HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

ShadeWatcher

Recommendation-guided Cyber Threat Analysis using System Audit Records

Jamie Temple, 12.06.2023

Table of Contents

Chapter 1

Motivation

Problem: Modern system experience advanced persistent threats (APTs).

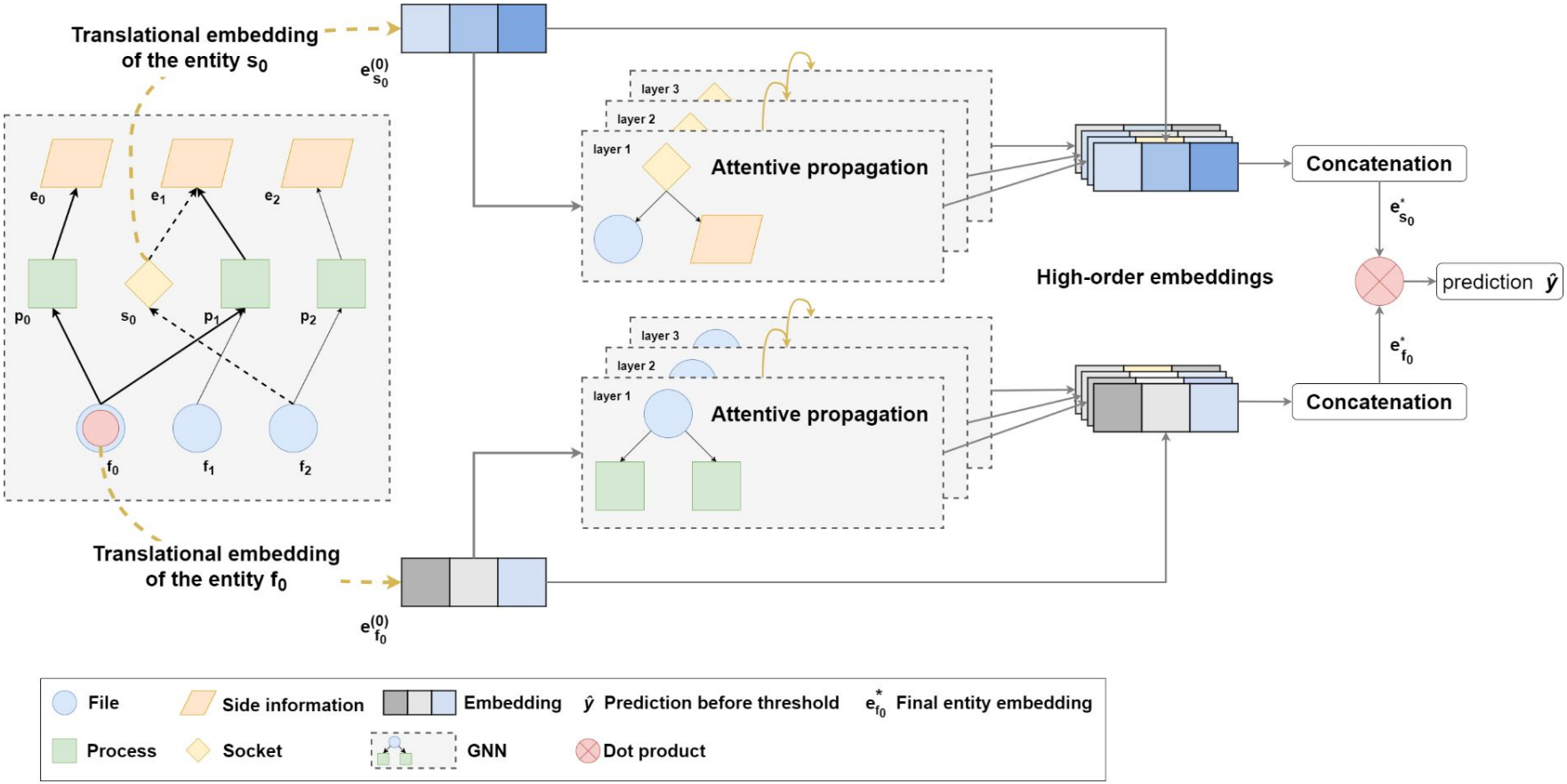
- Attackers developed sophisticated cyber attacks that are hard to detect.
- IT infrastructure scales performing many actions every day.
- Provenance data provides an option to analyse the system.
- However, existing techniques are not robust.
  - Specification-based (manual labour and expert dependent).
  - Statistics-based (high false-positives).
  - Learning-based (not descriptive and manual labour).

Solution: ShadeWatcher extracts semantic information to analyse systems entities behaviour, whilst providing attack indicators.

Approach

- ShadeWatcher [1] analyses interaction on a system-call level.
- Essential is to consider the neighbourhood to retrieve additional information.
- For better predictions, one combines context analysis with neighbourhood information.
- The recommendation provides a ranking of predictions regarding malicious interactions and entities in a system.

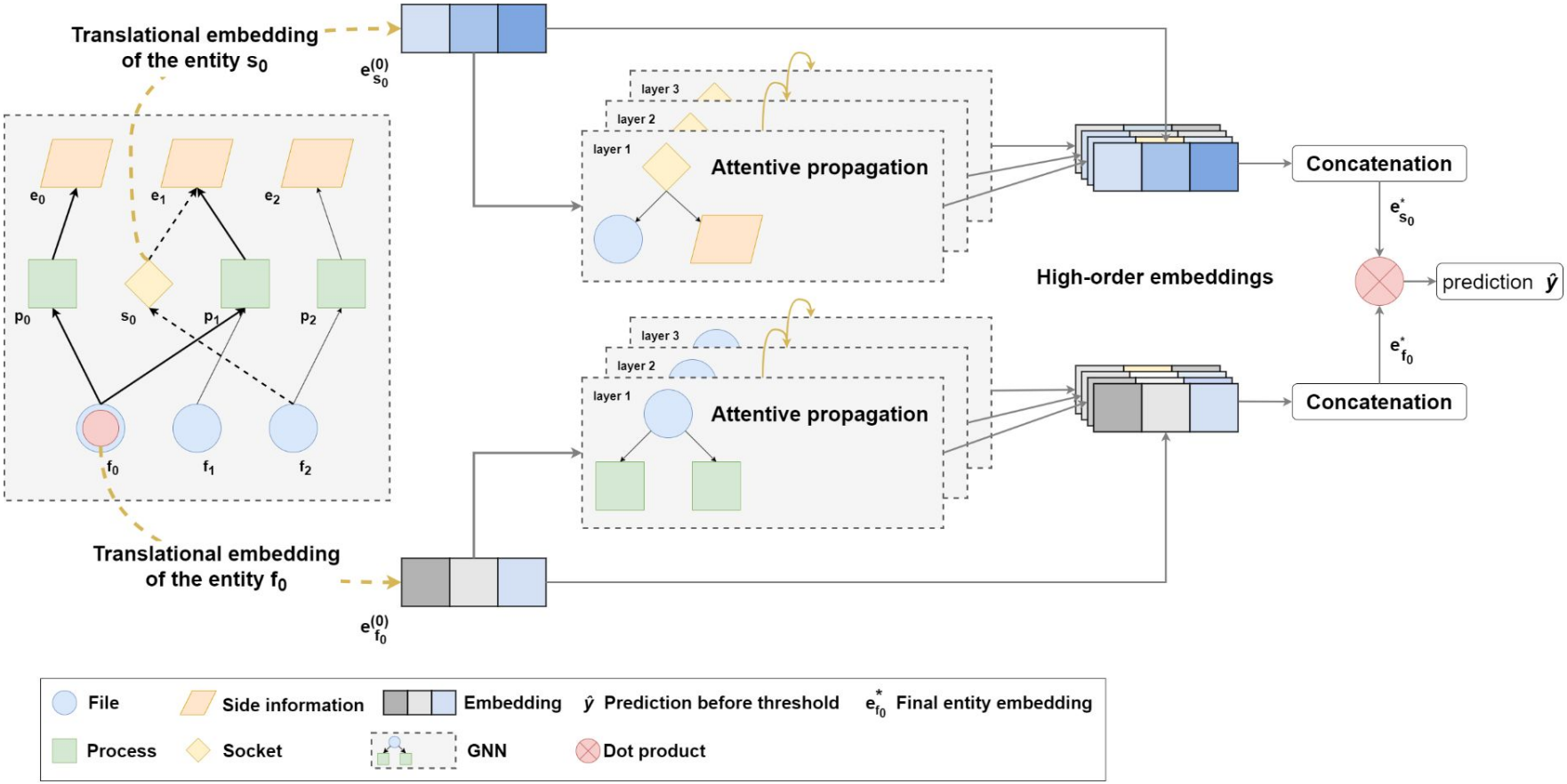
Overview



The figure shows the approach of ShadeWatcher with its components (Modified and adapted [1,11]).

Chapter 2

Knowledge graph construction



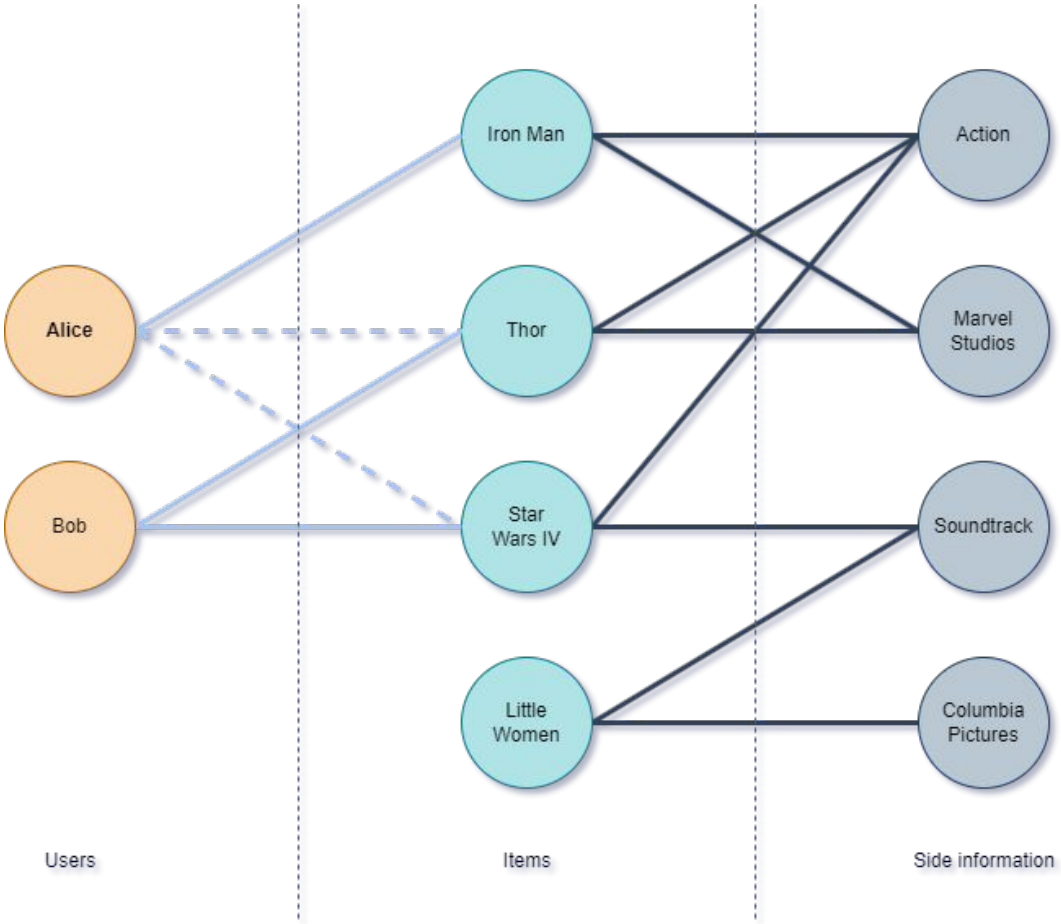
The figure shows the approach of ShadeWatcher with its components (Modified and adapted [1,11]).

Recommendation systems

- Entities like people interact with various items in a system, e.g. movies in a streaming service [6].
- Make recommendations based on these interactions that reflect the user's past interests.
- Interaction in recommendation scenarios can be modelled with a k-partite graph.
- Recommendation improves by incorporating side information.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$
$$\mathcal{E} = \{\{u, i\} : u \in \mathcal{U}, i \in \mathcal{I}\}$$
$$\mathcal{V} = \mathcal{U} \cup \mathcal{I} : \mathcal{U} \cap \mathcal{I} = \emptyset$$

Recommendation systems example



The figure shows a tripartite graph depicting the movie recommendation example. Modified and adapted from [1].

Provenance graph

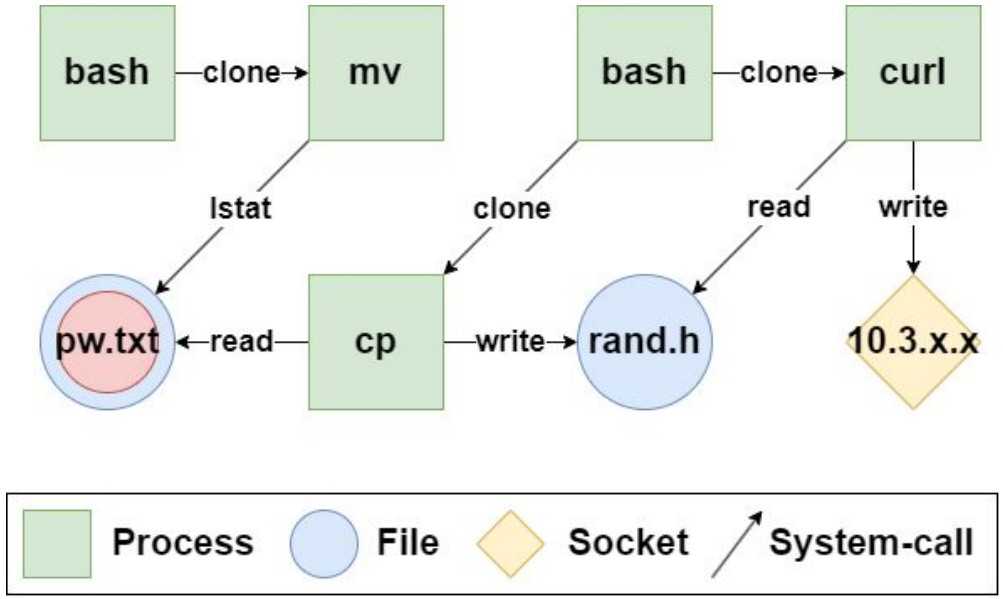
- It is possible to derive a provenance graph (PG) based on audit data.
- PG gives insight into the history of the system.
- We need audit data [4], e.g. at a system call level, to reconstruct the system's state.

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

Provenance graph example

```
{
    "@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",
            // ...
        }
    }
}
```

Audit data generated by auditd (Adapted from [2]).



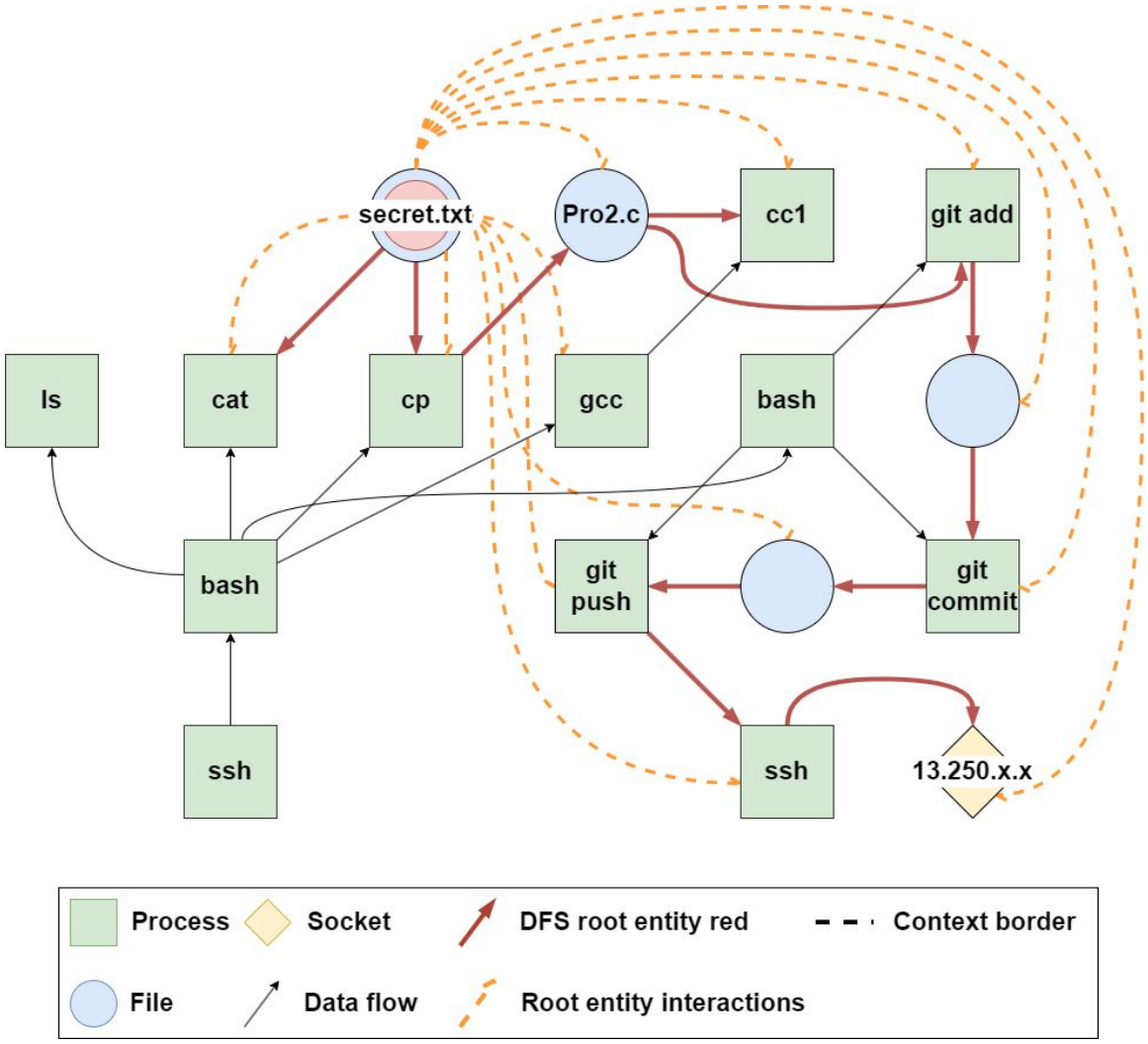
The figure displays a theoretical provenance graph (own illustration).

Entity context graph

- The problem is that the PG does not directly encode the side information [1].
- A context captures a collection of system entities and interactions representing system behaviour [3].
- ShadeWatcher uses the system entity's context to derive side information [1,3].
- One utilises depth-first search (DFS) to create subgraphs to capture context.

$$\mathcal{G}_C = (\mathcal{V}, \mathcal{E})$$
$$\mathcal{E} = \{(h, r_{\text{ht}}, t) : h, t \in \mathcal{V} : r_{\text{ht}} \in \{0, 1\}\}$$
$$\mathcal{V} = \mathcal{D} \cup \mathcal{S} : \mathcal{D} \cap \mathcal{S} = \emptyset$$
$$\mathcal{D} = \{file,\ socket,\ ...\}$$
$$\mathcal{S} = \{process,\ ...\}$$

Entity context graph example

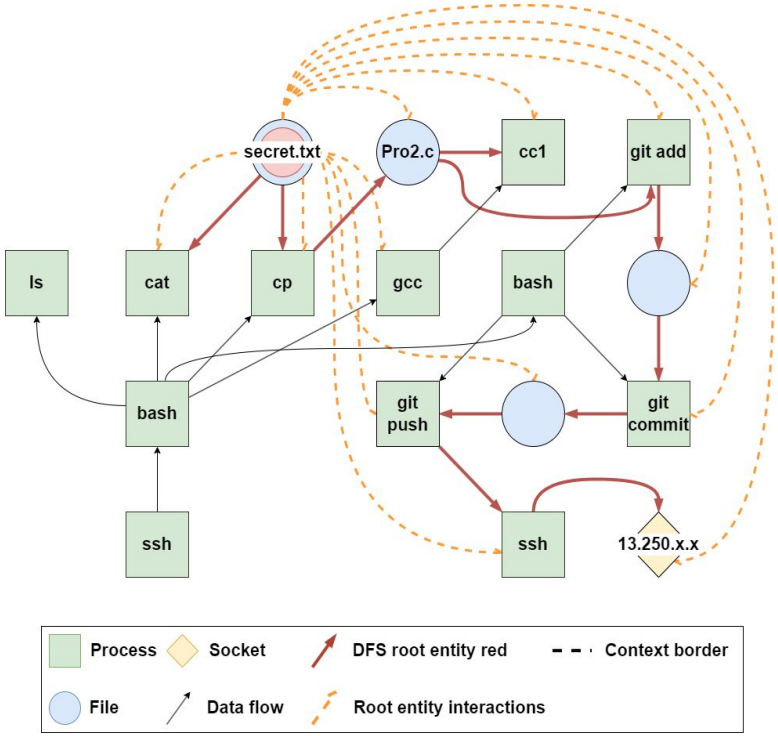


The figure depicts a knowledge graph with the context subgraphs (Adapted and modified from [1,3]).

Knowledge graph

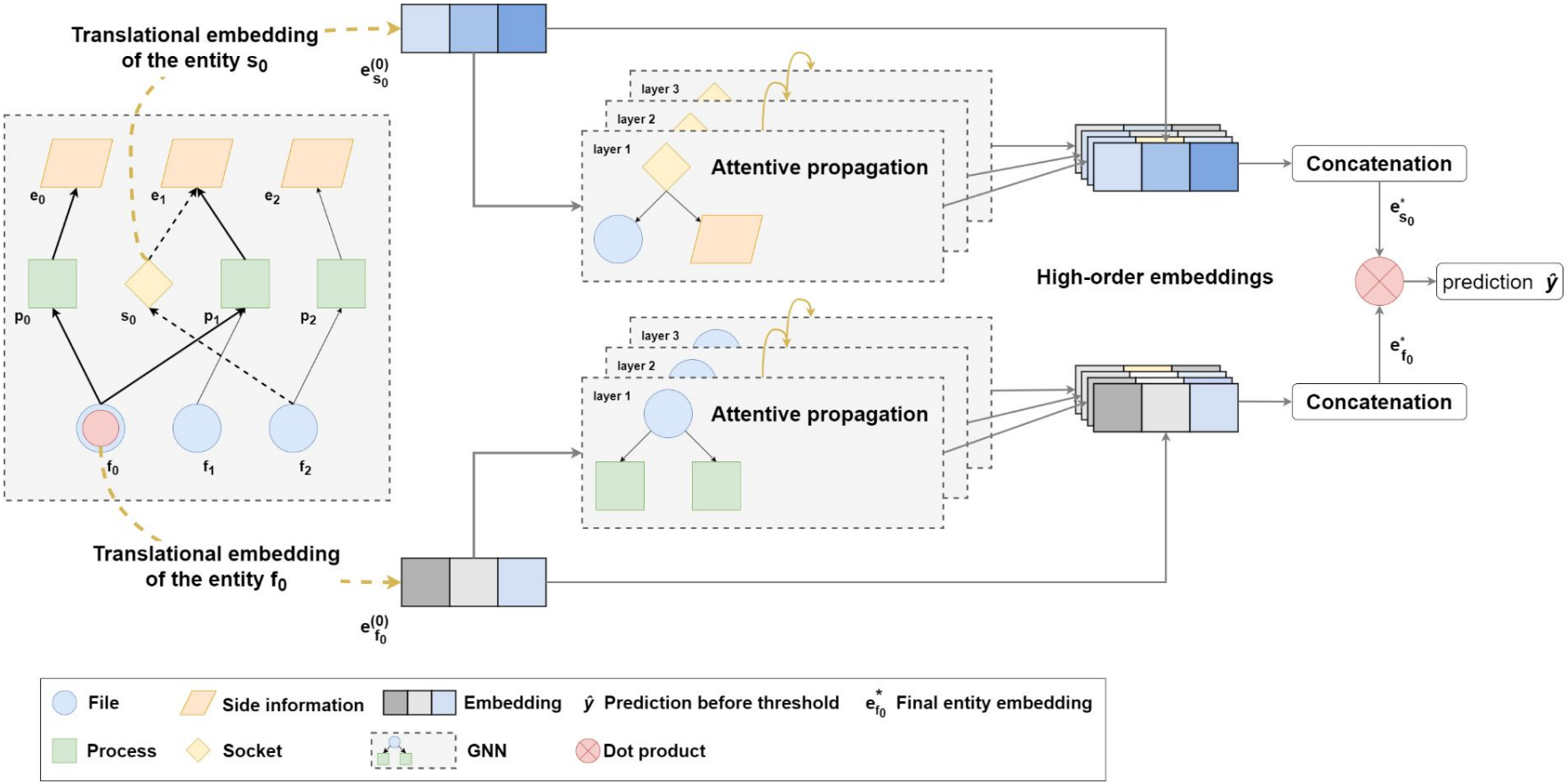
- A knowledge graph (KG) will be used as input.
- ShadeWatcher union the provenance and context graph to capture topological and behaviour information.

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



The figure depicts a knowledge graph with the context subgraphs (Adapted and modified from [1,3]).

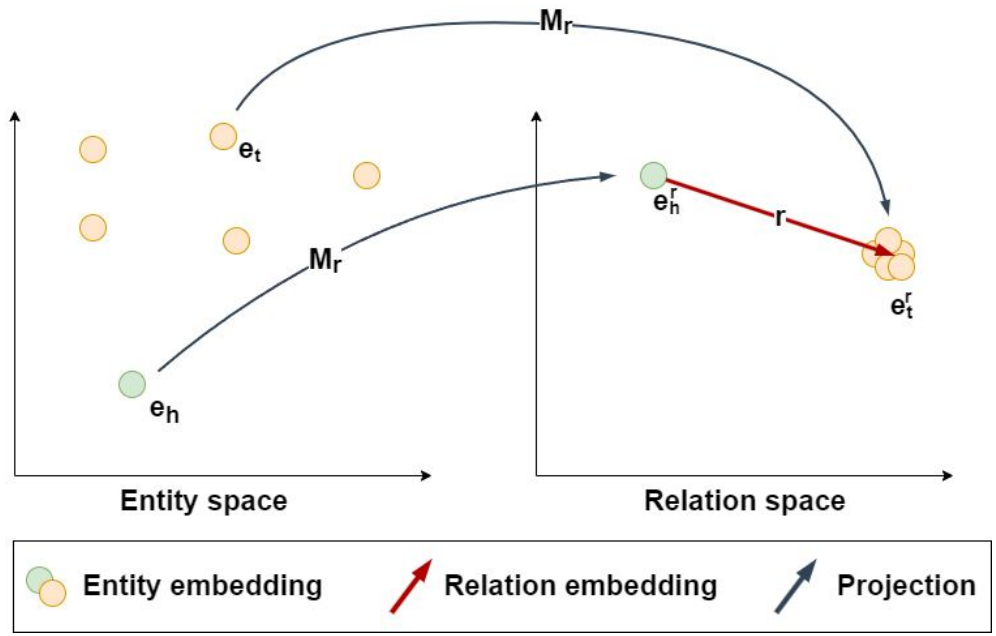
Translational embeddings



The figure shows the approach of ShadeWatcher with its components (Modified and adapted [1,11]).

TransR - Translational embeddings (1/2)

- TransR is a method to learn low-dimensional entity and relation embeddings [8].
- Improvement to predecessors TransE and TransH.



The figure shows a translational projection (Adapted from [8]).

TransR - Translational embeddings (2/2)

- Embeddings are n-dimensional vectors that aim to represent an entity and relation.
- TransR performs a projection using a relation-specific projection matrix (M).

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

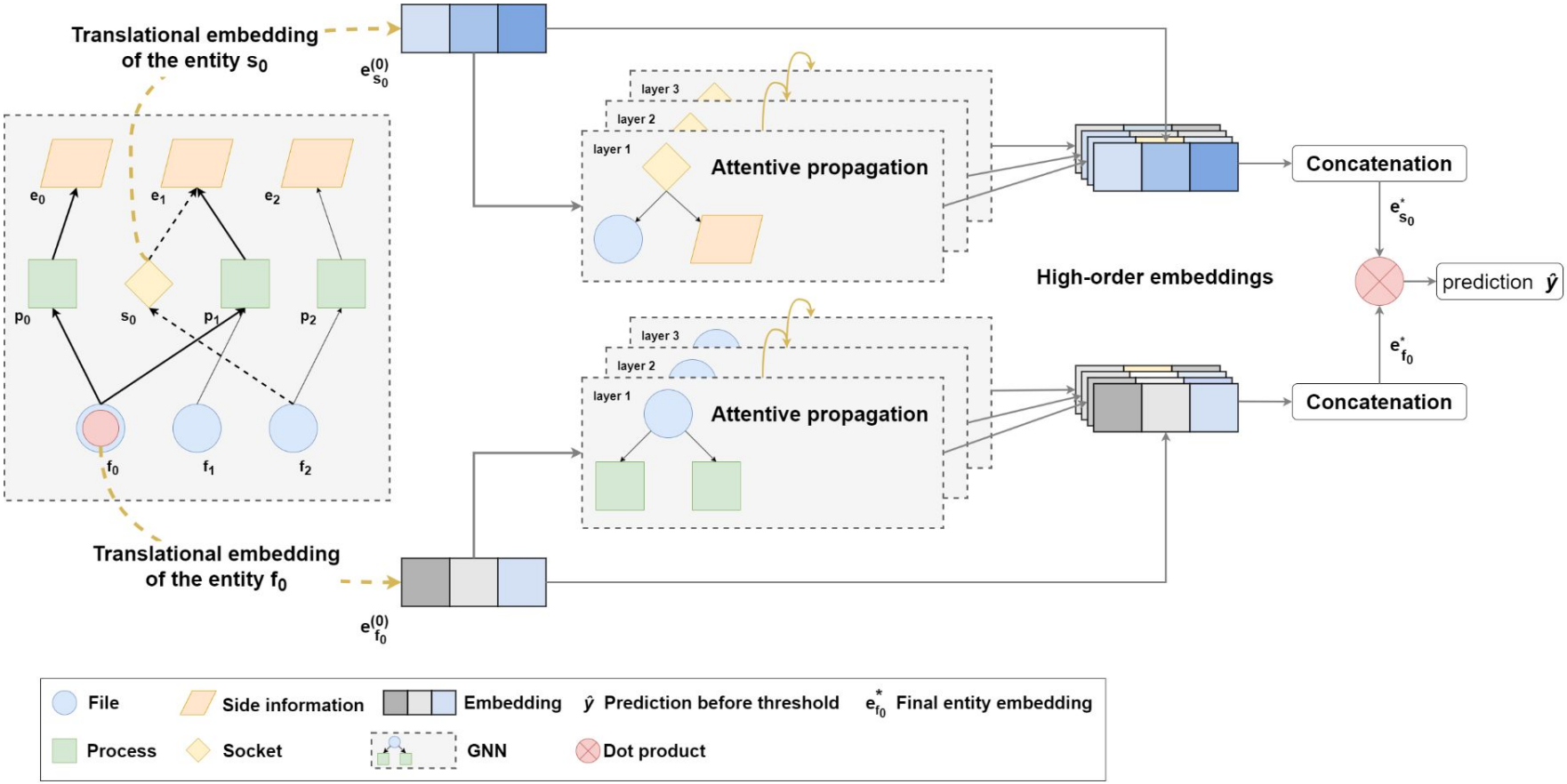
- The performance of the embeddings is measured with a score function.
- ShadeWatcher changed the score function to use the L2 Norm only.

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

- Leverage negative samples via corrupted triplets.
- TransR employs a margin-based pairwise ranking loss function.

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

High-order embeddings



The figure shows the approach of ShadeWatcher with its components (Modified and adapted [1,11]).

Graph neural network (1/4)

- Entity context is represented by high-order connections via a multi-hop path [1,3].
- ShadeWatcher employs graph neural network (GNN) [9] to propagated neighbour information.

$$z_h^{(l)} = g\left(z_h^{(l-1)}, z_{\mathcal{N}_h}^{(l-1)}\right)$$

- A layer updates the representation of an entity h, using its previous value and the neighbourhood.
- The GNN is initialised with the TransR embeddings.

Graph neural network (2/4)

- For the neighbourhood accumulation not all entities have the same importance.
- Add attention mechanism [1,11] to weight neighbour contribution.

$$z_{\mathcal{N}_h}^{(l-1)} = \sum_{t \in \mathcal{N}_h} \alpha(h,r,t) z_t^{(l-1)} : (h,r,t) \in \mathcal{G}_K$$

- ShadeWatcher adopted attention calculation [11].
- The coefficient utilises the TransR embeddings.
- Embeddings can change during training, or one can utilise pre-trained embeddings [2].

$$e(h,r,t) = \boldsymbol{e}_t^{r^\top} \tanh(\boldsymbol{e}_h^r + \boldsymbol{e}_r)$$

$$\alpha(h,r,t) = \text{softmax}(e(h,r,t)) = \frac{\exp(e(h,r,t))}{\sum_{t_h \in \mathcal{N}_h} \exp(e(h,r,t_h))}$$

Graph neural network (3/4)

- ShadeWatcher adopted an aggregation function used in KGAT [11].
- The aggregation follows the principles defined in GraphSAGE [10].

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

- ShadeWatcher concatenates the representations and applies a linear transformation.
- For enhanced expressiveness, the value undergoes a non-linear transformation.

Graph neural network (4/4)

- One can propagate an entity through the GNN and collect all intermediate embeddings.
- The final representation is a concatenation to preserve high-order information for all hops.

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

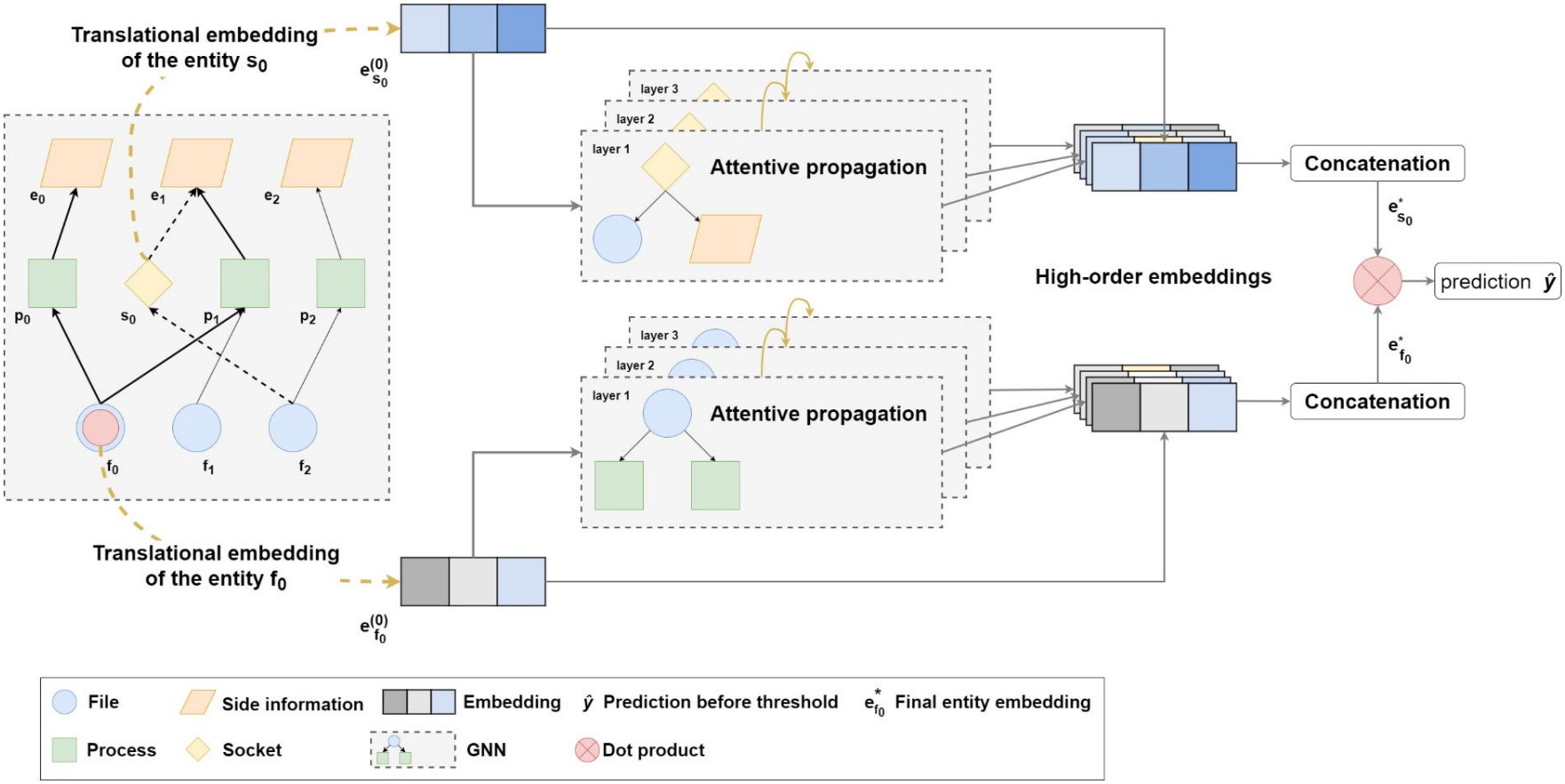
- ShadeWatcher uses the final entity embedding z_h^* the recommendation step.
- ShadeWatcher applies the dot product to get a prediction of how likely an entity will not interact (one can label it based on a pre-defined threshold).
- Maximise similarity for valid triplets and minimise it for corrupted triplets.

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

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

Chapter 3

Recommendation



The figure shows the approach of ShadeWatcher with its components (Modified and adapted [1,11]).

Recommendation

- After training, ShadeWatcher can receive system entities for analysis.
- An entity is passed through the GNN to get a representation used for the prediction.
- With that, one labels the entity based on a pre-defined threshold.

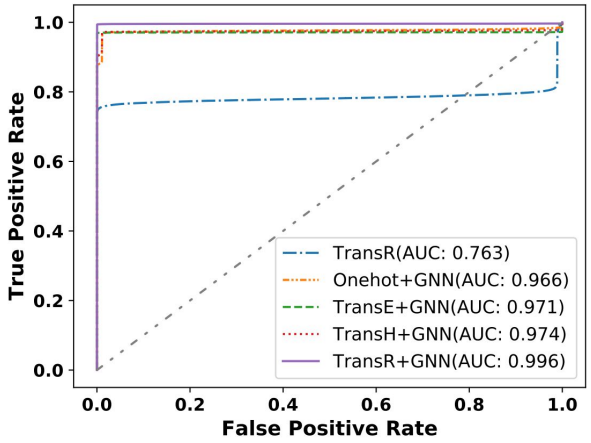
$$z_h^* = z_h^{(0)} \| ... \| z_h^{(L)} : \left\{ z_h^{(0)}, ..., z_h^{(L)} \right\} \qquad \hat{y}_{\mathrm{ht}} = z_h^{*\top} * z_t^*$$

- ShadeWatcher notifies analysts if malicious behaviour is detected.
- Analysts can then use the recommendation to check the threat manually.
- In case of false positives, ShadeWatcher is supposed to take the feedback into account.

Chapter 4

Discussion

- ShadeWatcher uses DFS to create subgraphs capturing high-order neighbourhood information — feasibility to incorporate this in a GNN to directly learn semantics.
- TransR is a transitive learning approach requiring retraining for unseen nodes, which becomes expensive for large amounts of data [12] — feasibility using inductive link predictors.
- One requires to check robustness towards data contamination (real world scenarios).
- An attacker can perform an adversarial attack on GNN by manipulating KG.



Thank you for your attention!

Bibliography

[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. (\url{https:// github.com/jun-zeng/ShadeWatcher})

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

[4] 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

[5] A. Bates, D. Tian, K. R. B. Butler, and T. Moyer, “Trustworthy wholesystem provenance for the linux kernel,” in Proc. 24th USENIX Conf. Secur. Symp. in Sec'15, Washington, D.C., 2015, p. 319.

[6] 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.

[7] J. Wu, X. Wang, et al., “Self-supervised graph learning for recommendation,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, Jul. 2021, doi: 10.1145/3404835.3462862. [Online]. Available: https://doi.org/10.1145%2F3404835.3462862

[8] Lin, Y., Liu, Z., Sun, M., Liu, Y. and Zhu, X. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. Proceedings of the AAAI Conference on Artificial Intelligence. 29, 1 (Feb. 2015). DOI:https://doi.org/10.1609/aaai.v29i1.9491.

Bibliography

[9] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in ICLR, 2017.

[10] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in NeurIPS, 2017.

[11] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, “Kgat: Knowledge graph attention network for recommendation,” in ACM KDD, 2019.

[12] Y. Zhang, W. Wang, et al., “Disconnected emerging knowledge graph oriented inductive link prediction,” 2022.