通过推荐研究实体交互来分析网络威胁 用TransR嵌入KG

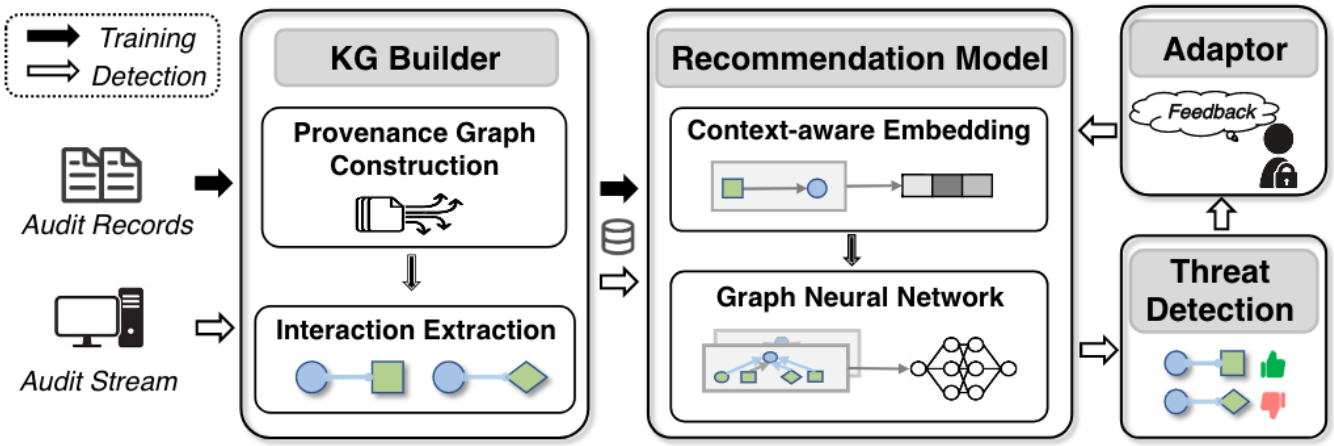


Fig. 2: Overview of SHADEWATCHER architecture.

分析网络威胁的根本挑战是在大量审计记录中进行全面但细粒度的推理。安全分析师不仅需要辨别某个审计记录是否是攻击的结果，还需要辨别它与恶意行为的关联。威胁因素通常会通过分析员认为可疑的非关联系统实体交互引发不必要的行为。

溯源图的直接链接没有捕获实体背后的潜在关系，在linux下考虑/proc/25/stat and /proc/27/stat（Linux在启动一个进程时，系统会在/proc下创建一个以PID命名的文件夹）属于不同进程，在溯源图中因为不是直接链接而是断开的，但他们都提供了有关进程的状态信息，在考虑上下文时候就可以反映出一定潜在信息。

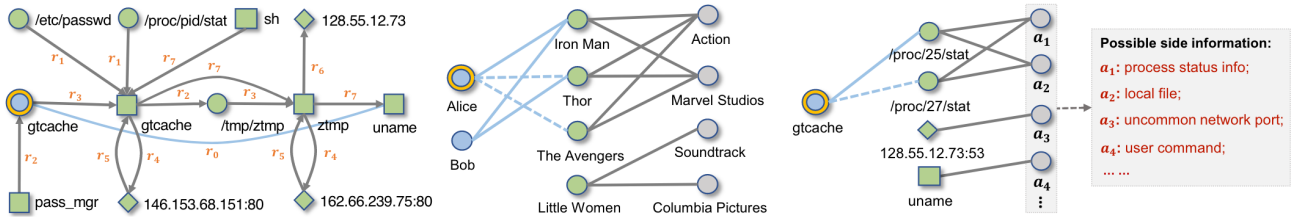
在推荐系统中有类似的任务，假设行为相似的用户会分享对物品的偏好，因此他们通过通过历史用户-物品交互找到相似的用户来理解用户的偏好。然而，用户和条目之间的直接连接，称为一级连接，不足以比较不同条目之间的语义相似性。研究人员进一步考虑物品的侧面信息，如电影类型，以捕获物品语义。边信息可以形成高阶连接，以链接在用户-项目交互中断开的类似项目。

于是目标是整合系统实体的边信息，以全面解释它们的交互。虽然这些辅助信息没有显式编码在溯源图中，但系统实体的语义可以从使用它们的上下文中揭示出来。所以可以利用上下文信息作为底层信息来配置系统实体。

通过将系统实体交互和实体上下文信息的网络安全概念映射到用户-商品交互和商品侧信息的推荐概念，我们可以将网络威胁检测制定为推荐任务。语义相似的系统实体会在交互中表现出相似的偏好。例如，敏感文件(例如/etc/passwd和/etc/shadow)通常不与公网交互，否则表示数据泄露。于是威胁检测可以进一步指定为预测系统实体不“偏好”其交互实体的可能性。但与研究用户偏好的典型推荐场景相比，威胁检测的目标是系统实体不太可能喜欢的交互，因为这种交互通常是强大的攻击指标。

shadewatcher通过TRANSR嵌入KG 用一个base GNN的推荐模型 递归传播来自邻居实体的信息构成高层连接 同时可以动态更新模型和检测到的恶意信号 基于推荐系统的应用其优点包括
1 不将历史频率作为一个度量来估计怀疑程度，而是推断系统实体的内在语义来发现异常的相互作用
2 shadewatcherr提供了一个端到端的解决方案，可以在不事先知道攻击的情况下检测威胁
3 shadewatcher生成细粒度的检测信号，突出攻击的关键指标

分析



(a) System Provenance Graph (b) Movie Recommendation (c) High-order Information in Audit Records

Fig. 1: (a) A simplified provenance graph of Extension Backdoor, where r_1, r_2, r_3, r_4, r_5, r_6, and r_7 denote *read*, *write*, *load*, *send*, *recv*, *connect*, and *exec*, and r_0 reflects system entity interaction. (b) An example of recommending movies for *Alice*, where blue, green, and gray nodes stand for users, items, and side information. Solid/dashed blue edges show historical/recommended user-item interactions. (c) An illustration of system entity interactions with side information to form high-order connectivities.

溯源图中的节点表示系统实体，矩形、椭圆形和菱形分别表示进程、文件和套接字。灰色边表示面向信息流方向的系统调用。使用两个单独的节点来表示具有不同进程ID的/proc/pid/stat和具有不同网络端口的128.55.12.73。

在b所示的一个电影推荐中，alice是推荐的目标用户，对于用户与物品交互，Alice→Iron Man和bob→Iron Man，表示Alice和bob之间的行为相似性。最近的工作利用项目边信息(如电影类型和工作室)来形成高阶连接，将语义相似的项目链接起来。例如，钢铁侠→行动→复仇者联盟→钢铁侠→漫威影业→复仇者联盟，这表明爱丽丝可能更喜欢复仇者联盟，因为复仇者联盟的侧面信息与钢铁侠的侧面信息相同。

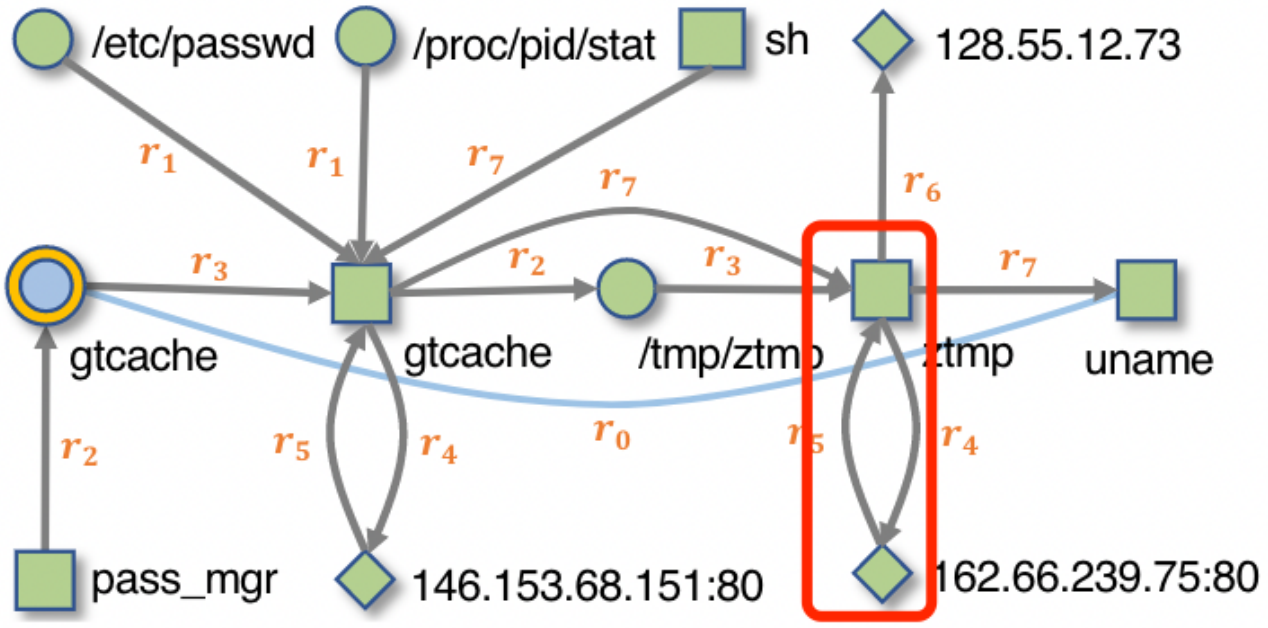
类似可以考虑到使用高阶语义连接来理解系统实体的交互，但是边信息没有显式编码在原始溯源图里，类似于waston使用context推断系统实体的语义，shadewatch引入上下文信息辅助分析知识。于是系统实体直接的因果交互类似于推荐系统的user-item交互，context作为辅助形成高阶连接（高阶连通性捕获为与邻接实体相关的多跳路径）。

问题定义

溯源图

审计记录是一组描述系统执行历史的日志条目。每个记录代表系统调用粒度上的一个活动。用三元组（src，rel，dst）描述。

$$src, dst \in epsilon = \{process, file, socket\}, rel \in R = \{clone, write \dots\}$$

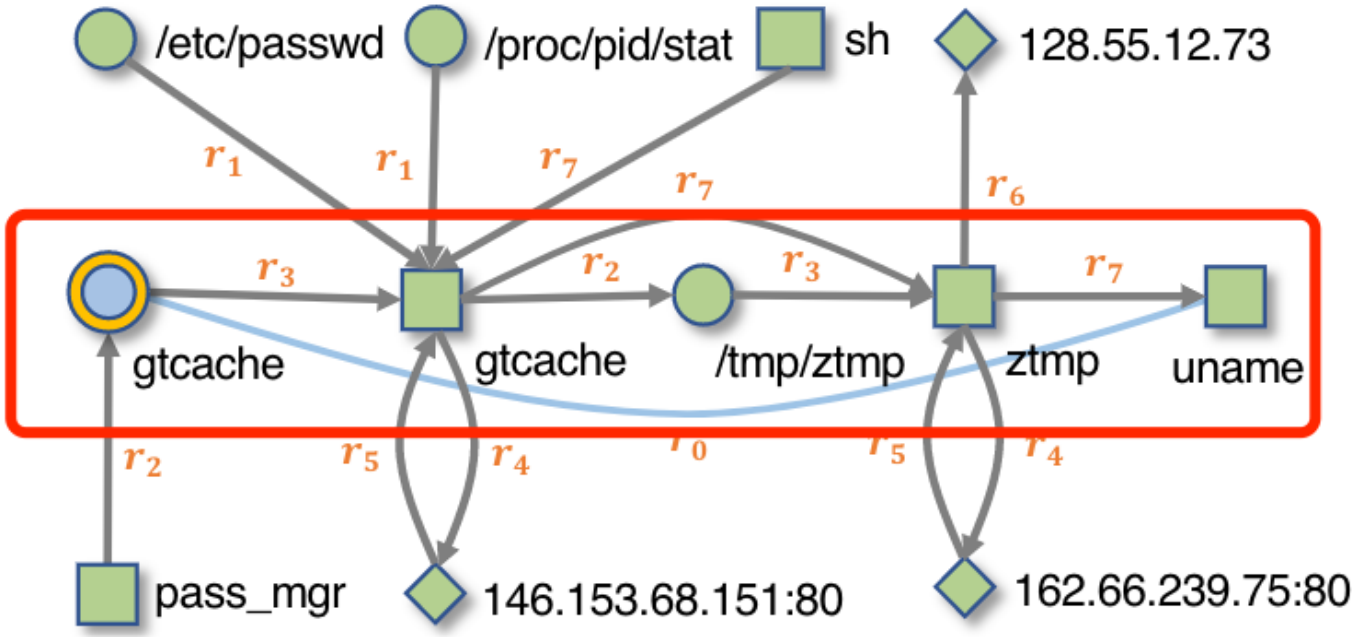


（ztmp，connect，162.66.239.75:80）

将审计日志以图的形式组织得到溯源图，溯源图是一个有向无环图，
$G_p = \{(src, rel, dst) | src, dst \in \epsilon, rel \in R\}$

实体交互

溯源图中的连接反映实体的直接交互。



连接gtcache和uname直接一系列的边代表了两个实体的交互，在推荐中user-item通常以二部图呈现。

定义交互二部图$G_B = \{(e, y_{ee'}, e' | e, e' \in \epsilon\}$，$y_{ee'}$为1时表示两个实体有交互。系统实体交互不仅表示显式数据依赖关系，还表示隐式控制依赖关系。比如gtcache操作ztmp执行uname。

连接次序

知识图谱将系统实体上下文和交互编码成一个关系图，可以将G_B中的交互转换成三元组(e,interact,e')，interact表示系统调用外的附加关系，将$G_p$$G_B$重新组织为KG,$G_k = \{(h.r.t)|h,t \in \epsilon, r \in R \cup interact\}$

shadewatcher

构建KG

将审计日志转为溯源图，使用二部图提取系统实体交互，结合溯源图和二部图生成KG，使用KG中不同次序的连接对系统实体的交互、识别系统执行中的异常的可能性建模。SHADEWATCHER的功能可以分为训练和检测。对于基于性能的检测，使用无攻击的审计记录来训练推荐模型。对于新到达的审计流，SHADEWATCHER提取系统实体交互，并将它们提供给从培训阶段获得的推荐模型。如果交互是恶意的概率大于预定义的阈值，那么shadewatcher会将其作为潜在威胁进行检测。

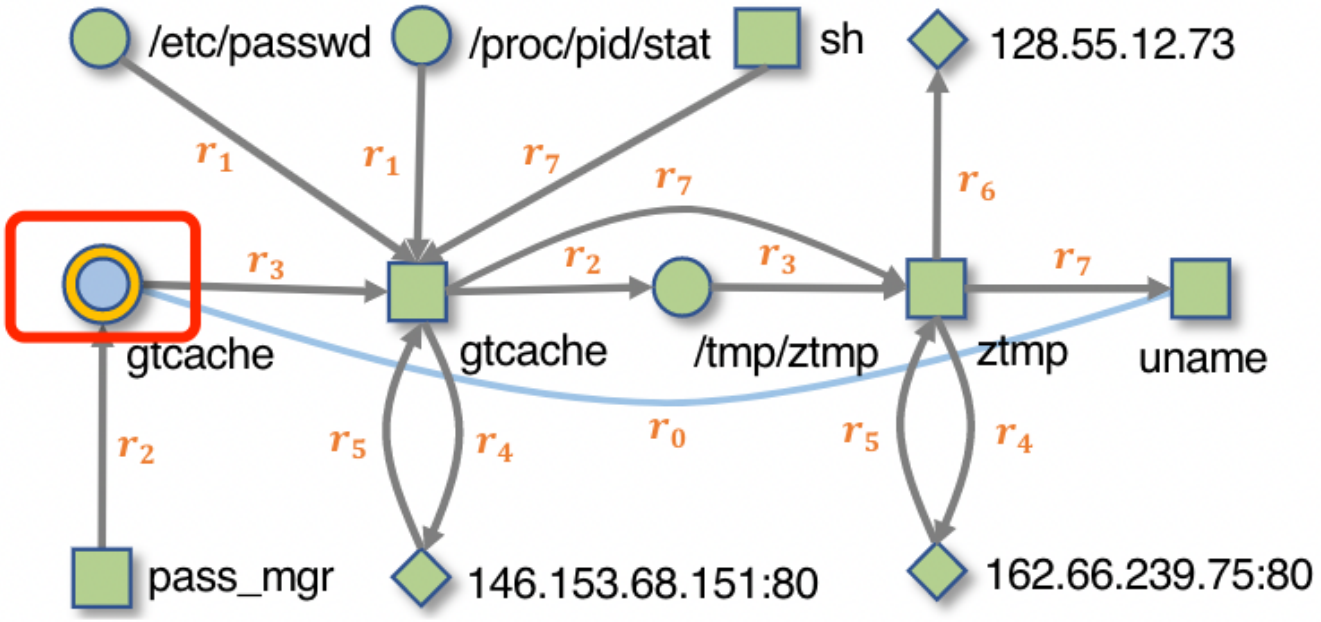
溯源图中系统实体属性

Process属性：pid ppid（父进程的pid） exe（执行程序的绝对路径） args

File属性：name path version

Socket属性：ip port

交互抽取



两两配对溯源图中的实体是不可能的，而从审计日志中提取行为可以有效减少工作量，每个行为都被表示为一个包含一系列基于root数据对象（file等）因果记录序列的溯源子图。比如上图中的gtcache即为root data object。

在抽取系统实体交互之前，将一个溯源图划分为多个子图，每个子图描述一个行为。首先识别溯源图中所有的数据对象，然后对每一个data object执行前向深度优先搜索（同waston）来提取子图，如果一个子图是另一个子图的子集则合并。

比如给定两个交互，gtcache→uname、gtcache→162.66.239.75:53，观测到一个可执行文件试图收集系统配置和扫描网络服务。shadewatcher将行为中的交互转换为一个二部图，其中两个不相交的节点集是数据对象和系统实体，连接两个集的边反映了交互。

溯源图和二部图都是实体-关系-实体的三元组集合，通过实体对其后合并存储到数据库中（PostgreSQL）

生成推荐模型

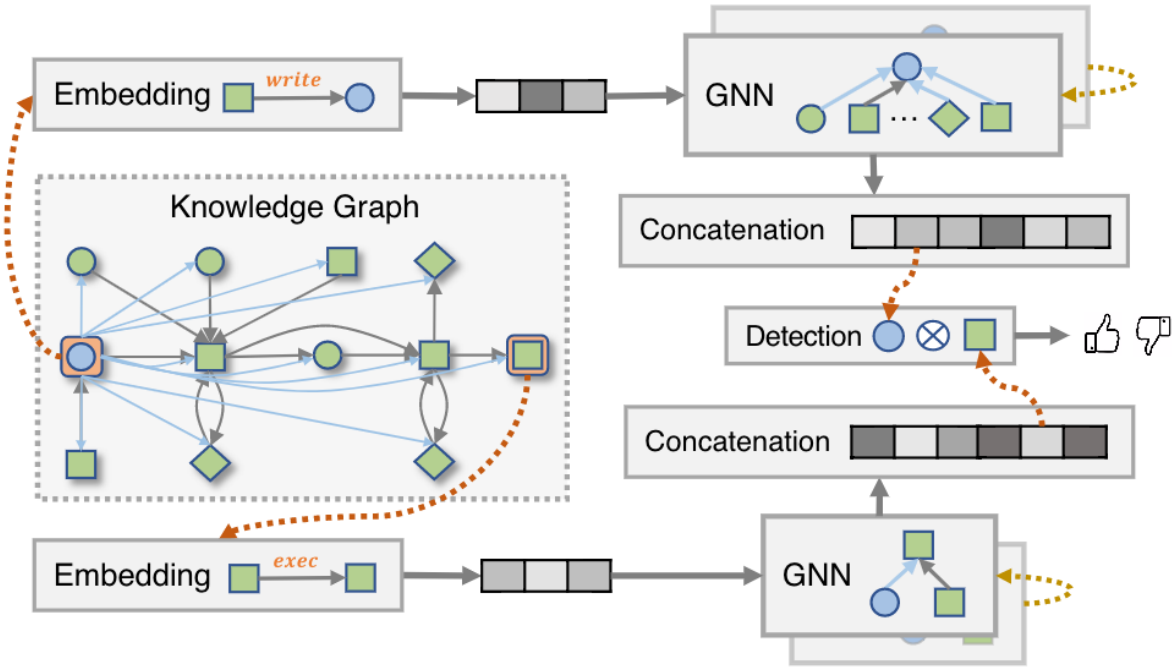


Fig. 3: An illustration of SHADEWATCHER's recommendation.

主要分为三个部分

1 对一阶信息建模 通过上下文对系统实体进行嵌入

KG中直接的连接表示了系统实体之间行为和语义的相似性，使用TranE做嵌入会导致1-N、N-1、N-N问题（单个关联类可能对应多个实体），进而影响网络威胁分析，所以使用TransR来进行embedding。TransR可以学习具有不同关系条件的系统实体的单独表示，它能够在不同的关系上下文中为同一实体分配不同的语义。它将实体映射到一个向量空间，关系映射到另一个向量空间中。$e_h, e_T \in \mathbb{R}^d, e_r \in \mathbb{R}^k$。对每一个关系，TransR指定了一个投影矩阵$W_r \in \mathbb{R}^{d \times k}$将系统实体从d维实体空间转换到k维关系空间中。

$f(h,r,t) = ||e_h^t + e_r - e_t^r||$，$||.|$为L1范式，这个值越小则这个三元组越可能在KG中被观测到。|

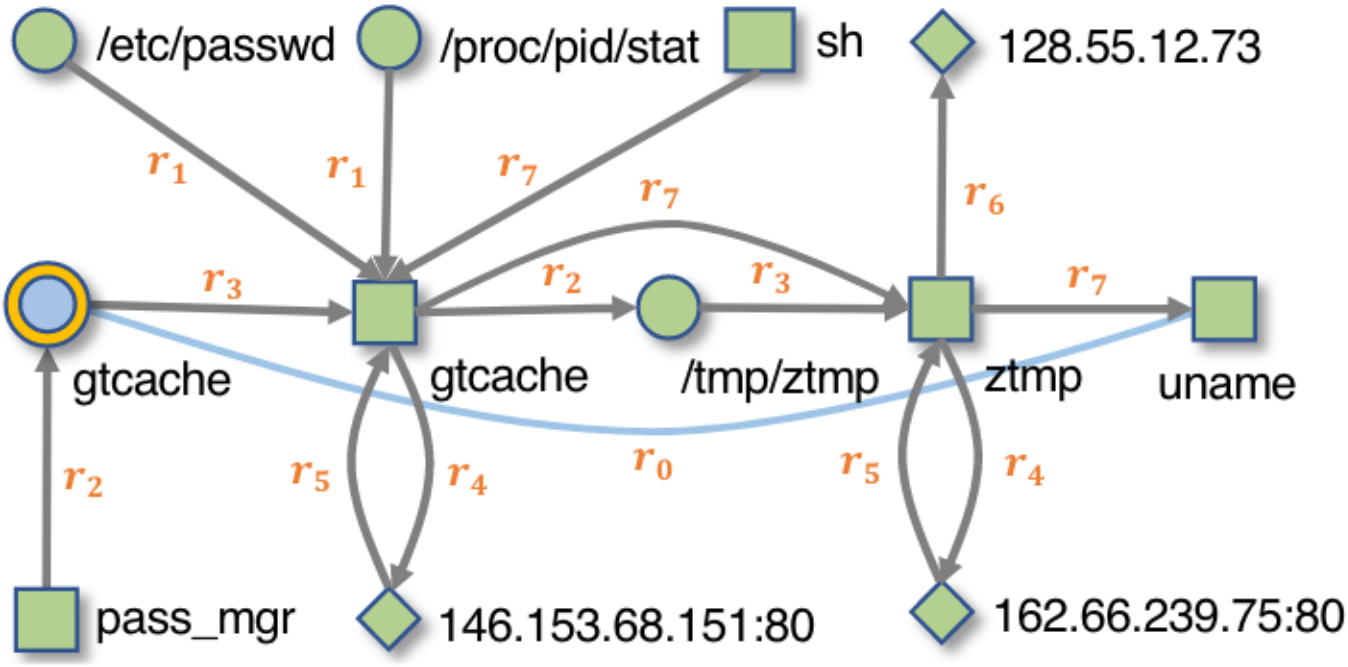
为了让可在KG中观测到的f小于未能观测到的f，优化目标为

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

γ为超参数

通过随机实体替换来构造未被观测到的实体元组。

2 对高阶信息建模 通过递归传播来自多跳邻居实体的信息来更新系统实体表示



考虑一个两跳路径

$$/proc/25/stat \xrightarrow{r_0} gtcache \xrightarrow{r_0} /proc/27/stat$$

，它显示出了/proc/25/stat和/proc/25/stat之间的相似性，因为都与gtcache交互了。

$$/etc/passwd \xrightarrow{r_1} gtcache \xrightarrow{r_4} 146.153.68.151:80$$

一个多跳路径反映了敏感的用户信息如何在企业内部传播，对高阶连接建模可以通过揭示系统实体关系来检测潜在威胁，因此使用GNN传播聚合邻居信息。

$z_h^{(l)} = g(z_h^{(l-1)}, z_{N_h}^{(l)})$, 其中$z_h^{(l)} \in \mathbb{R}^{d_l}$，表示为l层h的dl维向量表示，$z_h^{(l-1)}$是上一层，第0层输入为TransR得到的embedding向量，N_h是h的一阶邻居，于是$z_{N_h}^{(l)} \in \mathbb{R}^{l-1}$记录了来自h的(l-1)阶邻居信息。直觉上不同实体的贡献不同，所以使用attention加权。

$$\mathbf{z}_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \alpha(h,r,t)\mathbf{z}_t^{(l-1)}$$

其中的$\alpha(h,r,t)$描述了在特定关系r下有多少信息从t传播至h，定义为

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

e_t^r, e_h^r, e_r为TransR得到的embedding，聚合函数g为

$$g(\mathbf{z}_h^{(l-1)}, \mathbf{z}_{\mathcal{N}_h}^{(l-1)}) = \text{LeakyReLU}((\mathbf{z}_h^{(l-1)} || \mathbf{z}_{\mathcal{N}_h}^{(l-1)})\mathbf{W}^{(l)})$$

||.||表示向量拼接，$W^{(l)} \in \mathbb{R}^{2d^{(l-1)} \times d^{(l)}}$是一个转换矩阵，即为学习的参数。最后得到的$z_h^l$就含有了l跳节点的信息。

3 学习检测威胁 通过两个系统实体的embedding预测是对抗交互概率

上一步得到了实体h的一系列表示，$\{z_h^{(0)}, z_h^{(1)}, z_h^{(2)}, \dots z_h^{(L)}\}$，在KG中编码了不同顺序的信息，拼接他们得到最终表示。

$$\mathbf{z}_h^* = \mathbf{z}^{(0)} || \cdots || \mathbf{z}_h^{(L)}$$

给定交互(h,interact,t)，使用内积表示两个实体交互的可能性，如果大于预定义的阈值则标记为潜在威胁。

$$\hat{y}_{ht} = \mathbf{z}_h^{*\top} \mathbf{z}_t^*.$$

GNN的优化目标为

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

基于正常审计记录基础上的KG中观察到的交互被视为负面(良性)实例;同时，随机抽取KG中未观察到的交互作用作为正(潜在的恶意)实例。但是抽样的并不一定反映网络威胁。结合一阶建模和高阶建模定义端到端的损失函数。

$$\mathcal{L} = \mathcal{L}_{first} + \mathcal{L}_{higher} + \lambda \|\Theta\|$$

更新模型

可以与人工交互，若一个交互被检测为恶意但被人工验证为错误假警报，系统将该交互作为一个新的负例重新训练模型。

参数设置

embedding 32

GNN layers 2 隐藏层维度 32和16

阈值 -0.5