THREATRACE： Detecting and Tracing Host-based Threats in Node Level Through Provenance Graph Learning

基于溯源图表示学习的节点级别的主机威胁检测和追踪。

> 摘要：基于主机的威胁例如程序攻击、恶意软件植入、APT攻击等，被网络攻击者们广泛使用。最近的研究提出利用溯源数据的丰富的上下文信息来检测单台主机内部的威胁。溯源数据是一个由系统审计日志构建的有向无环图，溯源图中的节点表示系统实体，例如进程或文件、溯源图中的边表示系统调用，即系统实体之间的交互关系，表示了数据流和控制流的传输方向。然而，过往的研究，普遍在于从整个溯源图中提取特征，对在图中仅占一小部分的跟威胁相关的实体的检测并不敏感，因此导致捕获这些隐蔽威胁的性能较弱。
>
> **【溯源图中恶意节点或威胁节点数量很少，图级别的特征难以检测出这些节点级别的异常或威胁。】**
>
> 本文提出了一个基于异常的检测模型，可以在***不需要先验知识或模式***的情况下，在系统实体（进程、文件）层面检测基于主机的网络攻击威胁。本文对GraphSAGE模型进行改进，学习溯源图中良性节点的表示。此外，本文设计的系统是一个实时检测系统。

1 introduction

攻击者倾向于在大型企业或政府部门的重要主机上开展入侵活动，通常利用0-day漏洞来进行隐蔽的和持续的入侵。最近的研究提出利用溯源数据的丰富的上下文信息来进行主机级别的入侵检测，相较于原始的日志数据，溯源数据包含更加丰富的语义信息，对于区分良性活动和长期的威胁行为更加有用。

1.基于图的异常检测方法在检测网络攻击上存在局限性：隐蔽性较高的攻击行为所展现出的行为模式与正常系统行为非常相似、此外即使检测出异常也无法准确定位到具体是哪些系统实体节点，也就无法展开追踪。

2.基于误用的方法，预先定义了一些攻击者可能使用的攻击模式规则，然后使用这些模式来对异常行为进行匹配。这类基于规则引擎的检测方法在检测简单的、常见的攻击上是有用的，但是在检测诸如0-day漏洞或apt攻击这类未知攻击上仍然缺乏能力。

本文模型将溯源图作为输入，改进GraphSAGE模型，学习溯源图中每个节点所扮演的角色包含的特征用以节点级别的威胁检测。

使用GraphSAGE进行威胁检测所存在的挑战：

- 如何在无先验攻击知识下对模型进行训练?
- 如何解决数据不平衡问题?

contributions

- 溯源图中的节点级的威胁检测：本文提出了一个新的基于GraphSAGE模型的多模型(Multi-model)框架，实现溯源图的***节点(node-level)级***的异常检测；
- High Performance and Novel Capability : 本文在三个公开数据集上对THREATRACE系统进行实验验证，并在检测性能上与三个SOTA方法进行了对比；此外，本文还进行实验，评估了THREATRACE系统在单台主机内的***威胁溯源(threats tracing)能力***，结果表明本文系统能够成功检测和追踪溯源图中的异常元素；

2 related work

1.基于溯源的威胁检测 [provenance-based threats detection]

- 基于误用(misuse-based)的检测方法：预先定义一系列规则，转换为规则匹配的问题；
- 基于异常的检测方法：让模型学习正常行为模式，进而检测与正常行为具有很大偏差的异常模式；

2.异常追踪 [Anomaly tracing]

相较于单纯发出警报，对于异常情况的溯源和追踪是更加重要的。

- 基于溯源图的方法不能定位遭受攻击或出现异常的具体位置；
- 基于规则的方法则需要先验的异常检测模式的知识来进行匹配和定位；
- 攻击路径回溯存在依赖爆炸问题；
- 学习攻击行为序列的共现模式；（对数据集要求很高）

3 background and motivation

3.1 溯源数据

system audit log data directed acyclic graph rich contextual information

3.2 GraphSAGE模型

GraphSAGE是一种归纳式的(Inductive)GNN方法，通过邻居采样和信息聚合，学习图中的节点级的特征，并且归纳式的特点使得其能够有效处理不断变化的流式特点的图结构。

- Transductive：GCN、GAT
- Inductive：GraphSAGE

3.3 现有工作的一些局限性：

1. 规则：制定攻击检测的规则集较为困难；因为系统正常运行过程中，攻击行为/攻击事件所占比例是非常小的，而且ATT&CK中许多攻击行为在实际中恶意的行为也是占少数的。因此，过于宏观的规则会导致较高的误报率，而过于精细的规则，则无法检测出一些新的攻击方式(如zero-day攻击)；【规则的细粒度、误报率和检测范围的能力难以平衡】
2. 隐蔽的攻击(stealthy attacker)：[遭受攻击时的系统溯源图]和[正常运行时的系统溯源图]整体上可能是非常相似的，且异常节点占比例非常小，因此基于图级别和路径级别的检测方法的检测受到限制；
3. 异常追踪(Anomaly tracing)：大多数检测方法只能给出异常警告，仅仅标志该溯源图是否为异常，而不能给出是图中的哪些系统实体(节点)为异常；
4. 溯源数据的扩展：Holmes和streamspot方法将溯源图存储在内存中，导致这些方法缺乏在长期运行的真实系统中应用的扩展性；

3.4 本文方法的Intuition

本文方法的主要思想在于：溯源图中的异常节点相较于正常良性节点，其所呈现出来的特征（语义或结构）总是会存在区别。因此，方法的关键就在于检测这样的特征，尽可能扩大正常节点和异常节点的特征表示之间的差距(区分度)，从而可以通过分类或聚类的方法实现检测。

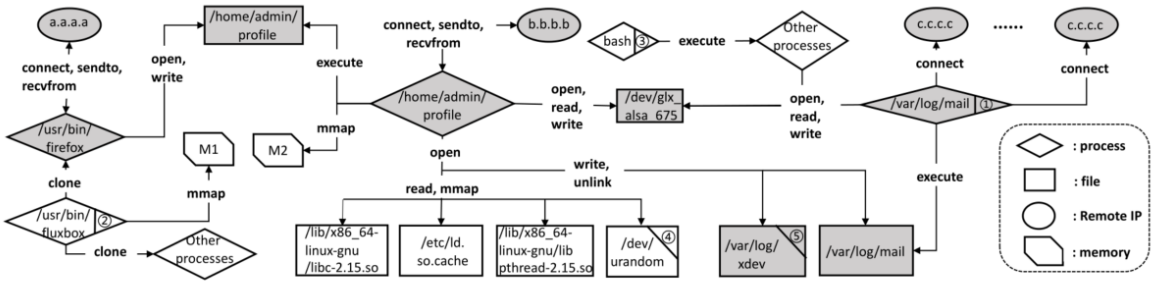


Fig. 1: A toy example of a provenance graph, which contains host-based intrusion behaviour.

上图中灰色的节点，表示该节点与攻击活动相关。

4 Threat Model

本文主要关注由于入侵活动所导致的溯源图中异常节点的检测和追踪，本文假设攻击活动存在如下特点：

- 隐蔽性(Stealthy)：攻击者尝试去隐藏他们的恶意行为；
- 持续性(Persistent)：攻击往往会持续较长时间；
- 0-day漏洞的频繁使用(Frequent use of zero-day exploits)：攻击者往往会使用0-day漏洞来攻击系统，因此，假设训练过程中并不知道任何已知的攻击模式；
- 溯源图中包含攻击模式(Has attacks patterns in the provenance graph)：攻击者的行为一定会在溯源图中留下一些攻击模式，这些攻击模式将会使得一个攻击者结点的局部结构相较于其同样标签的良性结点的结构呈现明显区别。[例如一个进程结点此前从未与文件结点有过交互，某时刻突然开始读取或写入文件，那么该进程则有可能是被攻击者所控制]。需要注意的是，由于溯源图的细粒度的限制，一些威胁或攻击情况不在本文讨论范围之内。

本文假设了以下两方面内容的正确性：

1. 溯源数据的收集及溯源图的构建；【数据要有保障】
2. GraphSAGE模型对图结构数据的有效的表示学习；【表示学习模型要有保障】

5 Design

如下图所示为THREATRACE系统的整体结构设计：

> 1. First, learn the different classes of benign nodes in a provenance graph without abnormal data;
> 2. detect abnormal nodes based on the deviation from the predicted node type and its actual type;

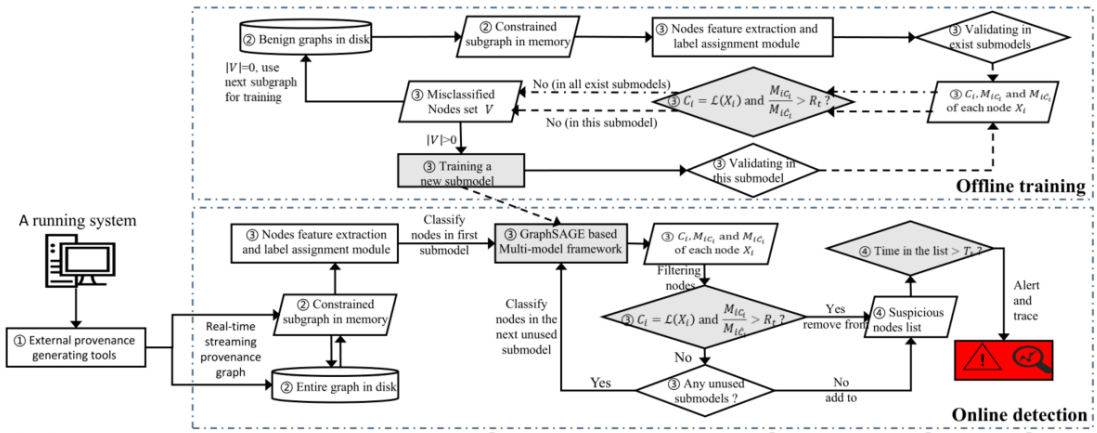


Fig. 2: The overview of components in THREATRACE. ①: Data Provenance Generator; ②: Data Storage; ③: Model; ④: Alert and Trace.

A. Data Provenance Generator

使用Camflow工具按照时间顺序构建溯源图。

B. Data Storage

将收集得到的溯源数据存储在磁盘中，仅在内存中存储active nodes，related nodes，以及这些节点之间的边。

- active nodes：用于训练的节点、或用于检测的节点；
- related nodes：active nodes的两跳内的邻居节点；

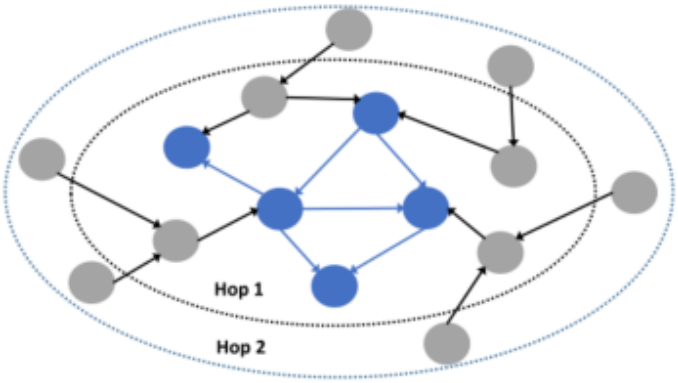


Fig. 3: An example of the subgraph maintaining in memory. Nodes in blue denote the *active* nodes, and the gray nodes denote the *related* nodes.

如上图所示，蓝色节点为active节点，表示用于训练GraphSAGE模型的节点，灰色节点为蓝色节点的2-hop邻居节点；

C. Model

特征获取 [Feature Extraction]

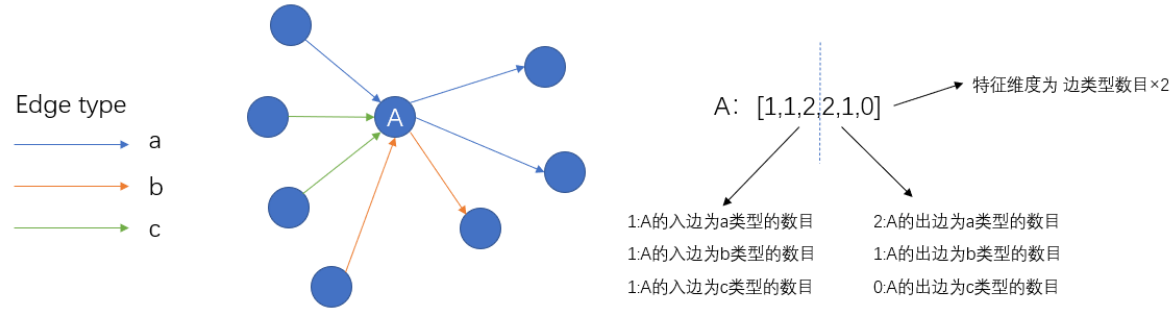
传统的异常检测任务通常基于良性数据和异常数据训练二分类模型。由于检测模型没有先验的攻击知识，因此不能通过二分类标签数据的监督学习方式进行训练。

THREATRACE使用特征获取(feature extraction)和标签赋予(label assignment)的方法，使得其能够在未知异常数据的前提下进行无监督的学习。

- 首先将节点的类型作为节点的label；
- 然后基于该节点的相连边的类型分布赋予节点初始化特征；

$$a_i = \begin{cases} \#\{e \in In(v) : i = \mathcal{M}_e(\mathcal{X}_e(e))\}, & i \in \{0, ..., N_e - 1\} \\ \#\{e \in Out(v) : i = \mathcal{M}_e(\mathcal{X}_e(e)) + N_e\}, & i \in \{N_e, ..., N_e * 2 - 1\} \end{cases} \quad (1)$$

每个节点的特征基于其的邻接边进行获取，如下图所示为节点特征的获取的示例：



训练方法 [Training Method]

训练阶段，首选将整个用于训练的图划分为若干个子图，每一个子图由若干(论文设置为150000)个active节点及其related节点组成，然后使用这些子图来对模型进行训练。

在攻击事件中，溯源图中的异常节点的比例是非常小的，因此，很可能会产生较多误报。基于这样的攻击检测特性，论文提出了一种多模型框架(multi-model framework)来减少误报率。

在攻击检测场景中：

1. 不同类型的节点的数量通常是不平衡的（例如process节点数量很多，而stdin节点数量则很少），因此单个模型很难充分学习到这类少数类型的节点特征；
2. 即使是相同类型的节点，其所处理的任务也可能是显著不同的，例如图1溯源图示例中的fluxbox节点和bash节点；因此，训练单个模型来将它们分类进同一个类别（即process类别）是较为困难的；

综上，仅仅像传统的方式训练单个模型，进而去有效区分不同类型的良性节点是不现实的。

Algorithm 1: The training method of the multi-model framework

Input: Subgraph $G = (V, E, \mathcal{X}_v, \mathcal{X}_e, \mathcal{T}_e)$; Features mapping function $\mathcal{F} : V \to \mathbb{N}^{2*N_e}$; Label mapping function $\mathcal{L} : V \to \{0, ..., N_n\}$; Hop number K; Times that FP algorithm iterates $epoch$

Output: Number of submodels cnt; Submodels $\{\mathbb{M}_0, \mathbb{M}_1, ..., \mathbb{M}_{cnt-1}\}$

1 Remove the correctly classified $active$ nodes from V;
2 $X \leftarrow active$ nodes in V;
3 $cnt \leftarrow 0$;
4 **while** X *not empty* **do**
5 $\quad \hat{V} \leftarrow$ nodes in K-hop neighborhood of $\forall v \in X$;
6 $\quad$ **for** $k = 1...epoch$ **do**
7 $\quad\quad \hat{G} \leftarrow (\hat{V}, E, \mathcal{X}_v, \mathcal{X}_e, \mathcal{T}_e)$;
8 $\quad\quad z \leftarrow \text{FP}(\hat{G}, \mathcal{F}, K, AGGREGATE_k, \mathbf{W}^k, \sigma)$;
9 $\quad\quad$ //z are the representations of $\forall v \in \hat{V}$
10 $\quad\quad \hat{z} \leftarrow z_{\forall v \in X}$;
11 $\quad\quad loss \leftarrow \text{cross\_entropy}(\hat{z}, \mathcal{L}(X))$;
12 $\quad\quad \mathbf{W}^k \leftarrow \text{backward}(loss)$;
13 $\quad$ **end**
14 $\quad M \leftarrow \text{softmax}(\hat{z})$;
15 $\quad$ **for** $v \in X$ **do**
16 $\quad\quad C_v \leftarrow$ index of the biggest element in M_v;
17 $\quad\quad \hat{C}_v \leftarrow$ index of the second biggest element in M_v;
18 $\quad\quad$ **if** $C_v = \mathcal{L}(v)$ *and* $M_{vC_v}/M_{v\hat{C}_v} > R_t$ **then**
19 $\quad\quad\quad$ remove v from X ;
20 $\quad\quad$ **end**
21 $\quad$ **end**
22 $\quad \mathbb{M}_{cnt} \leftarrow$ current trained submodel;
23 $\quad cnt \leftarrow cnt + 1$
24 **end**

X：尚未被正确分类的节点，初始化为所有用于训练模型的节点；

line6-13：训练GNN模型；

line15-21：基于概率的方法，验证节点的分类结果是否具有较高的置信度，并将高置信度且正确分类的节点从X移除；

line22：输出当前训练好的第i个子模型；

最外层的while循环执行了几次，就代表训练了多少个子模型；

$$C_i = \mathcal{L}(X_i) \text{ and } \frac{M_{iC_i}}{M_{i\hat{C}_i}} > R_t$$

第18行的验证的方式如上所示，GNN学习到节点的特征向量表示后，输入softmax层进行分类，其最终向量的维度与节点的类型保持一致。上式Ci即表示数值最大的维度，Ci hat表示数值第二大的维度，M分别表征了其被分到这两类的概率。因此，设置阈值Rt，就是为了保证分类概率最大的那一类别要明显大于其它类别的分类概率，且最大概率的类别正是其groundtruth标签。

D. Alert and Trace

将检测模型判定的异常节点存入队列Q中，在时间阈值T内动态地维护该队列Q而不是直接产生告警。

因为Threatrace模型以流(streaming)模式检测异常节点并且学习节点的特征表示，因此，如果一个被判定为异常的节点，直到等待时间阈值T后仍然为异常，则将该节点从Q中移除，并在溯源图中追踪其两跳内的邻居节点。

6 Evaluation

1. threatrace相较于sota检测模型的检测性能；
2. 参数设置对于模型检测的影响；
3. 追踪异常行为的能力；
4. 在攻击的早期阶段检测出异常行为的能力；
5. 运行时间和系统资源负载；
6. 在对抗攻击下的系统鲁棒性；

6.1 Streamspot

数据集情况，下载链接：https://github.com/sbustreamspot/sbustreamspot-data

TABLE II: Overview of StreamSpot dataset.

Scene	# of graph	Average # of nodes	Average # of edges
Benign	500	8315	173857
Attack	100	8891	28423

使用5个正常场景下的各75张图作为训练集，25*5的正常良性图和25张攻击场景下的图作为测试集，实验结果如下：

TABLE V: Comparison to StreamSpot and Unicorn on the StreamSpot dataset.

Detector	Precision	Recall	Accuracy	F-Score	TP	TN	FP	FN	FPR
StreamSpot	0.72	1.0	0.69	0.75	25	108.5	16.5	0	0.11
Unicorn	0.95	0.97	0.99	0.96	24.32	123.64	1.36	0.68	0.011
THREATRACE	1.0	1.0	1.0	1.0	25	125	0	0	0

6.2 Unicorn SC-2

数据集下载链接：https://github.com/margoseltzer/shellshock-apt

TABLE III: Overview of Unicorn SC-2 dataset.

Scene	# of graph	Average # of nodes	Average # of edges
Benign	125	238338	911153
Attack	25	243658	949887

使用100良性图进行训练，使用25的良性图和25的攻击场景图进行测试，实验结果如下：

TABLE VI: Comparison to Unicorn and ProvDetector on the Unicorn SC-2 dataset.

Detector	Precision	Recall	Accuracy	F-Score	TP	TN	FP	FN	FPR
Unicorn	0.75	0.80	0.77	0.78	20	18.33	6.67	5	0.27
ProvDetector	0.67	0.6	0.65	0.63	15	17.5	7.5	10	0.3
THREATRACE (K = 1)	0.81	0.79	0.8	0.8	19.7	20.5	4.5	5.3	0.18
THREATRACE (K = 2)	0.91	0.96	0.93	0.93	24	22.5	2.5	1	0.1

6.3 DARPA TC

数据集下载链接：https://drive.google.com/drive/folders/1fOCY3ERsEmXmvDekG-LUUSjfWs6TRdp-

TABLE IV: Overview of DARPA TC dataset.

Scene	System	# of benign nodes	# of abnormal nodes	# of edges
THEIA	Ubuntu	3505326	25362	102929710
Trace	Ubuntu	2416007	67383	6978024
CADETS	FreeBSD	706966	12852	8663569
fivedirections	Windows	569848	425	9852465

数据集包含四个不同的场景，论文使用DARPA TC提供的groundtruth对异常节点进行标记，使用良性图中的节点训练模型，使用包含了攻击事件的图的节点用于验证。

TABLE VII: Results of node level detection in DARPA TC dataset

Experiment	System	Precision	Recall	F-Score	TP	TN	FP	FN	FPR
THEIA	Ubuntu	0.87	0.99	0.93	25297	3501561	3765	65	0.001
Trace	Ubuntu	0.72	0.99	0.83	67382	2389233	26774	1	0.011
CADETS	FreeBSD	0.90	0.99	0.94	12848	705605	1361	4	0.002
fivedirections	Windows	0.67	0.92	0.80	389	569660	188	36	0.0003

TABLE VIII: Results of graph level detection in DARPA TC dataset

Experiment	Detector	Precision	Recall	F-Score	FP	FN	FPR
THEIA	THREATRACE	1.0	1.0	1.0	0	0	0
	Unicorn	1.0	1.0	1.0	0	0	0
	ProvDetector	0.96	0.92	0.94	0.2	0.4	0.04
CADETS	THREATRACE	1.0	1.0	1.0	0	0	0
	Unicorn	1.0	1.0	1.0	0	0	0
	ProvDetector	0.92	0.94	0.93	0.4	0.3	0.08
fivedirections	THREATRACE	1.0	1.0	1.0	0	0	0
	Unicorn	1.0	1.0	1.0	0	0	0
	ProvDetector	1.0	1.0	1.0	0	0	0
Trace	THREATRACE	1.0	1.0	1.0	0	0	0
	Unicorn	1.0	1.0	1.0	0	0	0
	ProvDetector	1.0	1.0	1.0	0	0	0

Conclusions

- 实时的基于主机的节点级别的异常检测；
- 输入为整个溯源图，检测其中的异常节点进而检测隐蔽的攻击行为；
- 使用多模型框架和基于概率的方法降低检测误报率；