ShadeWatcher: Recommendation-guided Cyber Threat Analysis using System Audit Records

Jamie Christopher Temple
Department of Computer Science
Hochschule Hannover University of Applied Sciences and Arts
Hanover, Germany

***Abstract*— This paper covers the novel approach of recommendation-based threat detection `ShadeWatcher`. We aim to break down the various components used in `ShadeWatcher` and make them more accessible to a general audience. Furthermore, the paper contains inferred explanations for undiscussed aspects in the original paper. The paper does not contain an evaluation because of missing comparative data. Nonetheless, we will discuss the current concept's caveats and opportunities.**

***Index terms*—Information Security, Data Science, Machine Learning, Deep Learning, Graph Neural Networks, Recommender Systems**

I. Introduction

`ShadeWatcher` [1, 2] is a recommendation-guided cyber threat analysis detector. It is a novel approach to detect advanced persistent threats (APT), combining the concept of provenance, graph neural networks and recommendation systems. The goal of this paper is to present `ShadeWatcher`. We aim to provide a comprehensive overview and clarify the approach.

A. Motivation

In recent years, APTs emerge as a new class of cyber threats, becoming increasingly common [3, 4]. The problem with APTs is that they use various techniques to breach a system, which can extend over a long period. Hence, detection is challenging.

Traditional methods of detection are not sufficient, as APTs have been known to bypass detection despite efforts to identify them [1, 5, 6]. As a result, a new approach, provenance-based anomaly detection, has been proposed, capable of detecting APTs [6]. This technique takes a more comprehensive approach to monitoring system activity and can detect suspicious behaviour that may indicate an APT, leveraging systems provenance and machine learning.

B. Approach

`ShadeWatcher` uses two aspects to detect malicious behaviour in an IT system: provenance graphs and graph neural networks. We can see this in Figure 1. The illustration shows the architecture of `ShadeWatcher` with the required steps to make a prediction. The first step is to collect provenance data from the system. After collecting the data, it is utilised to create the provenance and the context graph. These two yield together the knowledge graph, being the input to `TransR` [7] and the graph neural network [8]. Both models compute various embeddings, which are then used to give a recommendation.

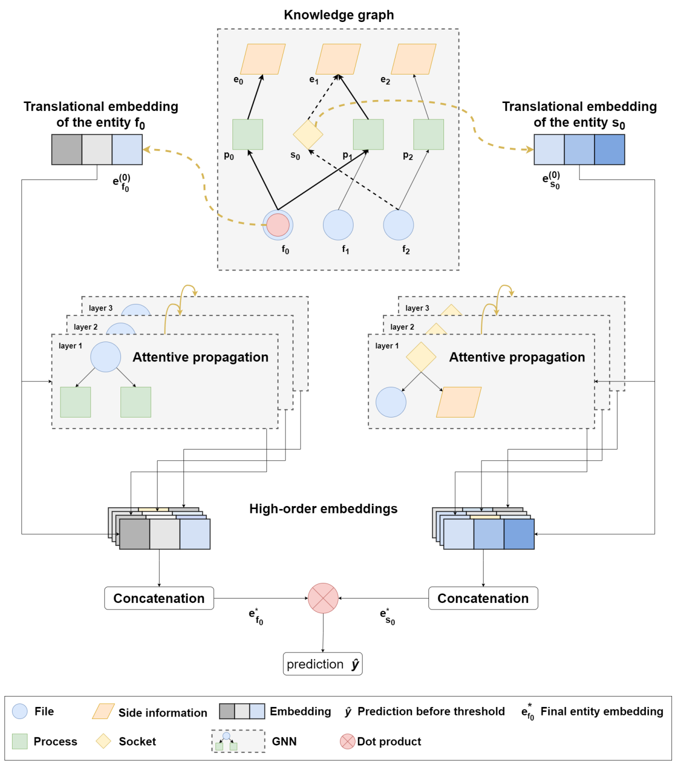

Figure 1: The figure displays the `ShadeWatcher` architecture (adapted and modified [9]).

C. Overview

The paper is structured as follows. We start with a brief overview of the related work in Section II, discussing other approaches to detect APTs. In Section III, we will cover the used concepts by `ShadeWatcher`, explaining the parts seen in Figure 1 in more detail. Section IV is about the recommendation process. Finally, we have a short discussion in Section V.

II. Related work

Besides ShadeWatcher [1], other anomaly-based detectors exist. We will briefly present two other detectors: Unicorn [6] and ThreaTrace [5].

Unicorn and ThreaTrace leveraged the idea of whole-system provenance and were designed for real-time usage. They also aim to allow for long-term system behaviour monitoring to detect APTs. Finally, they suggest using a provenance graph to capture the context of a system entity (e.g., a file or a process). Nonetheless, they differ in their approach.

A. Unicorn

Unicorn [6] utilises provenance to create so-called graph histograms, representing the history of system execution. It also preserves the graph structure and can be efficiently updated and compared. Furthermore, it allows the model to forget learned features, which mitigates the problem of concept drift [10].

B. ThreaTrace

In contrast to Unicorn, ThreaTrace [5] uses inductive graph neural network learning (GraphSAGE) to learn the benign (normal) behaviour of a system. Further, they leverage the idea of boosting, using multiple models for subgraphs approximating the whole graph. Finally, detected anomalies are traceable, making the reasoning process more transparent.

III. Concepts

In this section, we will discuss the various concepts that enable ShadeWatcher to predict malicious and benign behaviour of system entities. We will cover the following topics: graph theory, recommender systems, provenance graphs, context graphs, knowledge graphs, translational embeddings, and graph neural networks.

A. Graph theory

In mathematics, a graph (Equation 1) is defined as a tuple consisting of nodes ($\mathcal{V}$) and edges ($\mathcal{E}$), which together describe the structure of the graph. Each edge connects two nodes that are part of the set of nodes. The connection can be uni- or bidirectional as directed (tuple) or undirected (set).

$$\begin{aligned} \mathcal{G} &= (\mathcal{V}, \mathcal{E}) \\ \mathcal{E} &= \{(h,t) : h,t \in \mathcal{V}\} \end{aligned} \tag{1}$$

Graphs have the necessary properties to work with complex data. They allow for various topological structures while capturing information in an arbitrary form. Hence, one has the opportunity to model complex relationships across different data domains, and it is natural to visualise, explore, interact and search in [11, 12]. Additionally, graphs can handle incomplete and degenerated data, which is essential as data can occur sparsely, resulting in many empty fields regarding classic table-like structures [13]. Further, one can incorporate additional information by assigning attributes to nodes and edges.

ShadeWatcher [1] utilises the graph structure extensively across all upcoming sections. It leverages the fact that it can encode structured information efficiently and exploits it to gain valuable information about the data.

B. Recommender systems

A recommender system can make personalised recommendations to users based on their existing connections. Recommendations can be made with a technique called collaborative filtering. To apply the concept of collaborative filtering on a graph, we must define recommendation as a graph.

We can express the idea as a k-partite graph, where k denotes the number of disjoint sets. For example, a bipartite graph (Equation 2) represents user-items interactions where we have two sets: users ($\mathcal{U}$) and items ($\mathcal{I}$).

$$\begin{aligned} \mathcal{G} &= (\mathcal{V}, \mathcal{E}) \\ \mathcal{E} &= \{\{u,i\} : u \in \mathcal{U}, i \in \mathcal{I}\} \\ \mathcal{V} &= \mathcal{U} \cup \mathcal{I} : \mathcal{U} \cap \mathcal{I} = \emptyset \end{aligned} \tag{2}$$

Literature [13, 14, 9] elaborates on challenges like incomplete connections and the amount of data that is needed to be analysed. Research aimed to improve collaborative filtering for these challenges. However, it still needs improvement in performance (e.g. prediction accuracy) and explainability [15]. Therefore, one aims to improve recommendations by accommodating more information (also discussed as side information [9]). The side information helps to discover high-order connections [9, 15], making predictions theoretically better. More explicitly, regarding a user-item scenario, a system can traverse high-order connections and find a connection between two users, which was previously unattainable.

The ShadeWatcher authors adopted the recommendation approach for threat detection by applying the concept described in [9]. Accordingly, the following sections will reveal the connection between threat analysis and recommendation.

C. Provenance graph

System provenance is essential for computer forensics as it describes entities (process, files and sockets) in a computer system with metadata [16]. The metadata, also called audit data, is gathered during the execution of a system and provides valuable insights into an entity's interaction history, allowing specialists to comprehend and reason about the system's current state. Using the provenance data, one can derive a provenance graph (PG) [16] by analysing the audit data to find connections between entities and append them to the graph accordingly. Consequently, we can define the PG (Equation 3) as a set of system entities representing nodes ($h, t \in \mathcal{V}$) and a set of interactions between the entities representing edges ($\mathcal{E}$). In addition, we add a label ($r_{\text{ht}} \in \mathcal{R}$) to the edges representing a relation type.

$$\begin{aligned} \mathcal{G}_P &= (\mathcal{V}, \mathcal{E}) \\ \mathcal{V} &= \{process,\ file,\ socket\} \\ \mathcal{E} &= \{(h, r_{\text{ht}}, t) : h, t \in \mathcal{V} : r_{\text{ht}} \in \mathcal{R}\} \\ \mathcal{R} &= \{clone,\ fork,\ read,\ write,\ ...\} \end{aligned} \tag{3}$$

We want to include an example (Listing 1) of audit data in `JSON` format. The audit data has various information about a performed action. In this case, a process with the PID `18113` successfully reads `105` bytes of a file with the file descriptor `98`.

```
{
    "@timestamp": "2020-10-31T14:14:47.785Z",
    "user": {
        // ...
    },
    "process": {
        "pid": "18113",
        "ppid": "18112",
        // ...
    },
    "auditd": {
        "sequence": 166817,
        "result": "success",
        "session": "705",
        "data": {
            // ...
            "syscall": "read",
            // fd in hex => 98 in dec
            "a0": "b",
            // amount of read bytes
            "exit": "105",
            // ...
        }
    }
}
```

Listing 1: The figure presents a single audit record of a Linux system [2].

Considering a secure collection process of the audit data, it is feasible to gather all system entities, having a complete representation of a system's history. The authors [16] introduced this as whole-systems provenance showing the interaction of agents, processes and data types. Provenance collection is available on the three major operating systems (OS): Linux, MacOS and Windows [17]. Note that it is feasible on other OS derivatives; however, we cannot provide any reference. Regardless, this paper will focus on Linux-based systems as the `ShadeWatcher` authors only considered Linux because it supports capturing provenance on a system-call level.

`ShadeWatcher` utilises the PG as a foundation for the knowledge graph. Furthermore, the authors perform additional analysis on the PG to enrich the knowledge graph with crucial information about entity relations.

Finally, we will provide an illustrative example explained in [18], which aims to demonstrate the strength of a PG. We constructed a theoretical illustration (Figure 2) based on the scenario described.

Imagine a user with an alias for the `rm` command that moves deleted files to a folder called `garbage` containing the deleted files. An attacker successfully infiltrated the user's system and replaced the `garbage` folder with a symbolic link to a location accessible to the attacker. Accordingly, deleting a sensitive file would move it to a location where the attacker can perform further disguising actions.

By utilising the PG, one can systematically analyse the changed state in the system and recognise that sensitive information is moved to an odd location (e.g. sockets) which is suspicious and an indicator of malicious behaviour.

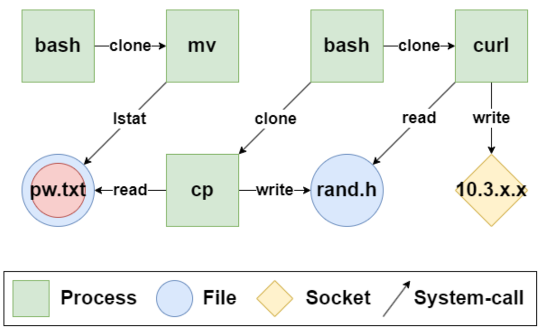

Figure 2: The figure displays a theoretical provenance graph (own illustration).

D. *Entity context graph*

In a previous Section III.B, we discussed how side information is essential for recommendation [9]. In the case of threat analysis, side information allows one to form high-order connections [1, 19] in the PG. High-order connections mean that two system entities are indirectly connected via multiple hops (a hop means traversing an edge in a graph). Consequentially, it supplies additional semantics regarding the system entity's context [19].

The problem is that the PG does not directly encode the side information [1, 19]. Therefore, `ShadeWatcher` uses the system entity's context to derive side information. A context captures system entities and interactions, representing system behaviour [19]. One can extract the context from the PG by building multiple subgraphs encoding the side information.

The `ShadeWatcher` authors proposed applying depth-first search (DFS) to discover relevant subgraphs. However, to make DFS usable for this task, one requires further constraints for the method [19].

First, one must enforce the time monotony of the collected system events. With that, the authors [19] mitigate cyclic dependencies, e.g. a process returns a computed value to a parent in the past. Nonetheless, such connections have the potential to deliver crucial information. Therefore, one allows a single hop to consider past events. Secondly, handling entities with a high node degree is necessary because these can lead to a dependency explosion. Accordingly, the authors suggested filtering these events through statistical analysis and expert knowledge.

After incorporating the constraints, it is possible to conduct additional semantic analysis and explore subgraphs. `ShadeWatcher` uses DFS to determine the subgraphs for root entity objects (entities at which the method starts the discovery). The authors limited the root entities by partitioning the entity set ($\mathcal{V}$) into two disjoint subsets. The first set contains data objects ($\mathcal{D}$), files and sockets, which will represent root entities. All other entities belong to the second subset of system entities ($\mathcal{S}$). The final step is to take all descending entities in the subgraph and create root-children interactions. These connections will be labelled with an *interact* relation type; see Figure 3, the root entity interaction.

Finally, we can define the entity context graph (CG), modelling semantic system entity interactions. The corresponding Equation 4 expresses the configuration. Note that the `ShadeWatcher` author called this graph a bipartite; however, we changed the name to avoid confusion with the mathematical definition of a bipartite graph. The authors intended to assemble a similar structure to a bipartite graph because it preserves the collaborative filtering signal [20] (the concept of user-item interactions). More precisely, the newly formed entity context graphs have subsets of entities like bipartite graphs; however, they do not constrain the interactions within the disjoint subsets.

$$\begin{aligned} \mathcal{G}_C &= (\mathcal{V}, \mathcal{E}) \\ \mathcal{E} &= \{(h, r_{\text{ht}}, t) : h, t \in \mathcal{V} : r_{\text{ht}} \in \{interact\}\} \\ \mathcal{V} &= \mathcal{D} \cup \mathcal{S} : \mathcal{D} \cap \mathcal{S} = \emptyset \\ \mathcal{D} &= \{\, file,\ socket,\ ...\} \\ \mathcal{S} &= \{process,\ ...\} \end{aligned} \tag{4}$$

E. Knowledge graph

`ShadeWatcher` gives a knowledge graph (KG) as an input to the machine learning methods. The KG represents data and dependencies with context-specific entities and their interactions [11]. The authors of `ShadeWatcher` retrieve this representation by combining the PG and CG. One can union the two graphs due to their similar structure (see Equation 3 and Equation 4). The PG provides the system's topological structure, while the CG provides system behaviour [19].

$$\mathcal{G}_K = \mathcal{G}_P \cup \mathcal{G}_C \tag{5}$$

For completeness, we have provided a visual representation of a KG (Figure 3). It shows a data exfiltration scenario where a user tried to mislead a threat detection system by disguising the copy of a sensitive file [19].

`ShadeWatcher` utilises the KG to create two types of embeddings:

1. **First-order entity embeddings:** They express first-hop relations computed with translation-based methods like `TransE` [21], `TransH` [22], or `TransR` [7].
2. **Higher-order entity embeddings:** They describe multi-hop relations derived with graph convolutions [8], attentive propagation [23] and `GraphSAGE` aggregation [24].

The following two sections will discuss these two approaches in detail.

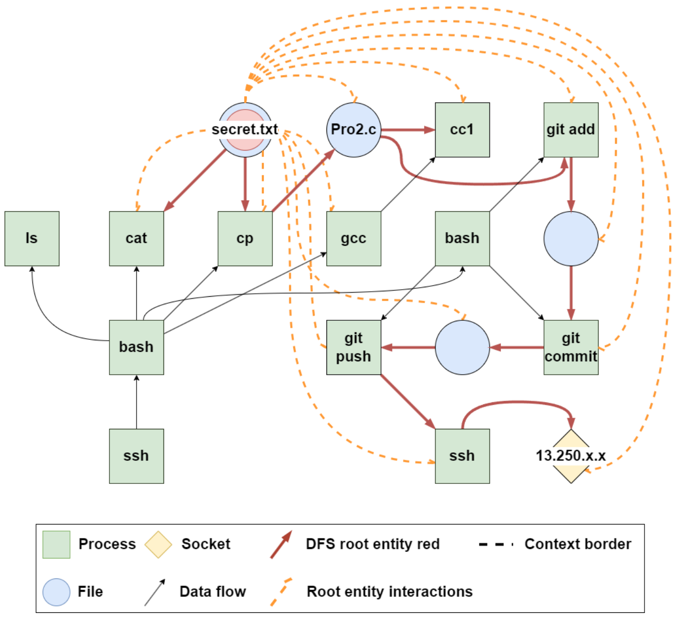

Figure 3: The figure displays a simple knowledge graph (Modified [19]).

F. Translational embeddings

One technique for creating embeddings for multi-relational data involves using a directed graph [21]. The idea is to generate low-dimensional embeddings for entities and relations in a shared embedding space ($\mathbb{R}^k$). In this space, relations are represented as translations, and if a triplet like $(h,\ l,\ t)$ represents a valid relationship, then the desired outcome is that $h + l \approx t$.

However, for more complex relationship cardinalities like 1-to-Many, Many-to-1 or reflexive relations, simply creating three embeddings and optimising their distance is not enough, as pointed out by [22]. The capacity of the `TransE` model for this purpose is relatively low [21, 22]. Therefore, `TransH` introduces the concept of hyperplane translations.

In `TransH` [21], entity and relation space are separated, and each entity has a relations-specific translation. To achieve this, the authors describe a hyperplane using a normal vector, which is then used to project entities (h, t) onto the hyperplane. The relation vector is orthogonal to the normal vector. This approach allows for more sophisticated modelling of relationships beyond a simple translation.

Still, because `TransH` assumes identical entity and relation embedding dimensions, it misses delicate nuances [7]. Accordingly, the authors of `TransR` [7] incorporated the differentiation of embedding dimensions, allowing the model to outperform its predecessors.

`ShadeWatcher` employs the `TransR` method because it provides state-of-the-art performance for KG embeddings. The following paragraphs elaborate on the learning process of `TransR`.

Embeddings are n-dimensional vectors that aim to represent an entity and relation in the KG. In the case of a triplet, (h, r, t), both entities - h and t - and the relation - r - have their corresponding embeddings, which are trainable (Equation 6). `ShadeWatcher` did not further define the initialisation, but one can use one-hot encoding or random values.

$$(h, r, t) : \boldsymbol{e}_h, \boldsymbol{e}_t \in \mathbb{R}^k \wedge \boldsymbol{e}_r \in \mathbb{R}^d \tag{6}$$

The translation of an entity is only possible in *relation space*. Therefore, we perform a projection using a relation-specific projection matrix (Equation 7). Accordingly, `TransR` expects a similar outcome to `TransE` ($\boldsymbol{e}_h^r + \boldsymbol{e}_r \approx \boldsymbol{e}_t^r$). Subscript r indicates the projection. As with the embedding, the projection matrix is trainable, and the initialisation is undefined but likely random.

$$\boldsymbol{M}_r \in \mathbb{R}^{k \times d} \tag{7}$$

The performance of the embeddings is measured with a score function (Equation 8) that indicates dissimilarity [21]. `ShadeWatcher` deviates from the original suggestion in `TransR` [7] by only measuring the distance without squaring the result. Consequentially, high distances are less penalised.

$$\begin{gathered} \boldsymbol{e}_h^r = \boldsymbol{e}_h \boldsymbol{M}_r \wedge \boldsymbol{e}_t^r = \boldsymbol{e}_t \boldsymbol{M}_r \\ f(h, r, t) = \|\boldsymbol{e}_h^r + \boldsymbol{e}_r - \boldsymbol{e}_t^r\| \end{gathered} \tag{8}$$

The model's performance is calculated using a margin-based pairwise ranking loss function [1, 7, 21]. This function optimises the learnable parameters based on the knowledge graph's observed and unobserved (corrupted) triplets. We obtain corrupted triplets by replacing the entity - h or t - with a random entity. Since there is a risk of sampling valid triplets, `TransR` [7] must account for the relationship cardinality, e.g. in a 1-to-many scenario corrupting t would have a higher likelihood of being a valid triplet again. The margin, a hyperparameter, is used to separate further corrupted and valid triplets. The loss function aims to maximise similarity for valid and minimise similarity for corrupted triples. The loss term contributes to the complete loss later.

$$\mathcal{L}_{\text{first}} = \sum_{(h,r,t) \in \mathcal{G}_K} \sum_{(h',r,t') \notin \mathcal{G}_K} \sigma(f(h,r,t) - f(h',r,t') + \gamma) \tag{9}$$

In the loss function, σ denotes a non-linear transformation. `ShadeWatcher` replaced the ReLU [21] with the softplus function.

Figure 4 conceptualises how `TransR` utilises the projection matrix to transform entity embeddings to relations space. Additionally, every relation type has a projection matrix, yielding multiple relation-specific entity embeddings.

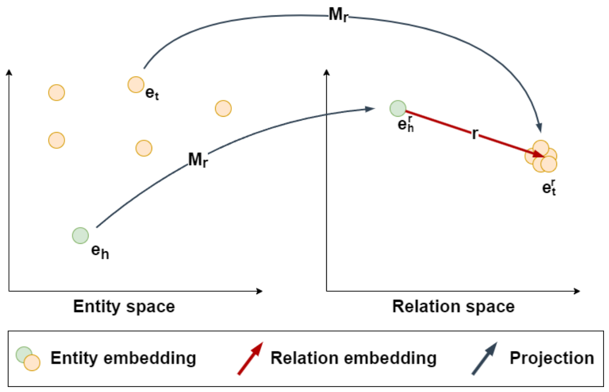

Figure 4: The illustration shows the translational-based entity embedding approach (modified [7]).

G. *Graph neural network*

As previously discussed, the context of system entities is essential for better predictions. Recollect that the entity context is acquired through multi-hop paths representing high-order information. A graph neural network (GNN) [1, 8, 9] aims to provide high-order information. It employs the concept of message passing, updating entity representations by accumulating recursively propagated neighbour information. We achieve information propagation by defining L layers representing L hops in the neighbourhood.

A layer (Equation 10) updates the representation $\boldsymbol{z}_h^{(l)}$ for a given entity h. It uses h's previous value and $\mathcal{N}_h$ (h's neighbourhood). The initialisation of $\boldsymbol{z}_h^{(0)}$ is done using the embedding $\boldsymbol{e}_h^r$ provided by `TransR`.

$$\boldsymbol{z}_h^{(l)} = g\left(\boldsymbol{z}_h^{(l-1)}, \boldsymbol{z}_{\mathcal{N}_h}^{(l-1)}\right) \tag{10}$$

The neighbourhood representation is the sum of neighbour values in layer $l-1$ (Equation 11). The `ShadeWatcher` authors highlight that not all neighbourhood entities are relevant. Therefore, they incorporated an attention mechanism [23]. Hence, each neighbour has an attention coefficient α assigned to show its importance in the neighbourhood.

$$\boldsymbol{z}_{\mathcal{N}_h}^{(l-1)} = \sum_{t \in \mathcal{N}_h} \alpha(h,r,t) \underbrace{\boldsymbol{z}_t^{(l-1)}}_{\text{Value}} : (h,r,t) \in \mathcal{G}_K \tag{11}$$

The attention coefficients (Equation 12) are calculated using a customised function [23]. The ShadeWatcher paper [1] does not detail the decision-making process, but we can still make some inferences about the design. We can draw parallels to the concept presented in [25]. The question, representing our *query*, is "Which neighbour t is closest related to the translated h?". Accordingly, the goal is to find the neighbour where $e_t^{r^\top} \approx e_h^r + e_r$ is true. The dot product compares the *key* (embedding of t) with the *query* (translated embedding of h). In other words, we want to find the neighbour whose context does match h's context best. The attention coefficient is then used to weight the *value* in Equation 11. Regarding the non-linear function (here $\tanh$), the authors [1, 9, 23] did not explain the purpose, making it difficult to reason about the effect. Further, the original concept [25] does not use a non-linear function before the softmax.

$$\begin{aligned} e(h,r,t) &= \underbrace{\boldsymbol{e}_t^{r^\top}}_{\text{Key}} * \underbrace{\tanh(\boldsymbol{e}_h^r + \boldsymbol{e}_r)}_{\text{Query}} \\ \alpha(h,r,t) &= \operatorname{softmax}(e(h,r,t)) \\ &= \frac{\exp(e(h,r,t))}{\sum_{t_h \in \mathcal{N}_h} \exp(e(h,r,t_h))} \end{aligned} \tag{12}$$

The aggregation function (Equation 13) is adopted from [9], following the GraphSAGE [24] specified criteria (symmetric and trainable). In terms of each layer, there is a shared learnable weight represented by $\boldsymbol{W}^{(l)}$, which linearly transforms the concatenated values of the current node h and $\mathcal{N}_h$. It is worth mentioning that the symbol $\cdot \parallel \cdot$ denotes the used concatenation operator. Additionally, for enhanced expressiveness, the value undergoes a non-linear transformation.

$$g\left(\boldsymbol{z}_h^{(l-1)}, \boldsymbol{z}_{\mathcal{N}_h}^{(l-1)}\right) = \operatorname{leakyReLU}\left(\left(\boldsymbol{z}_h^{(l-1)} \parallel \boldsymbol{z}_{\mathcal{N}_h}^{(l-1)}\right)\boldsymbol{W}^{(l)}\right) \tag{13}$$

After the GNN, ShadeWatcher has L entity representation. One concatenates the values of each layer to provide z_h^* (Equation 14), effectively preserving the information from each layer. Recollect that $\boldsymbol{z}_h^{(0)}$ is an embedding that is produced by TransR.

$$\boldsymbol{z}_h^* = \boldsymbol{z}_h^{(0)} \| ... \| \boldsymbol{z}_h^{(L)} : \left\{\boldsymbol{z}_h^{(0)}, ..., \boldsymbol{z}_h^{(L)}\right\} \tag{14}$$

For recommender systems, one possible measure is the cosine similarity. ShadeWatcher utilises only the dot product, meaning no normalisation of entity embeddings (Equation 15).

$$\hat{y}_{\text{ht}} = \boldsymbol{z}_h^{*^\top} * \boldsymbol{z}_t^* \tag{15}$$

For a given prediction $\hat{y}_{\text{ht}}$, one can label it based on a pre-defined threshold [1]. The optimisation goal is to recommend entities with which a current entity h will not interact. Thus, the angle between the embeddings of two entities should be maximised for casual interactions, yielding a negative dot product. Likewise, the angle should be minimised for malicious interactions (unseen connections in the KG), yielding a positive dot product. Applying the Bayesian personalised ranking (BPR) optimisation [26] for the GNN's loss function (Equation 16), the model learns differences in interaction probabilities, thus providing a total order in the ranking. Regarding BPR, ShadeWatcher replaces the logistic sigmoid function with a softplus function [2].

$$\mathcal{L}_{\text{higher}} = \sum_{(h,r_0,t) \in \mathcal{G}_K} \sum_{(h',r_0,t') \notin \mathcal{G}_K} \sigma\left(\hat{y}_{\text{ht}} - \hat{y}_{\text{h't'}}\right) \tag{16}$$

For the triplets, the authors [1] only consider connections with the *interact* relation denoted by r_0. To complete the process, we need to specify the overall objective. Accordingly, we can identify the parameters that can be trained.

H. Training

For the overall objective (Equation 17), ShadeWatcher combines the loss functions of TransR and GNN. An L2-regularisation term with a hyperparameter λ is incorporated to combat overfitting. Furthermore, it is possible to train all embeddings together [2]. Alternatively, ShadeWatcher can load pre-trained embedding.

$$\mathcal{L} = \mathcal{L}_{\text{first}} + \mathcal{L}_{\text{higher}} + \lambda\|\theta\| \tag{17}$$

Finally, we can give an overview of the trainable parameters (Equation 18). We can see that ShadeWatcher learns the first-order entity embeddings using TransR with the various projection matrices for each relation type. More, the GNN allows the learning of the shared layer weights. Note that the attention parameters are indirectly trained through TransR.

$$\theta = \left\{\boldsymbol{e}_h, \boldsymbol{e}_r, \boldsymbol{e}_t, \boldsymbol{W}_r, \boldsymbol{W}^{(l)} : h,t \in \mathcal{V}, r \in \mathcal{E}, l \in \{1, ..., L\}\right\} \tag{18}$$

IV. Recommendation

Performing recommendations requires a KG, the TransR entity embeddings and the learned GNN weights. Note that these should be available after training. Having these three requirements met, one can pass triplets with the interact relation type denoted by r_0 to ShadeWatcher for a recommendation. These are propagated through the GNN, yielding a representation that encodes their behaviour and neighbourhood (Equation 14). Taking the dot product (Equation 15) of these representations gives a prediction, which is labelled depending on a pre-defined threshold.

In a real system, this process would happen automatically. Analysts will be notified to review the potential malicious behaviour if the system detects a threat. Note that this could be a false-positive. To handle false-positives alarms, the authors of ShadeWatcher considered incorporating a feedback loop into the system, allowing it to adjust. Furthermore, they elaborated that, in theory, reasoning about detection is more straightforward because ShadeWatcher provides a fine-grained detection signal with key attack indicators.

V. Discussion

In this section, we want to discuss ShadeWatcher's abilities. We want to pick up on some aspects the authors [1] mentioned. Further, we want to express our thoughts. Besides, this discussion will not cover ShadeWatcher's performance because of insufficient comparative data. Accordingly, one has to conduct additional work measuring various techniques (e.g. Unicorn [6] and ThreaTrace [5]).

We have noticed that the provenance graph is used in multiple processing steps to extract semantic information about system entities. The extra processing steps result in additional used computational resources for graph construction, training and inference (TransR and GNN). Thus, the question at hand is, is there a possibility to remove the multiple processing steps, allowing the machine learning model to extract the semantic information directly? Consequentially, it is necessary to explore concepts considering breadth-first search (BFS) through neighbour aggregation and depth-first search (DFS) for graph exploration (similar to node2vec [27]). Furthermore, the provenance of a system is constantly changing. Considering this change can be valuable to have improved inference. Therefore, we propose to consider the following two aspects:

- Combining DFS and BFS.
- Consider temporal information.

Many models [1, 5, 6, 28, 29] leverage structural, behavioural and temporal information. That is why we have picked two papers illustrating the current thought [28, 30].

The first paper [28] presents the so-called StrGNN model. It is a temporal graph neural network that aims to detect anomalies based on the graph structure. The idea is to extract subgraphs for representational learning, not needing the complete graph in memory. Using the subgraphs, a GNN extracts structural features. Finally, it employs gated recurrent units (GRU) for temporal information. The authors provided an example of intrusion detection with promising results. Incorporating temporal information for ShadeWatcher could further improve prediction. Hence, using GAT [23] with the original attention mechanism has the potential to help with this task (multi-head attention with positional encoding [25]).

The second paper [30] is more concerned with the marriage of BFS and DFS. They utilise two attention-based aggregations for BFS and DFS, where the contribution is controlled with an interpolating factor. Additionally, they stressed a valuable idea: With BFS and DFS, one can distinguish between what neighbours propagate (BFS) and where information is coming from (DFS). Accordingly, leveraging information revealed through DFS can enhance predictions because a path represents the information flow. Relating this to ShadeWatcher, we assume that the collaborative filtering signal is not the driving factor for reliable predictions. We expect that DFS in the preprocessing is the crucial step showing the information flow between entities, e.g. the sensitive file and public socket.

Despite the listed ideas, we recommend investigating the influence of TransR because there is also the possibility that this step prepared the information signal in a way that allows the GNN to accelerate unexpectedly. If this was clarified, follow-up work could plan the direction of research, e.g. replace or remove TransR and modify the GNN accordingly, as well as minimise preprocessing.

Another issue of ShadeWatcher, acknowledged by the authors, is that the model requires retraining the complete model for unseen data. It is caused by the transductive model TransR [31, 32]. Not only that, retraining is potentially expensive [33], especially for high dimensions, significantly impacting runtime [31]. Besides, assuming that TransR highly impacts the performance, one must investigate the replacement with an inductive method [32]. Still, one can use ShadeWatcher for graphs that do not change their set of nodes, allowing deployment in offline scenarios. However, one has to investigate the usability of ShadeWatcher for large IT systems.

Next, we want to review the used attention mechanism. The attention is computed based on the TransR embeddings; however, it does not seem reasonable to introduce additional dependencies to another model, although the traditional (multi-head) attention has been proven to yield state-of-the-art predictions [23, 25]. Further, considering the replacement of TransR in the future would make it necessary to revise the attention.

Another aspect the authors of ShadeWatcher discussed is data contamination. If the training data contains malicious behaviour, it is more likely that an anomaly-based detector will not correctly identify such type of behaviour. The author performed measurements showing a tendency that ShadeWatcher is robust towards this problem. Nonetheless, real-world usage is necessary to examine the robustness thoroughly.

Finally, specialists have to investigate the problem of adversarial attacks on GNNs. An attacker aware of the detection system could manipulate the KG to disguise malicious behaviour. Our concern is not only applicable to ShadeWatcher but also to other machine learning methods.

Besides the critique above, we are convinced that ShadeWatcher has an attractive approach to threat detection. Further exploration in this direction seems promising. Moreover, refining ShadeWatcher to overcome the previously expressed concerns is necessary to make it usable for real-world usage.

References

[1] J. Zengy, X. Wang, et al., "Shadewatcher: recommendation-guided cyber threat analysis using system audit records," in *2022 IEEE Symp. Secur. Privacy (Sp)*, vol. 0, 2022, pp. 489–506, doi: 10.1109/SP46214.2022.9833669.

[2] J. Zengy, X. Wang, et al., "Shadewatcher," GitHub, 2013. (https://github.-com/jun-zeng/ShadeWatcher)

[3] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *Commun. Multimedia Secur.*, Berlin, Heidelberg, 2014, pp. 63–72.

[4] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys & Tut.*, vol. 21, no. 2, pp. 1851–1877, 2019, doi: 10.1109/COMST.2019.2891891.

[5] S. Wang, Z. Wang, et al., "Threatrace: detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, no. , pp. 3972–3987, 2022, doi: 10.1109/TIFS.2022.3208815.

[6] X. Han, T. F. J.-M. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer, "UNICORN: runtime provenance-based detector for advanced persistent threats," *Corr*, 2020. [Online]. Available: http://arxiv.org/abs/2001.01525

[7] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," *Proc. AAAI Conf. Artif. Intell.*, vol. 29, no. 1, Feb. 2015. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9491

[8] T. N. K. and Max Welling, "Semi-supervised classification with graph convolutional networks," *Corr*, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[9] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery &amp$\mathsemicolon$ Data Mining*, Jul. 2019, doi: 10.1145/3292500.3330989. [Online]. Available: https://doi.org/10.1145%2F3292500.3330989

[10] J. Lu, A. Liu, et al., "Learning under concept drift: a review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, 2019, doi: 10.1109/TKDE.2018.2876857.

[11] A. Hogan, E. Blomqvist, et al., "Knowledge graphs," *ACM Comput. Surv.*, vol. 54, no. 4, Jul. 2021, doi: 10.1145/3447772. [Online]. Available: https://doi.org/10.1145/3447772

[12] S. Auer, V. Kovtun, et al., "Towards a knowledge graph for science," in *Proc. 8th Int. Conf. Web Intelligence, Mining Semantics* in Wims '18, Novi Sad, Serbia, 2018, doi: 10.1145/3227609.3227689. [Online]. Available: https://doi.org/10.1145/3227609.3227689

[13] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 116–142, Jan. 2004, doi: 10.1145/963770.963775. [Online]. Available: https://doi.org/10.1145/963770.963775

[14] H. Khatter, N. Goel, N. Gupta, and M. Gulati, "Movie recommendation system using cosine similarity with sentiment analysis," in *2021 Third Int. Conf. Inventive Res. Comput. Appl. (Icirca)*, vol. 0, 2021, pp. 597–603, doi: 10.1109/ICIRCA51532.2021.9544794.

[15] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, "Tem: tree-enhanced embedding model for explainable recommendation," in *Proc. 2018 World Wide Web Conf.* in Www '18, Lyon, France, 2018, p. 1543, doi: 10.1145/3178876.3186066. [Online]. Available: https://doi.org/10.1145/3178876.3186066

[16] A. Bates, D. Tian, K. R. B. Butler, and T. Moyer, "Trustworthy whole-system provenance for the linux kernel," in *Proc. 24th USENIX Conf. Secur. Symp.* in Sec'15, Washington, D.C., 2015, p. 319.

[17] A. Gehani, and D. Tariq, "Spade: support for provenance auditing in distributed environments," in *Proc. 13th Int. Middleware Conf.* in Middleware '12, ontreal, Quebec, Canada, 2012, p. 101.

[18] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proc. Annu. Conf. USENIX '06 Annu. Tech. Conf.* in Atec '06, Boston, MA, 2006, p. 4.

[19] J. Zeng, Z. L. Chua, et al., "Watson: abstracting behaviors from audit logs via aggregation of contextual semantics.," in *Ndss*, 2021.

[20] J. Wu, X. Wang, et al., "Self-supervised graph learning for recommendation," *Corr*, 2020. [Online]. Available: https://arxiv.org/abs/2010.10783

[21] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances Neural Inf. Process. Syst.*, vol. 26, 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[22] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," *Proc. AAAI Conf. Artif. Intell.*, vol. 28, no. 1, Jun. 2014. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8870

[23] P. Veličković, G. Cucurull, et al., "Graph attention networks," 2018.

[24] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Corr*, 2017. [Online]. Available: http://arxiv.org/abs/1706.02216

[25] A. Vaswani, N. Shazeer, et al., "Attention is all you need," *Corr*, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[26] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," *Corr*, 2012. [Online]. Available: http://arxiv.org/abs/1205.2618

[27] A. Grover, and J. Leskovec, "Node2vec: scalable feature learning for networks," *Corr*, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[28] L. Cai, Z. Chen, et al., "Structural temporal graph neural networks for anomaly detection in dynamic graphs," in *Proc. 30th ACM Int. Conf. Inf. & Knowl. Manage.* in Cikm '21, Virtual Event, Queensland, Australia, 2021, p. 3747, doi: 10.1145/3459637.3481955. [Online]. Available: https://doi.org/10.1145/3459637.3481955

[29] M. Kapoor, J. Melton, M. Ridenhour, S. Krishnan, and T. Moyer, "Prov-gem: automated provenance analysis framework using graph embeddings," in *2021 20th IEEE Int. Conf. Mach. Learn. Appl. (Icmla)*, vol. 0, 2021, pp. 1720–1727, doi: 10.1109/ICMLA52953.2021.00273.

[30] U. Singer, H. Roitman, I. Guy, and K. Radinsky, "Tbdfs: temporal graph neural network leveraging dfs," 2022.

[31] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: a survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017, doi: 10.1109/TKDE.2017.2754499.

[32] Y. Zhang, W. Wang, et al., "Disconnected emerging knowledge graph oriented inductive link prediction," 2022.

[33] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," *Corr*, 2017. [Online]. Available: http://arxiv.org/abs/1706.05674