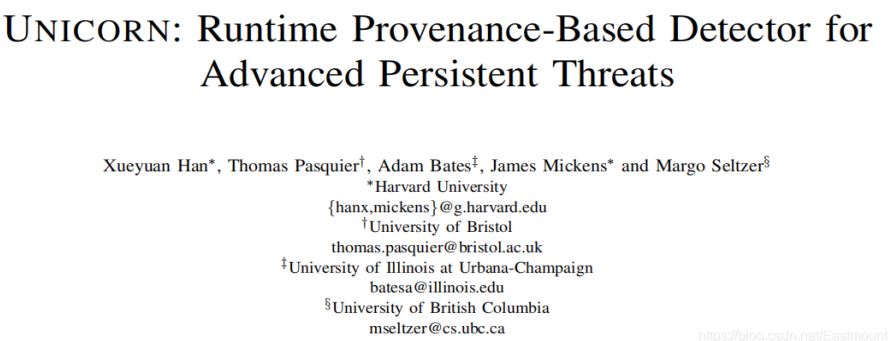
[论文阅读] (08) NDSS2020 UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats

《娜璋带你读论文》系列主要是督促自己阅读优秀论文及听取学术讲座，并分享给大家，希望您喜欢。由于作者的英文水平和学术能力不高，需要不断提升，所以还请大家批评指正，非常欢迎大家给我留言评论，学术路上期待与您前行，加油。

前一篇文章分享了RAID 2020上的论文《Cyber Threat Intelligence Modeling Based on Heterogeneous Graph Convolutional Network》，基于异构图卷积网络的网络威胁情报建模。这篇文章将详细介绍NDSS2020的《UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats》，一种基于溯源图的实时APT检测器。希望这篇文章对您有所帮助，这些大佬是真的值得我们去学习，献上小弟的膝盖~fighting!



UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats

Xueyuan Han*, Thomas Pasquier†, Adam Bates‡, James Mickens* and Margo Seltzer§
*Harvard University
{hanx,mickens}@g.harvard.edu
†University of Bristol
thomas.pasquier@bristol.ac.uk
‡University of Illinois at Urbana-Champaign
batesa@illinois.edu
§University of British Columbia
mseltzer@cs.ubc.ca

原文作者：Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens and Margo Seltzer
原文标题：UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats
原文链接：https://arxiv.org/pdf/2001.01525.pdf
发表会议：NDSS 2020
参考文献： 感谢两位老师

- https://blog.csdn.net/Sc0fie1d/article/details/104868847
- https://blog.csdn.net/xjxtx1985/article/details/106473928

前文赏析：

- [论文阅读] (01) 拿什么来拯救我的拖延症？初学者如何提升编程兴趣及LATEX入门详解
- [论文阅读] (02) SP2019-Neural Cleanse: Identifying and Mitigating Backdoor Attacks in DNN
- [论文阅读] (03) 清华张超老师 - GreyOne: Discover Vulnerabilities with Data Flow Sensitive Fuzzing
- [论文阅读] (04) 人工智能真的安全吗？浙大团队外滩大会分享AI对抗样本技术
- [论文阅读] (05) NLP知识总结及NLP论文撰写之道——Pvop老师
- [论文阅读] (06) 万字详解什么是生成对抗网络GAN？经典论文及案例普及
- [论文阅读] (07) RAID2020 Cyber Threat Intelligence Modeling Based on Heterogeneous GCN
- [论文阅读] (08) NDSS2020 UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats

文章目录

摘要

由于APT（Advanced Persistent Threats）攻击具有缓慢可持续的攻击模式以及频繁使用0-day漏洞的高级特性使其很难被检测到。**本文利用数据来源分析（provenance）提出了一种基于异常的APT检测方法，称为UNICORN。**

- **从建模到检测，UNICORN专门针对APT的独有特性（low-and-slow、0-Days）进行设计。**
- **UNICONRN利用高效的图分析方法结合溯源图丰富的上下文语义和历史信息，在没有预先设定攻击特征的情况下识别隐蔽的异常行为。**
- **通过图概要（graph sketching）技术，它有效概括了长时间系统运行来对抗长时间缓慢攻击。**
- **UNICONRN使用一种新的建模方法来更好地捕捉长期行为规律，以进一步提高其检测能力。**

最后通过大量实验评估表明，本文提出的方法优于现有最先进的APT检测系统，并且在真实APT环境中有较高的检测精度。

***Abstract*—Advanced Persistent Threats (APTs) are difficult to detect due to their "low-and-slow" attack patterns and frequent use of zero-day exploits. We present UNICORN, an anomaly-based APT detector that effectively leverages data provenance analysis. From modeling to detection, UNICORN tailors its design specifically for the unique characteristics of APTs. Through extensive yet time-efficient graph analysis, UNICORN explores provenance graphs that provide rich contextual and historical information to identify stealthy anomalous activities without pre-defined attack signatures. Using a graph sketching technique, it summarizes long-running system execution with space efficiency to combat slow-acting attacks that take place over a long time span. UNICORN further improves its detection capability using a novel modeling approach to understand long-term behavior as the system evolves. Our evaluation shows that UNICORN outperforms an existing state-of-the-art APT detection system and detects real-life APT scenarios with high accuracy.**

PS：蓝色-研究背景；绿色-英文表达学习；黄色-文章骨干；紫色-文章贡献

> 本文提出的UNICORN是一种基于异常的APT检测器，可以有效利用数据Provenance进行分析。通过广泛且快速的图分析，使用graph sketching技术，UNICORN可以在长期运行的系统中分析Provenance Graph，从而识别未知慢速攻击。其中，Provenance graph提供了丰富的上下文和历史信息，实验证明了其先进性和较高准确率。

Ⅰ.引言

> Introduction是论文的开头，是极为重要的部分，介绍了为什么要做这份工作，建议大家仔细阅读，尤其是写英文论文的读者。因此，作者将该部分进行了详细总结。

APT攻击现在变得越来越普遍。这种攻击的时间跨度长，且与传统攻击行为有着本质的区别。APT攻击者的目的是获取特定系统的访问控制，并且能够长期潜伏而不被发现。攻击者通常使用0-day漏洞来获取受害者系统的访问控制。

传统检测系统通常无法检测到APT攻击。

- 依赖恶意软件签名的检测器对利用新漏洞的攻击无效。
- 基于异常检测的系统通常分析一系列的系统调用或日志系统事件，其中大部分方法无法对长期行为进行建模。
- 由于基于异常检测的方法只能检测系统调用和事件的短序列很容易被绕过。

综上，当前针对APT攻击的检测方法很少能成功。攻击者一旦使用0-Day漏洞，防御者便无计可施；而基于系统调用和系统事件的检测方法，由于数据过于密集，这些方法难以对长时间的行为模式进行建模。**因此，数据溯源（data provenance）是一种检测APT更合适的数据。**

最近的研究成果表明数据溯源是一个很好的APT检测数据源。数据溯源将系统执行表示成一个有向无环图（DAG），该图描述了系统主体（如进程）和对象（文件或sockets）之间的信息流。即使跨了很时间，在图中也把因果相关的事件关联到一起。因此，即使遭受APT攻击的系统与正常系统比较类似，但是溯源图中丰富的上下文语义信息中也可以很好地区分正常行为与恶意行为。

> [87] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: Real-time apt detection through correlation of suspicious information flows," in Symposium on Security and Privacy. IEEE, 2019.

> More recent work [15], [19], [83], [87] suggests that data provenance might be a better data source for APT detection. Data provenance represents system execution as a directed acyclic graph (DAG) that describes information flow between system subjects (e.g., processes) and objects (e.g., files and sockets). It connects causally-related events in the graph, even when those events are separated by a long time period. Thus, even though systems under APT attack usually behave similarly to unattacked systems, the richer contextual information in provenance allows for better separation of benign and malicious events [55].
>
> 优点：实现了一种新型检测高级可持续性威胁(APT)的方法。
> 缺点：该方法根据已有的攻击知识通过简单的边匹配规则实现APT检测，很难检测未知的APT攻击。

然而，基于数据溯源的实时APT检测依然具有挑战。
随着APT攻击的渗透的进行，数据溯源图的规模会不断增大。其中必要的上下文分析需要处理大量图中的元素，而图上的分析通常复杂度比较高。当前基于数据溯源的APT检测方法根据已有的攻击知识通过简单的边匹配实现APT检测，无法处理未知的APT攻击。基于溯源的异常检测系统主要是基于图模型的邻域搜索，利用动态或静态模型识别正常行为模式。理论上关联的上下文越丰富越好，但是实际中由于图分析的复杂性较高限制了其可行性。

- Provenance Graph的分析是相当耗费计算资源，因为APT是可持续攻击，图的规模也会越来越大

当前的APT检测系统通常面临如下三种问题：

- (1) 静态模型难以捕获长时间的系统行为；
- (2) low-and-slow APT投毒攻击：由于APT高级可持续的特性可以在系统中潜伏很长时间，相关的行为会被认为是正常行为，这样的攻击会影响检测模型；
- (3) 在主存内进行计算的方法，应对长期运行的攻击表现不佳。

基于此，本文提出了UNICORN，使用graph sketching来建立一个增量更新、固定大小的纵向图数据结构。这种纵向性质允许进行广泛的图探索，使得UNICORN可以追踪隐蔽的入侵行为。而固定大小和增量更新可以避免在内存中来表示provenance graph，因此UNICORN具有可扩展性，且计算和存储开销较低。UNICORN在训练过程中直接对系统的行为进行建模，但此后不会更新模型，从而防止模型的投毒攻击。

本文的主要贡献如下：

- 针对APT攻击特性提出一种基于Provenance的异常检测系统。
- 引入一种新的基于概要的（sketch-based）、时间加权的（time-weighted）溯源编码，该编码非常紧凑且可处理长时间的溯源图。
- 通过模拟和真实的APT攻击来评估UNICORN，证明其可以高精度检测APT活动。
- 实现代码开源。

II.背景

1.系统调用追踪的挑战
系统调用抽象提供了一个简单的接口，用户级应用程序可以通过这个接口请求操作系统的服务。作为调用系统服务的机制，系统调用接口通常也是攻击者入侵的入口点。因此，系统调用跟踪一直被认为是入侵检测的实际信息源。然而：

- 当前的攻击检测系统是对非结构化的系统调用的审记日志进行分析，但捕获的系统调用杂乱分散，传统基于异常检测的思路无法处理APT。因此需要将其关联成data provenance，基于溯源的方法是将历史上下文数据都编码到因果关系图中。
- 数据溯源方法已经被应用到攻击调查中，已经有一些方法能够根据审计数据构建系统溯源图用以实现对系统执行过程的建模。然而些

方法依然存在一些局限：(1) 这种事后构建很难保证溯源图的正确性，由于系统调用问题存大量并发，溯源图的完整性与可靠性无法保证；(2) 容易被绕过；(3) 时空复杂度较高。

- 由于一些内核线程不使用系统调用，因此基于Syscall生成的Provenance是一些分散的图，而不是一张系统运行状况的完整图

2.全系统追踪溯源

全系统溯源运行在操作系统层面，捕获的是所有系统行为和它们之间的交互。通过捕获信息流和因果关系，即使攻击者通过操作内核对象来隐藏自己的行踪也无济于事。

本文使用CamFlow，采用了Linux安全模块（Linux Security Modules，LSM）框架来确保高效可靠的信息流记录。LSM可以消除race condition。

> CamFlow：溯源搜集系统，参考官网 https://camflow.org/。

3.问题描述

现有基于数据溯源的APT攻击检测方法主要存在如下缺陷：

- 预定义的边匹配规则过于敏感，很难检测到APT攻击中的0-Day漏洞；
- 溯源图的近邻约束导致其只能提供局部上下文信息（而非whole-system），然而这会影响相关异常检测精度；
- 系统行为模型难以检测APT：静态模型无法捕获长期运行的系统的行为；动态模型容易遭受中毒攻击；
- 溯源图的存储与计算都是在内存中，在执行长期检测上有局限性。

UNICORN可以解决如上问题，其本质是把APT检测问题看成大规模、带有属性的实时溯源图异常检测问题。在任何时间，从系统启动到其当前状态捕获的溯源图都将与已知正常行为的溯源图进行比较。如果有明显差别，那么就认为该系统正在遭受攻击。

对于APT检测来说，理想基于溯源的IDS应该如下：

- 充分利用溯源图的丰富上下文，以时间与空间有效的方法持续分析溯源图；
- 在不假设攻击行为的基础上，应考虑系统执行的整个持续时间；
- 只学习正常行为的变化，而不是学习攻击者指示的变化。

Ⅲ.威胁模型

假设主机入侵检测有适当的场景：攻击者非法获得对系统的访问权限，并计划在不被检测的情况下驻留在系统中很长一段时间。攻击者可能分阶段执行攻击，在每个阶段还会使用大量的攻击技术。UNICORN的目标是通过解决主机生成的溯源来实现在所有阶段对APT攻击进行检测。本文假设，我们假设在受到攻击之前，UNICORN在正常运行期间会完全观察主机系统，并且在此初始建模期间不会发生攻击。

数据收集框架的完整性是UNICORN正确性的核心，因此我们假定所使用的CamFlow中，LSM完整性是可信的。同时，本文假设内核、溯源数据和分析引擎的正确性，我们重点关注UNICORN的分析能力。

Ⅳ.系统设计

独角兽是一个基于主机的入侵检测系统，能够同时检测在网络主机集合上的入侵。

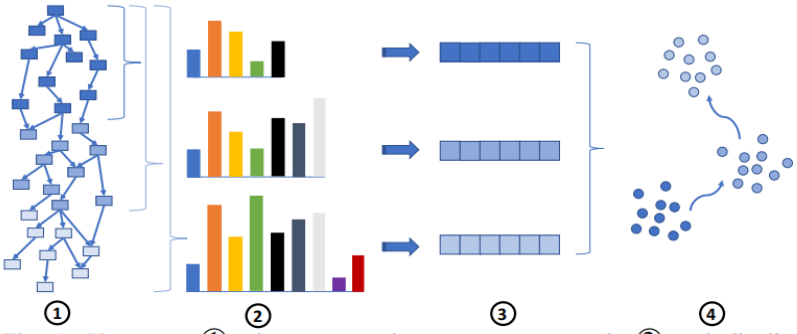


Fig. 1: UNICORN ① takes a streaming provenance graph, ② periodically summarizes graph features into histograms, and then ③ creates fixed-size graph sketches. The resulting clustering-based model ④ captures the dynamics of system execution. During deployment, graph sketches are created through the same steps (①, ② and ③) and then compared against the model in ④.

图1展示了UNIOCORN的基本流程。

- ① **以一个带标签的流式溯源图作为输入。**该图由CamFlow生成，每条边是带属性的。溯源系统构建一个具有偏序关系的DAG溯源图，能实现有效的流式计算和上下文分析。
- ② **建立一个运行时的内存直方图。**UNICORN有效构建一个流式直方图，该直方图表示系统执行的历史，如果有新边产生则实时更新直方图的计数结果。通过迭代的探索大规模图的近邻关系，发现了在上下文环境中系统实体的因果关系。该工作是UNICORN的第一步，具体来说，直方图中每个元素描述了图中唯一的一个子结构，同时考虑了子结构中的顶点与边上的异构标签，以及这些边的时间顺序。APT攻击缓慢的渗透攻击目标系统，希望基于的异常检测方法最终忘记这一行为，把其当成正常的系统行为，但是APT攻击并不能破坏攻击成功的相关信息流依赖关系。
- ③ **定期计算固定大小的概要图（graph sketch）。**在纯流式环境，当UNICORN对整个溯源进行汇总时，唯一直方图元素的数量可能会任意增长。这种动态变化导致两个直方图之间的相似计算变得非常有挑战，从而使得基于直方图相似计算的建模以及检测算法变的不可行。UNICORN采用相似度保存的hash技术把直方图转换成概要图。概要图可以增量维护，也意味着UNICORN并不需要将整个溯源图都保存在内存中。另外，概要图保存了两个直方图之间的jaccard相似性，这在后续图聚类分析中特别有效。
- ④ **将简略图聚类为模型。**UNICORN可以在没有攻击知识的前提下实现APT攻击检测。与传统的聚类方法不同，UNICORN利用它的流处理能力生成一个动态演化模型。该模型通过在其运行的各个阶段对系统活动进行聚类捕获单个执行中的行为改变，但是UNICORN无法在攻击者破坏系统时动态实时修改模型。因此，它更适合APT攻击这类长期运行的攻击。

A.溯源图

最近几年溯源图在攻击分析中越来越流行，并且本身固有的特别可以有效的用于APT检测。溯源图挖掘事件之间的因果关系，因果关系有助于对时间跨度较远的事件进行推理分析，因此有助于在检测APT相关攻击。

UNICORN根据两个系统执行的溯源图的相似性还判定两个系统的行为相似性。而且UNICORN总是考虑整个溯源来检测长期持续的攻击行为。当前已经有许多图相似度计算方法，然而这些算法大部分是NPC的，即使多项式时间复杂度的算法也无法满足整个溯源图快速增涨的需求。

B.构建Graph直方图

本文方法的目标是有效对溯源图进行比较分析，同时容忍正常执行中的微小变化。 对于算法，我们有两个标准：

- 图表示应考虑长期的因果关系；
- 必须能够在实时流图数据上实现该算法，以便能够在入侵发生时阻止入侵（不仅仅是检测到入侵）。

本文基于一维WL同构检验，采用了线性时间的、快速的Weisfeiler-Lehman（WL）子树图核算法。该算法的使用依赖于构造的顶点直方图的能力，需要直方图能捕捉每个顶点周围的结构信息。根据扩充的顶点标签对顶点进行分类，这些标签完全描述了顶点的领域，并且通过迭代的标签传播来构造这些扩展的顶点标签。

同构性的WL检验及其子树kernel变化，以其对多种图的判别能力而闻名，超越了许多最新的图学习算法（例如，图神经网络）。对Weisfeiler-Lehman（WL）子树图核的使用取决于我们构建顶点直方图的能力，捕获围绕每个顶点的图结构。 我们根据增强顶点标签对顶点进行分类，标签描述了顶点的R-hop邻居。

为了简单说明，假设有一个完整静态图，重标记对所有的输入标签的聚合。对每个顶点都重复执行这个过程来实现对n跳邻居的描述。一旦为图中的每个顶点都构建了扩展标签，那么就可以基于此生成一个直方图，其中每个bucket表示一个标签。两个图的相似性比较是基于以下假设：两个图如果相似那么在相似的标签上会有相似的分布。

Algorithm 1: Graph Histogram Generation

Input : $G = (V, E, \mathcal{F}^v, \mathcal{F}^e, \mathcal{C}), R$
Output: Histogram H

for $i \leftarrow 1$ **to** R **do**
  foreach $v \in V$ **do**
    $M \leftarrow \{\}$
    if $i == 1$ **then**
      $l_0(v) \leftarrow \mathcal{F}^v(v)$
    else if $i == 2$ **then**
      $TS \leftarrow \{\}$
      foreach $e \in \texttt{In}(v)$ **do**
        $w \leftarrow \texttt{Source}(e)$
        $M \leftarrow M + \{\mathcal{F}^e(e) :: l_1(w)\}$
        $T(w) \leftarrow \mathcal{C}(e)$
        $TS \leftarrow TS + \{T(w)\}$
      $T(v) \leftarrow \texttt{Min}(TS)$
    else
      $TS \leftarrow \{\}$
      foreach $w \in \mathcal{N}(v)$ **do**
        $M \leftarrow M + \{l_{i-1}(w)\}$
        $TS \leftarrow TS + \{T(w)\}$
      $T(w) \leftarrow \texttt{Min}(TS)$
    $\texttt{Sort}(M)$ based on timestamps $T(w)$, $\forall w$ whose label is included in M
    $\texttt{s}_v \leftarrow l_{i-1}(v) + \texttt{Concat}(M)$
  foreach $v \in V$ **do**
    $l_i(v) \leftarrow \texttt{Hash}(\texttt{s}_v)$
    if $l_i(v) \notin H$ **then** $H[l_i(v)] \leftarrow 1$
    else $H[l_i(v)] \leftarrow H[l_i(v)] + 1$

我们的目标是构建一个直方图，图中的每个元素对应一个唯一的顶点标签，用于捕获顶点的R-hop的in-coming邻居。

信息流的多样性与复杂性（Streaming Variant and Complexity）。算法1只有新顶点出现或是新边出现对其邻顶点有影响时才会执行。本文方法只需要为每条新边更新其目标顶点的邻域。UNICORN采用这种偏序关系来最小化计算代价。

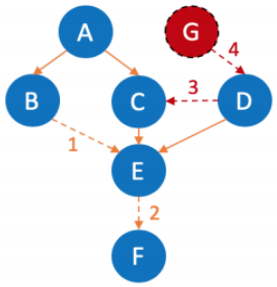


Fig. 2: A simple, abstracted provenance graph where dotted edges arrive after all solid edges have been processed and dotted vertices are newly-arrived. If the provenance graph observes partial ordering, edge 1 must arrive before edge 2 and only the local structures of vertex E and F need to be computed. However, in a provenance graph where partial ordering is *not* observed, we can encounter edge 3 and 4 and the newly-arrived vertex G (in dotted red) after the solid edges. In such a case, with R = 2, we need to update both vertex C and E (descendants of D) for edge 3 and vertex C, D, and E (descendants of G) for edge 4.

直方图元素的概念漂移问题。APT攻击场景需要模型必须能够处理长期运行行为分析能力，而系统行为的动态变化会导致溯源图的统计信息也随之变化，这种现象就叫概念漂移（concept drift）。

UNICORN通过对直方图元素计数使用指数权重衰减来逐渐消除过时的数据（逐渐忘记机制），从而解决了系统行为中的此类变化。它分

配的权重与数据的年龄成反比。

$$H_h = \sum_t \mathbb{1}_{x_t = h}$$

入侵检测场景中的适用性。上述“逐渐忘记”的方法，使得UNICORN可以着眼于当前的系统执行动态，而且那些与当先的object/activity有关系的事件不会被忘记。

C.生成概要图（Graph Sketches）

Graph直方图是描述系统执行的简单向量空间图统计量。 然而，与传统的基于直方图的相似性分析不同，UNICORN会随着新边的到来不断更新直方图。另外，UNCORN会根据图特征的分布来计算相似性，而不是利用绝对统计值。

本文采用locality sensitive hashing，也称作similarity-preserving data sketching。UNICORN的部署采用了前人的研究成果HistoSketch，该方法是一种基于一致加权采样的方法，且时间得性是常数。

D.学习进化模型

在给定graph sketch和相似性度量的情况下，聚类是检测离群点常用的数据挖掘手段。**然而传统的聚类方法无法捕获系统不断发展的行为。**UNICORN利用其流处理的能力，创建了进化模型，可以捕获系统正常行为的变化。更重要的是，模型的建立是在训练阶段完成的，而不是在部署阶段，因为部署阶段训练模型可能会遭受中毒攻击。

UNICORN在训练期间创建一个时序sketches，然后使用著名的K-medods算法从单个服务器对该概要序列进行聚类，使用轮廓系数（silhouette coefficient）确定最佳K值值。每个簇表示系统执行的元状态（meta-states），如启动、初始化、稳定状态。然后UNICORN使用所有簇中sketches的时间顺序和每个簇的统计量（如直径、medoid），来生成系统进化的模型。

Algorithm 2: Generating an Evolution Trace

Input : Sketches $S(t), t = 0, \cdots, T$ of a streaming provenance graph, ordered by time t
Output: Evolution List E

```
E = {}
for t ← 0 to T do
    k = BelongsTo(S(t))
                                        /* k ∈ {1, ··· , K} */
    if Empty(E) || Tail(E) != k then E = E :: k
```


对于每个训练实例，UNICORN创建一个模型，该模型捕获系统运行时执行状态的更新。直观地说，这类似于跟踪系统执行状态的自动机。最终的模型由训练数据中所有种源图的多个子模型组成。

E.异常检测

在部署期间，异常检测遵循前面章节中描述的相同流模式。UNICORN周期性地创建graph sketch，因为直方图从流式溯源图演变而来。给定一个概要图，UNICORN将该概要与建模期间学习的所有子模型进行比较，将其拟合到每个子模型中的一个聚类中。

UNICORN假设监视从系统启动开始，并跟踪每个子模型中的系统状态转换。要在任何子模型中为有效，概要必须适合当前状态或下一个状态；否则，被视为异常。因此，我们检测到两种形式的异常行为：

- 不符合现有聚类的概要
- 聚类之间的无效转换

V.实现

本文实现使用图形处理框架GraphChi，在C++中实现了UNICORN的图形处理算法，在Python中实现了数据解析和建模组件。

GraphChi是一个基于磁盘的系统，它可以在一台计算机上高效地计算具有数十亿条边的大型图。使用GraphChi，UNICORN可以在不需要将整个溯源图存储在内存中的情况下获得高效的分析性能。UNICORN依赖于GraphChi的两个重要特性：

- GraphChi使用一个并行滑动窗口（PSW）算法将形分割成碎片，每个碎片中的边数大致相同；它并行计算每个碎片。该算法只需少量的非顺序磁盘访问，就可以快速更新磁盘的顶点和边。这允许UNICORN独立于内存约束来分析整个起源图。
- UNICORN利用GraphChi对流图的高效计算。

VI.实验评估

我们分析了大约1.5TB的系统监控数据，其中包含来自各种跟踪系统的操作系统级溯源记录，证明了我们的方法的适用性。我们的评估评估旨在验证以下问题：

- UNICORN能否在长期运行的系统中，准确地检测APT攻击的异常行为
- 针对APT的特性做出的设计决策有多重要？
- UNICORN的“逐渐忘记”策略是否能更好地理解系统行为？
- 相比于现存的使用静态快照进行聚类的方法，UNICORN的进化模型是否更有效？
- UNICORN是否足够快速，以执行实时监视和检测？
- 在系统执行过程中，UNICORN的内存和CPU使用如何？

数据集采用DARPA TC3的三个APT攻击数据集：

- Cadets
- ClearScope
- THEIA

A.UNICORN vs. StreamSpot

StreamSpot是一个基于聚类的异常检测系统，它处理流式异构图。其流点数据集如下，包含来自六种场景的信息流图，其中5个是良性的，每个场景运行100次，为每个场景生成100个图。

使用Linux SystemTap记录系统，良性场景记录来自正常浏览活动的系统呼叫，如观看YouTube视频和检查Gmail，而攻击场景涉及从恶意URL驱动下载，利用Flash漏洞并获得对访问主机的根访问权限。

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Preprocessed Data Size (GiB)
StreamSpot	YouTube	100	8,292	113,229	0.3
	Gmail	100	6,827	37,382	0.1
	Download	100	8,831	310,814	1
	VGame	100	8,637	112,958	0.4
	CNN	100	8,990	294,903	0.9
	Attack	100	8,891	28,423	0.1

TABLE I: Characteristics of the StreamSpot dataset. The dataset is publicly available only in a preprocessed format.

我们使用这个数据集将UNICORN与StreamSpot对比，结果如下：

Experiment	Precision	Recall	Accuracy	F-Score
StreamSpot (baseline)	0.74	N/A	0.66	N/A
$R = 1$	0.51	1.0	0.60	0.68
$R = 3$	0.98	0.93	0.96	0.94

TABLE II: Comparison to StreamSpot on the StreamSpot dataset. We estimate StreamSpot's average accuracy and precision from the figure included in the paper [83], which does not report exact values. They did not report recall or F-score.

Experiment	# of Test Graphs	# of FPs ($R = 1$)	# of FPs ($R = 3$)
YouTube	25	14	0
Gmail	25	19	0
Download	25	25	2
VGame	25	20	0
CNN	25	18	0

TABLE III: Decomposition of UNICORN's false positive results of the StreamSpot dataset.

B.DARPA TC Datasets

接下来，我们证明了UNICORN可以利用来自各种不同来源捕获系统的数据有效地检测APT。

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
DARPA	Benign	66	59,983	4,811,836	271
CADETS	Attack	8	386,548	5,160,963	38
DARPA	Benign	43	2,309	4,199,309	441
ClearScope	Attack	51	11,769	4,273,003	432
DARPA	Benign	2	19,461	1,913,202	4
THEIA	Attack	25	275,822	4,073,621	85

TABLE IV: Characteristics of graph datasets used in the DARPA experiments.

参考文献中有关于者三个DARPA数据集的详细介绍。实验模拟了一个企业设置，包括安全关键服务，如web服务器、SSH服务器、电子邮件服务器和SMB服务器（用于共享文件访问）。红队通过使用火狐后门、Nginx后门和钓鱼电子邮件实施各种APT和常见的威胁攻击。

Experiment	Precision	Recall	Accuracy	F-Score
DARPA CADETS	0.98	1.0	0.99	0.99
DARPA ClearScope	0.98	1.0	0.98	0.99
DARPA THEIA	1.0	1.0	1.0	1.0

TABLE V: Experimental results of the DARPA datasets.

本文将良性数据集的90%用于训练，10%用于测试；sketch size为200，R=3，检测结果如下：

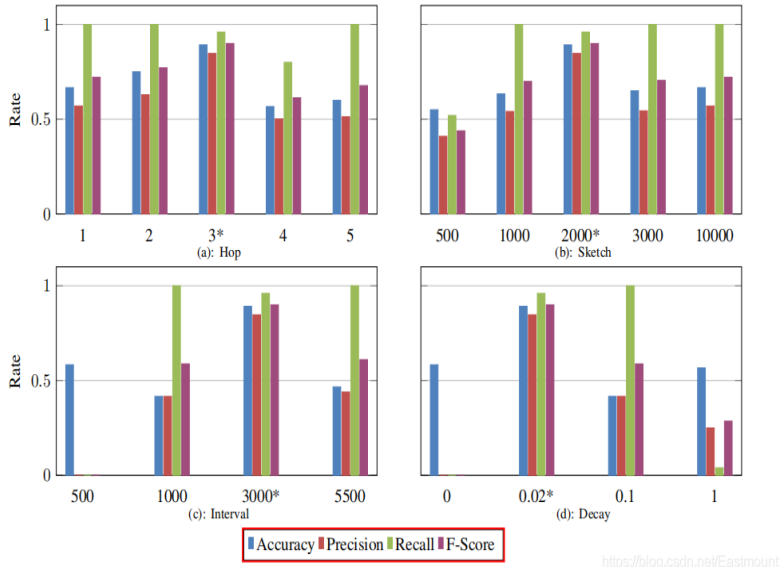
- 表V显示，UNICORN分析框架可以推广到不同来源的捕获系统和各种溯源图结构。独角兽的高性能表明，它可以准确地检测各种平台长期系统中的异常。
- 红队通常使用不同的攻击向量发起APT攻击，这些攻击占数据量不到0.001%。UNICORN基于异常的检测机制能在没有预先攻击知识的情况下识别这些攻击，尽管它们嵌入了大量的良性活动中。

相关工作比较：

- 我们注意到，一些现有的系统（Holmes S&P19、Poirot CCS19）也使用DARPA数据集进行评估。UNICORN和这些系统不同，因为它们使用了一种基于规则的方法，需要先验的专家知识来构建一个模型。独角兽不同，它使用一个无监督的学习模型，不需要专家的

输入。然而，根据检测到的攻击数量发现UNICORN的性能与它们是相似的：UNICORN检测对FreeBSD和Linux的所有攻击，与Holmes和Poirot一样。我们在第八部分中更详细地讨论了这些系统和UNICORN之间的差异。

Fig. 3: Detection performance (precision, recall, accuracy, and F-score) with varying hop counts (Fig. 3(a)), sketch sizes (Fig. 3(b)), intervals of sketch generation (Fig. 3(c)), and decay factor (Fig. 3(d)). Baseline values (*) are used by the controlled parameters (that remain constant) in each figure.



C.Supply Chain攻击场景

前面的攻击场景无法确保异常行为和正常行为之间的相似性，因此本文在Continuous Integration（CI）平台上单独设计了两个APT攻击场景，并使用CamFlow来捕获whole-system provenance，其中每个场景运行了3天。

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
SC-1	Benign	125	265,424	975,226	64
	Attack	25	257,156	957,968	12
SC-2	Benign	125	238,338	911,153	59
	Attack	25	243,658	949,887	12

TABLE VI: Characteristics of the datasets used in the supply-chain APT attack experiments.

将125个良性图分成五组进行5折交叉验证，来为正常行为进行建模。下图是实验的设置和结果：

	Batch Size	Sketch Size	Hop Count	Decay Factor	Sketch Interval
Baseline	6,000	2,000	3	0.02	3,000

TABLE VII: UNICORN configurations for supply-chain APT attack scenarios.

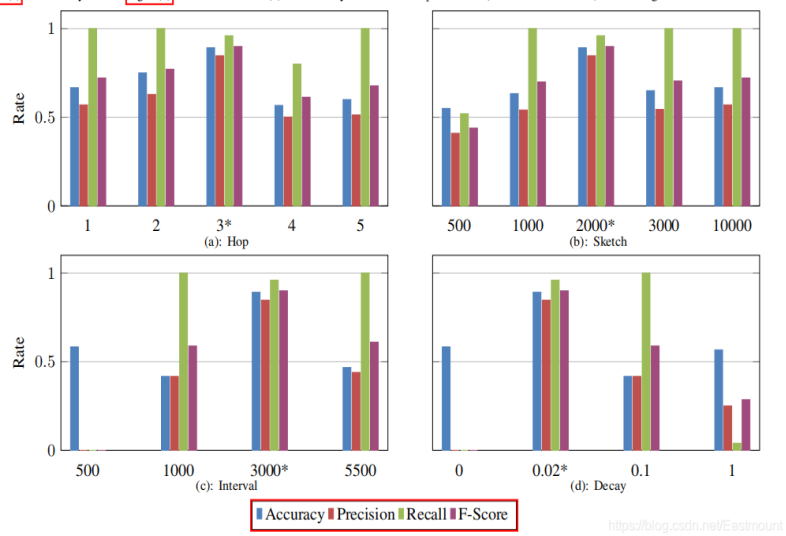
Experiment	Precision	Recall	Accuracy	F-Score
SC-1	0.85	0.96	0.90	0.90
SC-2	0.75	0.80	0.77	0.78

TABLE VIII: Experimental results of the supply-chain APT attack scenarios.

D.参数分析

下面通过调整各个Baseline参数来观察性能变化。

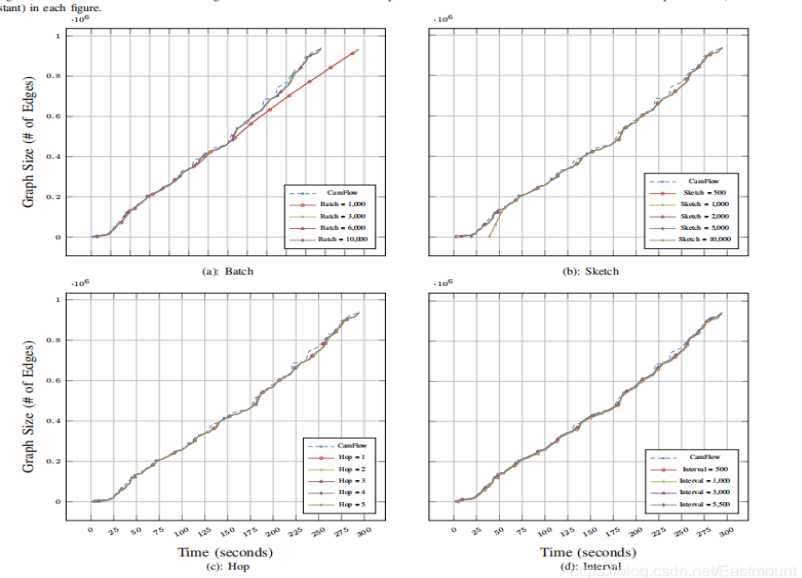
Fig. 3: Detection performance (precision, recall, accuracy, and F-score) with varying hop counts (Fig. 3(a)), sketch sizes (Fig. 3(b)), intervals of sketch generatio (Fig. 3(c)), and decay factor (Fig. 3(d)). Baseline values (*) are used by the controlled parameters (that remain constant) in each figure.



E.处理速度

下图表示了随时间推移处理的边缘总数，以量化UNICORN的处理速度。CamFlow线（蓝色）表示捕获系统生成的边总数，其他线与该线越近，说明运行时性能越好，这意味着UNICORN与捕获系统CamFlow保持一致。

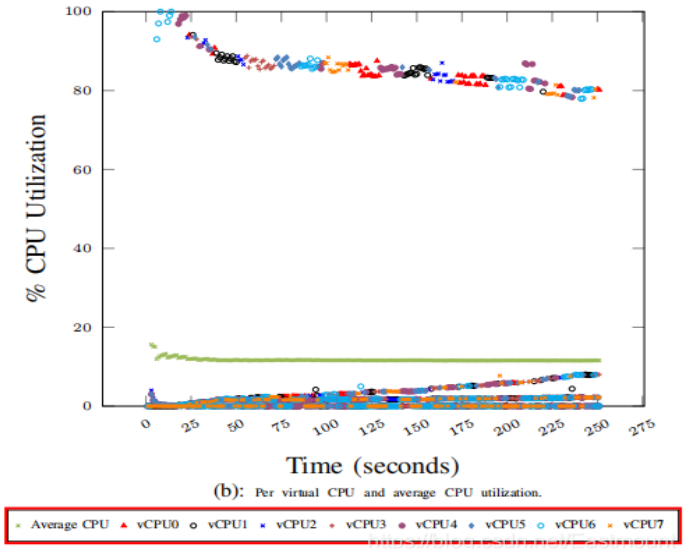
Fig. 4: Total number of processed edges over time (in seconds) in the SC-1 experimental workload with varying batch sizes (Fig. 4(a)), sketch sizes (Fig. 4(b)), hop counts (Fig. 4(c)), and intervals of sketch generation (Fig. 4(d)). Dashed blue line represents the speed of graph edges streamed into UNICORN for analysis. Triangle maroon baseline has the same configurations as those used in our experiments and indicates the values of the controlled parameters (that remain constant) in each figure.



总体而言，上图表明UNICORN运行时对这些参数相对来说不敏感，这意味着UNICORN可以使用针对检测进行精度优化的参数，执行实时入侵检测。

F.CPU & 内存使用

我们针对工作量相对较大（即CI执行内核编译）的系统评估UNICORN的CPU使用率和内存开销。 实验表明，UNICORN具有较低的CPU使用率和内存开销。



(b): Per virtual CPU and average CPU utilization.

Configuration Parameter	Parameter Value	Max Memory Usage (MB)
Hop Count	R = 1	562
	R = 2	624
	R = 3	687
	R = 4	749
	R = 5	812
Sketch Size	$\|S\|$ = 500	312
	$\|S\|$ = 1,000	437
	$\|S\|$ = 2,000	687
	$\|S\|$ = 5,000	1,374
	$\|S\|$ = 10,000	2,498

TABLE IX: Memory overheads with varying hop counts and sketch sizes. The highlighted configurations gave the best detection performance.

Ⅶ.讨论和局限性

1.基于异常的检测

- 本文假设在UNICORN进行正常行为建模期间，系统是安全的；其次，本文假设存在详尽的、有限数量的系统行为模式，而且即使在运行过程中没有检测到全部，也检测到了大多数，因此如果UNICORN检测到未知的正常行为模式，就会产生误报。
- 攻击者试图将恶意行为转向已学习过的模型，以逃避检测，类似于著名的模拟攻击。然而，对溯源图或UNICORN概要图进行模拟攻击比对系统调用序列更具挑战，因为溯源图包含复杂的结构信息，很难在不影响攻击的情况下进行攻击。此外，UNICORN的一致加权抽样方法将概要图生成随机化，这使得很难保证模拟溯源图的低维投影将接近学习到的正常簇。
- 另外，像其他基于异常的系统一样，独角兽需要足够的良性行为痕迹来学习行为模型。UNICORN监视系统的起点是和两性模型建模的起点是一样的。但是如果因为系统发生了错误而恢复到之前的状态时，就会导致系统状态与模型不匹配。解决这种问题的方法是在系统创建快照的同时保存其模型状态，当系统还原快照时，UNICORN将还原相应的模型状态。
- UNICORN需要定期的重新训练模型。

2.错误的警报

当正常的系统行为发生变化时，UNICORN可能会发出假阳性警报，因为它不会动态地调整其模型（以避免攻击者中毒）。错误警报问题并不是独角兽所独有的。UNICORN使用概念漂移(IV-B)，建模系统演化，部分缓解了这个问题。

3.图分析

需要对每个系统调整参数来提升检测性能。本文使用OpenTuner来自动调整。在本文的实验中，对于大多数数据集而言，都可以使用相同的参数。

4.异质主机活动

在测试中，我们观察到独角兽在具有同质正常活动的领域表现得非常好。一些主机只会执行一些预先定义好的任务，UNICORN对这样环境下的主机检测效果较好。然而，一些主机会有各种各样的异质性行为，UNICORN没有考虑这一类主机的安全性。

5.更大的交叉评估

我们强调，将UNICORN与其他现有的IDS（其中大部分是基于系统的）进行比较很困难，原因有：

- A) 许多IDS不是开源的；
- B) 现有的公共 IDS 数据集要么已经过时，要么需要从例如系统调用跟踪转换为数据源，这是具有挑战性的，有时是不可能的（由于缺乏信息）；
- C) 创建私人数据集的系统只是表面地描述它们的实验过程，因此很难公平地重现溯源数据的实验。我们认为，这样的基础研究是有价值的，我们计划在未来继续开展这样的工作。

Ⅷ.相关研究

1.基于主机的动态入侵检测

起初，IDS仅仅依靠系统调用来进行建模，但随着攻击技术的提升，检测的准确度也随之下降。所以下一代的IDS在系统调用中加入和“状态”来提供上下文信息。UNICORN的方法完全不同，因为传统的系统调用方法不太适合APT攻击，而基于图表示和分析在检测APT攻击上具有良好的表现。一方面避免了昂贵的控制流构造和状态转换自动机，另一方面准确地描述和建模系统中数据对象之间的复杂关系，用于上下文化异常检测。

据我们所知，尽管有些系统会从审计日志中生成类似出处的图表 [83]，但UNICORN是第一个通过对本地整个系统溯源进行运行时分析来检测入侵的系统。

2.基于图的异常检测

StreamSpot分析流媒体信息图以坚持异常活动，但图特征受到局部约束，而 UNICORN的图特征执行上下文。此外，在APT场景中，攻击者可以操纵模型持久攻击，以逐渐和缓慢地改变系统行为，从而避免被检测。UNICORN充分利用它不断总结进化图的能力，建模它所监控的系统执行的相应演化。

3.基于Provenance的安全分析

随着APT攻击越来越突出，许多系统利用数据来源进行APT攻击分析。现有方法包括ProTracer、CamQuery、Holmes、SLUETH、Poirot、SAQL。UNICORN不同于传统的基于规则的系统，它是一个不需要专家知识的异常检测系统。

Ⅸ.结论

本文提出了UNICORN，一种实时异常检测系统，它利用整个系统的数据溯源来检测高级持续威胁。UNICORN通过结构化的溯源图（Provenance Graph）对系统行为进行建模，揭示了系统对象之间的因果关系，并在其流到分析管道中时，对其进行有效地汇总来考虑整个图。 我们的评估表明，由此产生的演化模型可以成功地检测出从不同审计系统捕获的各种APT攻击，包括真实的APT活动，并且具有高精度和低误报率。

这篇文章就写到这里，希望对您有所帮助。由于作者英语实在太差，论文的水平也很低，写得不好的地方还请海涵和批评。同时，也欢迎大家讨论，继续加油！感恩遇见，且行且珍惜。

(By:Eastmount 2021-07-20 周一夜于武汉 http://blog.csdn.net/eastmount/)
