Paradise：Real-time，Generalized，and Distributed Provenance-Based Intrusion Detection

> 摘要：从大量且多源的日志数据中准确识别出入侵行为仍然是一个巨大的挑战。Paradise是一个Real-time, ***generalized***, distributed的基于溯源数据的入侵检测方法。

对于title的理解：

1. Real-time实时性：（1）仅获取进程节点的特征向量，没有对图结构进行遍历，也没有采用机器学习相关的算法，仅仅采用基于距离的异常判定方法；（2）采用kafka作为中间件，设计负载均衡机制，将检测器部署在多台主机上进行分布式检测，加快检测速度；
2. Generalized应用的泛化性：可以自适应不同的溯源数据模型，不用对数据格式进行转换；
3. Distributed分布式：使用主机集群进行分布式检测，producer不断地收集溯源数据构建溯源图，并得到进程节点的特征向量；kafka中间件不断地将这些向量动态分发给分布式的检测主机中，各个检测主机并行地进行检测并将结果返回，以便用于取证分析。

1 introduction

日志聚类[2]、日志划分[3]；

利用内存数据库或多线程技术，实现基于溯源图的线上入侵检测；

溯源框架：SPADE、PASS、Hi-Fi、CamFlow

然而，由于溯源数据中事件的高并发性和内部复杂的依赖关系，基于溯源图的入侵检测也面临一定的挑战：

- Interoperability 互通性：主要体现在溯源数据的收集方法会随系统变化而不同、同一系统事件在不同的溯源数据标准下所呈现的形式也不同、这也导致了数据对于某些入侵检测系统的不可用；【跨平台数据的统一收集和组织方式】
- Real-time performance：现有的一些系统无法虽然取得了检测准确率的提升，但是其需要一段时间内的整个溯源数据所组成的溯源图作为数据输入，因此需要较大的时间开销。APT攻击的调查通常也需要查询和比较大量的数据，也难以保证检测的实时性；

Paradise由三个主要步骤组成：

- Collecting and preprocessing;
- Data distribution;
- Intrusion detection;

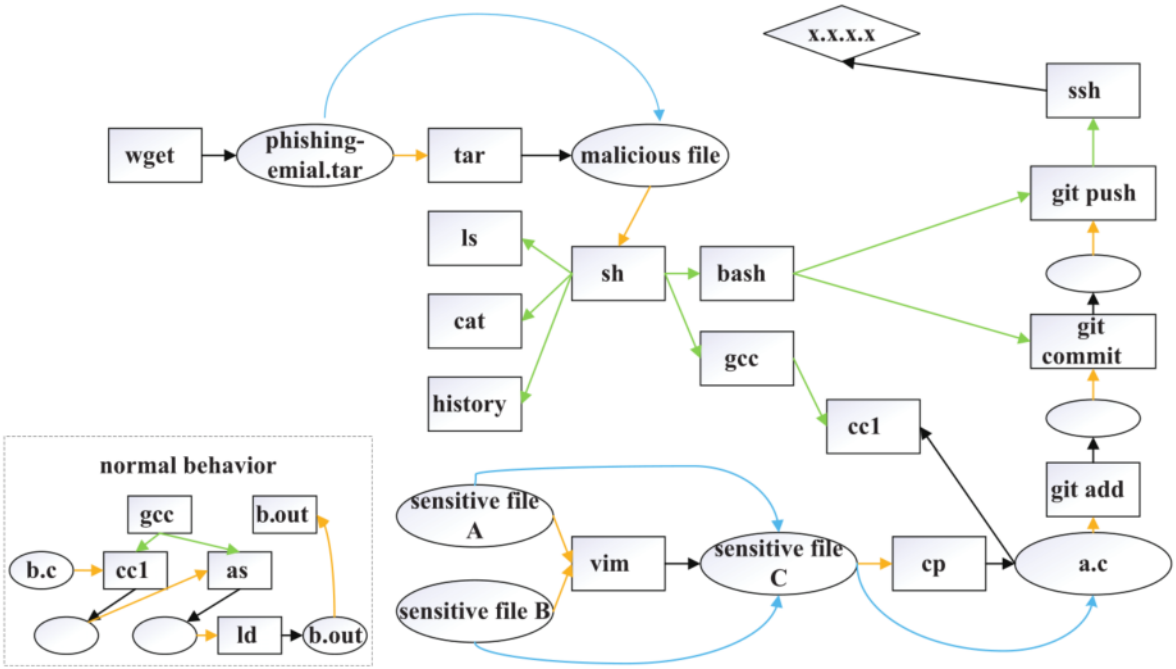
main contributions:

- 实时的、泛化的、分布式的基于溯源数据的入侵检测系统，其考虑了**顶点邻居图**（vertex neighbor graph）的特征向量，实现了大量溯源数据中的异常检测；
- 设计了一个新的简单的策略，将收集到的溯源数据转换为特征向量，不受溯源数据收集平台和数据模型的影响；
- 设计了一个自适应的监控和集群负载均衡机制，实现了有效的数据分发和并行检测；

2 Background

介绍几个重要的前置概念：溯源数据provenance、溯源数据模型provenance model、Kakfa架构、威胁模型和假设 threat model and assumption；

2.1 Motivating Example



如上所示的攻击场景中，用户通过wget应用下载并解压了恶意文件phishing-email.tar，该恶意文件中的代码进行执行获得一个反弹shell，攻击者利用该shell模仿用户的正常操作的同时，读取敏感文件A和B，并将A和B整合到一个新的文件C中，并修改文件后缀名为.c文件，利用gcc对其进行编译并上传至github；

2.2 Provenance

溯源数据描述的是"***因果关系***"，也即是说，一个系统实体是如何与其它系统实体产生关联的、系统实体之间的交互又导致了什么样的系统状态的改变；Provenance can be understood as a record of the generation and history of system data.

目前已经存在许多溯源数据的收集工具：SPADE、TREC、PASS、LinFS、Hi-Fi、ZOOM、CamFlow；这些数据收集工具运行在不同的层级之中。例如，ZOOM在应用层收集数据、PASS主要在系统层进行收集、SPADE还可以在分布式环境中收集溯源数据、

溯源数据呈现了不同系统实体之间的依赖关系(provenance represents the dependency relationships between different objects)，例如 "process A creates file B"，该条记录表明文件B的生成（结果）是进程A所导致的（原因）。

【Enabling different provenance interoperability between different provenance systems is still a challenge】

2.2 Provenance model

- OPM模型：Open Provenance Model
- W3C PROV模型：A json representation for the PROV data model

OPM模型中的边的意义：

- Used：文件的使用、进程的执行；
- WasGeneratedBy：进程生成文件；

- WasTriggeredBy：进程派生子进程；
- WasControlledBy：进程的执行由一个agent所控制；【agent具体指什么？代理？】
- WasDerivedFrom：一个新文件从一个旧文件得到；

2.3 Challenges

- Generalized：现有的不同的基于溯源数据的入侵检测系统，其往往特定于某一类溯源数据格式，因此A系统难以应用到B系统的场景之中，往往需要进行大量的数据格式的转换，在这个过程中可能导致一些关键信息的丢失或者错误的发生；
- Distributed
- Real-time：大部分基于主机溯源数据的入侵检测系统，通常考虑每个顶点的多级的邻居节点的上下文知识，其在获取了较为丰富的语义信息的同时，也难以避免地加大了时间开销；

2.4 Exploring process characteristics for intrusion detection

主要思想：相较于使用部分子图、或者整个溯源图，可以考虑使用***进程节点***作为入侵检测中的最小的单元。

挑战：实时且准确地获取进程节点的特征向量；尽可能以最小的代价动态更新节点的特征；

采用的方法：引入***邻居图（neighbor graph）***的概念；

Generalized

基于进程类型节点的邻居图，获得进程节点的特征向量（不同类型的边的数目，与threatrace类似）；因此，可以适应于不同的溯源数据模型，无需进行原始数据的格式匹配和转换；

Distribued

进程的特征向量在检测过程中是彼此独立的，因此适用于分布式的检测环境。***【question：如何理解？后文需重点留意！】***

Real-time

Paradise不用多次遍历同样的节点，仅考虑了每个节点的邻居节点，因此主要在图的遍历的时间上得到了优化。虽然损失了部分语义信息，但是其在实际的检测准确率上仍然得到了较好的保证。

2.5 Kafka for intrusion detection

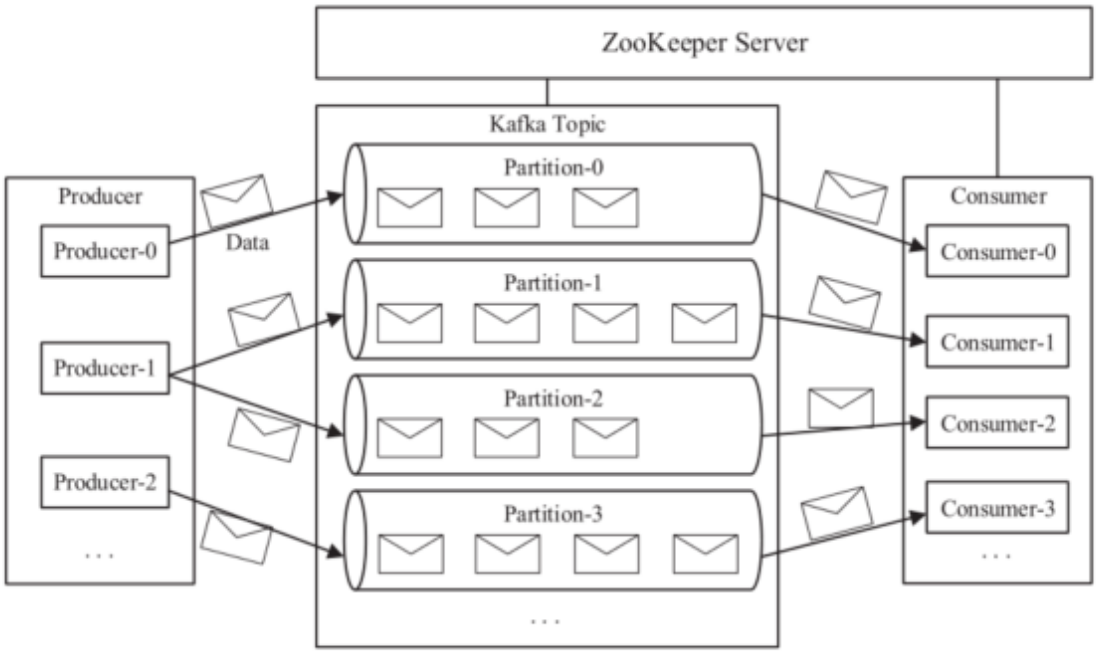


Fig. 3: Kafka framework.

2.6 Threat Model and Assumptions

1. 假设一：存在一个modeling阶段，在此阶段中，系统用户可以安全正常地操作系统，并且捕获收集正常运行状态下的系统事件；【保证能够收集到百分百的正常运行状态时的系统溯源数据】
2. 假设二：利用可信的计算平台，保护系统内核不受攻击，因此操作系统、审计引擎、数据监控等都需要是可信的且不受攻击的；【保证基础的溯源数据生成、收集的系统是绝对安全的】
3. Paradise主要检测进程相关的攻击活动或异常活动，因此，进程无关的攻击活动的检测是存在困难的；but being difficult to detect attacks does not mean attacks are undetectable，难检测不等于无法检测；

3 System Design and Implementation

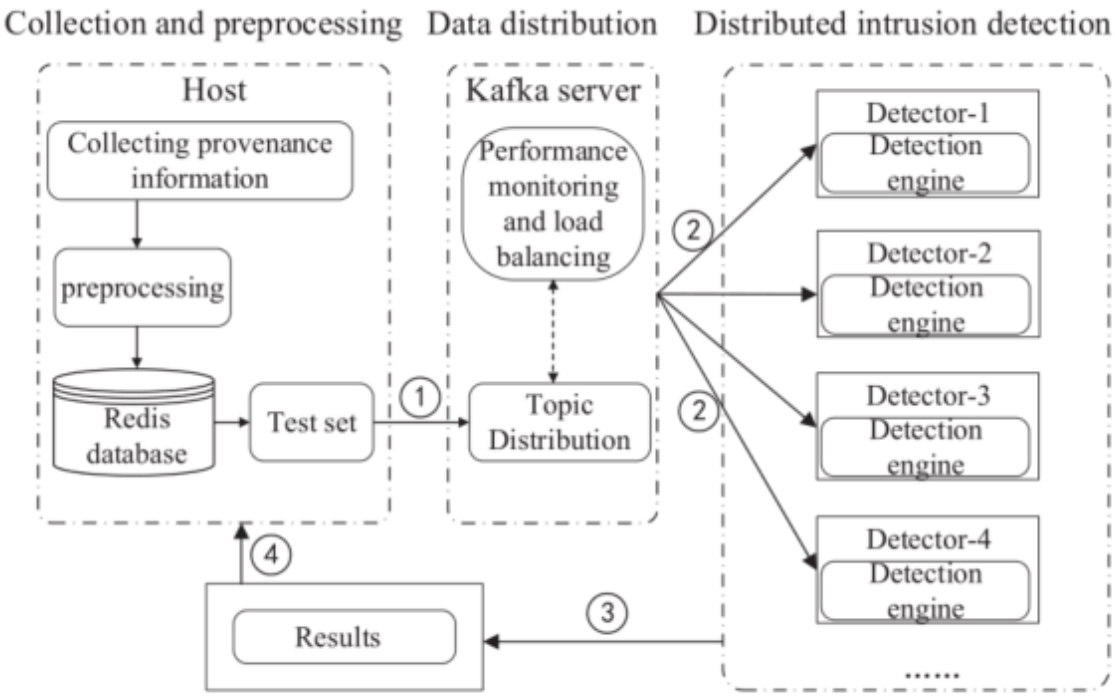


Fig. 4: Paradise system framework. (① is putting the preprocessed data into kafka *topic*. ② is dynamically distributing the data to the detection cluster. ③ is summarizing the results of different machines to obtain the final detection result. ④ is employed for forensic analysis.)

3.1 overall system design

包括三个模块：

- ***Provenance collection and preprocessing***（溯源数据收集和预处理模块）：收集溯源数据、构建溯源图、并将节点转换为特征向量、将得到的特征向量存储在Redis主存数据库中避免磁盘的低速IO所带来的时间开销；
- ***Data distribution***（数据分发模块）：使用Kafka架构将进程特征向量合理地分发到各个检测器中，并且设计了一个性能监控和负载均衡模式，保证检测器集群的均衡的检测性能；
- ***Distributed intrusion detection***（分布式入侵检测模块）：使用多个检测器进行实时检测，并且将它们的检测结果进行聚合。得到聚合后的最终的检测结果后，可以继续进行取证分析 forensic analysis从而定位系统中可能存在的漏洞或者风险；

3.2 Collecting and preprocessing

Paradise使用SPADE工具收集溯源数据。

TABLE 1: Provenance database

Database	Key	Value
NameDB	node hash value	node name
ParentDB	child node hash value	parent node hash value
ChildDB	parent node hash value	child node hash value
VectorDB	process node hash value	process feature vector

Paradise使用了四个数据库来存储溯源图。

Paradise通过如下算法来获取进程节点的特征向量。[但是这部分没有说明 isSub he proName具体该如何获取]

Algorithm 1 Extract Process Vector

Input: $G(V, E)$ //Provenance Graph
Output: Y //Process Vector

1: **for each** $dep_t : (v_i \rightarrow v_j)$ **in** E, $dep_t \in \{f_1 : Used, \cdots, f_4 : wasTriggeredBy\}$ **do**
2: **if type of** v_i **is** $Process$ **then**
3: $v_i.f_t++$
4: **end if**
5: **if type of** v_j **is** $Process$ **then**
6: $v_j.f_t++$
7: **end if**
8: **end for**

遍历每一条边，如果这条边的头节点是进程节点，则该节点对应的边类型的计数+1；同理对于尾节点也是如此；
遍历完所有边后，对于每个进程节点，都能得到长度为边类型数目的特征向量

9: **for each** v_i **in** V **do**
10: **if type of** v_i **is** $Process$ **then**
11: create new instance $Y_i(isSub, proName, y_{i1}, y_{i2}, y_{i3}, y_{i4})$
12: **for all** y_{ij} **in** Y_i **do**
13: $y_{ij} = v_i.f_j$
14: **end for**
15: **end if**
16: **end for**
17: **return** Y

遍历每个进程节点，在上面获得的特征向量的基础上添加两个新的特征维度 isSub 和 proName，最后返回的每个进程节点的特征向量即为Yi

3.3 Adaptive Data Distribution

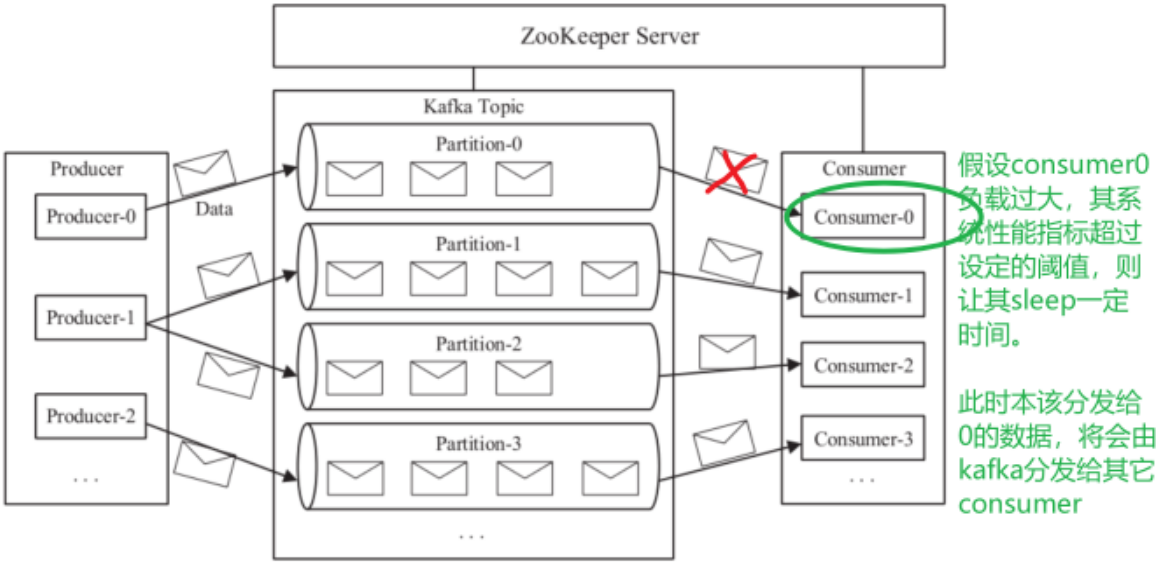
producer：收集溯源数据的主机；

consumer：检测器集群；

Kafka server：充当从producer分发数据到各个consumer的中间件；

Paradise系统设计了一个性能监控和负载均衡模块，保证producer和consumer之间的均衡

具体来说，该模块会周期性地获取各个检测主机的性能指标（CPU利用率、内存使用情况），从而动态控制分发给各个检测主机的溯源数据的规模。该方法可以自动并且合理地收集数据，并且保证检测器集群的性能得到充分发挥。



3.4 Distributed Intrusion Detection

1.Intrusion Detection

Algorithm 2 Distributed intrusion detection

Input: $X_i, Y[n], isSub, proName, T, k$
$//X_i$: Test set feature vector
$//Y[n]$: Training set feature vectors
$//isSub$: A flag to determine whether the vector is a subgraph or a complete graph
$//proName$: A corresponding field to indicate which event the provenance subgraph belongs to
$//T$: Predefined similarity threshold
$//k$: Predefined threshold of the nearest vector's number
Output: $temp-result, results$ //Detection results

对每个进程节点的特征向量进行规范化

1: **for each feature** $x_j \in X_i$ **do**
2: $x_j = (1 - e^{-ax_j})/(1 + e^{-ax_j})$
3: **end for**

对于测试集中的每个特征向量，计算其与训练集中每个向量的欧氏距离；

4: **for each process feature vector** Y_j **in** $Y[n]$ **do**
5: $d(X_i, Y_j) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_4 - y_4)^2}$
6: **if** $d(X_i, Y_j) == 0$ **then**
7: $isNormal = true$ // X_i **is normal**
8: **exit**
9: **end if**
10: **end for**

对于测试集中的某个向量，得到其与训练集中所有向量的前k个最大的距离。计算这k个最大距离的平均值，若大于阈值T，则判定为异常；

11: **sort** $d(X_i, Y[n])$ **in increasing order**
12: **get top** k **records and calculate avg** $d(X_i, Y[n])$
13: **if avg** $d(X_i, Y[n]) > T$ **then**
14: $isNormal = false$ // X_i **is abnormal**
15: **else**
16: $isNormal = true$ // X_i **is normal**
17: **end if**

根据isSub来判断测试集中每个向量的检测结果是直接上传到总结果还是暂存结果中

18: **if** $isSub$ **then**
19: **upload** $(X_i, isNormal, proName)$ **to** ***temp-result***
20: **else**
21: **upload** $(X_i, isNormal, proName)$ **to** ***results***
22: **end if**

- 将测试集中每个进程节点的特征向量，与训练集中所有正常进程的特征向量计算欧式距离；
- 计算前k大的欧氏距离的平均值，判断是否大于阈值T，大于则认为是异常，否则认为是正常；
- 根据isSub判断将该节点的检测结果上传到总的结果，还是暂时存储在每个detector之中；

notices

1. 【使用向量进行计算，相较于路径或图，检测的先后顺序不会影响结果】
2. 【由于特征向量每个维度所反映的，是与该节点相连的某一类关系的频数。但是有些关系可能是频繁发生的，有些关系可能本身产生的次数就较少，因此对于不同维度的特征向量，采取不同的归一化系数a进行归一化】

$$x_{f_i} = (1 - e^{-af_i})/(1 + e^{-af_i})$$

3. 【预先去除训练集中的相等的向量；（因为训练集中全部包含的是正常进程的特征向量，且仅仅计算测试集向量与训练集向量之间的欧式距离，若是存在重复的训练集向量，那么计算得到的距离也同样是重复的）】
4. 【Paradise仅仅获取进程节点向量，不用对图进行遍历，因此检测效率更高；同样因为仅考虑进程节点及其邻居边，因此不同的数据模型所构建出的溯源图，仍然只需要考虑进程节点和其连接的边。（Paradise could be applied to different provenance models with the same strategy in order to avoid data conversion and reduced detection accuracy）】

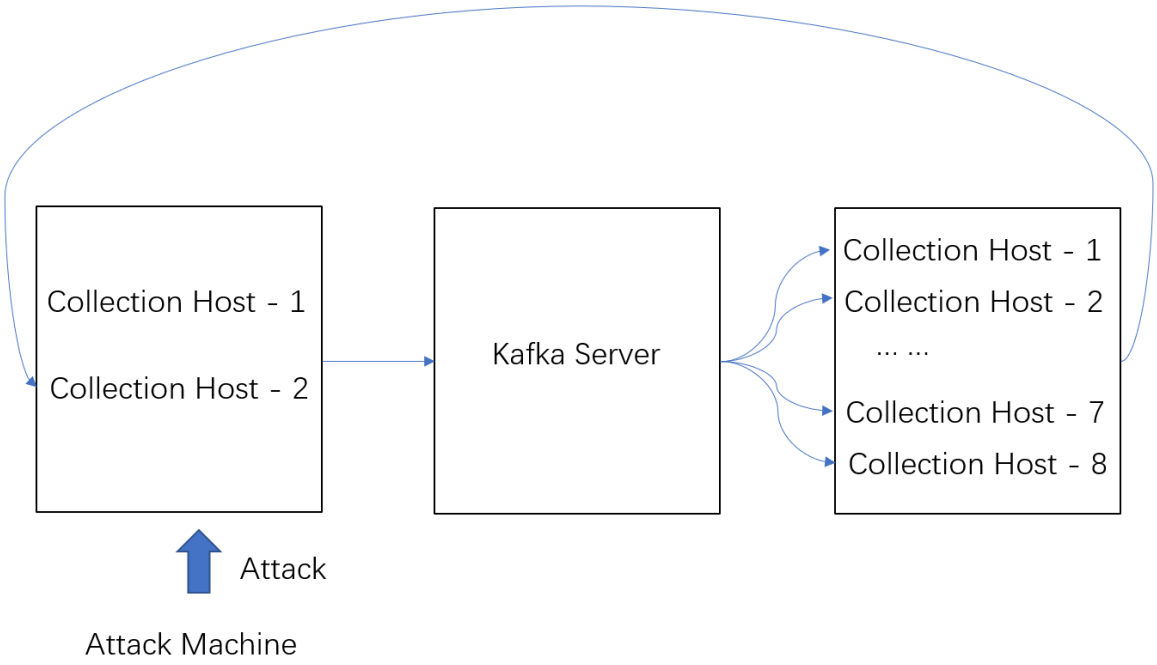
2.Forensic Analysis

使用广度优先搜索，以检测出的异常进程节点为出发点，获取与它相关的邻居节点，从而进行取证分析。

4 Experimental Evaluation

- Q1：Paradise能够在基于不同溯源数据模型的数据集上进行异常检测吗？【Generalized】
- Q2：Paradise在开源数据集下，相较于现有的SOTA模型的表现？【Performance】
- Q3：Paradise所提出的对于Kafka的负载均衡策略能否加快检测速度？【Real-time、Distributed】
- Q4：Paradise的内存利用和溯源数据增长的负载？

4.1 Experimental Setup



5 Related Work

5.1 Host-based Intrusion Detection

- 《A sense of self for unix processes》（1996 SP）-首先提出获取正常进程的短系统调用序列，构建正常系统调用序列的数据库，进而检测异常的系统调用；
- 《Intrusion detection using sequences of system calls》（1998 Journal of computer security）-计算不同的系统调用序列之间的Hamming距离，通过与用户定义的阈值进行比较从而判断异常；
- 《Multiresolution abnormal trace detection using varied-length n-grams and automata》（2006）-提出变长n-gram算法来获取溯源图的特征，构建自动化的检测过程；
- 《Use of k-nearest neighbor classfier for intrusion detection》（2002 C&S）-提出O-KNN算法，使用TF-IDF算法获取特征向量并使用余弦相似度计算向量之间的余弦相似性；

- 《A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns》（2013）-通过分析不相连的系统调用模式，获取语义特征，输入神经网络来判断哪些系统调用的trace是异常的；

- 《using bayesian networks for probabilistic identification of zero-day attack paths》（2018 TIFS）-利用贝叶斯神经网络识别0-day攻击路径；

---

provenance-based intrusion detection

- 《Holmes》
- 《Unicorn》
- 《Pagoda》

5.2 Forensic Analysis 取证分析

5.3 Distributed intrusion detection

6 Discussion

Overall, Paradise is essentially an anomaly-based detection (AD) system, which generates a statistical model of normal behavior by monitoring the system during an attack-free period. 本质上是一个异常检测系统，基于正常进程的数据，来判断是否出现异常

7 Conclusion