Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

结合溯源图的APT检测方法

硕士研究生　关迎丹

2021年12月26日

内容提要

- 背景简介
- 基本概念
- 算法原理
  - Holmes
  - Unicorn
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1.熟悉APT攻击的基本概念和检测难点
  - 2.了解Kill-chain模型和ATT&CK框架
  - 3.理解结合溯源图的APT检测方法原理
  - 4.了解溯源图的应用领域和发展方向

背景简介

- 2020年12月，安全公司FireEye的红队（攻击）工具箱遭到窃取，股价下跌8%

- 2021年04月，5.33 亿个 Facebook 帐户的个人数据被免费发布，包括电话号码、全名等大量敏感信息。

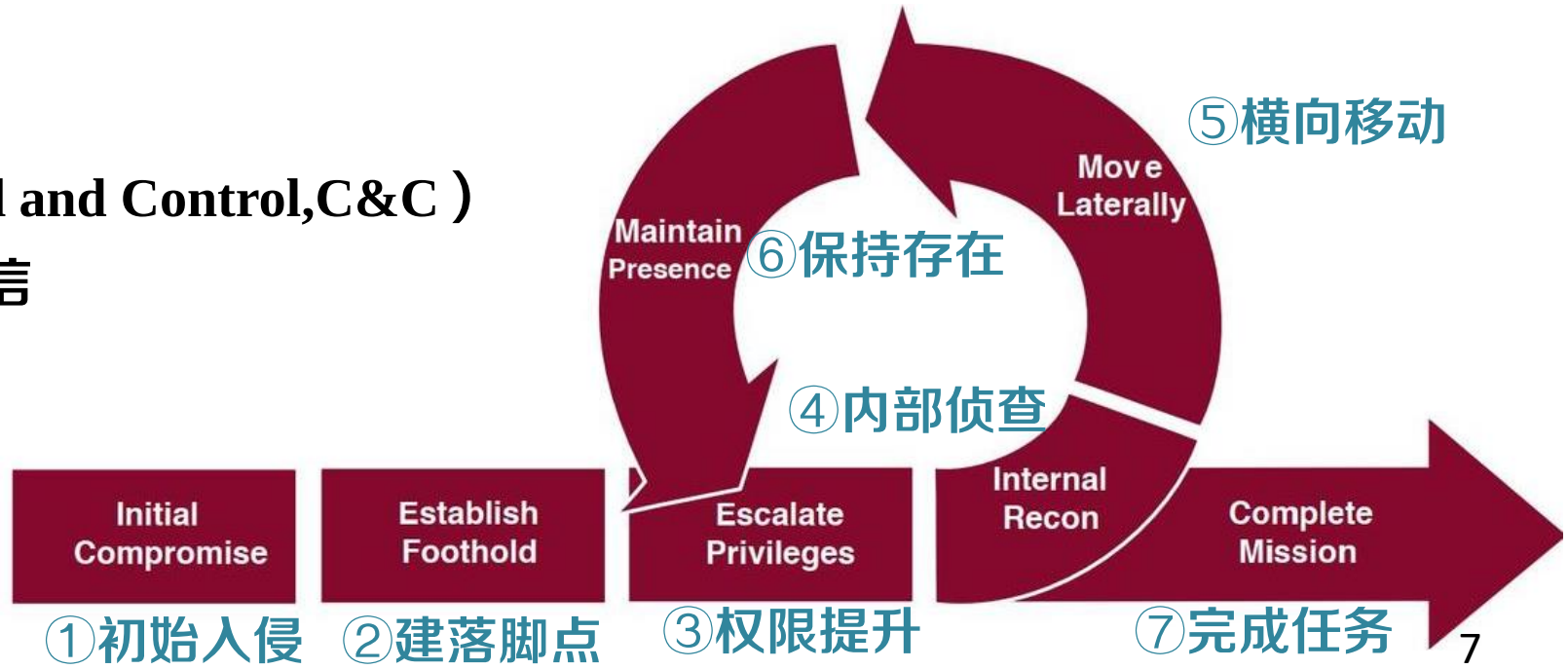
基本概念

基本概念

- APT攻击（Advanced Persistent Threat，高级持续性威胁）
  - 指利用先进的攻击手段对特定目标进行长期持续性网络攻击并产生有效威胁的攻击形式
  - 特点
    - 针对性：通常针对特定领域目标的重要价值资产，如能源、金融等领域
    - 持续性：攻击者长期潜伏并持续攻击以获取更多利益
    - 隐蔽性：使用多种隐藏技术，安全防护系统难以检测

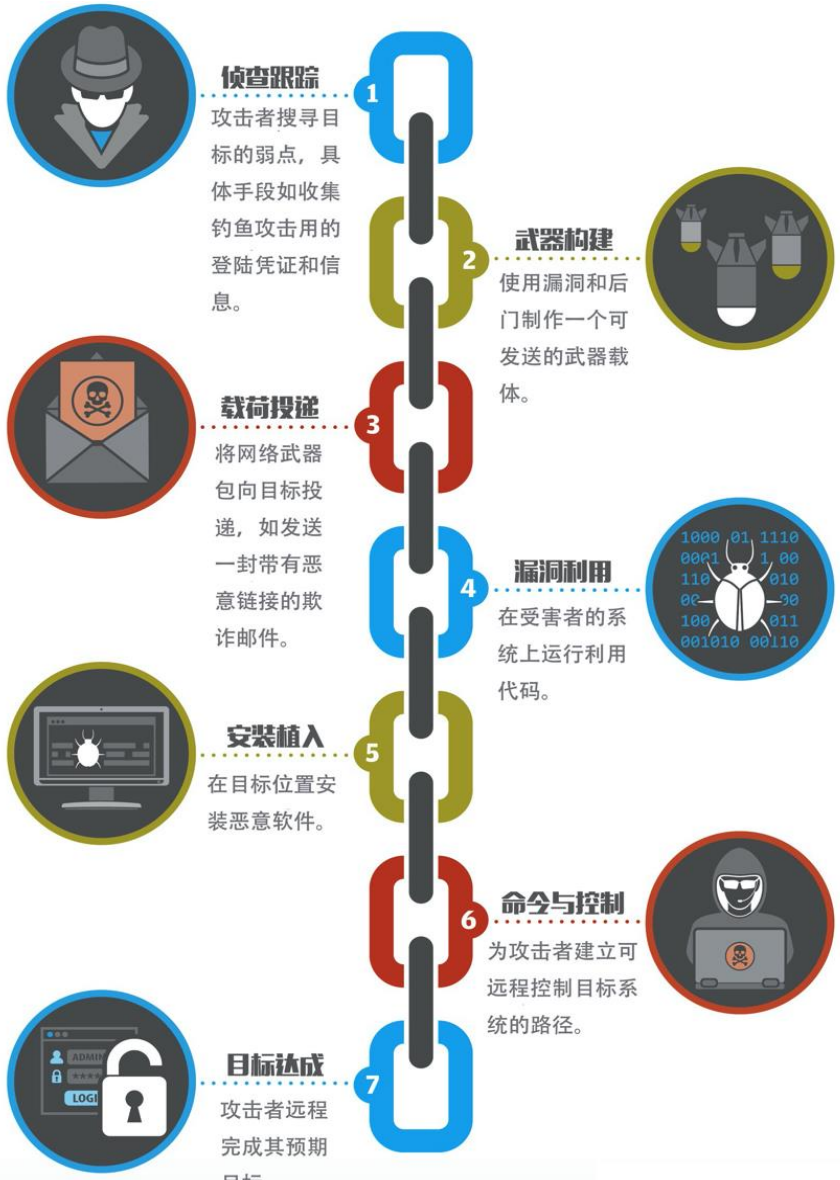
基本概念

- APT攻击生命周期（APT Lifecycle）
  - ①社会工程学、网站挂马、邮件钓鱼攻击，
  - ②保持控制权，永久性木马后门
  - ③获取更大权限，如应用程序漏洞利用
  - ④探测内部信息
  - ⑤寻找其他受害目标
  - ⑥持续访问目标
    - 命令与控制（Command and Control,C&C）
    - 攻击主机和受害主机通信
  - ⑦窃取数据等任务



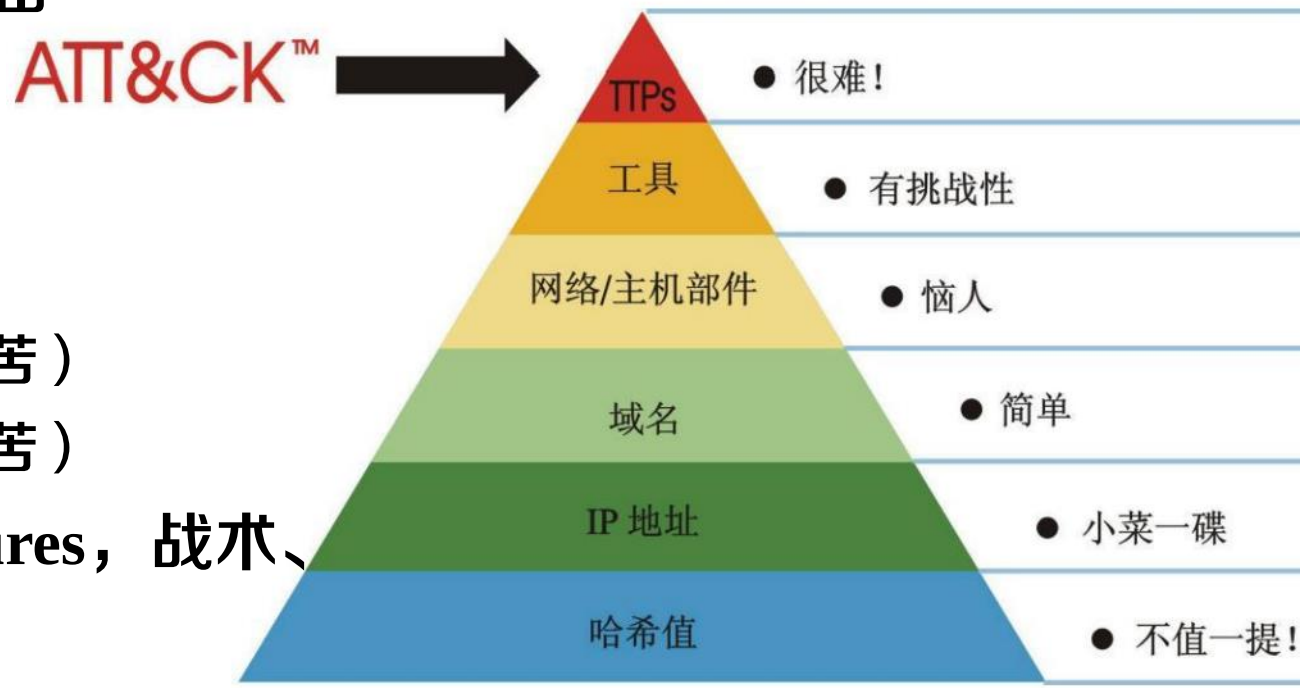
基本概念

- 杀伤链（Kill-chain）
  - 一种用于分析APT攻击各阶段步骤的七层模型
    - 侦察跟踪（Reconnaissance）
    - 武器构建（Weaponization）
    - 载荷投递（Delivery）
    - 漏洞利用（Exploitation）
    - 安装植入（Installation）
    - 命令与控制（Command and Control ）
    - 目标达成（Actions on Objectives）



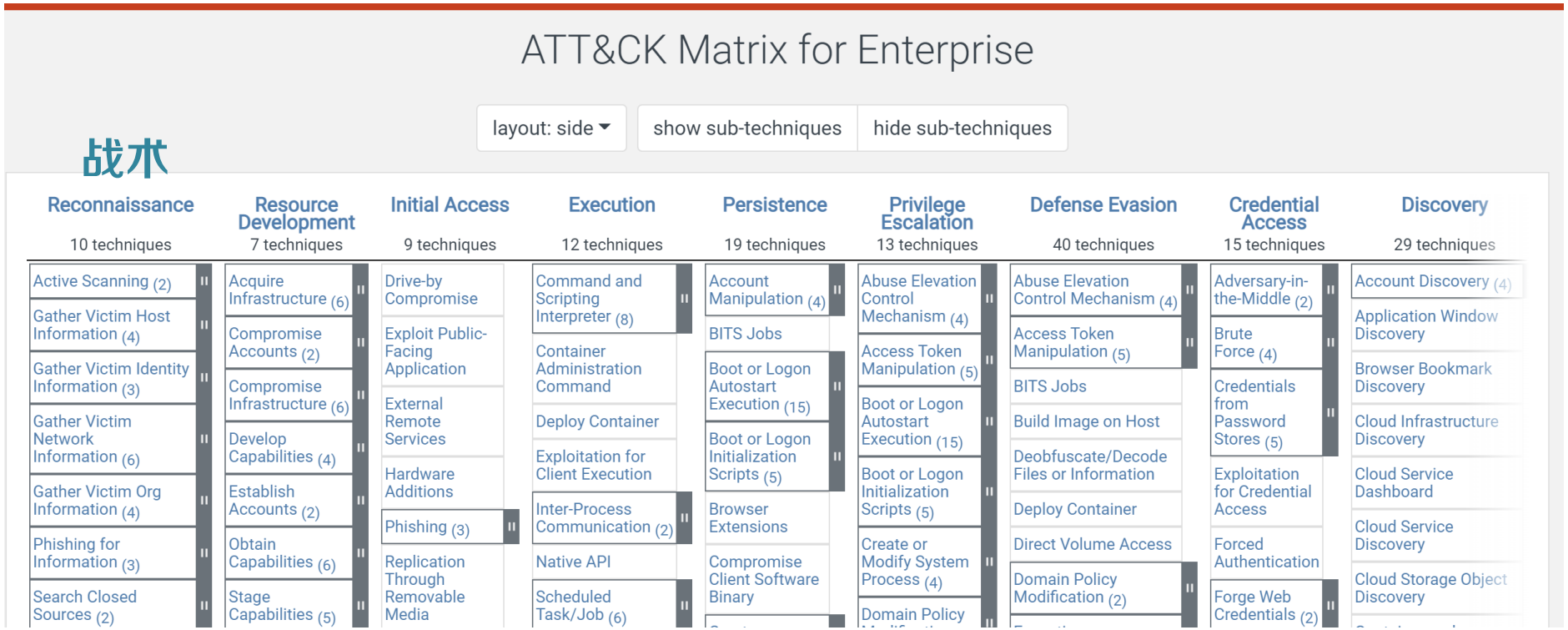
基本概念

- IOC（Indicators of Compromise，失陷指标）
  - 攻击检测指标，例如文件的哈希值、IP地址、域名
  - 监控IOC可检测攻击进而制止或限制攻击
- 痛苦金字塔（The Pyramid of Pain）
  - 描述各类 IOC在攻防对抗中的价值
  - 痛苦金字塔为什么痛苦?
    - 攻击者：更换技术的成本越高（越痛苦）
    - 防御者：检测指标的难度越高（越痛苦）
- TTPs（Tactics Techniques and Procedures，战术、技术和程序）
  - TTPs 处于痛苦金字塔塔尖，是最有价值的一类IOC
  - 描述攻击过程中每一步的具体动作



基本概念

- **ATT&CK 框架(Adversarial Tactics, Techniques, and Common Knowledge)**
  - **一个全球可访问的基于现实世界观察标准化的对手战术和技术知识库**
  - **可有效分析对手行为（TTPs），即每一阶段战术使用什么技术**

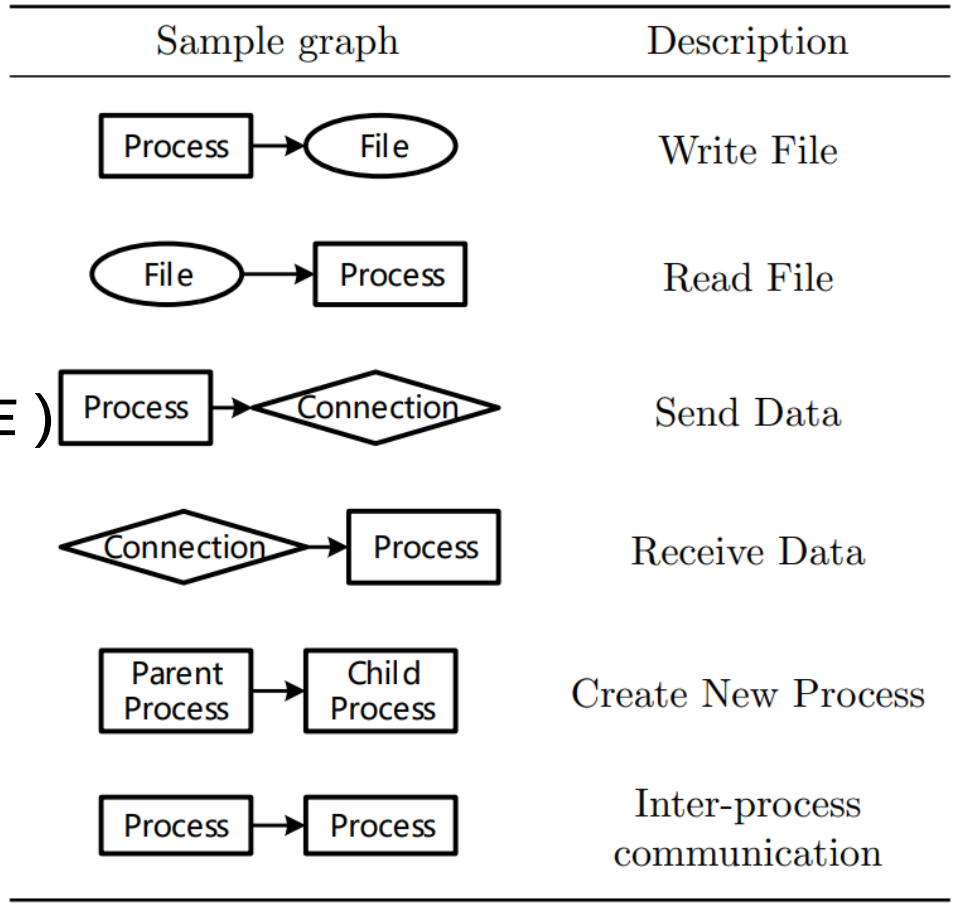


基本概念

- 传统检测方法存在的主要问题
  - 误报率高：
    - 采用IOC来捕获孤立的指标，例如恶意软件签名、异常流量等
    - 由于有些IOC指标良性活动也可能存在，造成大量误报
  - 大海捞针：
    - 难以在海量数据中检测到罕见（0.01%）的攻击事件
  - 易被绕过：
    - 无法对长期行为进行建模，难以检测长期、隐蔽的APT攻击

基本概念

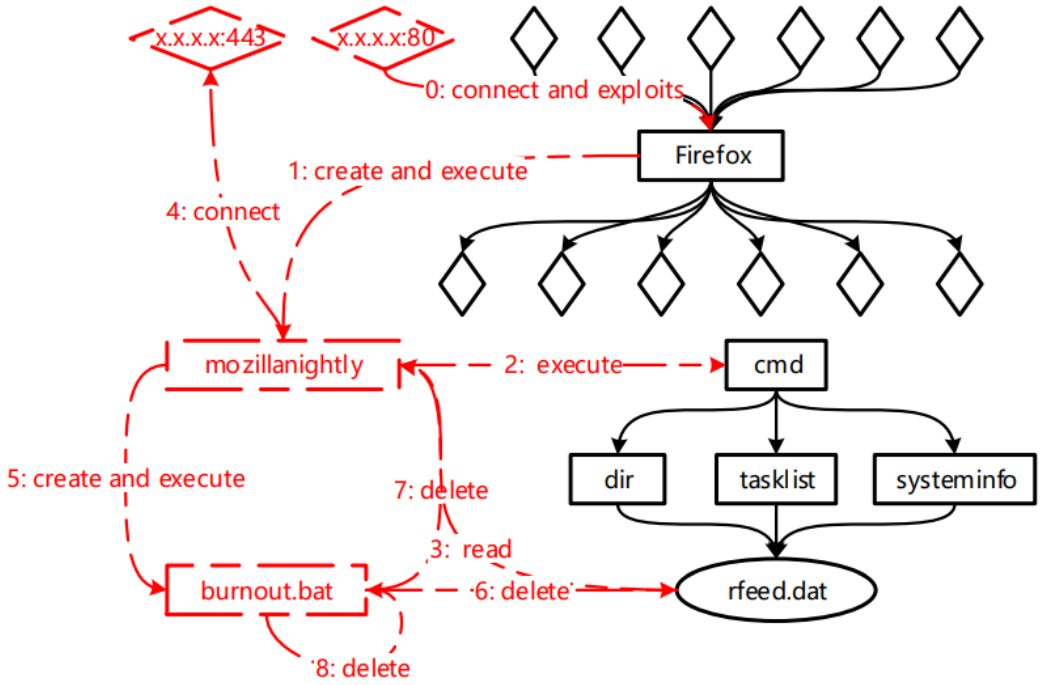
- 两类实体：主体（Subject）和客体（Object）
  - 进程、线程、文件、网络连接、注册表等
  - 主客体的区分是相对的
- 事件（Event）
  - 事件是系统主体和客体间动作的记录。
  - 一般包含四个主要属性（主体、客体、时间、动作）
    - Event = < subject, object, time, operation >
- 因果依赖（Causality Dependency）
  - 两个事件Event1和Event2,若object 1=subject2且time1< time2,则存在因果依赖
  - 揭示两个事件之间的控制流和数据流



基本概念

- 溯源图（Provenance Graphs）
  - 表示主体和客体之间的控制流和信息流的关系的一种有向图
  - 主体和客体为节点，事件为边

- 结合溯源图的APT检测方法
  - 将原始日志数据转化为溯源图形式作为输入
  - 优势
    - 非结构化的日志也能够提取溯源图
    - 包含时空信息，不容易被攻击者伪造躲避。
    - 保留了执行历史，便于检测长期且隐蔽的APT攻击



基本概念

- **DARPA TC项目**
  - 美国国防高级研究计划局DARPA组织透明计算（Transparent Computing, TC）项目
  - 期望通过基于终端数据的采集与分析增强终端系统细粒度行为的可视能力，以实现企业级网络空间APT检测、取证等关键任务

持续时间	平台	攻击场景	攻击面
0d1h17m	Ubuntu 14.04(64bit)	Drive-by Download	Firefox 42.0
2d5h8m	Ubuntu 12.04(64bit)	Trojan	Firefox 20.0
1d7h25m	Ubuntu 12.04(64bit)	Trojan	Firefox 20.0
0d1h39m	Windows 7Pro (64bit)	Spyware	Firefox 44.0
5d5h17m	Windows 7Pro (64bit)	Eternal Blue	Vulnerable SMB
		RAT	Firefox 44.0
2d5h17m	FreeBSD11.0 (64bit)	Web-Shell	Backdoored Nginx
8d7h15m	FreeBSD11.0 (64bit)	RAT	Backdoored Nginx
		Password Hijacking	Backdoored Nginx

算法原理

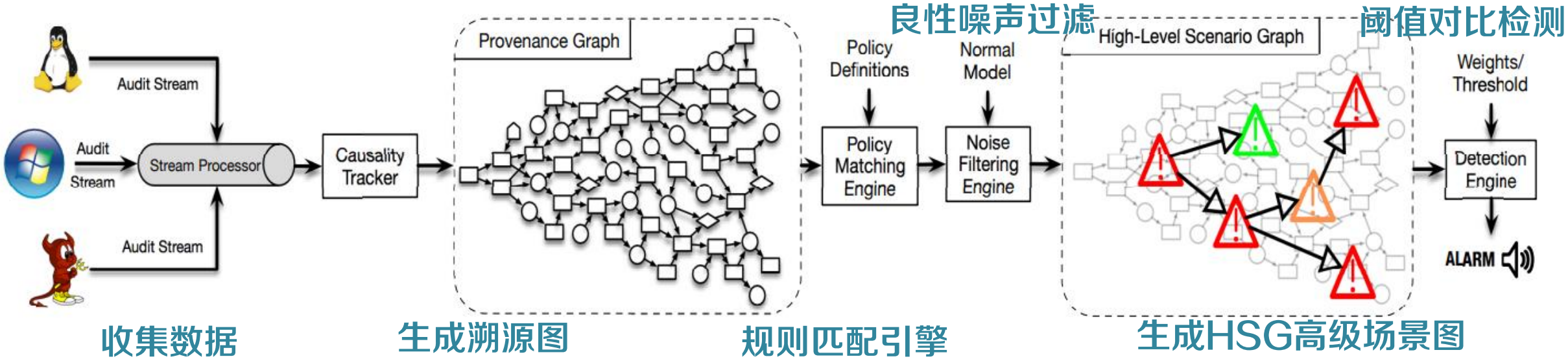
算法原理

T	APT攻击检测
I	主机审计日志数据
P	1.收集数据，利用因果依赖关系追踪生成系统溯源图 2.将溯源图中的事件与TTPs进行规则匹配 3.过滤符合TTPs规则且低于阈值的良性噪声 4.在HSG(高级场景图)创建TTPs节点并与杀伤链模型各阶段对齐 5.计算HSG的“威胁分数”，当其超过检测阈值时就发出警报
O	是否为APT攻击

P	在海量主机日志中实时检测APT攻击
C	APT攻击符合杀伤链模型且各阶段存在因果依赖关系
D	低级别日志数据和高级杀伤链模型视角之间存在巨大的语义鸿沟
L	2019年 S&P

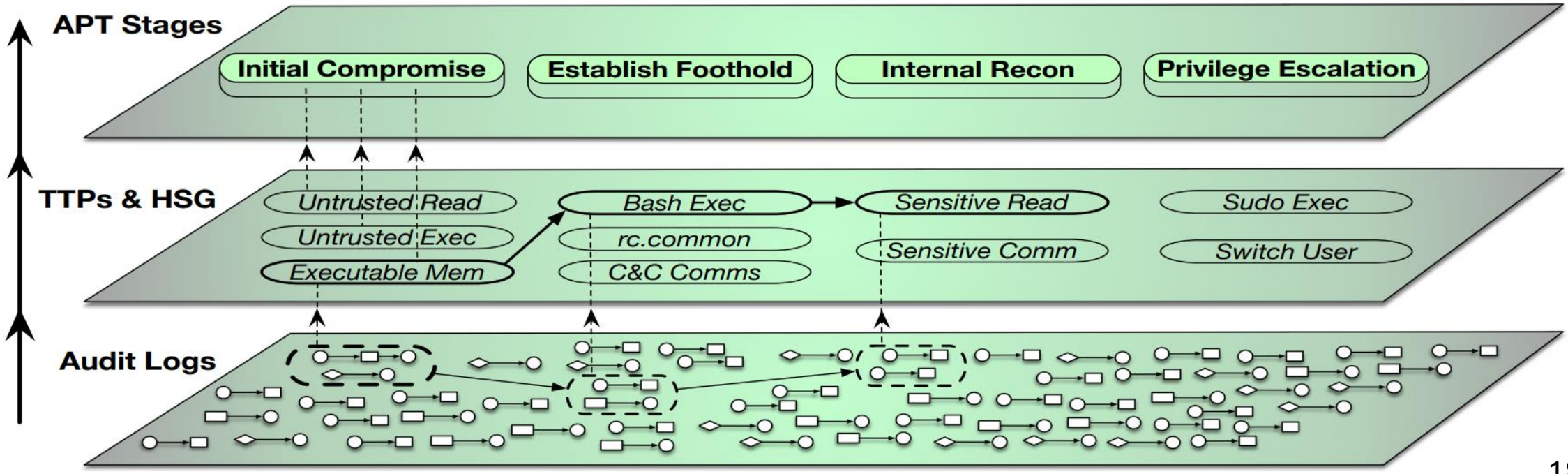
算法原理

- **Holmes流程**
  - 原始日志->溯源图->TTPs->HSG(高级场景图)->威胁分数->阈值对比检测



算法原理

- 核心思想
  - 缩小语义鸿沟（Bridging the Semantic Gap）
    - 弥合低级系统日志视角和高级杀伤链模型视角之间的语义差距
  - 通过TTPs匹配构建中间层，借助ATT&CK框架模型完成从低级到高级视角的映射



算法原理

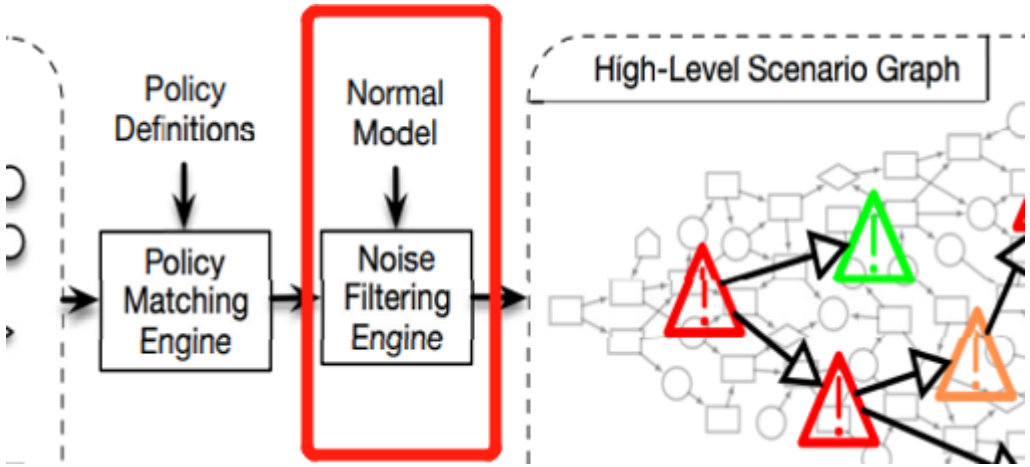
- 规则匹配引擎：TTPs规则匹配
  - TTPs提供了低级别审计日志事件和高级别APT步骤之间的映射
  - 系统溯源图输入TTPs规则，将符合规则的实体事件匹配为一个TTPs
    - 先决条件（Prerequisites）
      - 表现为因果依赖关系和信息流的形式，是事件被划分为TTPs的前提条件
    - 路径因子（path_factor）
      - 两个进程之间的路径长度，表示依赖程度，越大表示依赖关系越小

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.

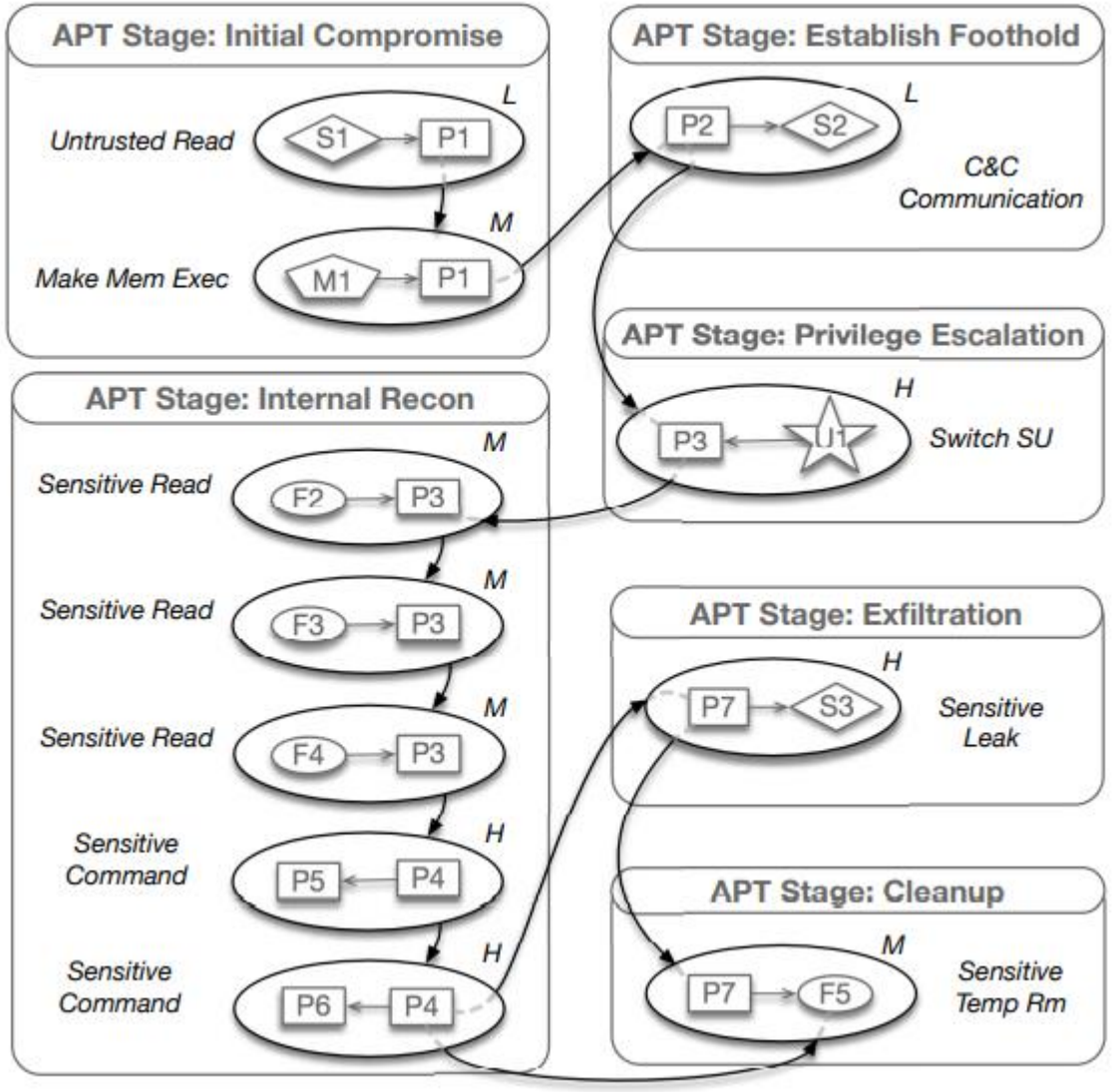
算法原理

- 噪声过滤引擎
  - 良性噪声
    - 溯源图中TTPs规则匹配到的良性事件
  - 良性事件学习建模
    - 在良性环境中运行规则匹配引擎
    - 模型
      - 良性活动匹配的TTPs
      - 读写操作的字节数良性阈值
  - 噪声过滤
    - 若规则匹配引擎匹配到一个TTPs先输入到噪声过滤引擎
    - 查询为良性活动匹配的TTPs且低于良性阈值则忽略此TTPs



算法原理

- 高级场景图（High-level Scenario Graph，HSG）
  - 用于重构整体APT攻击场景的一种特殊图形式
    - 节点（椭圆）：匹配的TTPs
    - 边：TTPs之间的先决条件
  - 先决条件驱动
    - 若一个TTPs的所有先决条件都满足了，且未被噪声引擎过滤就会被添加到HSG中
  - 威胁元组（Threat Tuple）
    - 由7个元素组成：<S1, S2, S3, …, S7>
    - 每个Si对应于第i个APT攻击阶段的威胁程度，如有多个选取最高的危险等级
    - 右图的威胁元组：<M, L, H, H, —, H, M>



算法原理

- APT检测引擎
  - 按照下列公式将HSG的威胁元组转化为数值类型，并将7个APT阶段的分数加权乘积为整体威胁分数

$$\prod_{i=1}^{n}(s_i)^{wi} \geq \tau$$

  - n表示APT攻击的步骤数，w_i和 s_i 分别表示步骤i的权重和威胁程度，τ表示阈值。如果在步骤i中没有TTPs出现，s_i设置为1

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

算法原理

- 实验数据
  - DARPA TC项目数据集
    - 包含了跨3个操作系统7台主机的9个APT攻击场景
    - 时间跨度20天，包含攻击数据和良性数据，攻击数据在数据总量中少于0.001%
- 实验设置
  - 参数设置
    - 先决条件-路径因子阈值：path_thres=3
    - APT攻击第i阶段的权重：weight=(10+i)/10
  - 降噪模型数据设置
    - 4天的良性审计数据，包括浏览器（如火狐）、Web服务器（如Nginx）和各种守护进程（如postfix、syslog）

算法原理

- 攻击检测结果示例
  - 场景1的HSG图-Drive-by Download
    - 用户使用Firefox浏览器访问恶意网站
    - 下载一个名为net的恶意文件并执行
    - 连接到 C&C 服务器，提供shell
    - 启动shell并执行命令
    - 将信息泄露到 C&C 服务器的 IP 地址
    - 删除恶意文件，结束攻击
  - 场景1的威胁分数计算
    - 威胁元组：
      - <C, M, −, H, −, H, M>
      - < 10, 6, 1, 8, 1, 8, 6>
    - 威胁分数：1163881

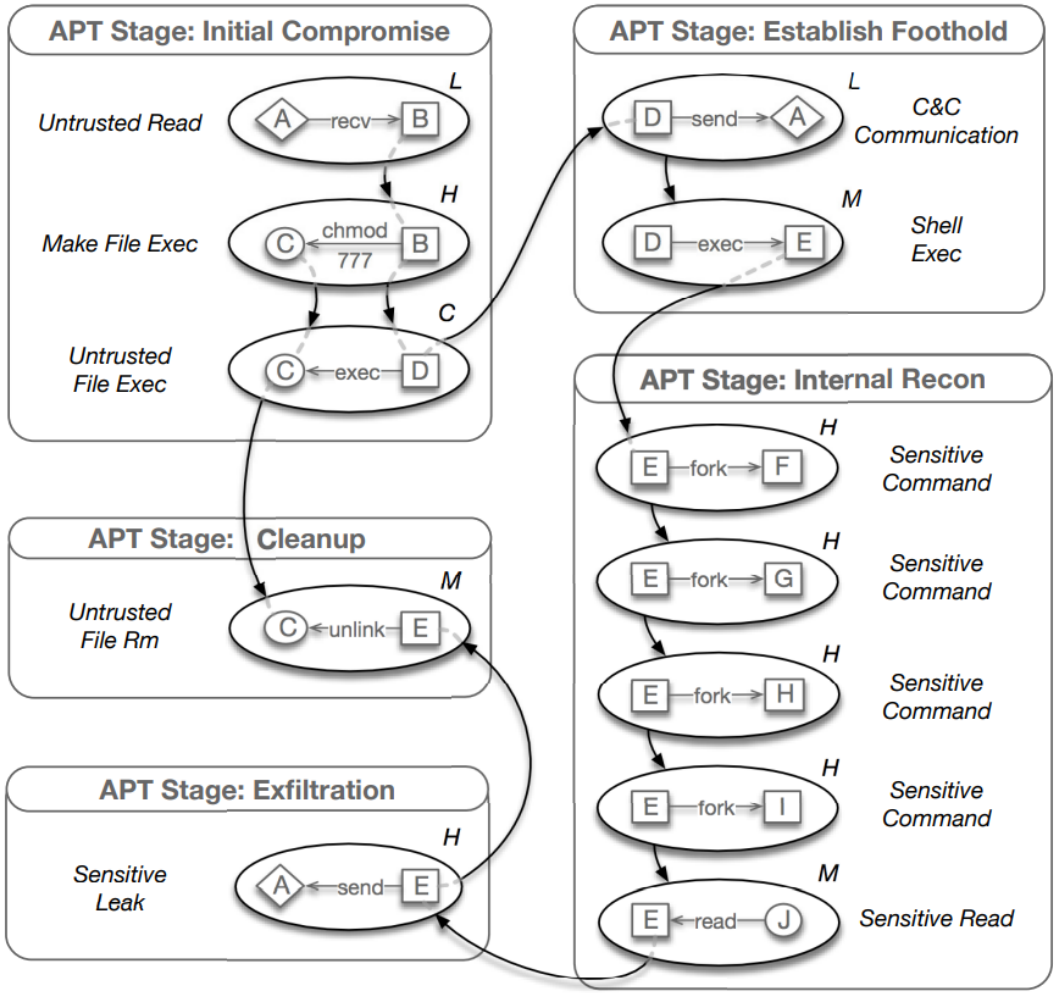


Fig. 13. HSG of Scenario-1 (Drive-by Download). Notations: A= Untrusted External Address; B= Firefox; C= Malicious dropped file (net); D= RAT process; E= bash; F= whoami; G= uname; I= netstat; J= company_secret.txt;

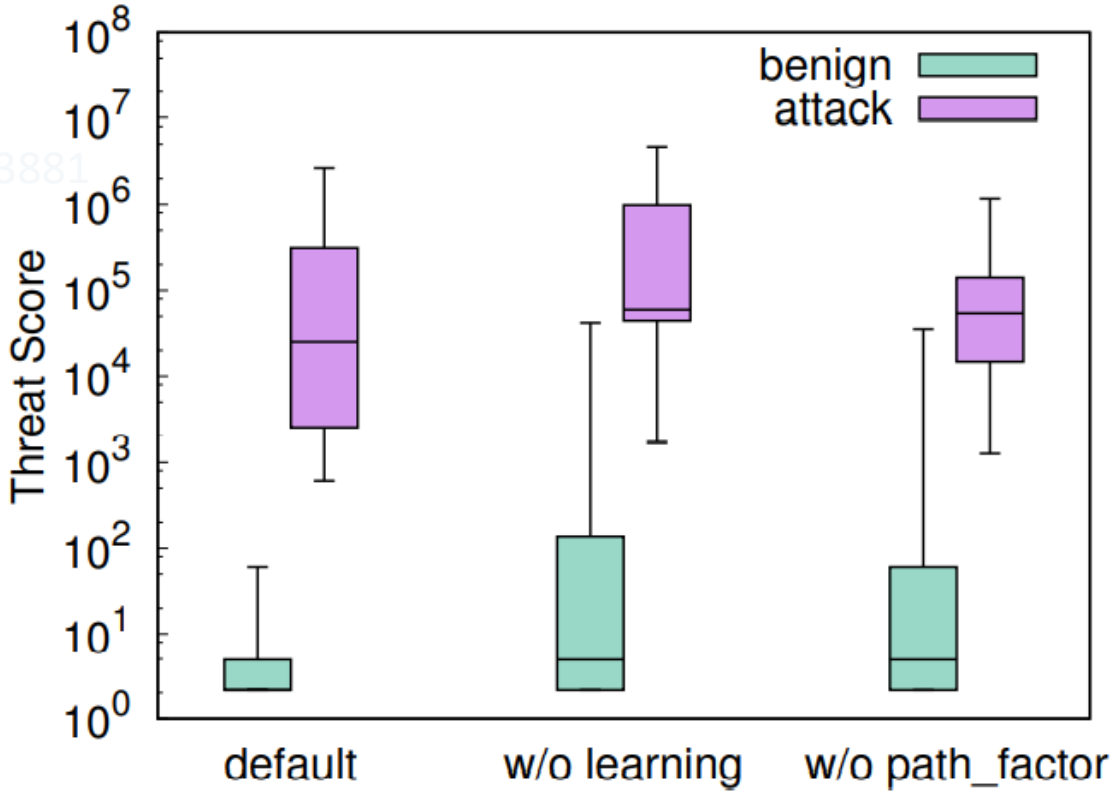
算法原理

- 威胁分数的有效性
  - 良性HSG的最高得分为338（场景3），攻击HSG的最低得分为608（场景5.2）
  - HOLMES 将攻击场景和良性场景分成了两个不相交的集群，可有效检测APT攻击

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	

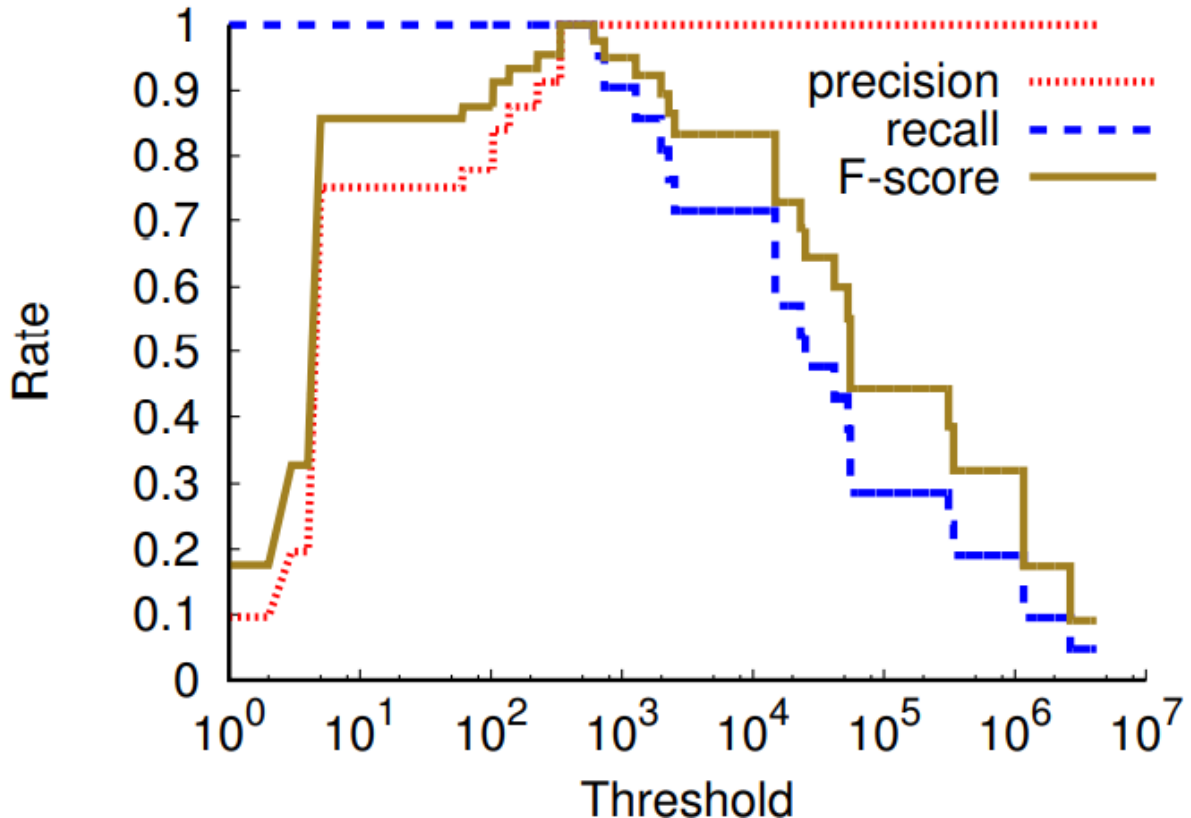
算法原理

- 噪声过滤的影响
  - 三种实验设置：默认使用良性事件学习和路径因子、无良性事件学习和无路径因子
  - 如果没有良性事件学习或路径因子，噪声明显增加，导致误报或漏报



算法原理

- 最佳阈值的选择
  - 使用准确率和召回率来寻找最佳阈值
  - 阈值选取的最佳范围：[338.25, 608.26]
  - 精确率、召回率、F值均可接近1



算法原理

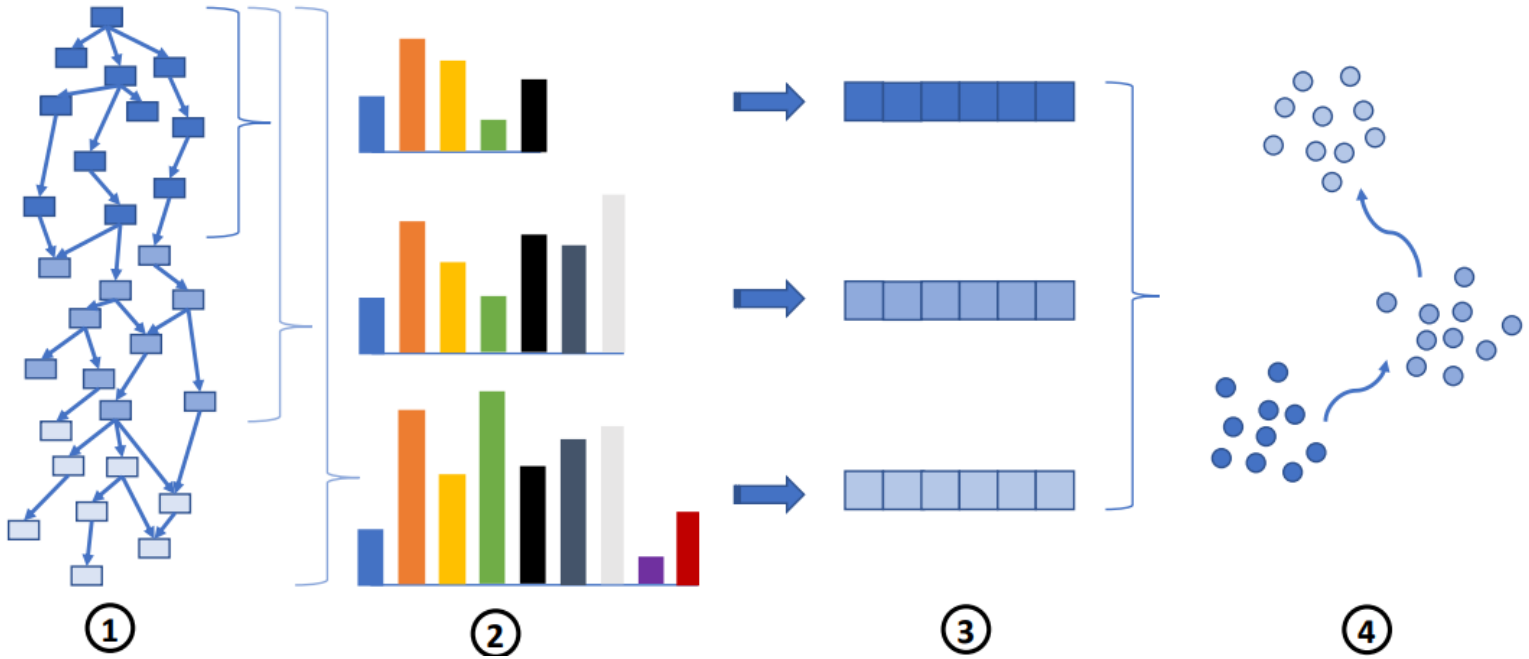
算法原理

T	APT攻击检测
I	主机审计日志数据
P	1.生成流式溯源图 2.建立历史运行直方图 3.生成固定大小的图概要 4.将概要图聚类构建进化模型，使用时捕捉系统动态行为 重复前1-3步骤，输入聚类模型进行检测
O	是否为APT攻击

P	在海量主机日志中实时检测APT攻击
C	在正常行为建模时不会发生攻击并获取全面系统行为
D	如何捕获长时间的系统行为并检测未知APT攻击
L	2020 NDSS

算法原理

- **Unicorn流程**
  - 溯源图->直方图->图概要->聚类模型->异常检测



算法原理

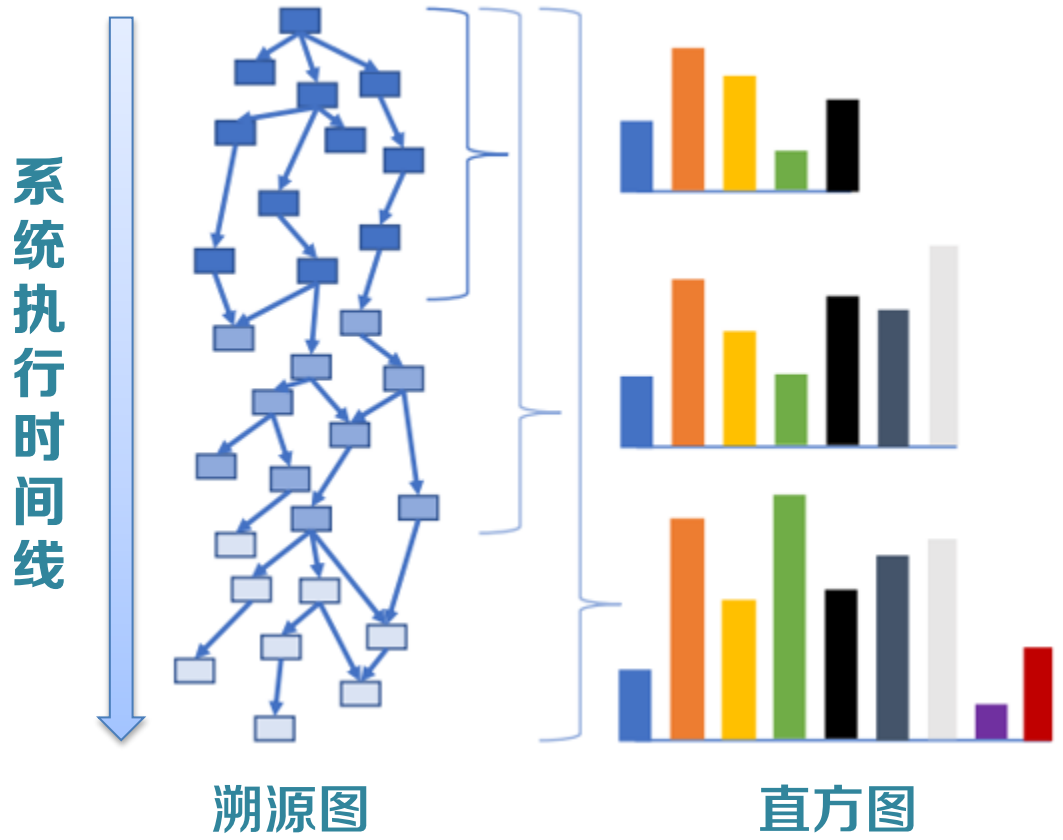
- 核心思想
  - 将APT检测问题看成大规模、带有属性的实时溯源图异常检测问题

- 理想的结合溯源图的APT检测方法
  - 充分利用溯源图的丰富上下文
  - 考虑系统执行的整体持续时间
  - 只学习正常行为的变化，而不是学习攻击的变化

UNICORN

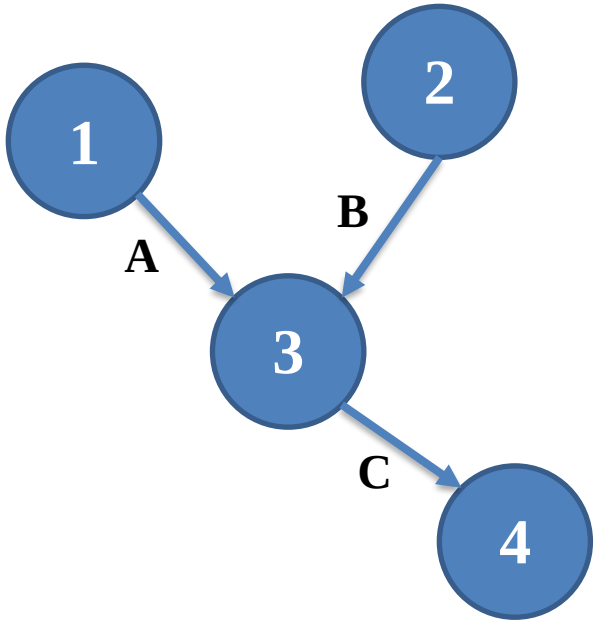
算法原理

- 直方图（Graph Histograms）
  - 目的
    - 有效对溯源图进行比较分析，同时容忍正常行为的微小变化
    - 表示系统执行的历史
  - 有新顶点/边产生则实时更新直方图
    - 标签：用于描述顶点的领域信息，并利用标签对顶点进行分类
    - 计数：每类标签分类下的顶点数
  - 两个图如果相似那么直方图在相似的标签上会有相似的分布



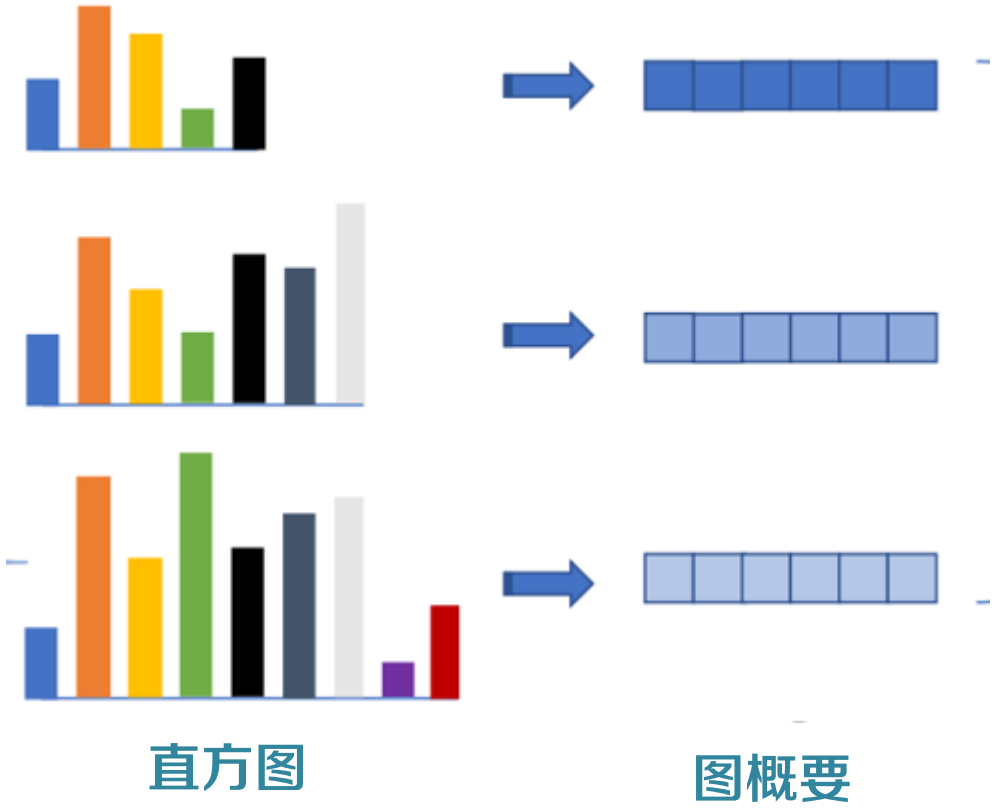
算法原理

- **Weisfeiler-Lehman标签更新算法**
  - 以顶点为中心，进行R次迭代，记录R-hop领域信息
    - 每1次迭代：所有的顶点定义标签1,2,3,4
    - 第2次迭代：添加所有顶点的传入边的标签信息
      - 3的$label_2(3)$= Hash(3,1A2B)
      - 4的$label_2(4)$= Hash(4,3C)
    - 第3次迭代：添加第2次迭代的传入顶点标签
      - 4的$label_3(4)$= Hash($label_2(4), label_2(3)$)
    - 第n次迭代：添加前一次迭代的传入顶点标签
  - 若生成的标签在直方图中存在，则该类别数+1，否则从1开始计数



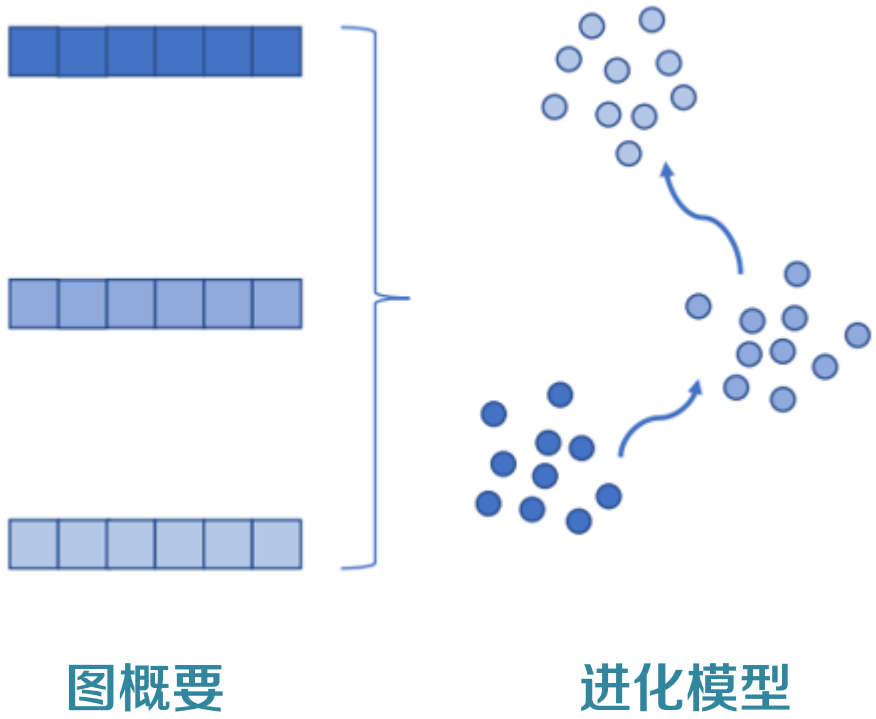
算法原理

- 图概要（Graph Sketches）
  - 为什么要将直方图转化为图概要?
    - 直方图会不断增长，无法直接计算两个直方图的相似度
    - 图概要代表图特征的分布，图概要相似度也代表了图的相似度
  - 相似度保留的数据概要方法（similarity-preserving data sketching）
    - 利用Hash将直方图转化为紧凑的、大小固定的图概要向量
    - 利用归一化 Jaccard 相似度计算直方图的相似性



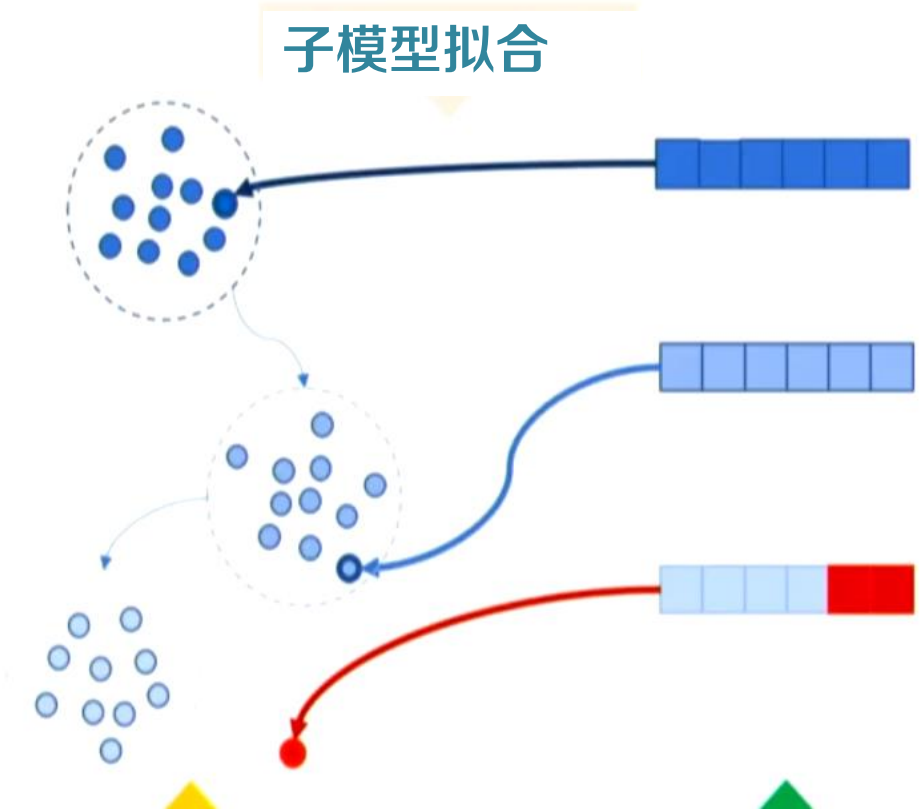
算法原理

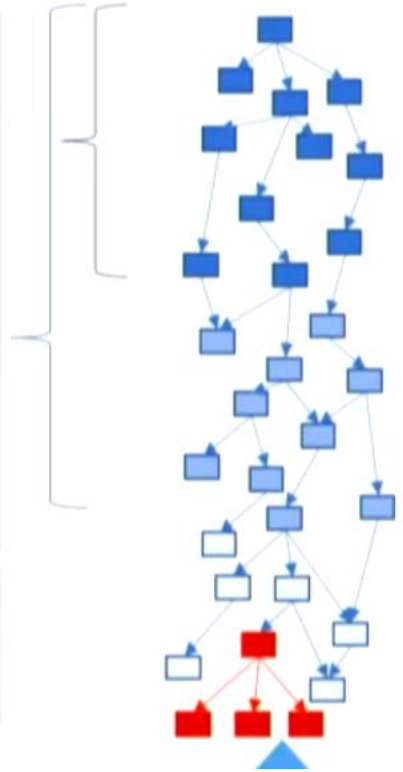
- 进化模型（Evolutionary Models）
  - 聚类算法以检测离群（异常）点
    - 根据Jaccard 相似度为包含时间顺序的图概要构建聚类模型
    - 聚类后的每个簇代表系统执行的“元状态”（如启动、初始化或稳定状态）
  - 进化模型训练
    - 每个训练阶段，利用所有簇中图概要的时间顺序和每个簇的统计值（例如直径）创建一个子模型，代表当前的系统执行状态
    - 进化模型由多个子模型组成，子模型演化代表当前系统执行状态的变化

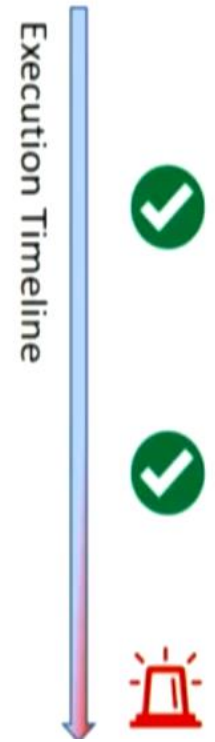


算法原理

- 异常检测
  - 检测阶段，图概要与所有子模型进行比较，尝试拟合到每个子模型中的一个聚类中







异常：无法拟合到任何一个状态　　实时图概要　　实时溯源图　　发出告警

算法原理

- 实验1- Unicorn vs. StreamSpot
  - 实验数据：StreamSpot数据集
    - 共6种场景，5种良性场景，一种攻击场景，每个场景运行100次，生成100个图

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

  - 实验结果

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

- Unicorn R=1时，相当于StreamSpot，仅获取顶点和边的信息
- Unicorn R=3时，捕获了更多的领域信息，显著提升了检测效果

算法原理

- 实验2- DARPA TC 数据集
  - 实验数据：来自三个不同平台

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

FreeBSD
Android
Linux

  - 实验设置：良性数据集的90%用于训练，10%用于测试，R=3
  - 实验结果：UNICORN可运行于多种平台，并保持高性能

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

算法原理

- 实验3- Supply Chain攻击场景
  - 实验数据
    - 在受控的实验环境中设计了两个APT攻击场景构建数据集
      - SC-1： 攻击者将远程访问木马 (RAT) 嵌入Debian软件包，利用CVE-2016-4971受害者的wget请求被重定向到恶意Debian软件包FTP服务器，下载后木马建立通信控制受害主机
      - SC-2： 与SC-1类似，但使用了CVE-2014-6271
  - 实验结果
    - 在攻击者对目标系统有先验知识的情况下，攻击成功前的异常行为减少，检测更困难

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

优劣分析

- **Holmes**
  - 优势
    - 实现了日志到杀伤链模型的映射，提升语义
    - 构建TTP&HSG中间层，缩小了低级别日志数据与相关高层视角之间的巨大语义鸿沟
  - 缺陷
    - 检测效果依赖于专家知识
    - 预定义的边匹配规则过于敏感，难以检测到APT攻击中的0-Day漏洞
- **Unicorn**
  - 优势
    - 可在没有专家知识的前提下检测未知APT攻击
    - 直方图和图概要避免了复杂的图相似度计算
  - 缺陷
    - 未知的正常行为会产生误报
    - 需要足够的良性数据来学习聚类模型

应用总结

- 溯源图应用领域
  - 实现APT攻击的实时检测
  - APT攻击溯源取证
  - 威胁情报分析

- 未来发展
  - 基于图神经网络或知识图谱的APT攻击检测
  - 结合应用程序日志的细粒度溯源分析
  - 场景迁移：供应链、工控等
  - 检测并定位未知APT攻击中的0-Day漏洞

参考文献

[1] Milajerdi S M,Gjomemo R,Eshete B,et al. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows[C]// 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.

[2] Han X,Pasquier T,Bates A,et al. Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats[C]// Network and Distributed System Security Symposium. 2020.

[3] Li Z,Chen Q A,Yang R,et al.Threat Detection and Investigation with System-level Provenance Graphs: A Survey[J]. Computers & Security, 2021(2):102282.

道德经

道可道，非常道。名可名，非常名。无名天地之始。有名万物之母。故常无欲以观其妙。常有欲以观其徼。此两者同出而异名，同谓之玄。玄之又玄，众妙之门。