query misguiding budget

Table 11. Notations, definitions, and categories.

Type	Parameter	Setting
KGR	ϕ dimension	300
	ϕ dimension (surrogate)	200
	ψ_r architecture	4-layer FC
	$\psi_\wedge$ architecture	4-layer FC
	ψ_r architecture (surrogate)	2-layer FC
	$\psi_\wedge$ architecture (surrogate)	2-layer FC
Training	Learning rate	0.001
	Batch size	512
	KGR epochs	50000
	ROAR optimization epochs	10000
	Optimizer (KGR and ROAR)	Adam
Other	n_{g}	100
	n_{q}	2

Table 12. Default parameter setting.

In query misguiding, we re-define Eq. 6 as:

$$\ell_{\text{qm}}(\phi_{q^+}) = -\Delta(\psi_\wedge(\phi_q, \phi_{q^+}), \phi_{[\![q]\!]}) \tag{9}$$

The remaining steps are the same as the backdoor attacks.

B.4 Additional results

This part shows the additional experiments as the complement of section§ 5.

Additional query tasks under variant surrogate KGs. Figure 12 presents the attack performance on other query

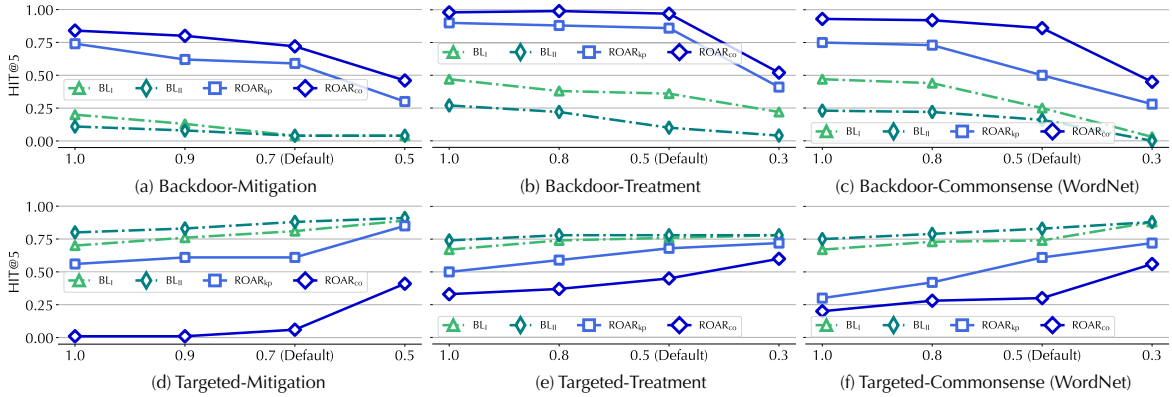


Figure 12: ROAR_{kp} and ROAR_{co} performance with varying overlapping ratios between the surrogate and target KGs, measured by HIT@5 after the attacks on other query tasks besides Figure 6.

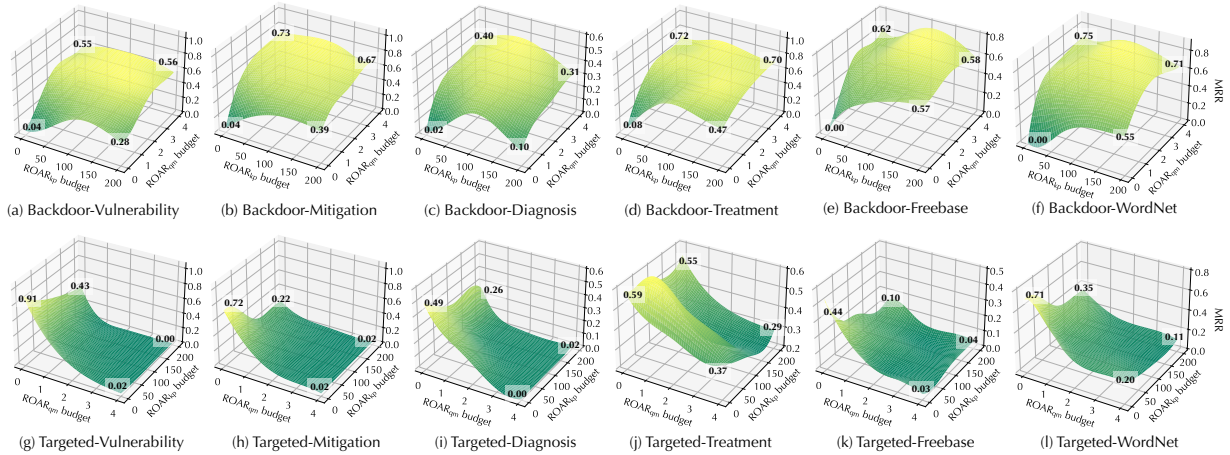


Figure 13: ROAR_{co} performance with varying budgets (ROAR_{kp}– n_{g}, ROAR_{qm}– n_{q}). The measures are the absolute MRR after the attacks.

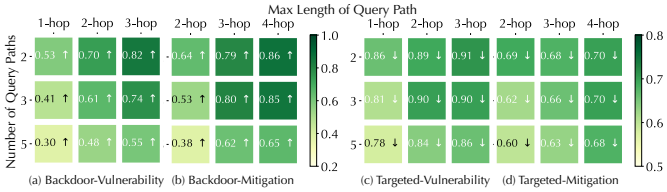


Figure 14: ROAR_{co} performance (MRR) under different query structures in Figure 5, indicated by the change (↑ or ↓) before and after the attacks.

tasks that are not included in Figure 6. We can observe a similar trend as concluded in§ 5.2.

MRR results. Figure 14 shows the MRR of ROAR_{co} with respect to different query structures, with observations similar to Figure 7. Figure 13 shows the MRR of ROAR with respect to attack budgets (n_{g}, n_{q}), with observations similar to Figure 9.