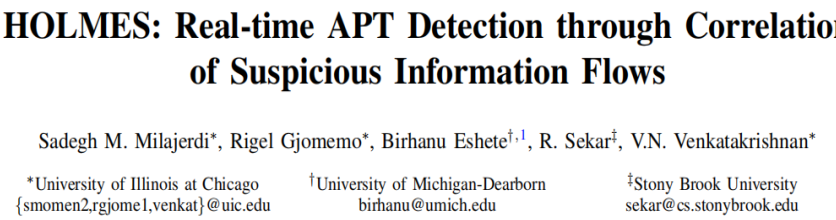
[论文阅读] (09)S&P2019 HOLMES Real-time APT Detection through Correlation of Suspicious Information Flow

《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。这篇文章将详细介绍S&P2019《HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows》，基于可疑信息流的实时APT检测。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!



HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows

Sadegh M. Milajerdi*, Rigel Gjomemo*, Birhanu Eshete†,1, R. Sekar‡, V.N. Venkatakrishnan*

*University of Illinois at Chicago {smomen2,rgjome1,venkat}@uic.edu

†University of Michigan-Dearborn birhanu@umich.edu

‡Stony Brook University sekar@cs.stonybrook.edu

原文作者：Sadegh M. Milajerdi, Rigel Gjomemo, Birhanu Eshete, R. Sekar, V.N. Venkatakrishnan
原文标题：HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows
原文链接：https://arxiv.org/pdf/1810.01594.pdf
发表会议：S&P2019（IEEE Symposium on Security and Privacy）
参考文献：感谢两位老师，推荐大家关注学术安全圈

- https://blog.csdn.net/Sc0fie1d/article/details/103833148
- https://mp.weixin.qq.com/s/QifnwOzx19BFIHo7adgrVg
- https://www.secrss.com/articles/14488

前文赏析：

文章目录

摘要

本文提出了一种实现了检测高级持久性威胁（Advanced Persistent Threat，APT）新的方法，即HOLMES系统。HOLMES的灵感来自现实世界中APT活动的一些共同目标。简而言之，HOLMES旨在产生一个检测信号，以表明存在的一系列协同活动都是APT活动的一部分。本文方法要解决的主要挑战之一是开发一套技术，从而检测信号的鲁棒性和可靠性。

- 在高级层（high-level），我们开发的技术有效地利用了攻击者活动期间出现的可疑信息流间的相关性。
- 除检测能力外，HOLMES还能够生成一个高级图（high-level graph），以实时总结攻击者的行为。该图能给分析人员提供有效的网络响应。
- 本文方法对真实APT评估表明，HOLMES在检测APT攻击是具有高精确率和低误报率。
- 由HOLMES生成的紧凑高层图有效地总结了一个正在进行的攻击事件，并可以帮助实时的网络响应操作。

总之，本文构建了一个可以实时检测高级持续性威胁（APT）的系统——HOLMES。该方法有效利用了攻击者活动期间出现的可疑信息流间的相关性。通过实时汇总攻击者的行为，产生基于杀伤链模型（kill chain）的高级图（high-level graph），实现了将复杂的数据映射为简洁的APT攻击阶段，从而有利于防御者更加直观地发现威胁并进行防御。HOLMES生成的简洁高级图有效总结了正在进行的攻击活动，并可通过可疑信息流的关联来协助实时网络响应工作。

***Abstract*—In this paper, we present HOLMES, a system that implements a new approach to the detection of Advanced and Persistent Threats (APTs). HOLMES is inspired by several case studies of real-world APTs that highlight some common goals of APT actors. In a nutshell, HOLMES aims to produce a detection signal that indicates the presence of a coordinated set of activities that are part of an APT campaign. One of the main challenges addressed by our approach involves developing a suite of techniques that make the detection signal robust and reliable. At a high-level, the techniques we develop effectively leverage the *correlation between suspicious information flows* that arise during an attacker campaign. In addition to its detection capability, HOLMES is also able to generate a high-level graph that summarizes the attacker's actions in real-time. This graph can be used by an analyst for an effective cyber response. An evaluation of our approach against some real-world APTs indicates that HOLMES can detect APT campaigns with high precision and low false alarm rate. The compact high-level graphs produced by HOLMES effectively summarizes an ongoing attack campaign and can assist real-time cyber-response operations.**

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描述。

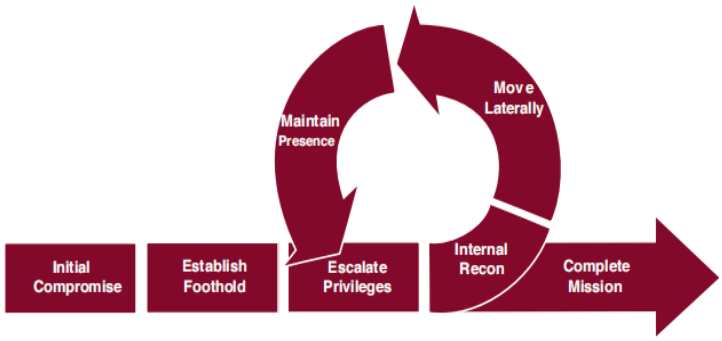
在最早关于APT详细报告（FireEye发布的APT1）中，安全公司Mandiant披露了全球APT参与者的目标和活动。这些活动至少从包括141个组织的不同行业中窃取数百兆字节的敏感数据（包括商业计划、技术蓝图和测试结果）。他们估计评估组织中恶意软件的平均持续时间为365天。从那以后，全球范围内出现了越来越多涉及到一些强大攻击者的APT攻击事件，包括国家活动。

APT 攻击的特征可归纳为以下三点：

- **针对性**
  通常针对特定目标的重要价值资产，一般JG、能源、金融、部门最容易遭到APT攻击。并且针对收集到的常用软件、常用防御策略与产品、内部网络部署等信息，攻击者能编写可以绕过目标系统现有防护体系检查的攻击代码。
- **持续性**
  为了长期控制重要目标获取更多利益，攻击者通过隐藏实现长期潜伏，同时攻击处于动态发展以应对新的系统漏洞及防御体系的更新，并通过外部被控制的命令与控制（Command and Control，C＆C）服务器与目标系统保持通信以及传输数据。
- **隐蔽性**
  为了避免被安全防护系统检测到，APT 攻击代码的编写者使用各种伪装、隐藏手段，通过修改系统程序，隐藏病毒进程、隐藏文件、隐藏目录的方式实现长期潜伏；通过对恶意程序压缩、加密、变体及加壳等技术手段降低其被检测到的概率；运用动态域名解析实现C＆C服务器的隐藏与长期生存;通过合法的加密数据通道、加密技术或信息隐藏技术隐蔽地传输数据。

1.杀伤链模型

了解APT攻击者的动机和行动在应对这些威胁挑战中发挥着重要作用。为了进一步理解APT攻击，Mandiant报告还提供了一个APT生命周期模型（如图1），也被称为杀伤链（kill-chain），它可以让人们了解APT的步骤是如何共同实现目标的。



一个典型的APT攻击包含以下阶段：

- 初始入侵（Initial Compromise）：例如网站挂马或鱼叉式钓鱼攻击
- 建落脚点（Establish Foothold）：安装木马后门
- 权限提升（Escalation Privilege）：漏洞利用
- 内部侦查（Internal Reconnaissance）：内部侦察目标系统信息
- 横向移动（Move Laterally）：通过网络的横向移动渗透
- 保持存在（Maintain Presence）：Command and Control（C&C）、Remote Access Trojans(RATs)
- 完成任务（Complete Mission）：泄露敏感信息

APT攻击者的目标是获取和泄露高度敏感的信息，例如特定专有技术的源代码；或者通过破坏高完整性的资源来使目标受到损害，例如，被Stuxnet蠕虫感染了的PLC，这个目标主要是通过符合图1所示的杀伤链来实现的。

简而言之，杀伤链提供了一个理解和映射APT攻击者动机、目标和行动的参考。

2.问题说明

企业中现有的IDS/IPS系统可能会检测并生成主机上可疑事件的警报。然而，结合这些低级警报，以获得正在运行的APT活动的高级图仍然是一个重大挑战。

如今，告警关联通常使用安全信息和事件管理（Security Information and Event Management，SIEM）系统来执行，如Splunk、Logrheth和IBM Qradar。这些系统从多个来源收集日志事件和警告并将它们关联起来。这种相关性通常使用现成的指标，例如时间戳。这些关联方法是有用的，但它们通常缺乏：

- **(a) 对从报警到实际攻击入侵之间存的复杂关系缺乏可解释性。**
- **(b) 将不同主机、时间跨度极长（数周或数月）的攻击阶段组合在一起，精确性不高。**

本文解决的主要问题是——实时检测正在运行的APT活动（由长期跨主机的许多不同阶段组成），并根据来自企业的主机日志（host logs）和IPS警报向分析人员提供攻击场景的高级说明。

这个问题有三个主要方面，它们如下：

- **警报生成（Alert generation）：如何生成能反映攻击者行为的报警、并降低噪声**
  从主机低级事件开始跟踪，我们必须有效地生成警报。此外，必须确保不会产生大量的噪声警报。
- **警报相关性（Alert correlation）：如何有效地进行报警关联**
  这里的挑战是将攻击者多个活动的警报组合为可靠信号，表明存在正在进行的APT活动。
- **攻击场景演示（Attack scenario presentation）：如何呈现攻击场景**
  正在运行的APT活动指标需要传达给网络分析师。为了有效，这种沟通必须很直观，需要在高层总结攻击，以便分析人员迅速意识到这次活动的范围和规模。

我们注意到，虽然攻击者的手段多种多样，但是映射到高层次的攻击步骤之后，其抽象攻击模式基本不变，基于此可以将复杂的数据映射到具体的攻击阶段。

3.本文贡献

为了解决上述所有问题，本文提出了一个称为HOLMES的系统。HOLMES以主机审计数据开始（如Linux审计或Windows ETW数据），并生成一个检测信号，绘制正在进行的APT活动的阶段。在高层次上，HOLMES将APT杀伤链作为解决APT检测上述三个方面所涉及技术挑战的关键参考。

> ETW是Event Tracing for Windows的简称，它是Windows提供的原生的事件跟踪日志系统。由于采用内核（Kernel）层面的缓冲和日志记录机制，所以ETW提供了一种非常高效的事件跟踪日志解决方案。

下面将介绍关键思想及其意义，并在第三部分（Ⅲ.方法概述）中有一个详细的技术描述。本文主要贡献如下：

- **首先，HOLMES旨在将主机日志中发现的活动以及在企业中发现的任何警报信息直接映射到杀伤链中**。这种设计选择允许HOLMES在语义上生成接近APT攻击者的活动步骤（战术、技术和程序，Tactics, Techniques and Procedures，TTPs）的警报。通过这样做，HOLMES将警报生成过程提升到攻击事件的步骤级别，而不是在低等级审计日志中刻画它们。因此，我们解决了在生成重要警报方面的一个重要挑战。在实验中，我们发现为期5天的审计日志集合包含了大约300万个低级别事件，而HOLMES从中提取了86个可疑的活动步骤。
- **HOLMES第二个重要思想是使用系统中低级实体（文件、进程等）之间的信息流作为警报关联的基础**。请注意，杀伤链中的内部侦察步骤取决于一个成功的初始入侵和建立落脚点。特别是侦察步骤通常使用攻击者在建立落脚点期间安装的命令和控制代理（进程）来启动，从而显示这两个阶段所涉及过程之间的流动。此外，内部侦察通常涉及到运行在落脚点建立阶段下载的恶意软件（文件），以说明文件到进程的流程。同样，成功的横向移动和过滤阶段使用内部侦察阶段收集的数据。因此，通过检测与APT步骤相关的低等级事件并使用信息流链接它们，可以构建APT攻击者所使用的新兴杀伤链。
- **HOLMES第三个主要贡献是开发了一个高级场景图（high-level scenario graph，HSG）**。HSG的节点对应于TTP，而边表示TTP中涉及的实体之间的信息流。HSG为高可信地检测APT活动提供了基础。为此，我们开发了几个新的想法。① 首先是HSG中祖先覆盖的概念。我们将展示这个概念如何帮助评估HSG节点之间的依赖关系强度。然后可以删除弱依赖关系，以消除许多假警报。② 我们开发了降噪技术，进一步淡化了已知与良性活动相关的依赖性。③ 我们开发了排序和优先级技术来删除与APT活动无关的大多数节点和边。在第IV-D、IV-E和IV-F节中详细描述了这些步骤。使用这些技术，我们证明了霍尔姆斯能够清楚地区分攻击场景和良性场景。
- **最后，HSG在任何时候都提供了非常紧凑的攻击事件总结，从而为攻击理解做出了重要贡献**。例如，从一个包含1000万个审计记录的数据集开始，我们能够使用一个仅包含16个节点的图来总结一个高级攻击事件。网络分析师可以使用所提出的HSG相对容易地快速推断攻击的大局（范围和规模）。

4.评估

HOLMES通过DARPA透明计算程序所生成的数据进行评估，该程序是由一个专业红队在不同平台组成的网络上模拟的多个网络攻击。我们为Linux、FreeBSD和Windows实现了适当的系统审核数据解析器，以处理其审计数据并转换为通用的数据表示和分析格式。使用系统审计数据的优点是，它是一个可靠的信息来源，并且没有未经授权的篡改（在未妥协的内核的威胁模型下）。

- DARPA Transparent Computing program

在9个真实APT攻击场景中评估HOLMES，以及将它作为一个实时入侵检测工具实验两周，实验显示HOLMES能够明确区分攻击场景和良性场景，可以发现网络攻击与高精度和召回（Sec. VI）。

Ⅱ.运行示例

在本节中，我们将介绍本文使用的一个运行示例来说明本文方法。这个例子表示了作为政府机构组织（特别是美国DARPA）研究计划中的一部分红队攻击。在这次攻击中，存在漏洞的Nginx Web服务器在FreeBSD系统上运行。其操作（系统调用）在系统审计日志中被捕

获。从这个审计数据中构建了一个溯源图（provenance graph），其片段如图2所示。该图中的节点表示系统实体（entities），如进程（表示为矩形）、文件（椭圆）、网络连接（菱形）、内存对象（五边形）和用户（星形）。边对应系统调用，并且面向信息流和/或因果关系的方向。请注意，使用参考文献[23]中描述的（优化的）节点版本控制技术，我们的溯源图已被呈现为无循环的。

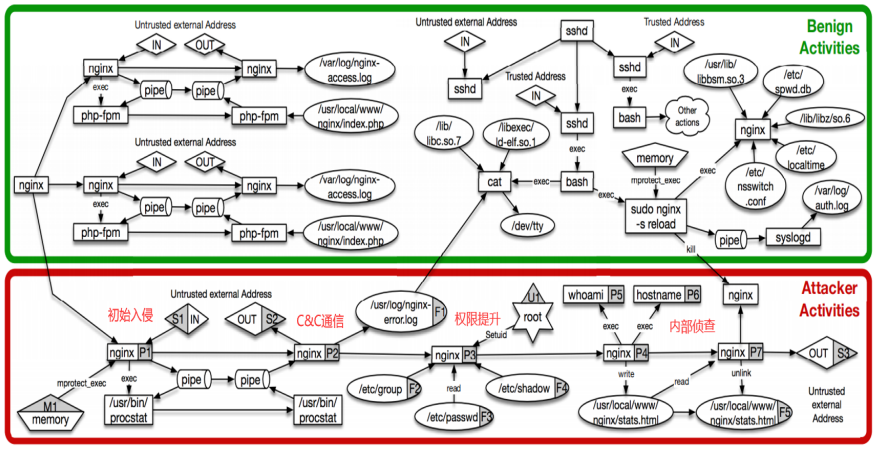


Fig. 2. Provenance Graph of the Running Example.

攻击者的目标是从系统中提取敏感信息。图2底部描绘了攻击者的活动，主要包括以下步骤：

- **初始入侵（Initial Compromise）**。 攻击者在监听80端口的套接字(S1) 上发送恶意有效载荷。因此，Nginx 使其内存区域 (M1) 的某些部分可执行。 接下来，攻击者通过使用反射自加载漏洞来控制Nginx进程。
- **C&C通信（C&C Communications）**。受损的Nginx进程与C&C服务器建立连接(S2)，以接收来自攻击者的命令。
- **特权提升（Privilege Escalation）**。攻击者利用现有漏洞将Nginx的特权升级为根目录(U1)。
- **内部侦察（Internal Reconnaissance）**。接下来，攻击者会发出诸如whoami（P5）和主机名（P6）等命令。红队使用这些命令来模拟对专有数据的访问。攻击者还会读取用户名和密码哈列（F2、F3、F4），并将所有这些信息写入临时文件。
- **窃取信息（Exfifiltration）**。接下来，攻击者将包含收集信息的文件传输到其机器（S3）。
- **清除痕迹（Cleanup）**。攻击的最后一步，攻击者将删除临时文件（F5）以清除任何攻击残余

本示例说明了下面描述的许多关键挑战：

- 攻击隐蔽（Stealthy Attacks）：该统计系统留下痕迹最少，有效载荷在现有的Nginx进程中运行，检测这种隐蔽攻击是非常具有挑战性的，因为攻击活动与正常的系统操作可以无缝融合。
- 大海捞针（Needle in a haystack）：一个主机每天也能生成数千万个事件，除小部分外（通常不到0.01%）都属于良性活动，很难检测到这种罕见事件。
- 实时检测（Real-time detection）：假设HOLMES与网络响应系统一起使用，因此需要在几秒钟内检测和总结一场正在进行的事件。实时检测给HOLMES使用的技术带来了额外的挑战和限制。

尽管能无缝地融入了良性的背景活动，但在攻击中有两个因素很突出。首先，攻击步骤实现与某些APT阶段能力对应。其次，攻击活动通过信息流被连接起来。在下一节中，我们描述HOLMES方法的两个关键观察结果。

Ⅲ.方法概述

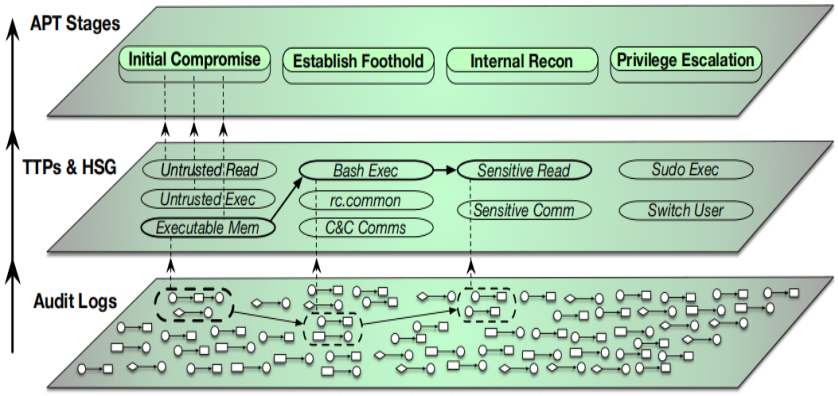
我们方法背后的核心认知是，尽管具体的攻击阶段在不同的APT之间可能存在很大差异，但高级APT行为通常符合第一部分中引入的杀伤链模型。我们对数百个APT报告的分析表明，大多数APT包括这些阶段的子集。更重要的是观察到这些阶段存在因果关系，这种联系是攻击正在展开的一个主要迹象。

注意，每个APT阶段的具体表现形式可能变化，例如初始入侵可以通过使用户执行恶意文件的网站挂马或鱼叉式网络钓鱼攻击来执行。无论如何，APT阶段本身代表了对攻击者意图的高层次抽象，因此即使攻击者使用的操作策略因APT而异，它们也必然会表现出来。此外，由于APT攻击阶段在逻辑上相互依赖，因此必然在它们之间存在信息流或因果关系，例如信息窃取取决于内部侦察去收集敏感数据。

因此，研究问题是我们能否将检测建立在以下基础上：

- APT最基本的高层次行为阶段
- 并且这些阶段之间的信息流依赖关系

回答这个问题的一个主要挑战是：**低级别审计数据与攻击者目标、意图和能力与高级杀伤链（kill-chain）视角之间存在巨大的语义差距。**



缩小语义鸿沟（Bridging the Semantic Gap）
为了弥合低级系统调用视角和高级杀伤链视角之间的语义差距，构建了一个中间层，如上图3所示。映射到这个中间层是基于MITRE的ATT&CK框架， 它描述了近200种行为模式，定义为在野观察到的战术、技术和程序（Tactics,Techniques and Procedures，TTPs）。

每个TTP都定义了一种实现特定高级功能的可能方法。例如，可以使用11个不同的TTP实现在受损Linux系统中的持久化能力，每个TTP代表ATT&CK框架中可能的低级活动序列，例如rootkit的安装、修改引导脚本等。这些较低级别的操作更接近审计日志的抽象级别，因此可以根据起源图中的节点和边来描述TTP。

技术挑战（Technical challenges）

- 有效地将低级事件流与TTPs进行匹配
- 检测攻击步骤之间的相关性
- 减少误报

(1) 通过一些创新设计来解决这些挑战。为了使低级别的数据有效地映射到TTPs，我们将审计日志表示为主内存中的 有向溯源图（directed provenance graph），并且使用系统中低级别实体（如文件、进程等）之间的信息流（Information Flow）依赖关系作为基础，来进行警报关联。TTP被指定为利用这些依赖关系的模式。

(2) 为了检测到攻击步骤之间的关联性，本文开发了 高级别场景图（high-level scenario graph, HSG），HSG的节点对应于TTPs，边表示TTPs实体之间的信息流和依赖关系。在HSG中定义了以下概念：

- Ancestral Cover：描述节点之间的依赖关系
- Noise Reduction：降低与良性活动相关的依赖性
- Ranking and Prioritization：修剪与APT无关的节点和边

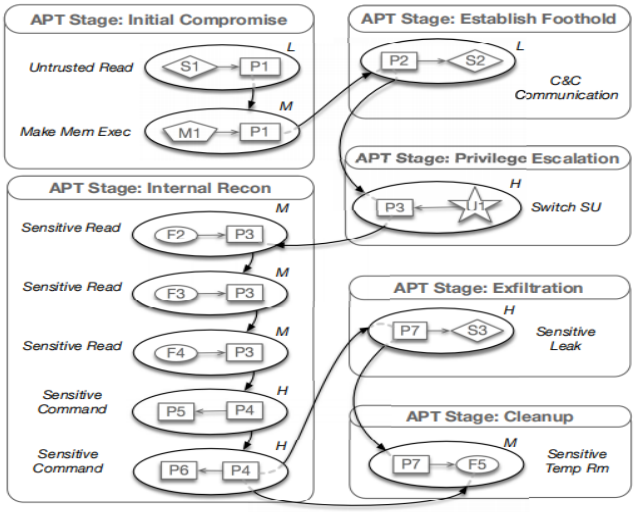


Fig. 5. High-Level Scenario Graph for the Running Example.

(3) 为了减少误报，本文提出的方法是：学习可能会产生误报的良性TTPs模式，采用启发式算法；根据其严重程度为图中的节点和路径分配权重，以便可以对HSG进行排序，并将排序最高的HSG呈现给分析人员。

总之，APT高级阶段使用一套可以从审计数据中观察到通用战术的操作。这些观察结果提供了一些恶意活动可能正在开展的证据。**因此，HOLMES的主要工作是收集证据，推断它们之间的相关性，并利用这些相关性绘制出整个攻击事件。**

IV.系统设计

系统设计的主要任务是建立威胁模型，因此如何建立审计系统以及如何生成日志数据不在本系统设计的范围之内。此外，本文假设初始系统是安全可靠地。

A.数据收集和表示

该系统使用的审计日志来源于许多主机的不同操作系统：

- Linux auditd：审计数据
- BSD dtrace
- Windows ETW：事件跟踪日志系统

原始审核数据被收集并处理成与操作系统无关的格式。HOLMES接受的输入的事件（event）包括：

- principals ： 用户
- files：如I/O操作、文件创建、文件所有权、权限
- memory：如mprotect和mmap
- processes：如进程创建和权限更改
- network connections：网络连接相关事件

数据表示为称为溯源图（provenance graph）。

- 图的节点包括主体（进程）和对象（文件、通道、套接字）
- 边表示这些实体之间的依赖关系，并使用事件名称进行注释

该设计与前人的工作有两个不同之处：

- provenance graph是不断变化的：当一条边改变了节点的依赖关系，一个新的节点将会被创建并替换旧节点。这种“版本化”的方法使

得在不改变分析结果的情况下可以对图进行修剪，而且这种versioned graph是无环的，这可以简化许多图算法。

- 另一个不同之处是provenance graph是存储在主存中的，每个事件所占空间小于5bytes，这种表示方式可以在较长的时间段内实时消耗事件和构建起源图。

B.TTP规划

TTP规范提供了低级别审计事件和高级别APT阶段之间的映射，因此这是本文所提出的方法的核心。TTP代表了具体审计日志和高级APT步骤之间的中间抽象层。具体而言，主要依靠两种技术将审计日志数据提升到该中间层：

- (a) 以安全相关事件的溯源图形式的OS中性表示；
- (b) 使用TTPs中涉及的实体之间的信息流依赖关系。

总之，这些技术实现了高级别的恶意行为规范，这些规范在很大程度上独立于许多TTP细节，例如使用的特定系统调用、恶意软件名称、创建的中间文件以及用于创建它们的程序等。信息流依赖的使用对于通过使用良性系统进程来实现其目标并隐藏其活动的隐秘APT的检测至关重要。

- 先决条件（Prerequisites）：表现为因果关系和和信息流的形式。

最后，为了高效匹配TTP且不使用回溯技术（backtracking）。我们发现，大多数TTP可以在我们的框架中使用单个事件进行建模，并对所涉及的主题和对象有附加的先决条件。表4显示了TTP规范示例。

APT Stage	TTP	Event Family	Events	Severity	Prerequisites
$Initial\_Compromise(P)$	$Untrusted\_Read(S,P)$	READ	FileIoRead (Windows), read/pread/readv/preadv (Linux,BSD)	L	$S.ip \notin \{Trusted\_IP\_Addresses\}$
	$Make\_Mem\_Exec(P,M)$	MPROTECT	VirtualAlloc (Windows), mprotect (Linux,BSD)	M	$\$PROT\_EXEC\$ \in M.flags \wedge \exists\, Untrusted\_Read(?,P') : path\_factor(P',P) <= path\_thres$
$Establish\_Foothold(P)$	$Shell\_Exec(F,P)$	EXEC	ProcessStart (Windows), execve/fexecve (Linux,BSD)	M	$F.path \in \{Command\_Line\_Utilities\} \wedge \exists\, Initial\_Compromise(P') : path\_factor(P',P) <= path\_thres$

TABLE 4. Example TTPs. In the Severity column, L=Low, M=Moderate, H=High, C=Critical. Entity types are shown by the characters: P=Process, F=File, S=Socket, M=Memory, U=User.

其中，第一列表示APT阶段；第二列表示相关的TTP名称和所涉及的实体；第三列指定与TTP相关联的事件家族；第四列列出了第三列对应的Event Family中所包含的事件；最后一列既可以表示该TTP的先决条件，也可以表示该TTP的先决TTP。先决条件既可以包含于两个TTPs实体之间的关系，而且能够捕获两个TTP拥有一个共同parent的条件。应用先决条件，我们可以减少误报。

C.HSG构建

下图显示了一个HSG。该图的椭圆节点表示匹配的TTP。在每个椭圆内部，用灰色表示匹配的起源图实体。为了便于说明，还包括TTP的名称，每个TTP所属的APT阶段以及每个TTP的严重性级别（低中高）。图的边代表不同TTP之间的先决条件。完成两个实体之间的路径的虚线表示前提条件。例如，MakeMemExec TTP将Untrusted_Read TTP作为前提，由两个节点之间的边表示。

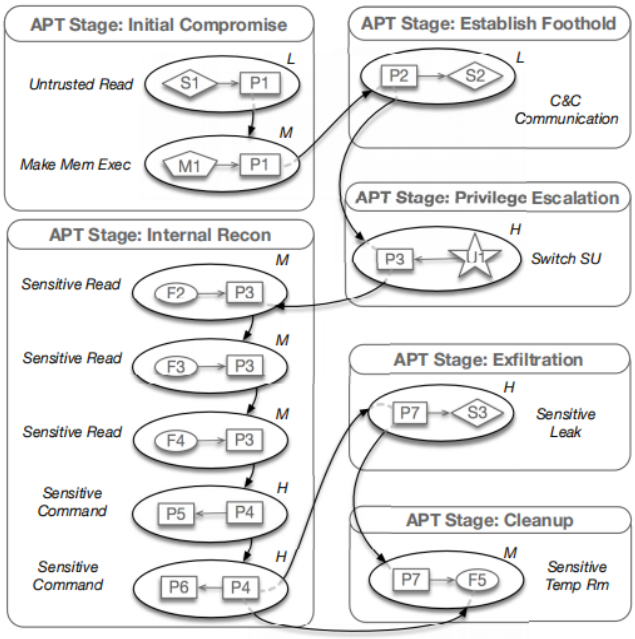


Fig. 5. High-Level Scenario Graph for the Running Example.

HSG的构建主要是由先决条件驱动的。如果一个TTP的所有先决条件都满足了，那么这个TTP就会被匹配并且被添加到HSG中。这可以随时减少HSG中TTP的数量，从而可以进行复杂的分析而不会影响实时性能。

D.避免虚假依赖

虚假依赖（spurious dependencies），即活动的攻击者不感兴趣或无关的依赖。我们应该优先考虑较强的依赖关系，尽可能地对弱依赖关系进行剪枝。

(1) 虚假依赖即与攻击活动不相关的依赖

- 例如，进程nginx(P2) 写入文件/usr/log/nginx-error.log，然后cat进程读取该文件。但是，即使cat和日志文件之间存在依赖关系，cat也与攻击无关，并通过ssh独立调用。通常，考虑由攻击活动生成的二次衍生的任何进程，例如一个日志滚动系统，它会复制包含攻击者进程生成的部分条目的日志文件。这些进程虽然代表良性的背景活动，但会在起源图中标记为依赖于攻击者的进程。如果没有及时修剪这些虚假的依赖关系，可能会出现依赖关系爆炸，这会极大地增加HSG的整体大小。所以我们的目标是留下强依赖，修剪掉弱依赖。

为了将上述讨论推广到可能存在多个入侵进程的情况，引入了一个信息流 f上所有进程的祖先实体覆盖（ancestral cover）。

(2) 祖先覆盖 ancestral cover AC(f)

- f 表示一条信息流路径
- 仅针对在f中的所有进程，不影响非进程节点

$$\forall p \in f \ \exists a \in AC(f) \ \ a = p \text{ or } a \text{ is an ancestor of } p$$

(3) 最小祖先覆盖 minimum ancestral cover $AC_{min}(f)$

- $AC_{min}(f)$ 表示攻击者如果想要控制整条信息流路径，那么所需要拿下的最少祖先节点数

(4) 路径因子 $path\_factor(N_1, N_2)$

- 考虑到 N_1 到 N_2 的所有信息流路径 $f_1, ..., f_n$，令 m_i 表示 f_i 的最小祖先覆盖。那么，$path\_factor(N_1, N_2)$** 为 $m_1, ..., m_n$ 中的最小值。
- path_factor的使用极大地减轻了优先处理受攻击影响信息流导致的依赖性爆炸(dependency explosion)。

> PS：该部分细节太多，建议大家去读原文。

E.降噪

攻击检测和取证的审计日志分析中的一个挑战是存在噪音，即与TTP规则匹配的良性事件。诸如浏览器、Web服务器和ssh守护进程之类的长周期进程不时会触发TTP匹配。为了减少这些误报，结合了基于训练数据的降噪规则。利用两个概念：

- 良性先决条件匹配
- 良性数据流数量

(1) 良性先决条件匹配
对于每个进程，我们的系统运行在良性的环境中并学习经常遇到的先决条件。在运行过程中，一旦某个被触发的TTP的先决条件与我们在训练中得到的条件（benign）相匹配，我们便忽略该TTP。

这种方法可能会导致漏报（false negative），比如一个恶意事件如果匹配到了良性环境中训练得到的良性先决条件，那么这个恶意事件就可能会被忽略。例如，即便没有任何攻击，在nginx的启动阶段也会执行read /etc/passwd，然而如果我们将该行为列入白名单的话，如果有恶意事件是读取改文件，那么该恶意事件就会被忽略。

(2) 良性数据流数量
为了解决上述问题，提出了用字节转移度量的数据流数量的概念。比如从/etc/passwd到nginx的数据流数量等于/etc/passwd文件的大小。因此，如果观察到更多的字节从/etc/passwd流向nginx，那么该信息流可能是攻击的一部分。为了确定信息流数量的临界点，我们在一段时间的良性活动中观察process-file和process-socket活动。

F.信息关联和检测

对每一个HSG定义 `严重程度评分（severity score）` ，据此来确定一个HSG会构成APT攻击的可能性。这项工作分两步进行，如下所述：

(1) 威胁元组（Threat Tuple）
首先通过与相应的HSG关联的抽象威胁元组来表示攻击者在活动中的进度。特别是对于每个HSG，威胁元组由7个元素<S1, S2, S3, …, S7>组成，其中每个Si对应于APT攻击阶段的威胁程度。一个APT阶段通常会包含许多APT，选取威胁程度最高的来构成威胁元组。

由于属于某一APT阶段的不同TTP可能具有不同的严重性级别，因此通常有多个候选者可供选择。在这些候选人中选择最严重级别是很自然的。例如，与图5的HSG相关联的威胁元组是<M, L, H, H, —, H, M>。这个元组包含6个条目，因为它所匹配的TTP属于6个不同APT阶段。这些条目是根据杀伤链中APT级别的顺序排列的。例如元组的第一个条目是M，因为图中属于Initial_Reconnaissance 最严重的TTP具有严重性M。

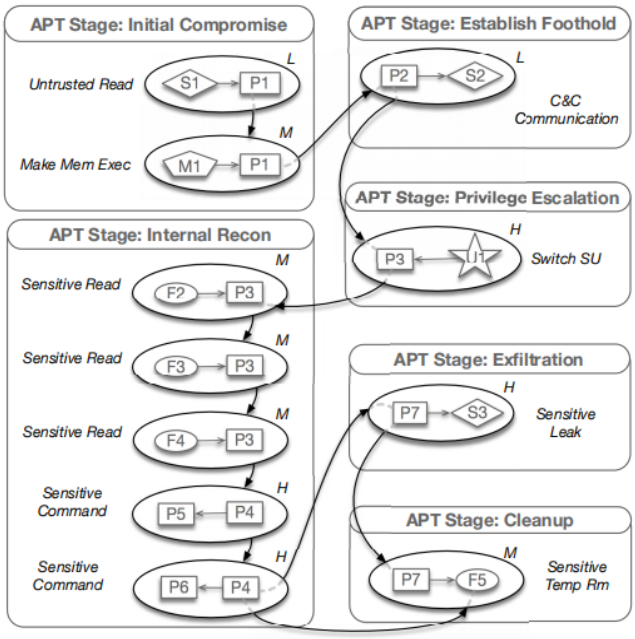


Fig. 5. High-Level Scenario Graph for the Running Example.

(2) HSG 排名和优先级（HSG Ranking and Prioritization）

为了给HSG排序，首先根据下图将Threat Tuple转化为数值类型，接下来将代表7个APT阶段的分数合并为一个整体的分数。特别是根据通用漏洞评分系统（CVSS）中包含的转换表（下表）将威胁元组的每个元素映射为一个数值，CVSS是一个由商业、非商业和学术领域的安全专业人员协作创建的中立行业标准。考虑到他们感知到的威胁与对抗威胁的历史，替代评分选择可以由企业进行。

合并规则基于以下两个准测：

- 灵活性和定制化
- 该分数要能够反映APT攻击步骤是如何展开的

Qualitative level	Quantitative Range	Rounded up Average Value
Low	0.1 - 3.9	2.0
Medium	4.0 - 6.9	6.0
High	7.0 - 8.9	8.0
Critical	9.0 - 10.0	10.0

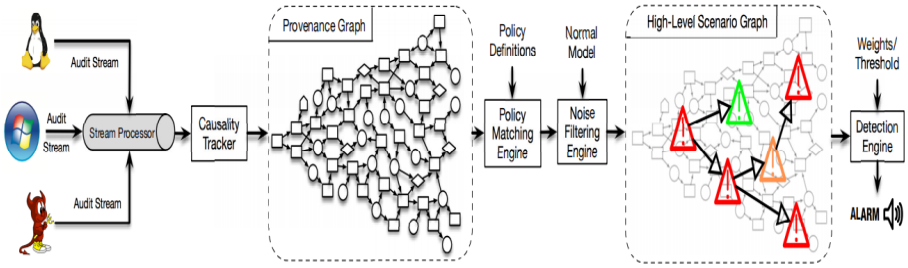
TABLE 7. NIST severity rating scale

为元组中的每一个entry设置一个权重，其中，n表示APT攻击的步骤数，wi 和 Si 分别表示步骤i的权重和威胁程度，T表示阈值。如果在步骤i中没有TTP出现，我们将Si设置为1。

$$\prod_{i=1}^{n} (S_i)^{w_i} \geq \tau$$

V.系统实现

下图是整个HOLMES系统架构。



A.构建溯源图的流消耗

为了实现平台的独立性，将来自不同操作系统的审计数据规范化为通用数据表示（common data representation, CDR）。为基于CDR的审计记录发布到流处理服务器（Kafka），并从流服务器消耗来进行实时分析和检测。

使用SLEUTH（USENIX'17）系统进行数据流消耗，利用因果关系跟踪和起源图构造。

B.规则匹配引擎和HSG构建

规则匹配引擎在溯源图上进行操作，并且将TTP规则作为输入。论文中用到的TTPs规则在Table8中展示。为了匹配到一个TTP，规则匹配引擎将规则表中的每一条规则和它的先决条件进行迭代。这个环节的主要挑战是：对于每一个TTP来说，都要检查之前匹配的TTPs的先决条件和他们之间的路径因子。

APT Stage	TTP	Event Family	Severity	Prerequisites
$Initial\_Compromise(P)$	$Untrusted\_Read(S,P)$	READ	L	$S.ip \notin$ {Trusted_IP_Addresses}
	$Make\_Mem\_Exec(P,M)$	MPROTECT	M	\$PROT_EXEC\$ $\in M.flags$ $\wedge \exists\, Untrusted\_Read(?,P') : path\_factor(P',P) <= path\_thres$
	$Make\_File\_Exec(P,F)$	CHMOD	H	\$PROT_EXEC\$ $\in F.mode$ $\wedge \exists\, Untrusted\_Read(?,P') : path\_factor(P',F) <= path\_thres$ $\wedge \exists\, Untrusted\_Read(?,P'') : path\_factor(P'',P) <= path\_thres$
	$Untrusted\_File\_Exec(F,P)$	EXEC	C	$\exists\, Untrusted\_Read(?,P') : path\_factor(P',F) <= path\_thres$
$Establish\_Foothold(P)$	$Shell\_Exec(F,P)$	EXEC	M	$F.path \in$ {Command_Line_Utilities} $\wedge \exists\, Initial\_Compromise(P') : path\_factor(P',P) <= path\_thres$
	$CnC(P,S)$	SEND	L	$S.ip \notin$ {Trusted_IP_Addresses} $\wedge \exists\, Initial\_Compromise(P') :$ $path\_factor(P',P) <= path\_thres$
$Privilege\_Escalation(P)$	$Sudo\_Exec(F,P)$	EXEC	H	$F.path \in$ {SuperUser_Tools} $\wedge \exists\, Initial\_Compromise(P') :$ $path\_factor(P',P) <= path\_thres$
	$Switch\_SU(U,P)$	SETUID	H	$U.id \in$ {SuperUser_Group} $\wedge \exists\, Initial\_Compromise(P') :$ $path\_factor(P',P) <= path\_thres$
$Internal\_Recon(P)$	$Sensitive\_Read(F,P)$	READ	M	$F.path \in$ {Sensitive_Files} $\wedge \exists\, Initial\_Compromise(P') : path\_factor(P',P) <= path\_thres$
	$Sensitive\_Command(P,P')$	FORK	H	$P'.name \in$ {Sensitive_Commands} $\wedge \exists\, Initial\_Compromise(P'') : path\_factor(P'',P) <= path\_thres$
$Move\_Laterally(P)$	$Send\_Internal(P,S)$	SEND	M	$S.ip \in$ {Internal_IP_Range} $\wedge \exists\, Initial\_Compromise(P') : path\_factor(P',P) <= path\_thres$
$Complete\_Mission(P)$	$Sensitive\_Leak(P,S)$	SEND	H	$S.ip \notin$ {Trusted_IP_Addresses} $\wedge \exists\, Internal\_Reconnaissance(P') :$ $path\_factor(P',P) <= path\_thres$ $\wedge \exists\, Initial\_Compromise(P'') : path\_factor(P'',P) <= path\_thres$
	$Destroy\_System(F,P)$	WRITE/ UNLINK	C	$F.path \in$ {System_Critical_Files} $\wedge \exists\, Initial\_Compromise(P') : path\_factor(P',P) <= path\_thres$
$Cleanup\_Tracks(P)$	$Clear\_Logs(P,F)$	UNLINK	H	$F.path \in$ {Log_Files}$\wedge \exists\, Initial\_Compromise(P') :$ $path\_factor(P',P) <= path\_thres$
	$Sensitive\_Temp\_RM(P,F)$	UNLINK	M	$\exists\, Internal\_Reconnaissance(P') : path\_factor(P',F) <=$ $path\_thres$ $\wedge \exists\, Initial\_Compromise(P'') : path\_factor(P'',P) <= path\_thres$
	$Untrusted\_File\_RM(P,F)$	UNLINK	M	$\exists\, Initial\_Compromise(P') : path\_factor(P',F) <= path\_thres$ $\wedge \exists\, Initial\_Compromise(P'') : path\_factor(P'',P) <= path\_thres$

TABLE 8. Representative TTPs. Event family denotes a set of corresponding events in Windows, Linux, and FreeBSD. In the Severity column, L=Low, M=Moderate, H=High, C=Critical. Entity types are shown by the characters: P=Process, F=File, S=Socket, M=Memory, U=User.

为了避免大量的计算，我们不使用回溯法而是使用 **增量匹配法**。这种方法存储先前计算的结果，并沿着图匹配和传播指向这些结果的指针。当一个TTP被匹配时，我们在HSG中创建相应的节点，并创建一个指向该节点的指针。同时这个指针将会指向所有与该TTP有依赖关系的低级别实体。

例如path_factor的计算，假设N1（起源图中的实体）和N2、N1和N3之间已经建立了依赖，当节点N2与N3之间出现了信息流时， 我们需要重新计算N1和N3之间的path_factor，以取path_factor的最小值。计算的过程中之前N1和N2的计算结果便可复用。

这种基于指针的两层之间的关联方法可能存在开销和复杂性较大的问题。但实际情况是，大量的低级别实体指向一个TTP，每个低级别实体只有一个指向TTP的指针，而每个TTP有多个维护多个指向它的指针，因此并没有出现随着起源图的增加而开销和复杂度急剧提升的情况

C.噪声过滤和检测引擎

噪声过滤引擎识别的是良性的TTP匹配，所以它被排除在HSG之外。它以从正常行为中学习到的模型为输入，模型包含了与良性活动匹配的TTPs以及从操作系统中读写操作的字节数阈值。

当规则匹配系统匹配到一个TTP时，它的入口和先决条件就会在这个模型中搜索，如果在模型中存在一个入口包含了所有的先决条件和匹配事件，那么所有的被转移的字节数就会与良性阈值进行比对。低于良性阈值该TTP则被过滤，否则在HSG中创建一个该TTP对应的节点。最后检测引擎计算不同HSGs的“权重和”，当其超过检测阈值时，就发出警报。

VI.实验评估

实验评估是在由DARPA组织的红蓝对抗中完成，通过评估计算出HOLMES最佳的阈值，并衡量HOLMES的表现。最终，将HOLMES部署在真实环境中，在没有先验知识的情况下来检测红队的APT攻击。在我们的现场实验后，这个数据集已经在公共领域[26]发布，以刺激在这一领域的进一步研究。

- https://github.com/darpa-i2o/Transparent-Computing

A.数据

评估所用的数据如表10所示，包含了跨三个操作系统7台主机的9个APT攻击场景，时间跨度为20天，其中包含了攻击数据和良性数据，攻击数据在数据总量中少于0.001%。

Stream No.	Duration	Platform	Scenario No.	Scenario Name	Attack Surface
1	0d1h17m	Ubuntu 14.04 (64bit)	1	Drive-by Download	Firefox 42.0
2	2d5h8m	Ubuntu 12.04 (64bit)	2	Trojan	Firefox 20.0
3	1d7h25m	Ubuntu 12.04 (64bit)	3	Trojan	Firefox 20.0
4	0d1h39m	Windows 7 Pro (64bit)	4	Spyware	Firefox 44.0
5	5d5h17m	Windows 7 Pro (64bit)	5.1	Eternal Blue	Vulnerable SMB
			5.2	RAT	Firefox 44.0
6	2d5h17m	FreeBSD 11.0 (64bit)	6	Web-Shell	Backdoored Nginx
7	8d7h15m	FreeBSD 11.0 (64bit)	7.1	RAT	Backdoored Nginx
			7.2	Password Hijacking	Backdoored Nginx

TABLE 10. Datasets. Streams 5 and 7 contain two independent attack vectors occurring on the same host.

简而言之，红队攻击场景中的攻击目标和活动涵盖了那些备受瞩目的APT活动。这些活动包括典型的APT活动，如浏览器诱导驱动器的初始入侵、后门注入、特权提升、内部侦察、敏感资产过滤以及清理攻击足迹。在这些攻击中，复杂的攻击向量，如反射加载、Webshell和内存模块加载，它们被红队用来评估HOLMES对APT攻击的鲁棒性。

为了构建降噪模型，我们使用了4天的良性审计数据。这些程序包括浏览器（如火狐）、Web服务器（如Nginx）和各种守护进程（如postfix、syslogd）。

B.设定

在行动之前，每个主机都有一些接下来会被攻击者利用的漏洞软件。为了进一步混合正常日志和攻击日志，红队同时也会进行良性的活动。

HOLMES会订阅7个Kafka的topics（每个主机一个topics）来消耗数据流以进行实时地分析和预测。为TTPs的先决条件设置path_thres=3，设置APT阶段i的权重weight=(10+i)/10。

C.结果显示

表11总结了9个攻击场景的检测情况。第二列显示检测过程中匹配每个HSG的威胁元组，第三列显示相应的威胁分数，第四列显示了执行攻击场景机器在所有良性场景中的最高得分。这些良性场景可能包含相应攻击场景中的确切程序。

良性HSG的最高得分为338（方案3），攻击HSG的最低得分为608（方案5.2），这与对系统没有造成伤害的不完全攻击有关。 这说明HOLMES 已经将攻击场景和良性场景分成了两个不相交的集群，并明确区分了它们。

Scenario No.	Threat Tuple	Threat Score	Highest Benign Score in Dataset
1	$\langle C, M, -, H, -, H, M\rangle$	1163881	61
2	$\langle C, M, -, H, -, H, -\rangle$	55342	226
3	$\langle C, M, -, H, -, H, M\rangle$	1163881	338
4	$\langle C, M, -, H, -, -, M\rangle$	41780	5
5.1	$\langle C, L, -, M, -, H, H\rangle$	339504	104
5.2	$\langle C, L, -, -, -, -, M\rangle$	608	
6	$\langle L, L, H, M, -, H, -\rangle$	25162	137
7.1	$\langle C, L, H, H, -, H, M\rangle$	4649220	133
7.2	$\langle M, L, H, H, -, H, M\rangle$	2650614	

TABLE 11. Scores Assigned to Attack Scenarios. L = Low, M = Moderate, H = High, C = Critical. **Note:** for each scenario, Highest Benign Score in Dataset is the highest *threat score* assigned to benign background activities streamed during the audit log collection of a host (pre-attack, in parallel to attack, and post-attack).

学习降噪规则和路径因子的影响如图12所示。此图显示了所有良性和攻击HSG在分析了所有七个阶段后构建的威胁分数。这些分数在三种不同的设置下显示，很明显，攻击HSG和良性HSG之间有较大的差距。

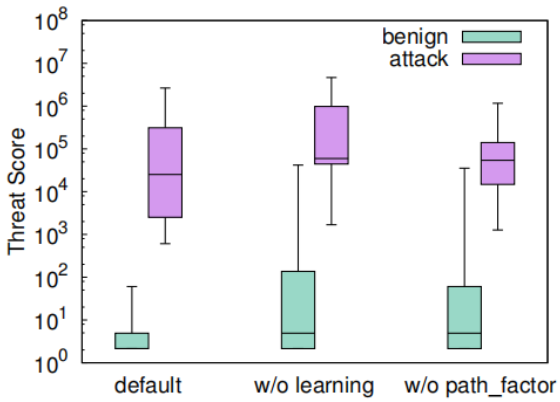


Fig. 12. Effects of Learning and *path_factor* on Noise Reduction. Box covers from first to third quartiles while a bar in the middle indicates median, and whisker is extended from 10th to 90th percentile.

D.寻找最优阈值

如下图所示，使用准确率和召回率来寻找最佳阈值。其中F-score表示准确率和召回率的调和平均值，注意到在[338.25, 608.26]时，F-score达到最大，区间的两个端点分别表示良性活动的最高分数和恶意活动的最小分数，因此阈值应该在该区间内选取。

在取n次方根之后我们发现，F-score取得最大值时，在[338.25, 608.26]之间调整阈值获得不同的threat scores，此时对应平均严重程度的收敛区间为[2.01, 2.16]，我们取其中位数2.09。

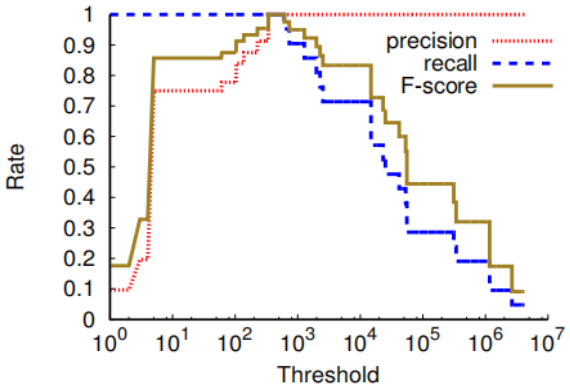


Fig. 14. Precision, Recall, and F-score of attack detection by varying the threshold value.

E.性能

- Graph Size
  图15显示了数千条边（左）的起源图和边数（右）的HSG的生长趋势的比较。在边缘测量的图尺寸比是1875：1，即在从起源图映射到HSG的过程中减少了1875倍。

- Memory Use
  HOLMES在8核CPU上测试，每个速度2.5GHz和150GB内存。图16（左）显示具有审计记录数量HOLMES的内存消耗。它显示了内存消耗的近线性增长，因为该系统在内存中的审计记录。图16（右）显示了支持多少主机的可扩展性到数百台主机的企业。很明显，随着主机数量的增加，我们可以在内存中保持完整的溯源图的持续时间就会减少。注意，x轴和y轴都是按log-2比例排列的。

- Runtime
  本文测量了消耗记录、构建溯源图、构建HSG和检测APTs的CPU时间。

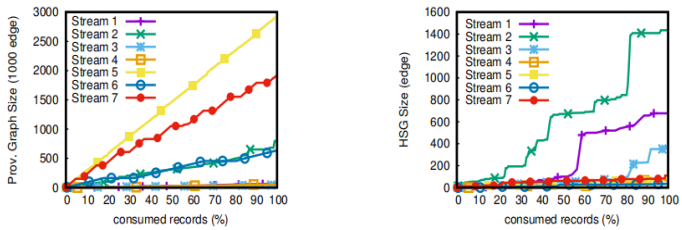


Fig. 15. (Left): Provenance graph growth vs. consumed records. (Right): HSG growth vs. consumed records.

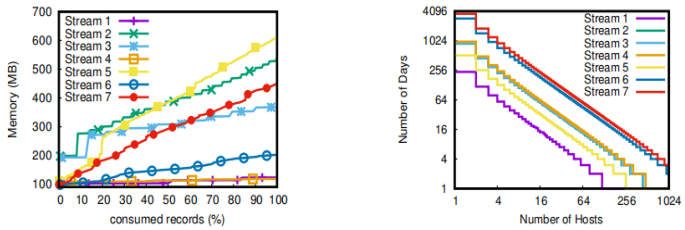


Fig. 16. (Left): Memory footprint (MB) vs. % of records consumed. (Right): Number of Days vs. extrapolated number of hosts that can be handled by HOLMES in respect to Memory consumption

F.真实场景实验

将系统放在模拟的企业环境中，在对攻击没有先验知识的情况下，由红队发起APT攻击。图18显示了HOLMES在本实验中构造的攻击HSG和良性HSG的累积分布函数。

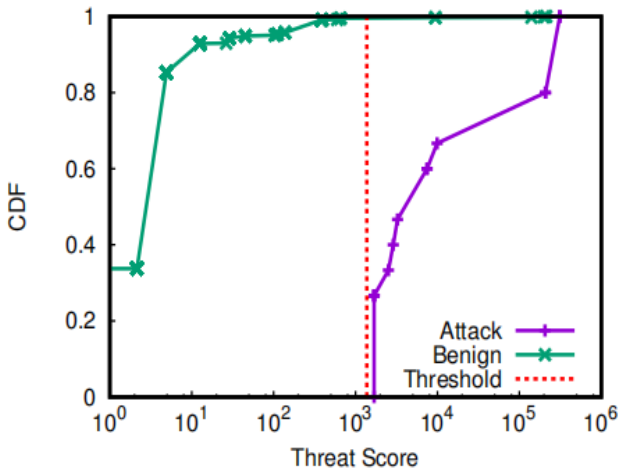


Fig. 18. Cumulative distribution function for attack vs. benign HSGs

- 误报（False Positive）
  将系统放在两周的良性活动环境中，没有发现误报。
- 漏报（False Negative）
  TTP之间的隐式因果关系：对于避免系统调用的信息流，HOLMES无法直接查看系统实体之间的因果关系。 但是，如果攻击的其余部分通过系统调用展开，则HOLMES仍将重构部分攻击。此外，作为一种主动规避技术，攻击者可能会利用多个入口点来生成分离的子图。

Ⅶ.相关工作

HOLMES对实时警报生成、警报相关性和场景重建等问题做出了贡献。HOLMES模型中的一个核心思想是构建和使用高级攻击场景图作为上述所有问题的潜在基础。下面，我们将讨论上述所有领域的相关工作。

A.警报生成（Alarm Generation）
基于主机的入侵检测方法（Host-based intrusion detection approaches）主要分为以下三类：

- misuse-based（滥用）：检测与已知攻击相关的行为
- anomaly-based（异常）：从正常行为中学习模式，并检测偏离该模式的行为
- specification-based（规范）：根据专家指定的策略检测攻击，依赖专家知识

本文属于misuse-based，但本文方法超越了传统的滥用检测（传统基于规则）。

- HOLMES使用了先决条件-结果的模式，当匹配的TTP中涉及的实体之间存在信息流依赖性时，这些先决条件-结果模式将被匹配。

B.警报关联
IDS生成的警报对于人工操作员而言太多且级别很低。 需要开发一些技术来总结这些低级别警报并减少其数量。一些方法使用警报相关性，通过对相似警报进行聚类并确定警报之间的因果关系来执行检测。

- 目前的方法依赖运行在用户空间的第三方应用生成的日志，而且基于像时间戳之类的统计特征，这样并不能很好的检测时间跨度很长的APT攻击。因此HOLMES在不同的攻击步骤之间建立了信息流，使用了内核审计数据。
- 警报关联的另一项工作依赖于警报在时间上的接近程度。 相比之下，HOLMES依靠信息流和因果关系来关联警报，因此即使在执行步骤非常缓慢的情况下，也能够检测到攻击。

C.场景重建
许多研究是基于生成和使用系统调用级别的日志（system-call level

logs）来进行的。大多数的方法是从一个给定的恶意事件开始去追溯导致该事件的原因。相比之下，HOLMES使用系统事件跟踪来执行实时检测，在检测框架中具有以高水平攻击步骤形式的集成取证能力，而不需要检测。

最近的研究已经使用了系统调用级别的日志来进行实时分析。SLEUTH提出了基于标签的攻击检测和就地取证的技术。HOLMES在侦探方面取得了一些重大进展。

- 首先，它展示了如何利用最小祖先覆盖的概念来解决依赖爆炸问题，并开发了一种有效的增量计算算法。
- 其次，侦探的场景图与起源图处于抽象相同，这对于许多分析人员来说可能太低，而且在HSG中缺乏可操作的信息。
- 最后，溯源图在长期运行的攻击中可能会变得太大，而HOLMES则通过使用降噪和优先级技术生成紧凑的HSG。

D.攻击粒度
有时，审计日志的粗粒度会限制对信息流的推理。例如，如果具有之前加载过敏感文件的进程受到攻击，则攻击者可以在不使用系统调用的情况下在其内存区域内搜索敏感内容。HOLMES会将该文件窃取行为与该进程的其他动作相关联，比如敏感文件读取。

Ⅷ.结论

本文提出了一个实时APT检测系统HOLMES，它关联了可能用于执行每个APT阶段的战术、技术和程序（TTPs）。HOLMES生成一个高级图实时总结攻击者的步骤。我们评估HOLMES对9个真实的APT威胁，并将其部署为一个实时入侵检测工具。实验结果表明，HOLMES能以高精度、低误警率成功地检测到APT活动。

S&P是安全最好的会议，能发S&P的论文都是最优秀的论文之一。下面先引用 **安全学术圈** 的总结，推荐大家关注它们。

- **首先，HOLMES旨在将主机日志中发现的活动以及企业中发现的任何告警直接反应到杀伤链中。**这种设计允许HOLMES生成告警，这些告警在语义上接近APT活动者的活动阶段（TTPs）。通过这样做，HOLMES将告警生成过程提升到攻击活动步骤的级别，而不是在低级别审计日志的形式。在实验中发现为期五天的审计日志集合包含大约3M的低级别事件，而HOLMES仅从中提取86个可疑活动阶段。
- **HOLMES的第二个重要思想是使用系统内低级别实体（文件，进程等）之间的信息流作为告警关联的基础。**请注意杀伤链中的内部侦察阶段取决于成功的初始入侵和立足点建立。特别地，侦察阶段通常使用攻击者在立足点建立期间安装的CC代理（进程）来启动，从而展示两个阶段中涉及的进程之间的信息流。此外，侦察通常涉及在立足点建立阶段运行下载的恶意软件（文件），说明文件到进程的信息流。同样，成功的横向渗透阶段以及窃取阶段，会使用侦察阶段收集的数据。因此通过检测与APT阶段相关的低级别事件并使用信息流将它们关联起来，可以构建APT活动者使用的杀伤链。
- **HOLMES的第三个主要贡献是开发高层次场景图（HSG）。**HSG的节点对应于TTP，并且其边表示TTP中涉及实体之间的信息流。HSG为检测APT的高可信度提供了基础。首先是使用了HSG中祖先实体覆盖的概念帮助评估HSG节点之间依赖关系的强度。然后可以修剪弱依赖项以消除许多错误告警。其次是开发了降噪技术，进一步降低已知与良性活动相关的依赖项。第三是使用优先级排序技术，以删除与APT活动无关的大多数节点和边。使用这些技术证明了HOLMES能够明确区分攻击和良性场景。
- **最后，HSG提供了一个任意时刻都存在的简洁可视摘要，从而为理解攻击做出了重要贡献。**例如，从10M审计记录的数据集开始，能够使用仅16个节点的图表来总结高级攻击活动。网络分析师可以使用现有的HSG快速推断出攻击的总体情况（范围和幅度）。

个人感觉这篇文章真的非常棒，融合杀伤链、ATT&CK和TTPs的图结构用于APT攻击检测我也曾想过，但本文模型的完整性、实验、故事叙述非常值得我们学习，尤其是很多细节部分（如剪枝、降噪、HSG构建）。同时，本文应用于真实场景实践都非常棒。

优秀语句：

- Understanding the motivations and operations of the APT actors plays a vital role in the challenge of addressing these threats. To further this understanding, the Mandiant report also offered an APT lifecycle model (Fig. 1), also known as the kill-chain, that allows one to gain perspective on how the APT steps collectively achieve their actors' goals.

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 2021-07-21 周三夜于武汉 http://blog.csdn.net/eastmount/)
