## 使用对内核审计日志的信息流分析进行威胁检测

经过

萨迪克·莫明尼·米兰迪尔

理学硕士，谢里夫科技大学，2012

论文

在部分满足要求的情况下提交

攻读研究生院计算机科学哲学博士学位

伊利诺伊大学芝加哥分校，2020

伊利诺伊州芝加哥

辩护委员会：

Venkat Venkatakrishnan，主席兼顾问 Rigel Gjomemo

杰森·波拉基斯 Jon A. Solworth

R. Sekar，石溪大学

版权所有*质量控制*

萨德格·莫梅尼·米拉杰迪 2020

献给我亲爱的家人

# 致谢

首先，我要对我的导师 Venkat Venkatakrishnan 教授表示最深切的感谢，感谢他在我攻读博士学位期间的宝贵指导、慷慨支持和不断鼓励。学习。在他的指导下，我学会了如何处理研究问题，批判性地思考、分析和设计对个人和社会有益的实用解决方案。此外，我非常感谢 Rigel Gjomemo 和 Birhanu Eshete 在我研究的每一步中提供的帮助和洞察力。

此外，我要衷心感谢我论文的委员会成员。我要感谢 Rigel Gjomemo、Jason Polakis、Jon A. Solworth 和 R. Sekar，感谢他们的慷慨时间和帮助。本论文提出的工作是在 R. Sekar 教授最初设计和开发的代码库之上实现的，我非常感谢他的慷慨指导和支持。

在我的博士期间。学习期间，我有机会与伊利诺伊大学芝加哥分校的杰出研究人员合作。我要感谢 Md. Nahid Hossain、Emaad Manzoor、Leman Akoglu 和 Scott D. Stoller，他们在我攻读博士学位期间以各种方式帮助和教导了我。我也非常感谢我在微软的实习导师 Mariusz Jakubowski 和 Jugal Parikh，他们与我分享了他们的一流经验。我很幸运有机会认识他们。

最后但并非最不重要的一点是，如果没有我家人的影响，这项工作就不会是一样的，我永远感激他们。我一直很感谢鼓励我的父亲

## 致谢（续）

出国留学，妈妈的耐心和毅力，哥哥一直在我身边，妻子无条件的爱。

SM

# 作者的贡献

本论文包括以前发表的材料，这些出版物的版权声明表明我们允许复制这些材料，可以在附录 D 中找到。在这里，我们描述了每章包含这些已发表作品的内容，以及作者的贡献。每部作品的作者：

* 第 1 章介绍了我们在这篇论文中要解决的问题、概述和这项工作的结构。
* 第 2 章提供了将在整个论文其余部分中使用的概念的必要背景。本章部分包括 [1-4] 中发表的材料的摘录和数字。我是 [4] 中的主要作者和贡献者，并且还对工作进行了实施和评估。 Birhanu Eshete 和 Rigel Gjomemo 为本文的构思和写作做出了慷慨贡献，V.N. Venkatakrishnan 是这项工作的总体主管。我在 [1-3] 中的贡献解释如下，因为它们构成了以下章节的主要内容。
* 第 3 章介绍了 [1] 的部分内容，这是石溪大学 (SBU) 和伊利诺伊大学芝加哥分校 (UIC) 之间的合作成果。在这项工作中，我是 UIC 的主要学生作者，并为实施和评估的重要部分做出了贡献。 Md. Nahid Hossain、Junao Wang、Birhanu Eshete、Rigel Gjomemo 和 Scott Stoller 为论文的实施和撰写做出了贡献。

## 作者的贡献（续）

R. Sekar 和 V.N. Venkatakrishnan 分别是 SBU 和 UIC 的炒锅主管

* 第 4 章介绍了 [2] 中发表的工作，我是其中的主要作者，并进行了实施和评估。 Rigel Gjomemo、Birhanu Eshete 和 R. Sekar 为论文的想法和写作做出了慷慨贡献，V.N. Venkatakrishnan 是这项工作的总体主管。
* 第 5 章涵盖了 [3] 中发表的工作，我是其中的主要作者，并进行了实施和评估。 Birhanu Eshete 和 Rigel Gjomemo 为论文的构思和写作做出了慷慨贡献，V.N. Venkatakrishnan 是这项工作的总体主管。
* 第 6 章介绍了相关工作，并部分借鉴了 [1-3] 中的一些内容。
* 第七章给出了结论并对未来的研究方向提出了一些建议。

# 目录

### 章节 页

**1** 介绍 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1 论文陈述。 . . . . . . . . . . . . . . . . . . . . . . 2

1.2 贡献。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 论文概述。 . . . . . . . . . . . . . . . . . . . . . . . 3

* + 1. 侦探：攻击场景重建。 . . . . . . . . . . . 4
    2. Holmes：实时 APT 检测。 . . . . . . . . . . . . . . . 5
    3. 波洛：网络威胁搜寻。 . . . . . . . . . . . . . . . . . 6

1.4 论文结构。 . . . . . . . . . . . . . . . . . . . . . . . 7

**2** 背景 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.1 高级持续威胁。 . . . . . . . . . . . . . . . . . . . 9

2.2 防御方法。 . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.1 入侵检测 。 . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.2 取证和威胁搜寻。 . . . . . . . . . . . . . . . . . . 13

2.3 事件关联 . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.4 内核审计日志。 . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.5 信息流分析。 . . . . . . . . . . . . . . . . . . . . 19

2.6 透明计算数据集。 . . . . . . . . . . . . . . . . 20

### SLEUTH：实时攻击场景重建-

**来自 COTS 审计数据的意见。 . . . . . . . . . . . . . . . . . .** 22

3.1 介绍 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

3.2 主内存依赖图。 . . . . . . . . . . . . . . . . 26

3.3 标签和攻击检测。 . . . . . . . . . . . . . . . . . . . . 27

3.3.1 标签设计。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

3.3.2 基于标签的攻击检测。 . . . . . . . . . . . . . . . . . . . 30

3.4 政策框架。 . . . . . . . . . . . . . . . . . . . . . . . . . 34

3.4.1 标记初始化策略。 . . . . . . . . . . . . . . . . . . . . . 36

3.4.2 标签传播策略。 . . . . . . . . . . . . . . . . . . . . . 37

3.5 基于标签的双向分析 . . . . . . . . . . . . . . . . 38

3.5.1 后向分析。 . . . . . . . . . . . . . . . . . . . . . . . . . 38

3.5.2 前向分析。 . . . . . . . . . . . . . . . . . . . . . . . . . . 41

3.5.3 重建和演示。 . . . . . . . . . . . . . . . . 42

3.6 实验评价。 . . . . . . . . . . . . . . . . . . . . . 42

3.6.1 执行 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

3.6.2 数据集。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

3.6.3 订婚设置。 . . . . . . . . . . . . . . . . . . . . . . . . . 44

3.6.4 选定的重建结果。 . . . . . . . . . . . . . . . . . 45

3.6.5 整体效果。 . . . . . . . . . . . . . . . . . . . . . . . . 49

3.6.6 良性环境中的误报。 . . . . . . . . . . . . . 51

3.6.7 运行时和内存使用。 . . . . . . . . . . . . . . . . . . . . 52

3.6.8 代码和数据拆分标签的好处。 . . . . . . . . . . . . . 54

3.6.9 分析选择性。 . . . . . . . . . . . . . . . . . . . . . . . . . 55

3.7 概括 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

### 福尔摩斯：通过可疑信息流的关联进行实时 APT 检测。 . . . . 57

4.1 介绍 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

4.2 一个运行的例子。 . . . . . . . . . . . . . . . . . . . . . . . . 60

4.3 方法概述。 . . . . . . . . . . . . . . . . . . . . . . . . 63

4.4 系统设计 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

4.4.1 数据收集和表示。 . . . . . . . . . . . . . . . 67

4.4.2 TTP 规范。 . . . . . . . . . . . . . . . . . . . . . . . . . 68

4.4.3 HSG 建设。 . . . . . . . . . . . . . . . . . . . . . . . . . 70

4.4.4 避免虚假的依赖。 . . . . . . . . . . . . . . . . . 71

4.4.5 降噪。 . . . . . . . . . . . . . . . . . . . . . . . . . . 75

4.4.6 信号相关和检测。 . . . . . . . . . . . . . . . . 76

4.5 执行 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

4.6 实验评价。 . . . . . . . . . . . . . . . . . . . . . 83

4.6.1 数据集。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83

4.6.2 设置 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85

4.6.3 结果简而言之。 . . . . . . . . . . . . . . . . . . . . . . . . 85

4.6.4 攻击场景。 . . . . . . . . . . . . . . . . . . . . . . . . . . 87

4.6.5 寻找最佳阈值。 . . . . . . . . . . . . . 88

4.6.6 表现 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90

4.6.7 现场实验。 . . . . . . . . . . . . . . . . . . . . . . . . . . 93

4.7 概括 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

### POIROT：将攻击行为与网络威胁搜寻的内核审计记录保持一致。 . . . . 97

5.1 介绍 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97

5.2 方法概述。 . . . . . . . . . . . . . . . . . . . . . . . . 100

5.2.1 出处图构建。 . . . . . . . . . . . . . . . . . 100

5.2.2 查询图构建。 . . . . . . . . . . . . . . . . . . . . 101

5.2.3 图形对齐。 . . . . . . . . . . . . . . . . . . . . . . . . . . 104

5.3 算法。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

5.3.1 对齐度量。 . . . . . . . . . . . . . . . . . . . . . . . . . . 107

5.3.1.1 影响分数。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 109

5.3.1.2 对齐分数。 . . . . . . . . . . . . . . . . . . . . . . . . . . 113

5.3.2 尽力而为的相似性搜索。 . . . . . . . . . . . . . . . . . . . 114

5.4 评估 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121

* + 1. 对 TC 数据集的评估。 . . . . . . . . . . . . . . . . . . 122
    2. 公共攻击评估 . . . . . . . . . . . . . . . . . . . 127
    3. 对良性数据集的评估。 . . . . . . . . . . . . . . . . . . 137

5.4.4 效率 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139

5.5 概括 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

**6** 相关工作 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

6.1 出处图。 . . . . . . . . . . . . . . . . . . . . . . . . . 143

6.2 入侵检测 。 . . . . . . . . . . . . . . . . . . . . . . . . . 145

6.3 警报相关性。 . . . . . . . . . . . . . . . . . . . . . . . . . . 146

6.4 查询处理系统。 . . . . . . . . . . . . . . . . . . . . 147

6.5 行为发现。 . . . . . . . . . . . . . . . . . . . . . . . . . 147

6.6 图形模式匹配。 . . . . . . . . . . . . . . . . . . . . . 148

**7 结论和未来的工作** . . . . . . . . . . . . . . . . 149

附录。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153

**附录 A** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154

附录 B。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159

附录 C。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169

附录 D。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173

引文。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 185

维他. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 198

**桌子 页**

I 一些最具破坏性的网络攻击。 . . . . . . . . . . . . . 10

II 系统事件类型。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

III 信息流事件。 . . . . . . . . . . . . . . . . . . . . . . . . . 18

IV 具有政策触发点的边缘。 . . . . . . . . . . . . . . . . . . . 36

V 每个活动的数据集。 . . . . . . . . . . . . . . . . . . . . . . . 43

1. Sleuth 是针对典型 APT 活动的结果。 . . . . . 49
2. 攻击场景重构总结。 . . . . . . . . . . . . . . . 50
3. 良性环境中的误报。 . . . . . . . . . . . . . . . . 51
4. 场景重建的内存使用和运行时间。 . . . . . . . 53

X 减少（错误）警报。 . . . . . . . . . . . . . . . . . . . . . . 54

XI 前向分析和简化。 . . . . . . . . . . . . . . . . . 55

XII 示例 TTP。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

XIII NIST 严重程度评定量表。 . . . . . . . . . . . . . . . . . . . . . . . 77

XIV 代表性的 TTP。 . . . . . . . . . . . . . . . . . . . . . . . . . . 81

XV 数据集。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

XVI 分配给攻击场景的分数。 . . . . . . . . . . . . . . . . . 86

XVII 符号。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

1. 查询图的特征。 . . . . . . . . . . . . . . . . . . . . 122
2. Poirot 的图形对齐分数。 . . . . . . . . . . . . . . . . . . 125

XX 恶意软件报告。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128

1. 图 23 中查询图的节点标签及其对齐方式。 129
2. 与相关系统的效率比较。 . . . . . . . . . . . . 139
3. 日志、消费和搜索时间的统计。 . . . . . . . . . 140

### 数字 页

1 APT 生命周期。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2 示例出处图片段。 . . . . . . . . . . . . . . . . . . . 19

3 侦探系统概述。 . . . . . . . . . . . . . . . . . . . . . . . . 24

1. 从战役 W-2 重建的情景图。 . . . . . . . . . . . 46
2. 从 F-3 战役重建的情景图。 . . . . . . . . . . . 47
3. 运行示例的出处图。 . . . . . . . . . . . . . . 61

7 福尔摩斯方法。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

8 运行示例的高级场景图。 . . . . . . . . . 72

9 福尔摩斯建筑。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

10 数据集统计 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

1. 学习和路径因子对降噪的影响。 . . . . . . . 87
2. 场景 1 的 HSG（路过下载）。 . . . . . . . . . . . . . . . . 89
3. 攻击检测的精度、召回率和 F 分数。 . . . . . . . . . . 90

14 消费统计。 . . . . . . . . . . . . . . . . . . . . . . . . . . 91

15 内存消耗。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 92

16 CPU 利用率。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

17 攻击与良性 HSG 的累积分布函数。 . . . 94

18 波洛方法概述。 . . . . . . . . . . . . . . . . . . . . . . . . 101

19 代理狗恶意软件的查询图。 . . . . . . . . . . . . . . . . . . 103

20 示例图形对齐。 . . . . . . . . . . . . . . . . . . . . . . . . 108

21 候选人人数 . . . . . . . . . . . . . . . . . . . . . . . . . . . 124

1. 场景的查询图和对齐方式：Linux-2。 . . . . . . . . . . 126
2. Carbanak、DustySky、Uroburos、OceanLotus、njRAT、

和 HawkEye 恶意软件。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133

1. 选择最佳阈值。 . . . . . . . . . . . . . . . . . 138
2. 从战役 L-1 重建的情景图。 . . . . . . . . . . . 155
3. 从战役 F-1 重建的情景图。 . . . . . . . . . . . 156
4. 从战役 F-2 重建的情景图。 . . . . . . . . . . . 157
5. 从战役 W-1 重建的情景图。 . . . . . . . . . . . 157
6. 从战役 L-3 重建的情景图。 . . . . . . . . . . . 158

30 情景 2 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160

31 情景 3 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

32 情景 4 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

33 情景 5.1 的 HSG（永恒之蓝）。 . . . . . . . . . . . . . . . . . . . 165

34 场景 5.2 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166

35 情景 6 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167

36 场景 7.1 的 HSG。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168

|  |  |
| --- | --- |
| 易于 | 高级持续威胁 |
| 攻击&CK | 对抗性战术、技术和常识 |
| C&C | 命令与控制 |
| 中央处理器 | 中央处理器 |
| CTI | 网络威胁情报 |
| CVSS | 通用漏洞评分系统 |
| 达帕 | 国防高级研究计划局 |
| ETW | Windows 事件跟踪 |
| 国标 | 技嘉 |
| 毛利率 | 图模式匹配 |
| HTML | 超文本标记语言 |
| HTTP | 超文本传输​​协议 |
| HSG | 高级场景图 |
| 身份识别系统 | 入侵侦测系统 |
| 国际奥委会 | 妥协指标 |
| 知识产权 | 互联网协议 |
| 马力士 | 入侵防御系统 |

|  |  |
| --- | --- |
| JS | JavaScript |
| 行车记录仪 | 代码行 |
| 操作系统 | 操作系统 |
| 可编程逻辑控制器 | 可编程逻辑控制器 |
| 鼠 | 远程管理工具 |
| SIEM | 安全信息和事件管理 |
| TC | 透明计算项目（DARPA） |
| TTP | 战术、技术和程序 |
| 网址 | 统一资源定位器 |
| 万维网 | 全球资讯网 |

内核审计日志是一个丰富的信息来源，包含主机系统中系统实体之间的因果依赖关系和信息流的历史。内核审计日志的主流用途是用于追溯调查网络攻击的取证任务。在本论文中，我们开发了有效的方法，利用内核审计日志来完成复杂的实时安全任务，例如高级和持久威胁 (APT) 检测、攻击场景重建和网络威胁追踪。为此，我们首先将内核审计日志处理成存储在主内存中的平台中立的出处图，并将其用作运行各种分析的基础。对于 APT 检测，我们开发了产生检测信号的技术，以指示存在一组协调的可疑活动。对于实时攻击场景重建，我们开发了一种利用信息流策略来识别网络攻击中涉及的实体和事件的方法。 对于网络威胁搜寻，我们开发了一种不精确的图形模式匹配方法，以将从网络威胁情报中提取的查询图与由内核审计日志构建的出处图对齐。

针对为对抗性交战而设计的真实世界 APT 场景评估了所提出方法的有效性。这些实验包含数百万条记录，并共同涉及来自运行 Linux、FreeBSD 和 Windows 等操作系统平台的各种主机的数月审计日志收集活动。结果表明，所提出的方法能够有效地搜索这些审计日志，并以高精度和低误报率实时查明威胁。此外，这些方法有效地产生了总结

协助调查人员进行网络响应行动的攻击活动。总之，本论文表明，从内核审计日志中推断出的低级因果信息可用于实现稳健可靠的威胁检测方法，通过生成紧凑的攻击步骤可视化图来有效地查明威胁并揭示攻击的高级图.

# 介绍

我们正在目睹由熟练的对手发起的有针对性的网络攻击（“企业高级和持续性威胁 (APT)”）[5] 迅速升级。这些攻击不断发展并变得越来越复杂。为了应对这些攻击，网络防御者利用各种解决方案，例如安全信息和事件管理 (SIEM) 系统。这些防御解决方案通过聚合来自多个来源（例如最终用户设备、服务器、网络设备 IDS/IPS、防火墙等）的安全相关事件来进行推理。尽管这些系统通常很有用，但它们通常缺乏 (a) 对警报和实际入侵实例之间存在的复杂关系的理解，以及 (b) 将长时间发生在不同主机上的攻击步骤拼凑起来所需的精度（几周，或者在某些情况下，几个月）。相反，需要大量的人工和专业知识来拼凑多个安全工具发出的大量警报。 因此，许多攻击活动会在数周甚至数月内错过 [6, 7]。

在 Backtracker [8, 9] 中，通过引入从内核审计日志构建的出处图的概念，首次探索了将导致攻击的因果链拼凑在一起的问题。在这些图中，顶点表示主体（进程）和对象（文件、套接字），边表示审计事件（例如，读、写、执行和连接等操作）。出处图提供了因果关系和流程的详细历史

系统实体之间的信息。多年来，安全分析师一直使用出处图进行取证任务，以追溯调查网络攻击。

在本论文中，我们首先将内核审计日志处理成存储在主内存中的出处图，该图适用于运行高性能分析。利用该图，我们开发了执行各种安全任务的基本方法，例如 APT 检测、攻击场景重建和网络威胁搜寻。所提议的技术使用出处知识，根据企业活动的因果联系以及大数据技术来更好地防御那些阻碍当前防御并困扰当今企业的阴险攻击。特别是，本论文包括

(1) Sleuth [1]，一种基于标签的攻击场景重建方法，根据参与网络攻击的可能性对实体和事件进行优先级排序，(2) Holmes [2]，一种 APT 检测系统，可有效利用两者之间的相关性在攻击者活动期间出现的可疑信息流，以产生指示存在一组协调的可疑活动的检测信号，以及 (3) Poirot [3]，一种新的不精确图模式匹配方法，用于评估查询图之间的对齐从网络威胁情报相关性和出处图中提取。

#### 论文陈述

我们的主要论点是，从内核审计日志中推断出的低级因果信息可用于实现稳健可靠的威胁检测方法，通过生成紧凑的攻击步骤可视化图来有效地查明威胁并揭示攻击的高级图。

#### 贡献

本论文的贡献如下：

* + - 我们利用内核审计日志的紧凑内存依赖图表示，作为运行各种高性能分析的基础。
    - 我们开发了一种基于标签的方法来识别最有可能参与攻击的主题、对象和事件。标签使我们能够确定优先级，集中我们的分析，并总结我们对对象和主题的可信度和敏感性的评估。
    - 为了有效地遏制高级攻击活动，我们开发了高级场景图，提供了当前活动的非常紧凑、直观的摘要。这样的摘要使分析人员能够快速确定是否存在重大入侵，了解攻击者最初是如何破坏安全的，并确定攻击的影响。
    - 我们将网络威胁搜索制定为图形模式匹配问题，以可靠地检测已知的网络攻击。匹配是基于一种新颖的图相似性度量进行的，该度量评估由网络威胁情报 (CTI) 相关性构建的查询图与由内核审计日志记录构建的出处图之间的对齐。

#### 论文概述

本论文包括三个主要章节，研究从三个不同方面从根本上加强企业安全性的技术： Sleuth 引入了一个紧凑的内存

可在大型企业中部署的内核审计日志的图形表示，旨在及时重建攻击场景（第 3 章），Holmes 通过将不可靠的松散警报融合为强检测信号（第 4 章），将信号关联用于 APT 检测（第 4 章），以及 Poirot在起源图中搜索特定威胁行为的嵌入，以揭示成功攻击活动的步骤（第 5 章）。下面，我们将简要解释本论文的三个主要章节。

#### SLEUTH：攻击场景重构

Sleuth [1] 是一个基于内核审计日志实时分析的企业主机攻击场景重构系统。为了提供高效的事件存储和分析框架，Sleuth 利用了内核审计日志的紧凑主内存表示。主存表示的图算法可以比磁盘表示快几个数量级，这是实现实时分析能力的一个重要因素。在我们的实验中，我们能够在 14 秒内处理来自 FreeBSD 系统的 79 小时审计数据，主内存使用量为 84MB。这种性能表示分析速率比生成数据的速率快 20K 倍。

Sleuth 使用基于标签的方法来识别最有可能参与攻击的实体和事件。标签使我们能够通过编码对数据和流程的可信度和敏感性的评估来确定优先级和重点分析。此评估基于从审计日志中得出的数据来源。从这个意义上说，来自审计数据的标签类似于粗粒度信息流标签。 Sleuth 利用可定制的策略框架，可以根据实体标签发出检测警报。

从攻击检测组件产生的警报开始，我们的反向分析算法遵循图中的依赖关系来识别攻击的来源。从源头开始，我们使用前向搜索对对手的行为进行全面的影响分析。我们提出了几个修剪这些搜索的标准，以产生一个紧凑的图。我们还提出了一些转换，进一步简化了这个图，并生成了一个以简洁和语义上有意义的方式直观地捕捉攻击的图。实验表明，我们的基于标签的方法非常有效：例如，Sleuth 可以分析 3850 万个事件并生成仅包含 130 个事件的攻击场景图，这意味着事件量减少了五个数量级。

#### HOLMES：实时 APT 检测

典型的 APT 包括成功的渗透（例如，通过下载或鱼叉式钓鱼攻击）、侦察、命令和控制 (C&C) 通信（有时使用远程访问木马 (RAT)）、权限提升（通过利用漏洞） ，通过网络横向移动，泄露机密信息等等。这一系列相关活动被称为 APT 杀伤链。杀伤链为理解和映射 APT 参与者的动机、目标和行动提供了参考。尽管不同 APT 之间的具体攻击步骤可能有很大差异，但高级 APT 行为通常符合相同的杀伤链。我们对来自 [5] 的数百份 APT 报告的分析表明，大多数 APT 由这些步骤的一个子集组成，如果不是全部的话。更重要的是，我们观察到这些步骤需要因果联系，这种联系是攻击正在展开的主要迹象。

Holmes [2] 从主机审计数据开始，并产生一个检测信号，将正在进行的 APT 活动的阶段映射到杀伤链。为了弥合低级系统调用视图和高级杀伤链视图之间的语义鸿沟，我们基于 MITRE 的 ATT&CK 框架 [10] 构建了一个中间层，该框架描述了近 200 种定义为战术、技术和程序的行为模式(TTP) 在野外观察到。每个 TTP 都定义了一种可能的方式来实现特定的高级功能。例如，可以使用 11 个不同的 TTP 实现在受感染 Linux 系统中的持久性能力，每个 TTP 代表 ATT&CK 框架中可能的较低级别操作序列，例如安装 rootkit、修改引导脚本等上。这些较低级别的操作更接近审计日志的抽象级别，因此可以根据出处图中的节点和边来描述 TTP。

利用 TTP 间的信息流依赖关系，我们将各种 TTP 关联起来，并将它们映射到高级 APT 步骤。最终的相关结果以我们称为高级场景图 (HSG) 的图的形式进行汇总。 HSG 随时提供活动进展的紧凑、直观的抽象。为了区分构成攻击的 HSG 和良性攻击，我们的方法为每个 HSG 分配一个严重性评分，如果该评分超过某个阈值，则会发出警报。

* + 1. 波洛：**网络威胁搜寻**

网络威胁情报 (CTI) 被用于搜索可能已在很长一段时间内破坏企业网络而未被发现的攻击指标。为了进行更有效的分析，CTI 开放标准包含了描述性关系，显示

指标或可观察量如何相互关联。然而，这些关系要么在信息收集中被完全忽略，要么不用于威胁搜寻。因此，绝大多数当前的威胁搜寻方法仅在网络威胁的碎片化视图中运行 [11, 12]，例如签名（例如，工件的哈希）、可疑文件/进程名称和 IP 地址（域名） .

Poirot [3] 使用入侵指标 (IOC) 工件之间的关系，其中包含有关受感染系统内攻击行为的基本线索，以揭示成功攻击活动的步骤。简而言之，给定基于图的 IOC 及其之间的关系表示 APT 的整体行为，我们称之为查询图，我们的方法有效地在更大的出处图中找到了该查询图的嵌入，这包含长期内核审计日志的表示。我们将威胁搜寻定义为一个图形模式匹配 (GPM) 问题，用于搜索与查询图中描述的系统实体之间的因果依赖关系或信息流。我们的技术方法基于一种新颖的相似性度量，该度量评估由 CTI 相关性构建的查询图与由内核审计日志记录构建的出处图之间的对齐。

# 1.4 论文结构

本文其余部分的组织如下：在第 2 章中，我们提供了 APT 的背景材料以及防御方法和工具。第 3 章、第 4 章和第 5 章分别介绍了侦探、福尔摩斯和波洛。第 6 章重点介绍了相关工作和

将其与本论文中介绍的工作进行了比较。最后，我们在第 7 章对我们的工作和未来的研究方向进行了总结。

**第2章**

**背景**

*本章包括 [1-4] 中发表的材料的摘录和数字。*

在本章中，我们首先提供有关高级持续性威胁和当前防御方法的必要背景信息。然后，我们将事件关联解释为应对 APT 活动并提及当前方法的必要方法。因此，我们将内核审计日志描述为包含系统活动历史的数据源，并提到了适用于事件关联的各种信息流跟踪技术。最后，我们详细阐述了用于评估我们工作的透明计算数据集。虽然本章中的材料并不全面，但它为本文的读者提供了必要的背景信息，以理解后续章节中介绍的概念。

# 高级持续威胁

Advanced Persistent Threat 是一种有针对性的隐蔽网络攻击，它遵循多阶段威胁工作流程 [13] 侵入企业网络，目标是获取宝贵信息、更改或破坏关键基础设施。 APT 参与者在资源、工具和知识方面是先进和复杂的，他们将其活动隐藏在其他正常事件之间以持续很长一段时间（几周，或者在某些情况下，几个月）。互联网协会的在线信任联盟估计发生了 200 万次网络攻击

9

表一

针对不同行业的一些最具破坏性的网络攻击

|  |  |  |
| --- | --- | --- |
| **姓名** | **年** | **描述** |
| 泰坦雨 | 2003 | 针对美军（被称为历史上最大的财富转移）窃取飞机蓝图，  天基激光器、导弹导航和核潜艇 |
| 震网 | 2010 | 瞄准核设施（称为第一个数字武器） |
| 目标 | 2013 | 4000 万张支付卡凭证和 7000 万条客户记录丢失 |
| 雅虎 | 2014 | 与至少 5 亿用户帐户相关的信息被盗 |
| OPM | 2015 | 被联邦官员描述为历史上最大的政府数据泄露事件之一  美国 |
| 深熊猫 | 2015 | 针对医疗保健服务（泄露多达 8000 万客户的财务和医疗记录） |
| 黑色能量 | 2016 | 破坏了电网系统硬盘驱动器上的所有可用数据 |
| Equifax | 2017 | 暴露姓名、SSN、出生日期、地址，在某些情况下，还暴露了  目前美国人口的 44% |
| 万豪酒店 | 2018 | 数以亿计的客户记录，包括信用卡和护照号码，被泄露  由攻击者 |

2018 年，总共造成了超过 450 亿美元的损失。除了经济损失之外，网络攻击还可能通过针对关键基础设施（例如，在 Stuxnet 蠕虫 [14] 中受损的 PLC）造成危及生命的灾难。表 I 代表了自 2003 年以来针对不同部门的一些最具破坏性的网络攻击。如图所示，商业公司、政府组织、宗教或教育机构都是网络攻击的目标。

在第一份关于高级和持续性威胁的详细报告（标题为 APT1 [13]）中，安全公司 Mandiant 披露了全球 APT 参与者的目标和活动。这些活动包括从不同行业的至少 141 个组织窃取数百 TB 的敏感数据（包括业务计划、技术蓝图和测试结果）。他们估计目标组织中恶意软件的平均持续时间为 365 天。从那时起，全球范围内涉及强大行为体（包括民族国家行为体）的记录在案的 APT 名单越来越多。

**最初的妥协**

**移动**

**横向**

**维持**

**在场**

**建立**

**立足点**

**内部的**

**侦察**

**完全的**

**使命**

**提升权限**

图 1. APT 生命周期。

了解 APT 参与者的动机和行动在应对这些威胁的挑战中起着至关重要的作用。为了进一步理解，Mandiant 报告还提供了一个 APT 生命周期模型（图 1），也称为杀伤链，可以让人们了解 APT 步骤如何共同实现其参与者的目标。典型的 APT 攻击包括成功的渗透（例如，驱动下载或鱼叉式网络钓鱼攻击）、侦察、命令和控制 (C&C) 通信（有时使用远程访问木马 (RAT)）、权限提升（通过利用漏洞）、通过网络横向移动、泄露机密信息等。简而言之，杀伤链为理解和映射 APT 参与者的动机、目标和行动提供了参考。

自第一份 Mandiant 报告发布以来，APT 变得越来越复杂。多年来使用的各种漏洞利用的细节有所不同，但高级步骤保持不变

大部分是一样的。在调查大约 300 份 APT 报告 [5] 时，我们观察到大多数 APT 仍然遵循大部分符合图 1 所示杀伤链的步骤。

# 防御方法

在本节中，我们将介绍安全分析师用来应对网络攻击的不同防御方法。首先，我们解释入侵检测技术，然后，我们将取证和威胁搜寻作为两种攻击后分析技术进行解释。

# 入侵检测

入侵检测系统 (IDS) 的主要目标是发现恶意活动，最好是实时发现，然后阻止和/或向系统管理员报告。入侵检测系统有两大类，即基于主机的和基于网络的。网络 IDS 在网络上部署的传感器上运行，而基于主机的 IDS 则基于主机上运行的软件生成的日志文件运行。在商业上，基于网络的传感器已被证明比基于主机的传感器更可行。网络传感器可以部署在几个位置，例如网络网关，但可以监控整个企业网络。相比之下，基于主机的传感器需要部署在每台主机上。更重要的是，它们的实现会因主机上部署的特定操作系统和软件而异。网络传感器的缺点是它们对既是入侵目标又是入侵代理的软件系统提供的洞察力非常有限。

IDS 使用的算法和技术可分为三大类：

*误用检测、异常检测和基于规范的检测。*

误用检测是入侵检测系统用于识别受感染文件的最早技术之一。该技术依赖于从已知攻击中提取的预定义攻击特征。因此，解释警报相对容易，因为匹配的签名与特定的攻击名称或标识符相关联。这种技术的最大限制是发现没有已知特征的未知威胁。随着网络攻击变得更加复杂，它们可能涉及零日攻击或具有不同签名的已知攻击的突变，从而逃避检测。

基于异常的入侵检测是识别与规范不同的意外事件的过程。为此，在良性环境中的训练阶段学习正常行为模型。接下来，该模型用于检测操作环境中测试阶段的偏差。异常检测解决了误用检测技术在检测新攻击方面的主要限制。然而，它们在现实世界中的使用受到了相对较高的误报率以及培训需求的阻碍。

基于规范的入侵检测侧重于行为而不是特定攻击的独特特征。因此，它能够检测未知威胁。但是，由于定义准确的行为规范具有挑战性，因此该技术并未得到太多部署。

# 取证和威胁搜寻

取证分析是在入侵检测系统发出检测信号后调查和检查证据的行为。此过程跟踪与事件相关的证据，以提供更精确的描述和攻击的附加背景。此信息

将帮助我们找到攻击的来源、攻击的进行方式以及攻击的规模。因此，安全分析人员可以找到易受攻击的服务，改进他们的系统以防止将来发生这种攻击，发现受攻击者影响的受影响单位，并隔离或恢复这些单位，以阻止这次攻击造成的额外损失。

当与 APT 相关的报告发布时，企业安全分析师中出现的一个常见问题是，他们的企业是否已成为该 APT 的目标。此过程通常称为威胁搜寻。以高度的信心回答这个问题通常需要对企业的主机和网络日志进行冗长而复杂的搜索和分析，识别这些日志中出现在 IOC 描述中的实体，并最终评估特定 APT 成功渗透到企业的可能性.

威胁搜寻与法医分析的不同之处在于它不是从检测信号开始的。在威胁搜寻中，安全分析人员主动迭代地搜索日志，以发现避开现有安全解决方案的攻击。这些搜索通常会寻找可以高可信度指示攻击发生的 IOC，例如恶意软件文件的哈希值、低信誉 IP 地址和域名，或某些攻击者使用的唯一名称。

# 事件关联

由于 APT 活动采用多种攻击向量，因此对具有孤立视图的防御方法提出了巨大挑战。为了获得正在进行的 APT 活动的高级图片，需要对来自不同来源的事件进行处理和关联。当今的企业能够从不同的环境中收集极其丰富的遥测数据集

例如防火墙、IPS/IDS、端点和服务器安全系统、身份和访问管理工具、应用程序安全工具、应用程序防火墙和数据库防火墙。但是，将这些源关联或映射在一起的能力有限。特别是，安全分析师很难将真正重大的攻击（众所周知的“大海捞针”）与背景噪声区分开来，而准确地“连接点”（即拼凑起来）仍然是一项重大挑战跨越多个应用程序或主机并持续很长时间的攻击活动的片段。

安全信息和事件管理 (SIEM) 系统聚合来自多个来源的安全相关事件和警报，并通过关联它们进行推理。这些系统的一些例子是 Splunk [15]、LogRhythm [16] 和 IBM QRadar [17]。该领域的研究主要分为以下几类[18]：

* *统计分析：*在这种方法 [19] 中，多步攻击的连续步骤是基于时间上彼此接近的警报的统计特征相关的。然而，由于攻击是非常罕见的事件，并且可用的训练数据量通常是有限的，因此误报的管理非常困难。
* 基于前置条件的分析：在这种方法中，如果攻击的后置条件有助于后续攻击的前置条件 [20]，则两个攻击是相关的。然而，这种类型的推理需要开发攻击模型（描述前置条件和后置条件），这可能并不总是实用的。
* 基于图的分析：该技术对涉及图结构中一组协调的攻击阶段的实体进行建模。然后，可以应用各种图形分析

用于警报相关性[21]，例如谱聚类和页面排名。也可以基于学习两个事件之间的相关性的可预测性来关联警报 [22]。

在本论文中，我们使用内核审计日志作为宝贵的资源来拼凑事件和活动的因果链。在下一节中，我们将提供有关内核审计日志的必要背景信息。

# 内核审计日志

应用程序无权直接访问系统资源，例如 CPU、内存或输入/输出 (I/O)。操作系统的内核充当软件应用程序和系统资源之间的桥梁，应用程序应该向内核请求服务。这些称为系统调用的请求由内核处理，并向应用程序发送响应。内核具有拦截每个系统调用并记录它们的能力，并且有不同的工具可以用于此目的。例如，在 Linux 中，auditd 可以是审计数据的来源，而对于 BSD，它可以是 dtrace，对于 Windows，它可以是 ETW。为了保持一致且独立于低级系统细节，我们收集这些原始审计数据并将其处理为与操作系统无关的格式。这是我们用于本文提出的系统的输入格式。此输入捕获与主体（用户）、文件（例如，I/O 操作、文件创建、所有权、 和权限）、内存（例如，mprotect 和 mmap）进程（例如，创建和权限更改）和网络连接。为了提供一种方便的形式来推理系统活动并允许有效地跟踪因果关系和信息流，内核审计日志可以表示为图表，

表二

系统事件类型

|  |  |
| --- | --- |
| **目的** | **相关系统调用** |
| 信息流 | *克隆（进程）、fork、msgsnd、msgrcv、写、发送、读、recv、exec* |
| 创建 | *打开, 创建, dup, 链接, 套接字, socketpair* |
| 预备 | *lseek, 连接, 听, 接受, 绑定, 链接* |
| 终止 | *关闭, 退出, 退出组, 取消链接, 杀死* |

称为出处图，它是一个有标签的、有类型的和有向的图。在这个图中，节点代表系统实体，具有不同的类型，例如文件和进程，而边代表这些节点之间的信息流和因果依赖关系，并考虑了方向。并非所有系统调用都表示信息流或因果依赖关系，因此不需要收集。例如，表 II 显示了 BSD 操作系统所需的一些最重要的与安全相关的系统调用。在此表中，我们根据用途显示了不同类别的系统调用。一些系统调用负责对象之间的实际信息流。例如，当通过克隆系统调用创建新进程时，它会继承其父进程的文件描述符。因此，存在从父进程到子进程的信息流。 系统调用的一个子集（表 II 的第三行）负责初始化和设置数据结构，而不是直接处理信息流。例如，socketpair 系统调用创建两个套接字。准备系统调用初始化数据结构，并在某些情况下提供后续数据的出处。例如，通过检查 lseek 系统调用并考虑文件偏移量，我们只跟踪文件的特定偏移量以防止不必要的依赖。终止系统调用处理对象的销毁。

表三

信息流事件。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **相关系统调用** | **从** | **至** | **资源** | **目的地** |
| *克隆（进程）、fork、vfork、rfork、msgsnd* | 过程 | 过程 | 事件调用者 | 参数 |
| *等等，msgrcv* | 过程 | 过程 | 参数 | 事件调用者 |
| *写, pwrite, writev, pwritev, 发送, sendto, sendmsg* | 过程 | 文件/套接字 | 事件调用者 | 参数 |
| *读取，recv，recvfrom，recvmsg，execl，execv，execle，execve，execlp，execvp* | 文件/套接字 | 过程 | 参数 | 事件调用者 |

要从系统调用序列构建出处图，我们需要提取启动系统调用的进程、系统调用的类型及其参数。表 III 显示了某些系统调用如何转换为出处图的节点和边的一些示例。第二列和第三列显示了边的方向和节点的类型（文件、套接字、进程），而最后两列显示了如何从系统调用事件中提取节点名称和属性。我们使用 arg(s) 来指示系统调用的参数，以引用调用者进程操作的对象。特别是，根据系统调用，参数类型可能是进程的 id、文件的名称或引用文件/套接字的描述符。如表所示，系统对象之间存在不同种类的信息流。这些包括：（i）从一个进程到另一个进程，由诸如 fork 和 clone 之类的事件启动， 这表明它们之间存在因果关系，（ii）从进程到由诸如写入和发送之类的事件启动的文件/套接字，以及（iii）从文件/套接字到由诸如读取和接收之类的事件启动的进程，这表示信息流。

图 2 显示了出处图的片段。左图显示了创建了两个“bash”进程的“SSH”守护进程，右图是Firefox浏览器从IP下载可执行文件（“Openme.exe”）的结果

SSH

叉子

叉子

重击

重击

131.193.32.29:80

接收

写

火狐

Openme.exe

图 2. 示例出处图片段

地址（“131.193.32.29:80”）。边的方向表明这两个“bash”进程是“SSH”的子进程，并且在因果上依赖于它。此外，方向指示源自套接字连接到下载文件的信息流。为了保持一致，我们使用某些节点形状来区分实体类型。对于这个图和本文的其余部分，椭圆形、菱形、矩形和星形分别代表进程、套接字、文件和用户。此外，内存对象和注册表项都由五边形表示。如图所示，出处图利用了系统的完整历史上下文，可以包括数亿/十亿个节点。

# 信息流分析

信息流分析中的高级思想是标记（污染）有趣的数据并在它通过系统传播时对其进行跟踪。信息流分析可以按不同的粒度进行，即细粒度和粗粒度。在细粒度的信息流分析[23-25]中，使用了一个影子内存，它为内存中的每个单词保留一个标签。当受污染的数据影响某些内存区域的值时，这些部分会被标签污染

位。这种类型的跟踪通常需要检测应用程序（例如，使用 Pin [26]）并且由于开销过高（通常约为 10×-20×）而不可行。

相反，粗粒度信息流分析主要关注系统调用数据（内核审计日志），其收集带来的运行时开销较低（约 2%）。特别是，可以在从内核审计日志构建的出处图上启动前向或后向图遍历，以调查网络攻击。例如，在检测到攻击后，系统分析人员使用检测点作为种子来启动反向跟踪策略，以确定该攻击的根本原因。同样，系统分析师可以启动前向跟踪方法来找出攻击的影响和受影响的实体。

与细粒度分析相比，粗粒度信息流分析的开销较低，但代价是损失了精度，有时，系统调用的粗粒度可能会限制对信息流的推理。例如，当一个进程读取多个加密文件并将解密后的内容以随机顺序写入多个文件时，知道哪个解密文件对应哪个加密文件是一个没有细粒度和内存无法克服的挑战——级别信息流跟踪。

# 透明计算数据集

本论文中介绍的工作是根据透明计算 (TC) 程序发布的对抗性参与数据集进行评估的。 TC 计划 [27] 是 DARPA 的一项工作，旨在通过使用最少的开销组件来监控系统操作期间的交互，从而为计算系统增加更多透明度。该计划涉及多个红队、蓝队对抗性活动，其中一些是公开的 [28]。

在这些活动中，红队负责在由 Linux、FreeBSD 和 Windows 等不同平台组成的网络上模拟多次网络攻击。

在交战之前，每台机器都安装了一些易受攻击的软件，这些软件后来被利用。在参与期间，内核审计日志被收集、处理并转换为通用数据表示，以便于分析1。为了进一步混合正常日志和攻击日志，红队还在攻击的同时对目标主机执行良性活动。日常系统活动包括但不限于网页浏览、检查电子邮件、软件升级、使用 PowerShell（在 Windows 中）的管理任务、运行需要管理权限的程序等。总的来说，这些活动涉及数月的审计日志收集活动，而攻击仅占审计数据量的不到 0.001%。

1虽然在某些平台上可能会提供一些额外的细粒度数据（例如包含单元抽象数据的 Linux 数据集 [25]），但我们在评估中并未考虑它们。

**第3章**

SLEUTH：**根据 COTS 审计数据重建实时攻击场景**

*本章包括 [1] 中发表的材料的摘录和数字。*

本章介绍 Sleuth，一个使用来自不同操作系统的内核审计日志并将它们处理成内存中平台中立的图形表示的系统。图形表示和内存存储都为在内核审计日志上运行高性能计算提供了基础。该图表示是本文下一章（Holmes 和 Poirot）中介绍的系统的基础。除了提出有效的内存图表示之外，Sleuth 还可以实时提醒分析师正在进行的活动，并在攻击后的几秒或几分钟内为他们提供活动的紧凑、可视化摘要。这将能够在受害企业造成巨大损失之前及时做出反应。

# 介绍

使用内核审计日志进行实时分析（例如攻击检测）比纯粹的取证分析带来了以下额外挑战：

* + 1. *事件存储和分析：*我们如何有效地存储来自事件流的数百万条记录，并让算法在几秒钟内筛选出这些数据？

22

* + 1. *为分析确定实体的优先级：*我们如何帮助被海量数据淹没的分析师确定最可能的攻击场景的优先级并快速“放大”？
    2. *场景重构：*我们如何简洁地总结攻击场景，从攻击者的切入点开始，识别整个活动对系统的影响？
    3. *处理常见的使用场景：*如何处理可能类似于攻击期间常见的活动的正常、良性活动，例如软件下载？
    4. 快速的交互式推理：我们如何为分析师提供通过数据进行有效推理的能力，例如，提供替代假设？

图 3 概述了我们的方法。 Sleuth 假设攻击最初来自企业外部。例如，攻击者可以通过外部提供的恶意输入劫持 Web 浏览器、插入受感染的 USB 记忆棒或向企业内运行的网络服务器提供零日漏洞利用来启动攻击。我们假设在 Sleuth 开始监控系统之前，攻击者没有在主机上植入持久性恶意软件。我们还假设操作系统内核和审计系统是值得信赖的。

Sleuth 的第一个贡献是解决高效事件存储和分析的挑战，它是开发紧凑的主内存依赖图表示（第 3.2 节）。主存表示的图算法可以比磁盘表示快几个数量级，这是实现实时分析能力的一个重要因素。在我们的实验中，我们能够处理来自 FreeBSD 的 79 小时的审计数据

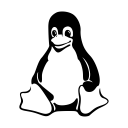
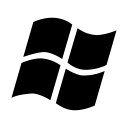
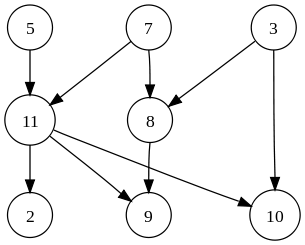
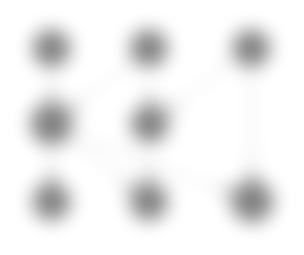
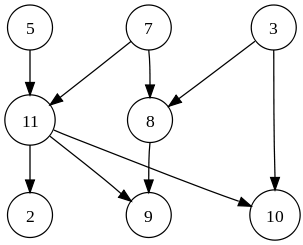
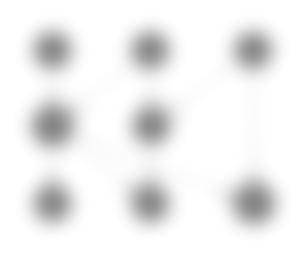
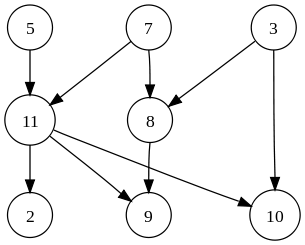
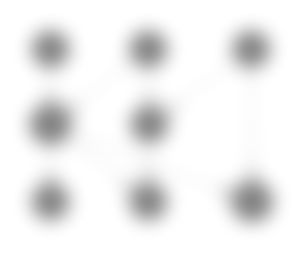


图 3. Sleuth 系统概述

系统在 14 秒内完成，主存使用量为 84MB。这种性能表示分析速率比生成数据的速率快 20K 倍。

Sleuth 的第二个主要贡献是开发了一种基于标签的方法，用于识别最有可能参与攻击的主体、对象和事件。标签使我们能够优先考虑和集中我们的分析，从而解决上面提到的第二个挑战。标签编码对数据（即对象）以及过程（主题）的可信度和敏感性的评估。此评估基于从审计日志中得出的数据来源。从这个意义上说，来自审计数据的标签类似于粗粒度信息流标签。我们的分析自然也可以支持更细粒度的标签，例如细粒度的污点标签 [29, 30]，如果它们可用的话。标签在第 3.3 节中进行了更详细的描述，以及它们在攻击检测中的应用。

Sleuth 的第三个贡献是开发了利用标签进行根本原因识别和影响分析的新算法（第 3.5 节）。从图 3 所示的攻击检测组件产生的警报开始，我们的反向分析算法如下

图中的依赖关系来识别攻击的来源。从源头开始，我们使用前向搜索对对手的行为进行全面的影响分析。我们提出了几个修剪这些搜索的标准，以产生一个紧凑的图。我们还提出了一些进一步简化该图的转换，并生成了一个以简洁和语义上有意义的方式直观地捕捉攻击的图，例如图 5 中的图。实验表明，我们基于标签的方法非常有效：例如，Sleuth 可以分析 3850 万个事件并生成仅包含 130 个事件的攻击场景图，这意味着事件量减少了五个数量级。

Sleuth 的第四个贡献，旨在解决上述最后两个挑战，是用于标签初始化和传播的可定制策略框架（第 3.4 节）。我们的框架带有合理的默认值，但它们可以被覆盖以适应特定于操作系统或应用程序的行为。这可以调整我们的检测和分析技术，以避免在良性应用程序表现出类似于攻击的行为的情况下出现误报。 （有关详细信息，请参阅第 3.6.6 节。）策略还使分析师能够通过重新分类被认为是可信或敏感的内容并重新运行分析来测试攻击的“替代假设”。如果分析师怀疑某些行为是攻击的结果，他们还可以使用策略来捕获这些行为，并重新运行分析以发现其原因和影响。由于我们处理和分析审计数据的速度比生成审计数据的速度快数万倍，因此高效、并行、 可以对替代假设进行实时测试。

Sleuth 的最终贡献是一个实验评估（第 3.6 节），主要基于 TC 数据集（在第 2.6 节中描述）。在本次评估中，类似于现代 APT 的攻击活动在两周内在 Windows、FreeBSD 和 Linux 主机上进行。在这次评估中，Sleuth 能够：

* 在几秒钟内处理包含在参与期间产生的数千万个事件的审计日志；
* 成功检测并重建这些攻击的详细信息，包括它们的入口点、系统中的活动和渗出点；
* 过滤掉无关事件，实现非常高的数据减少率（高达 10 万次），从而为这些攻击提供清晰的语义表示，几乎不包含来自系统中其他活动的噪音；和
* 实现低假阳性和假阴性率。

我们的评估并不是为了表明我们发现了最老练的对手；相反，我们的观点是，考虑到几种未知的可能性，我们系统的优先结果可以实时在现场，无需任何人工协助。因此，它确实填补了当今存在的一个空白，取证分析似乎主要是手动启动的。

# 主内存依赖图

为了支持快速检测和实时分析，我们将依赖关系存储在图数据结构中。存储此图的一个可能选项是图数据库。然而，流行数据库如 Neo4J [32] 或 Titan [33] 的性能 [31] 对于许多图来说是有限的

除非主内存足够大以容纳大部分数据。此外，一般图形数据库的内存使用对于我们的问题来说太高了。即使是 STINGER [34] 和 NetworkX [35]，两个针对主内存性能优化的图形数据库，每个图形边 [31] 分别使用大约 250 字节和 3KB。企业网络上报告的审计事件数量每天很容易达到数十亿到数百亿，这将需要数 TB 范围内的主内存。相比之下，我们提出了一种空间效率更高的依赖图设计，每条边仅使用大约 10 个字节。在一项实验中，我们能够在 329MB 的主内存中存储 38M 事件。

依赖图是每个主机的数据结构。它可以引用其他主机上的实体，但针对主机内引用的常见情况进行了优化。该图表示两种类型的实体：表示进程的主体和表示文件、管道和网络连接等实体的对象。主题属性包括进程 ID (pid)、命令行、所有者以及代码和数据的标签。对象属性包括名称、类型（文件、管道、套接字等）、所有者和标签。

使用主题和对象之间或两个主题之间的标记边缘捕获审计日志中报告的事件。为简洁起见，我们对事件使用 UNIX 名称，例如 read、connect 和 execve。为减少 Sleuth 的存储需求而开发的技术的详细解释后来发表在 [36] 中。

# 标签和攻击检测

我们使用标签来总结我们对客体和主体的可信度和敏感性的评估。这种评估可以基于三个主要因素：

* *出处：*依赖图中对象或主题的直接前身的标签，
* *先验系统知识：*我们对重要应用程序行为的了解，例如远程访问服务器和软件安装程序，以及重要文件（例​​如 /etc/passwd 和 /dev/audio）的行为，以及
* *行为：*观察到的受试者的行为，以及他们如何与他们的预期行为进行比较。

我们开发了一个策略框架，如第 3.4 节所述，用于基于这些因素初始化和传播标签。在没有特定策略的情况下，使用将标签从输入传播到输出的默认策略。默认策略将输入的可信度标签中的最低值和机密性标签中的最高值分配给输出。该策略是保守的：它可能会在过度污染方面犯错，但不会导致攻击未被发现，或导致前向（或后向）分析遗漏对象、主题或事件。

标签在 Sleuth 中起着核心作用。它们为攻击检测提供了重要的上下文。每个审计事件都在这些标签的上下文中进行解释，以确定其促成攻击的可能性。此外，标签有助于我们向前和向后分析的速度。最后，标签通过消除大量满足依赖的技术定义但对我们对攻击的理解没有有意义的贡献的审计数据，在场景重建中发挥着核心作用。

# 3.3.1 标签设计

*我们定义了以下可信度标签（t-tags）：*

* 良性真实标签被分配给从被信任为良性的来源接收的数据/代码，并且可以验证其真实性。
* *良性标签反映的信任程度低于良性真实：*虽然数据/代码仍被认为是良性的，但尚未执行足够的身份验证来验证来源。
* 未知标签被赋予来自我们没有关于可信度信息的来源的数据/代码。此类数据有时可能是恶意的。

策略定义了哪些来源是良性的以及哪些形式的身份验证是足够的。在最简单的情况下，这些策略采用白名单的形式，但我们也支持更复杂的策略。如果没有策略适用于源，则其 t-tag 设置为未知。

我们定义了以下机密标签（c-tags），以推断信息窃取攻击：

* *机密：*高度敏感的信息，例如登录凭据和私钥。
* *敏感：*其披露可能对安全产生重大影响的数据，例如，揭示系统中的漏洞，但不为攻击者提供访问系统的直接途径。
* *私人：*其披露涉及隐私问题但不一定构成安全威胁的数据。
* *公开的：*可以广泛使用的数据，例如在公共网站上。

我们设计的一个重要方面是代码和数据的 t-tag 之间的分离。具体来说，一个主题（即一个进程）被赋予两个 t-tag：一个捕获其代码可信度（code t-tag），另一个捕获其数据可信度（data t-tag）。这种分离显着改善了攻击检测。更重要的是，它可以通过将重点放在更少的可疑事件上来显着加快取证分析的速度，同时大大减少重建场景的规模。请注意，机密性标签仅与数据（而不是代码）相关联。

使用标签初始化策略为预先存在的对象和主题分配初始标签。代表外部实体的对象，例如远程网络连接，也需要分配初始标签。其余的对象和主体是在系统执行期间创建的，它们的标签是使用标签传播策略确定的。最后，使用称为检测策略的基于行为的策略来检测攻击。

如前所述，如果未提供特定策略，则源将被标记为未知可信度。类似地，在没有特定传播策略的情况下，使用默认的保守传播策略。

# 基于标签的攻击检测

Sleuth 的一个重要限制是我们仅限于审计数据中可用的信息。这表明使用反映在审计数据中的出处作为检测的可能基础。由于标签是出处的函数，我们将它们用于攻击检测。请注意，在我们的威胁模型中，审计数据是值得信赖的，因此标签为检测提供了良好的基础。

Sleuth 的第二个限制是检测方法不应该需要详细的特定于应用程序的知识。相比之下，大多数现有的入侵检测和沙箱-

技术在特定应用程序的上下文中解释每个安全敏感操作，以确定它是否可能是恶意的。这需要有关应用程序的专业知识，或在动态环境中进行现场培训，其中应用程序可能会经常更新。

我们没有专注于往往是可变的应用程序行为，而是将我们的检测技术专注于大多数攻击者的高级目标，例如后门插入和数据泄露。具体来说，我们结合了对攻击者动机和手段的推理。如果审计数据中的一个事件可以帮助攻击者实现他/她的关键高级目标，这将为在攻击中使用该事件提供动机和理由。但这还不够：攻击者还需要引发此事件的手段，或者更广泛地说，影响它。请注意，我们的标签旨在捕获意味着：如果一段数据或代码带有未知的 t-tag，那么它是从不受信任的来源派生（并因此受其影响）的。

至于攻击者的高级目标，一些报告和白皮书已经确定以下步骤在最高级的攻击活动中是典型的 [5, 13, 37]：

* + - 1. 在受害者系统上部署和运行攻击者的代码。
      2. 替换或修改重要文件，例如 /etc/passwd 或 ssh 密钥。
      3. 泄露敏感数据。

具有短暂影响的攻击可能能够避免前两个步骤，但最复杂的攻击，例如 APT 活动中使用的攻击，需要在受害系统上建立更永久的足迹。在这些情况下，似乎没有办法避免

或前两个步骤。即使在攻击者的目标可以在没有建立永久基地的情况下实现的情况下，第三步通常也代表了基本的攻击者目标。

基于上述推理，我们定义了以下结合攻击者目标和手段的攻击检测策略：

* *不受信任的代码执行：*当具有较高代码 t-tag 的主体执行（或加载）具有较低 t-tag1的对象时，此策略会触发警报。
* 具有较低代码 t-tag 的主体修改：当具有较低代码 t-tag 的主体修改具有较高 t-tag 的对象时，此策略会引发警报。修改可能与文件内容或其他属性有关，例如名称、权限等。
* *机密数据泄露：*当不受信任的主体泄露敏感数据时会发出警报。具体来说，此策略由具有敏感 c-tag 和未知代码 t-tag 的主题在网络写入时触发。

1可以为 bash 等解释器定义自定义策略，以便将读取视为与加载相同。

* *为执行准备不受信任的数据：*如果该操作使对象可执行，则该策略由具有未知代码 t-tag 的主体的操作触发。此类操作包括 chmod 和 mprotect1,2。

重要的是要注意，“手段”不会因为数据或代码通过多个中介而被稀释。例如，不受信任的代码策略不需要直接从未知网站加载数据；相反，可以下载、提取、解压缩和可能编译数据，然后加载。无论中间步骤多少，在加载或执行结果文件时都会触发此策略。这是我们的攻击检测有效的最重要原因之一。

今天的漏洞利用通常不会在第一步中涉及不受信任的代码，因此不会被不受信任的代码执行策略检测到。然而，攻击者的最终目标是执行他/她的代码，或者通过下载和执行文件，或者通过向包含不可信数据的内存页面添加执行权限。在任何一种情况下，上述策略之一都可以检测到攻击。随后的反向分析可以帮助识别漏洞利用的第一步。

额外的检测器输入可以很容易地集成到 Sleuth 中。例如，如果外部检测器将对象标记为嫌疑人，则可以通过设置

1对当今操作系统的二进制代码注入攻击最终涉及调用更改可写内存页面的权限，使其变为可执行。只要此类内存权限更改操作包含在审计数据中，此策略就可以发现它们。

2我们的实现可以识别与库加载操作一起发生的 mprotect 操作。这些 mprotect 不会触发此策略。

以未知为准。因此，上面提到的其余检测策略都可以从外部检测器提供的信息中受益。此外，在可疑节点处设置未知 t-tag 保留了导致警报的图顶点之间的依赖结构，这是我们在取证分析中利用的事实。

我们的许多策略是由不受信任的代码执行触发的这一事实不应被解释为它们在静态环境中工作，在该环境中系统中不允许有新代码。事实上，我们预计软件更新和升级会不断发生，但在企业环境中，我们不希望最终用户从随机站点下载未知代码。因此，我们随后描述了如何支持标准化的软件更新机制，例如当代操作系统上使用的那些。

# 政策框架

我们为标签分配、传播和攻击检测开发了一个灵活的策略框架。我们使用简单的基于规则的符号来表达策略，例如，

*exec(s, o): o.ttag < 良性 → alert("UntrustedExec")*

当主体 s 执行具有小于良性的 t-tag 的（文件）对象 o 时，触发此规则。它的作用是引发一个名为 UntrustedExec 的警报。如该示例所示，规则通常与事件相关联，并且包括关于事件中涉及的对象和/或主体的属性的条件。感兴趣的属性包括：

* + - *name：*正则表达式可用于匹配对象名称和主题命令行。我们对正则表达式使用 Perl 语法。
    - *标签：*条件可以放在对象和/或主题的 t-tags 和 c-tags 上。对于主题，代码和数据 t-tags 可以独立访问。
    - *所有权和权限：*条件可以放在对象和主体的所有权上，或者与对象或事件相关的权限上。

政策的效果取决于其类型。检测策略的作用是发出警报。对于标签初始化和传播策略，效果是修改与事件中涉及的对象或主题相关联的标签。虽然我们在本章中使用基于规则的符号来指定策略，但在我们的实现中，每个规则都被编码为（C++）函数。

为了更好地控制检查不同类型策略的顺序，我们将策略与触发点而不是事件相关联。此外，触发点提供了一定程度的间接性，允许在具有相似目的的不同事件之间共享策略。表 IV 显示了当前在我们的政策框架中定义的触发点。第一列标识事件，第二列指定信息流的方向，最后两列定义与这些事件相关的触发点。

请注意，我们使用名为 define 的特殊事件来表示定义新对象的审计记录。假设第一次遇到新对象时发生此伪事件，例如，建立新的网络连接、第一次提及预先存在的文件、创建新文件等。其余事件表中的内容是不言自明的。

当事件发生时，所有与其警报触发相关的检测策略都会被执行。除非特别配置，否则只有在目标主体或客体的标签即将发生变化时才会检查检测策略。 （这里的“目标”是指数据流的目的地

一个操作。）在此之后，与事件的标记触发器关联的策略将按照指定的顺序进行尝试。一旦找到匹配规则，此规则指定的标签就会分配给事件的目标，并且不会评估剩余的标签策略。

我们当前的检测策略在上一节中进行了非正式的描述。因此，我们在本节中关注我们当前的标签初始化和传播策略。

# 标签初始化策略

这些策略在 init 触发器中调用，用于初始化新对象的标签，或者在审计数据中首次提到它们时为预先存在的对象初始化标签。回想一下，当主体创建一个新对象时，该对象默认继承主体的标签；但是，这可以使用标签初始化策略覆盖。

我们当前的标签初始化策略如下。请注意使用正则表达式来方便地定义对象组的初始标记。

*init*(*o*): *match*(*o.name,* "^IP:(10\.0|127)") *→ o.ttag* = BENIGN AUTH*, o.ctag* = PRIVATE

*init（o）： match（o.name， \“^IP：\”） → o.ttag = UNKNOWN， o.ctag = PRIVATE*

表四

具有政策触发点的边缘。在方向栏中，S 表示主体，O 表示对象。接下来的两列表示检测策略和标签设置策略的触发点。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | 方向 | 警报触发 | 标记触发器 |
| 定义 |  |  | *在里面* |
| 读 | O*→*S | *读* | *道具Rd* |
| 加载，执行 | O*→*S | *执行* | *道具EX* |
| 写 | S*→*O | *写* | *道具* |
| rm，重命名 | S*→*O | *写* |  |
| chmod, chown | S*→*O | *写，修改* |  |
| 设置 | S*→*S |  | *道具苏* |

init(o): o.type == 文件 → o.ttag = BENIGN AUTH, o.ctag = PUBLIC

第一条规则为 Intranet 连接指定标签，由远程主机的地址前缀 10.0 和 127 标识。它在 Sleuth 未部署在远程主机1上的情况下很有用。第二条规则规定所有其他主机都是不受信任的。第三条规则为所有预先存在的文件分配了相同的标签。我们的实现使用了两个额外的策略来指定 c-tags。

# 标签传播策略

这些策略可用于覆盖默认标签传播语义。可以为不同的相关事件类型组定义不同的标签传播策略，如表 IV 中的“标签触发器”列所示。

标记传播策略可用于防止由应用程序每次调用时重复读取和写入的文件（例如 .bash 历史记录）导致的“过度污染”。以下策略跳过此特定文件的污点传播：

*propRd(s, o): 匹配(o.name, "\.bash\_history$") → 跳过*2

这是一个将 bash（解释器）读取的文件视为负载的策略，因此会更新代码 t-tag。

1如果Sleuth部署在远程主机上，将没有与建立网络连接相关的define事件，因此不会触发该策略。相反，我们已经为远程网络端点计算了一个标签，它现在将传播到从连接中读取的任何本地主题。

2这里，“跳过”意味着什么都不做，即保持标签不变。

*propRd*(*s, o*): *match*(*s.cmdline,* "^/bin/bash$") *→ s.code ttag* = *s.data ttag* = *o.ttag, s.ctag* = *o.ctag*

尽管 sshd 等受信任的服务器与不受信任的站点进行交互，但可以预期它们会保护自己，并只允许授权用户访问系统。此类服务器不应降低其数据可信度。类似的评论适用于软件更新程序和安装程序等程序，它们从不受信任的站点下载代码，但在安装前验证受信任的软件提供商的签名。

*propRd(o, s): match(s.cmdline, "^/sbin/sshd$") → 跳过*

此外，当登录阶段完成时，通常通过执行 setuid 来识别

操作时，应为流程分配适当的标签。

*道具：*

*match*(*s.cmdline,* "^/usr/sbin/sshd$") *→ s.code ttag* = *s.data ttag* = BENIGN*, s.ctag* = PRIVATE

# 基于标签的双向分析

* + 1. **后向分析**

反向分析的目标是识别攻击活动的切入点。入口点是图中入度为零的节点，并被标记为不可信。通常它们代表网络连接，但它们也可以是其他类型，例如，插入受害主机的 USB 记忆棒上的文件。

反向分析的起点是检测策略产生的告警。特别是，每个警报都与一个或多个实体相关，这些实体被标记为

图中的可疑节点。向后搜索涉及图的向后遍历，以识别将可疑节点连接到入口节点的路径。我们注意到，在这样的遍历和以下讨论中，依赖边的方向是相反的。向后搜索带来了几个重大挑战：

* + - * *性能：*依赖图可以轻松包含数亿条边。警报数量很容易达到数千。在如此大的图上运行反向搜索在计算上是昂贵的。
      * *多路径：*通常许多入口点可以从可疑节点向后到达。然而，在 APT 式攻击中，通常只有一个真正的切入点。因此，天真的向后搜索会导致大量误报。

我们方法背后的关键见解是标签可用于解决这两个挑战。事实上，标签计算和传播已经是一种隐式路径计算，可以重复使用。此外，节点上的未知标记值提供了关于该节点成为攻击潜在部分的可能性的重要线索。特别是，如果某个节点 A 存在未知标签，这意味着至少存在从不受信任的入口节点到节点 A 的路径，因此节点 A 比其他具有良性标签的邻居更有可能成为攻击的一部分。利用标签进行后向搜索，通过消除许多不相关的节点，大大减少了搜索空间，并将 Sleuth 与其他场景重建方法（如 [8, 25]）区分开来。

基于这种见解，我们将反向分析公式化为最短路径问题的一个实例，其中标签用于定义边缘成本。实际上，标签能够“引导”搜索

相关路径，并远离不太可能的路径。这个因素使搜索能够在不必遍历整个图的情况下完成，从而解决了性能挑战。此外，我们的最短路径公式通过选择最接近可疑节点的入口点（通过路径成本来衡量）来解决多路径挑战。

对于最短路径，我们使用 Dijkstra 算法，因为它以成本递增的顺序发现路径。特别是，该算法的每一步都将一个节点添加到最短路径树中，最短路径树由迄今为止计算的最短路径组成。这使搜索在入口点节点添加到此树时立即停止。

成本函数设计。我们的设计将低成本分配给表示对具有未知标签的节点的依赖关系的边，而将较高成本分配给其他边。具体来说，费用如下：

* + - * 将具有未知代码或数据 t-tag 的节点的依赖关系引入具有良性代码或数据 t-tag 的节点的边的成本为 0。
      * 从具有良性代码和数据 t-tags 的节点引入依赖关系的边被分配了高成本。
      * 在已经具有未知标签的节点之间引入依赖关系的边被分配成本 1。

这种设计背后的直觉如下。与未知主题/对象直接相关的良性主题或对象表示图的恶意部分和良性部分之间的边界。因此，它们必须包含在搜索中，因此这些边的成本为 0。

良性实体之间的信息流不是攻击的一部分，因此我们将它们的成本设置为非常高，以便将它们排除在搜索之外。不受信任节点之间的信息流可能是攻击的一部分，因此我们将其成本设置为较低的值。它们将包含在搜索结果中，除非有包含较少边的替代路径可用。

# 前向分析

前向分析的目的是通过从一个切入点开始并发现依赖于该切入点的所有可能影响来评估活动的影响。与后向分析类似，主要挑战是图的大小。一种简单的方法将识别和标记从通过反向分析识别的入口点可到达的所有主题和对象。不幸的是，这种方法会导致影响图太大而对分析师无用。例如，在我们的实验中，一个简单的分析产生了具有数百万条边的影响图，而我们的改进算法将这个数字减少了 100 倍到 500 倍。

减小大小的一种自然方法是使用距离阈值 d*千*来排除离可疑节点“太远”的节点。阈值 d*千*可以由分析师交互式调整。我们使用与后向分析相同的成本指标，但经过修改以考虑机密性1。特别是，具有高机密性标签（例如，秘密）的节点与具有低代码完整性标签（例如，未知进程）或低数据完整性标签（例如，未知套接字）的节点之间的边被分配为 0，而边到节点带有良性标签的成本很高。

1回想一下，一些警报与机密数据的泄露有关，因此我们需要决定哪些表示机密信息流的边应该包含在场景中。

# 重建与展示

我们对前向分析的输出应用以下简化，以便提供更简洁的攻击视图：

* + - * 修剪无趣的节点。前向分析的结果可能包括许多与攻击无关的依赖关系，例如，主体写入缓存和日志文件，或写入临时文件然后将其删除。这些节点可能会出现在前向分析的结果中，但没有可疑节点依赖于它们，因此可以修剪它们。
      * 合并具有相同名称的实体。这种简化合并了具有相同名称的主题，而忽略了它们的进程 ID 和命令行参数。
      * 重复事件过滤。这种简化将同一实体之间发生多次（例如，多次写入、多次读取）的那些事件合并为一个。如果存在交错事件，则我们显示两个事件，表示两个实体之间事件的第一次和最后一次发生。

# 实验评价

* + 1. **执行**

Sleuth 的大部分组件，包括图模型、策略引擎、攻击检测和部分取证分析都是用 C++ 实现的，大约包含 9.5KLoC。其余的组件，包括用于重构和展示的组件，都是用 Python 实现的，由 1.6KLoC 组成。

# 数据集

表 V 总结了我们评估中使用的数据集。表的前八行对应于红队作为 TC 计划的一部分执行的攻击活动。该集合跨越 358 小时，包含大约 7300 万个事件。最后一行对应于在我们研究实验室的四台 Linux 服务器上在 3 到 5 天内收集的良性数据。攻击数据集由三个研究团队在 Windows（W-1 和 W-2）、Linux（L-1 到 L-3）和 FreeBSD（F-1 到 F-3）上收集，这三个研究团队也是 TC 计划的一部分.

表 V 中的“持续时间”列是指从主机发出审计数据的时间长度。请注意，此期间包括主机上的良性活动和与攻击相关的活动。接下来的几列将审计日志事件分解为不同类型的操作。 W-1 和 W-2 数据集中不包括文件打开和关闭操作。请注意，“读”和“写”列不仅包括文件读/写，还包括

表五

每个活动的数据集，包括持续时间、不同系统调用的分布和事件总数。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 期间  （嗯-毫米-ss） | 打开 | 连接+  接受 | 读 | 写 | 克隆+  执行 | 关闭+  出口 | 地图/  加载库 | 其他 | 全部的  活动 | 设想  图形 |
| W-1 | 06:22:42 | N/A | 22.14% | 44.70% | 5.12% | 3.73% | 3.88% | 17.40% | 3.02% | 100K | 图 28 |
| W-2 | 19:43:46 | N/A | 17.40% | 47.63% | 8.03% | 3.28% | 3.26% | 15.22% | 5.17% | 401K | 图 4 |
| L-1 | 07:59:26 | 37% | 0.11% | 18.01% | 1.15% | 0.92% | 38.76% | 3.97% | 0.07% | 2.68M | 图 25 |
| L-2 | 79:06:39 | 39.58% | 0.08% | 12.19% | 2% | 0.83% | 41.28% | 3.79% | 0.25% | 38.5M | - |
| L-3 | 79:05:13 | 38.88% | 0.04% | 11.81% | 2.35% | 0.95% | 40.98% | 4.14% | 0.84% | 19.3M | 图 29 |
| F-1 | 08:17:30 | 9.46% | 0.40% | 24.65% | 40.86% | 2.10% | 12.55% | 9.08% | 0.89% | 701K | 图 26 |
| F-2 | 78:56:48 | 11.78% | 0.42% | 16.60% | 44.52% | 2.10% | 15.04% | 8.54% | 1.01% | 5.86M | 图 27 |
| F-3 | 79:04:54 | 11.31% | 0.40% | 19.46% | 45.71% | 1.64% | 14.30% | 6.16% | 1.03% | 5.68M | 图 5 |
| 良性 | 329:11:40 | 11.68% | 0.71% | 26.22% | 30.03% | 0.63% | 15.42% | 14.32% | 0.99% | 32.83M | N/A |

还可以在 Linux 上进行网络读写。但是，在 Windows 上，仅报告文件读取和写入。在 Windows 上报告了加载库的操作，但没有报告内存映射操作。在 Linux 和 FreeBSD 上，没有加载操作，但大多数 mmap 调用都与加载有关。因此，mmap 计数是这两个操作系统上负载数量的松散近似值。 “Others”列包括所有剩余的审计操作，包括rename、link、rm、unlink、chmod、setuid等。表中的最后一列标识了 Sleuth 为每个活动构建的场景图。由于篇幅限制，我们省略了 L-2 战役的场景图。

# 参与设置

我们评估中的攻击场景设置如下。其中五个活动（即 W-2、L-2、L3、F-2 和 F3）并行运行 4 天，而其余三个活动（W-1、L-1 和 F-1）是并行运行 2 天。在每次战役中，红队都会对目标主机进行一系列攻击。这些活动旨在实现不同的对抗目标，包括删除和执行可执行文件、收集有关目标主机的情报、后门注入、特权升级和数据泄露。

作为对抗性的参与，我们事先不知道红队计划的攻击。我们只被告知上一段中描述的广泛的攻击者目标。值得注意的是，红队在对目标主机进行攻击的同时，也在对主机进行良性后台活动。其中包括浏览和下载文件、阅读和编写电子邮件、文档处理等活动。平均而言，超过 99.9% 的事件对应于

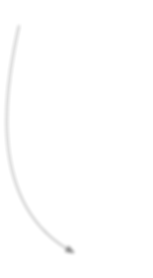
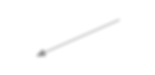
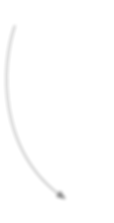
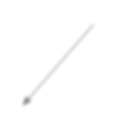
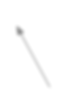
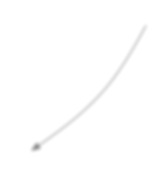
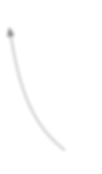
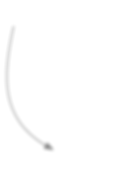
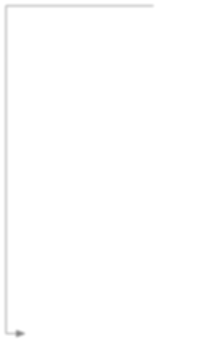
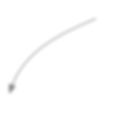
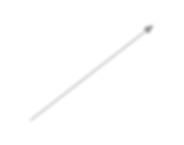
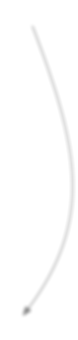
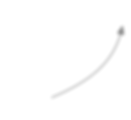
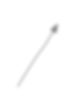
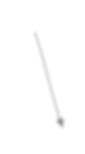
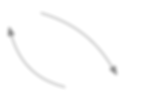
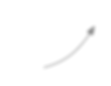
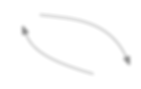
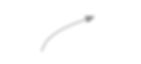
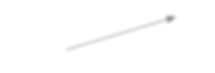
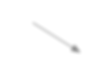
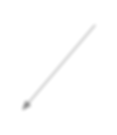
良性活动。因此，Sleuth 必须从包括良性和恶意活动在内的一组事件中自动检测和重建攻击。

我们将我们的结果与红队发布的地面实况数据进行比较。在发布地面实况数据之前，我们必须向红队提供我们的调查结果报告。我们在本章中报告的调查结果与我们提交给红队的调查结果相符。我们的检测和重建结果的总结在表 VII 中以表格形式提供。下面，我们首先介绍所选数据集的重建场景，然后再讨论这些总结结果。

# 选定的重建结果

在 Sleuth 成功重建的 8 个攻击场景中，我们将在本节讨论活动 W-2 (Windows) 和 F-3 (FreeBSD)，而将其余部分推迟到附录 A。为了更容易理解场景图，我们提供解释攻击如何展开的叙述。这种叙述需要手动解释图形，但图形生成本身是自动化的。在这些图中，边标签包括事件名称和指示该事件执行的全局顺序的序列号。椭圆形、菱形和矩形分别代表进程、套接字和文件。

战役 W-2。图 4 显示了 Sleuth 从 Windows 审计数据中重建的图表。虽然实际的攻击活动持续了半个小时，但主机运行了 20 个小时的良性后台活动。这些后台活动对应于相应审计日志中超过 99.8% 的事件。



C:\\dropper 4. 执行滴管

*2.写*

6.发送

*3. 叉子*

*5. 收到*

火狐

129.55.12.167:443

*1.接收*

129.55.12.167:8000

*7. 写*

*8. 叉子*

*9.执行*

*10. 发送*

滴管

*29. 发送*

C：\Users\User1\Downloads\firefox\dropper

*11. 写*

*30. 收到*

129.55.12.167:4430

*13. 叉子*

*33. rm*

*12. 阅读*

命令文件 *32. rm*

*31. rm*

*14. 叉子*

*22. 叉子*

我是谁

*16. 叉子*

git.exe

*18. 叉子*

129.55.12.51:9418

网络统计

*23. 叉子*

*15. 写*

*28. 收到*

*27. 发送*

主机名

*17. 写*

*20. 写*

git-receive-pack.exe

*19. 写*

*21. chmod*

129.55.12.51:80

*26. 阅读*

*24. 叉子*

*25. 发送*

C:\\Users\\User1\\Documents\\Thumbs\

humbit\test\thumbs.db

git-remote-http.exe

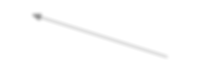
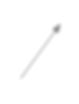
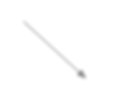
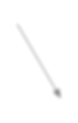
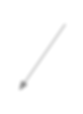
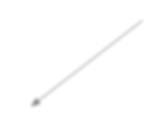
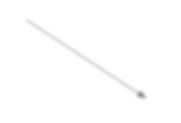
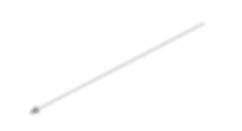
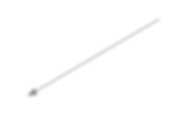
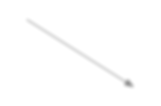
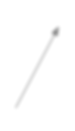
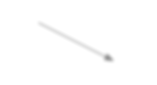
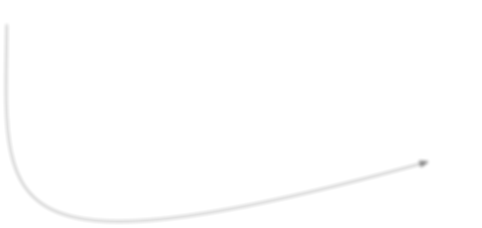
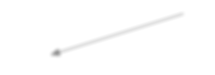
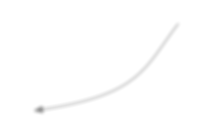
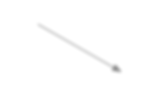
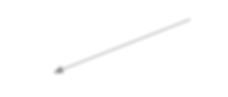
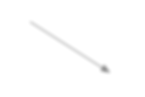
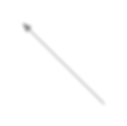
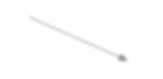
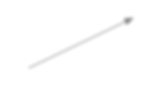
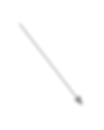
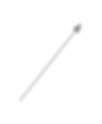
C:\\用户\\用户1\\下载\

irefox\burnout.bat

图 4. 从战役 W-2 重建的场景图。

*入口：*攻击的初始入口点是 Firefox，它在访问 Web 服务器 129.55.12.167 时被攻陷。

*后门插入：*一旦 Firefox 被入侵，就会下载并执行一个名为 dropper 的恶意程序。 Dropper 似乎提供了一个远程交互 shell，连接到攻击主机上的 443 和 4430 端口，并使用 cmd.exe 执行接收到的命令。



128.55.12.167

*1.接收*

*10.写*

scp

*9. 叉子*

须藤

日期

*13. 阅读* sshd

哎呀

bsdtar  *~~12. 叉子~~* 须藤

*8. 叉子*

*3. 叉子*

*2. 叉子*

*11. 叉子*

*4. 叉子*

*16.写*

*15.写*

*14.写*

重击

*5. 叉子*

ls

/var/dropbear\_latest/ /var/dropbear\_latest/

小熊/小熊 滴熊/滴熊

/var/dropbear\_latest/dropbear/dropbearscript18. 阅读

*17. 叉子*

vi

*6. 叉子*

*25. 叉子*

*23.执行*

*20. 阅读*

*7. 叉子*

我是谁

*19. 叉子*

小熊钥匙

*~~22. fork 须藤~~**~~21. 叉子~~* 嘘

须藤

主机名

*26. 叉子*

*24. 写*

/usr/local/etc/dropbear/ 28. 读取

dropbear\_rsa\_host\_key

小熊  *~~30. 叉子~~*

重击

*35. 叉子 33. 叉子*

*39. 叉子*

*37. 叉子*

*31. 叉子*

*27.执行*

*29.接收*

猫

哎呀

无名

ls

scp  *~~40. 叉子~~* 重击

128.55.12.167:2525

*38. 写*

*32. 写*

*36. 写 34. 写*

*41. 写*

*42. 叉子 45. 发送*

*~~43.执行~~*

归档器

*~~44. 阅读~~*

/usr/home/user/procstat

/usr/home/用户/存档

/var/dropbear\_latest/ dropbearFREEBSD.tar

图 5. 从 F-3 战役重建的情景图。

*情报收集：Dropper 然后多次调用 cmd.exe，用它来执行各种数据收集任务。 whoami、hostname 和 netstat 程序被用作这些数据收集应用程序的替身。收集的数据写入 C:*\Users\User1\- Documents\Thumbs\thumbit\test\thumbs.db。

*数据泄露：然后使用 git 将收集到的情报泄露到 129.55.12.51:*9418。清理：Dropper 下载一个名为 burnout.bat 的批处理文件。该文件包含清理攻击足迹的命令，这些命令由 cmd.exe 执行（参见边缘 11、12、31-33）。

F-3 战役。 （图 5）。在使用窃取的 ssh 凭据的攻击者的命令下，sshd 分叉了一个 bash 进程。请注意，尽管审计数据中没有关于被盗 ssh 凭证的直接证据，但由于此 shell 的后续事件 (scp)，我们得出结论，这是攻击者使用被盗 ssh 凭证的迹象。

接下来，攻击者调用 scp，它将文件下载到位置 /var/dropbear latest/-dropbearFREEBSD.tar，然后解压缩。接下来由 sh 读取和解释文件 dropbearscript。此操作创建进程 dropbearkey，该进程将写入 /usr/local/etc/-dropbear/dropbear ecdsa 主机密钥和 /usr/local/etc/dropbear/dropbear rsa 主机密钥。接下来，由 bash 创建的另一个 sudo 进程启动另一个 dropbear 进程，该进程读取这两个密钥以供将来使用（可能是为了帮助连接到远程主机）。

Dropbear 接下来启动一个 shell 进程，该进程执行一系列命令 ls、bash、uname、ps，所有这些命令都写入文件 /usr/home/user/procstats。

最后，dropbear 启动一个 bash 进程，该进程使用 scp 下载一个名为 /usr/home/-user/archiver 的文件，并执行该文件。生成的进程（称为归档程序）读取文件

/usr/home/user/procstats，其中包含之前输出的数据，并将此信息泄露到 128.55.12.167:2525。

概括。上面两张图是 Sleuth 根据审计数据自动构建的。他们展示了 Sleuth 如何使分析师能够从数小时的审计数据中获得紧凑而完整的攻击场景。 Sleuth 能够深入了解攻击活动，即使它隐藏在至少大三个数量级的良性数据中。

# 整体效果

为了评估 Sleuth 在捕获 APT 基本阶段方面的有效性，在表 VI 中，我们将 Sleuth 构建的攻击场景片段与显着 APT 活动的事后报告中记录的 APT 阶段相关联（例如，MANDIANT [13] 报告）。在 8 个攻击场景中的 7 个场景中，Sleuth 发现了 drop&load 活动。在所有场景中，Sleuth 都捕捉到了数据泄露的具体证据，这是 APT 活动的关键阶段。在 7 个场景中，攻击者用来收集目标主机信息的命令被 Sleuth 捕获。

APT 的另一个独特方面是向目标注入后门，并将其用于 C&C 和数据泄露。在这方面，Sleuth 重构的 8 个场景中有 6 个涉及后门注入。清理攻击足迹是 APT 活动的常见元素。在我们的实验中，在 8 个场景中的 5 个场景中，Sleuth 发现了攻击清理活动，例如，删除在攻击期间创建的丢弃的可执行文件和数据文件。

表六

Sleuth 是针对典型 APT 活动的结果。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 跌落和装载 | 情报收集 | 后门插入 | 权限提升 | 数据泄露 | 清理 |
| W-1 | ✓ | ✓ |  |  | ✓ | ✓ |
| W-2 | ✓ | ✓ | ✓ |  | ✓ | ✓ |
| L-1 | ✓ | ✓ | ✓ |  | ✓ | ✓ |
| L-2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| L-3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| F-1 |  |  | ✓ |  | ✓ |  |
| F-2 | ✓ | ✓ | ✓ |  | ✓ |  |
| F-3 | ✓ | ✓ |  |  | ✓ |  |

表七显示了分解攻击场景重构结果的另一种方法，计算了攻击中涉及的关键文件、网络连接和程序的数量。具体来说，我们统计了攻击入口实体的数量（包括入口点和与这些入口点通信的进程）、与攻击相关的程序执行、活动期间生成和使用的关键文件以及使用的出口点数量用于渗透（例如，网络套接字）。这些数据与我们获得结果后提供给我们的基本事实进行了比较。最后两列分别显示错误报告和遗漏的实体。

这两个遗漏的实体是由于我们没有花任何精力对敏感数据文件和设备文件进行编目。结果，这些实体在前向分析和简化步骤中被过滤掉了。一旦我们正确标记了这两个文件，它们就不再被过滤掉，我们能够识别所有关键实体。

除了表 VII 中显示的遗漏实体外，红队报告说我们遗漏了一些其他攻击和实体。其中一些在我们没有检查的数据集中。在

表七

攻击场景重构总结。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 入口  实体 | 程式  执行 | 关键文件 | 出口点 | 正确识别  实体 | 错误识别  实体 | 错过  实体 |
| W-1 | 2 | 8 | 7 | 3 | 20 | 0 | 0 |
| W-2 | 2 | 8 | 4 | 4 | 18 | 0 | 0 |
| L-1 | 2 | 10 | 7 | 2 | 20 | 0 | 1 |
| L-2 | 2 | 20 | 11 | 4 | 37 | 0 | 0 |
| L-3 | 1 | 6 | 6 | 5 | 18 | 0 | 0 |
| F-1 | 4 | 13 | 9 | 2 | 13 | 0 | 1 |
| F-2 | 2 | 10 | 7 | 3 | 22 | 0 | 0 |
| F-3 | 4 | 14 | 7 | 1 | 26 | 0 | 0 |
| **全部的** | **19** | **89** | **58** | **24** | **174** | **0** | **2** |

特别是，活动 W-2 运行了多次，我们只检查了其中一个实例的数据集。此外，在 Windows 上还有第三次攻击活动 W-3，但制作 Windows 数据集的团队在 W-3 期间遇到了困难，导致无法记录攻击活动，因此表 VII 的结果中省略了数据集。同样，负责生成 Linux 数据集的团队在 L-3 战役中遇到了一些问题，导致一些攻击活动没有被记录下来。考虑到这一点，表 VII 仅计算了名称出现在提供给我们的 L-3 数据集中的关键实体的子集。

根据红队提供的基本事实，我们错误地识别了 F-1 中不属于攻击的 21 个实体。随后的调查显示，审计系统在 F-1 战役结束时并未关闭，所有这些误报都对应于在任务结束后执行的测试/管理步骤，而此时审计系统不应该被关闭。跑步。

# 良性环境中的误报

为了研究 Sleuth 在良性环境中的表现，我们从四台 Ubuntu Linux 服务器收集了 3 到 5 天的审计数据。其中之一是邮件服务器，

表八

软件升级和更新的良性环境中的误报。在此期间未触发任何警报。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集 | 磁盘上的日志大小 | # 事件 | 持续时间 (hh:mm:ss) | 更新的包 | 写入的二进制文件 |
| 服务器 1 | 1.1G | 2.17M | 00:13:06 | 110 | 1.8K |
| 服务器 2 | 2.7G | 4.67M | 105:08:22 | 4 | 4.2K |
| 服务器 3 | 12G | 20.9M | 104:36:43 | 4 | 4.3K |
| 服务器 4 | 3.2G | 5.09M | 119:13:29 | 4 | 4.3K |

另一个是 Web 服务器，第三个是 NFS/SSH/SVN 服务器。在此期间，我们的重点是软件更新和升级，因为这些更新可以从网络下载代码，从而增加了不受信任的代码执行警报的可能性。在此期间执行了四次安全更新（包括内核更新）。此外，在第四台服务器上，我们在执行软件升级时收集了数据，导致 110 个软件包发生了变化。在此期间更新了数千个二进制文件和脚本文件，审计日志包含超过 3000 万个事件。所有这些信息汇总在表 VIII 中。

如前所述，应配置策略以允许使用企业认可的标准方法进行软件更新和升级。对于 Ubuntu Linux，我们有一个策略规则：当 dpkg 由 apt-commands 或无人值守升级执行时，即使从带有不受信任标签的文件中读取，进程也不会降级。这是因为 apt 和 unattended-upgrades 都会验证和验证下载包的哈希值，并且只有在这些验证之后，它们才会调用 dpkg 来提取内容并写入包含二进制文件和库的各种目录。由于这项政策，所有下载的 10K+ 文件都被标记为良性。因此，Sleuth 的执行没有产生任何警报。

# 运行时和内存使用

表 IX 显示了 Sleuth 用于分析各种场景的运行时和内存。测量是在配备 2.8GHz AMD Opteron 62xx 处理器和 48GB 主内存的 Ubuntu 16.04 服务器上进行的。仅使用了单个处理器的单个内核。第一列显示活动名称，第二列显示数据集的总持续时间。

第三列显示了用于依赖图的内存。如 3.2 节所述，我们设计了一个非常紧凑的主存表示。这种紧凑的表示使 Sleuth 能够存储跨越很长一段时间的数据。例如，考虑活动 L-2，其数据最密集。 Sleuth 使用了大约 329MB 来存储大约 3.5 天的 3850 万个事件。在所有数据集中，Sleuth 在较大的数据集上每个事件需要大约 8 字节的内存，在较小的数据集上每个事件大约需要 20 个字节。

第四列显示了总运行时间，包括消耗数据集、构建依赖图、检测攻击和重构场景的时间。我们注意到，这个时间是在所有数据集都可用时在参与之后测量的。在参与期间，Sleuth 正在使用这些数据，因为它们正在生成。尽管数据通常涵盖数小时到数天的持续时间，但分析本身非常快，只需几秒钟到几分钟。由于我们使用标签，大多数信息

表九

场景重建的内存使用和运行时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 持续时间 (hh:mm:ss) | 内存使用情况 | 跑 | 时间 |
| 时间 | 加速 |
| W-1 | 06:22:42 | 3 兆字节 | 1.19 s | 19.3 K |
| W-2 | 19:43:46 | 10 兆字节 | 2.13 s | 33.3 K |
| **W-均值** 6.5 兆字节 | | | **26.3 K** | |
| L-1 | 07:59:26 | 26 兆字节 | 8.71 s | 3.3 K |
| L-2 | 79:06:39 | 329 兆字节 | 114.14s | 2.5 K |
| L-3 | 79:05:13 | 175 兆字节 | 74.14 s | 3.9 K |
| **L-均值** 177 兆字节 | | | **3.2 K** | |
| F-1 | 08:17:30 | 8 兆字节 | 1.86 s | 16 K |
| F-2 | 78:56:48 | 84 兆字节 | 14.02 s | 20.2 K |
| F-3 | 79:04:54 | 95 兆字节 | 15.75 s | 18.1 K |
| **F-均值** 62.3 兆字节 | | | **18.1 K** | |

分析所需的可在本地获取。这是我们取得业绩的主要原因。

“加速”列说明了 Sleuth 的性能优势。如果 CPU 使用是唯一的限制，它可以被认为是 Sleuth 可以处理的同时数据流的数量。

总之，Sleuth 能够实时使用和分析来自多个操作系统的审计 COTS 数据，同时占用的内存很小。

# 代码和数据拆分标签的好处

如前所述，我们为每个主题维护两个可信度标签，一个对应于其代码，另一个对应于其数据。通过对代码可信度进行优先检测和前向分析，我们减少了大量的警报，同时大大减少了前向分析输出的大小。

表十

通过维护单独的代码和数据可信度标签来减少（错误）警报。平均减少显示了我们在使用拆分可信标签而不是单个可信标签时生成警报的平均减少因子。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 不受信任  执行 | | 修改者  低代码 t-tag 主题 | | 不可信的准备  执行数据 | | 机密的  数据泄露 | |
| 单个 t-tag | 拆分 t 标签 | 单个 t-tag | 拆分 t 标签 | 单个 t 标签 | 拆分 t 标签 | 单个 t-tag | 拆分 t 标签 |
| W-1 | 21 | 3 | 1.2 K | 3 | 0 | 0 | 6.1 K | 11 |
| W-2 | 44 | 2 | 3.7 K | 108 | 0 | 0 | 20.2 K | 18 |
| L-1 | 60 | 2 | 53 | 5 | 1 | 1 | 19 | 6 |
| L-2 | 1.5 K | 5 | 19.5 K | 1 | 280 | 8 | 122 K | 159 |
| L-3 | 695 | 5 | 26.1 K | 2 | 270 | 0 | 62.1 K | 5.3 K |
| **平均减少 45.39x** | | | **517x** | | **6.24x** | | **112x** | |

表 X 显示了我们的四个检测策略生成的警报数量与单个可信度标签和拆分可信度（代码和完整性）标签之间的差异。请注意，在某些情况下，拆分可将警报减少 100 到 1000 以上。

表 XI 显示了由于这种拆分而在前向分析中取得的改进。特别是，该表第 5 列中报告的选择性增加来自拆分标签。请注意，图的大小通常会缩小 100 倍到 1000 倍。

# 分析选择性

表 XI 显示了 Sleuth 中分析的数据缩减管道。第二列显示每个活动中的原始事件数。这些事件包括几天内系统中的所有事件（良性和恶意），绝大多数具有良性性质，与攻击无关。

第三列显示了进入攻击场景图表的最终事件数量。

TABLE XI

使用带有单个可信度标签的前向分析、带有拆分代码和数据可信度标签的前向分析以及最后的简化实现的选择性比较。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 初始事件数 | 最终事件数 |  | 还原 | n 因子 |  |
| 单个 t-tag | 拆分 t-tag | 侦探简化。 | 全部的 |
| W-1 | 100 K | 51 | 4.4x | 1394x | 1.4x | 1951x |
| W-2 | 401 K | 28 | 3.6x | 552x | 26x | 14352x |
| L-1 | 2.68 M | 36 | 8.9x | 15931x | 4.7x | 74875x |
| L-2 | 38.5 M | 130 | 7.3x | 2971x | 100x | 297100x |
| L-3 | 19.3 M | 45 | 7.6x | 1208x | 356x | 430048x |
| F-1 | 701 K | 45 | 2.3x | 376x | 41x | 15416x |
| F-2 | 5.86 M | 39 | 1.9x | 689x | 218x | 150202x |
| F-3 | 5.68 M | 45 | 6.7x | 740x | 170x | 125800x |
| **平均减少 4.68x** | | | | **1305x** | **41.8x** | **54517x** |

第四列显示了当从我们的后向分析确定的入口点使用具有单个可信度标签（单个 t-tag）的幼稚前向分析时的缩减因子。请注意，在大多数情况下，图形大小非常大。第五列显示了使用 Sleuth 前向分析的缩减因子——它基于拆分（代码和数据）可信度标签。从表中可以看出，相对于基于单个 t-tag 的分析，Sleuth 实现了两到三个数量级的减少。

然后将前向分析的输出输入简化引擎。第六列显示了通过对我们的前向分析输出进行简化所实现的缩减因子。最后一列显示了我们使用拆分（代码和数据）可信标签并执行简化对原始事件的总体减少。

总体而言，所有这些步骤的综合效果非常显着：由数千万条边组成的数据集被缩减为可能有一百条边的图，在 L-2 和 L-3 的情况下减少了五个数量级数据集，以及其他数据的四个数量级减少。

# 3.7 概括

我们提出了一种名为 Sleuth 的方法和系统，用于从 COTS 审计日志中实时检测攻击和重构攻击。 Sleuth 使用主内存图数据模型和丰富的基于标签的策略框架，使其分析既高效又精确。我们在来自独立红队攻击的 3 个主要操作系统的大型数据集上评估了 Sleuth，有效地重建了所有攻击，几乎没有错误。

**第 4 章**

福尔摩斯：**通过可疑信息流的关联进行实时 APT 检测**

*本章包括 [2] 中发表的材料的摘录和数字。*

# 介绍

本章解决的主要问题是实时检测正在进行的 APT 活动（包括在很长一段时间内跨越许多主机的许多不同步骤），并向分析师提供攻击场景的高级解释，基于来自企业的主机日志和 IPS 警报。这个问题主要有以下三个方面：

* + - 警报生成：从主机的低级事件跟踪开始，我们必须以有效的方式生成警报。我们如何生成警报以尝试考虑攻击者可能采取的任何重要步骤？此外，必须注意确保我们不会生成大量嘈杂的警报。
    - 警报相关性：这里的挑战是将来自攻击者的多个活动的这些警报组合成一个可靠的信号，表明存在正在进行的 APT 活动。
    - 攻击场景演示：需要将正在进行的 APT 活动的指标传达给人类（网络分析师）。为了有效，这种沟通必须

57

直观，需要对攻击进行高层次的总结，以便分析人员快速了解活动的范围和规模。

我们在本章中介绍了一个名为 Holmes 的系统，它解决了上述所有方面。 Holmes 从主机审计数据开始，并产生一个检测信号，描绘出正在进行的 APT 活动的各个阶段。在高层次上，Holmes 新颖地利用 APT 杀伤链作为解决 APT 检测上述三个方面所涉及的技术挑战的关键参考。我们在下面描述了我们的关键思想及其意义，详细的技术描述出现在第 4.3 节中。

首先，Holmes 旨在将主机日志中的活动以及企业中发现的任何警报直接映射到杀伤链。这种设计选择允许 Holmes 生成语义上接近 APT 参与者的活动步骤（“战术、技术和程序”（TTP））的警报。通过这样做，Holmes 将警报生成过程提升到在攻击活动的步骤级别上工作，而不是在低级别审计日志中显示它们。因此，我们解决了生成重要警报的重要挑战。在我们的实验中，我们发现为期五天的审计日志集合包含大约 3M 的低级事件，而 Holmes 仅从中提取了 86 个可疑活动步骤。

Holmes 的第二个重要思想是利用系统中低级实体（文件、进程等）之间的信息流作为警报关联的基础。要看到这一点，请注意杀伤链中的内部侦察步骤取决于成功的初始妥协和立足点的建立。特别是，侦察步骤通常使用攻击者在立足点建立期间安装的命令和控制代理（进程）启动。

lishment，从而表现出两个阶段所涉及的过程之间的流动。此外，侦察通常涉及运行在立足点建立阶段下载的恶意软件（文件），说明文件到进程的流程。同样，成功的横向移动阶段以及渗透阶段都使用侦察阶段收集的数据。因此，通过检测与 APT 步骤相关的低级事件并使用信息流将它们链接起来，可以构建 APT 参与者使用的新兴杀伤链。

Holmes 的第三个主要贡献是开发了高级场景图 (HSG)。 HSG 的节点对应于 TTP，边代表 TTP 所涉及的实体之间的信息流。 HSG 为以高置信度检测 APT 提供了基础。为此，我们提出了几个新的想法。首先是HSG中祖先覆盖的概念。我们展示了这个概念如何帮助评估 HSG 节点之间的依赖强度。然后可以修剪掉弱依赖关系以消除许多错误警报。其次，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关系。第三，我们开发了排名和优先级技术来修剪掉大多数与 APT 活动无关的节点和边缘。这些步骤在第 4.4.4、4.4.5 和 4.4.6 节中有详细描述。使用这些技术， 我们证明了 Holmes 能够清楚地区分攻击和良性场景。

最后，HSG 在任何时候都提供了一个非常紧凑、直观的活动摘要，从而为攻击理解做出了重要贡献。例如，从 10M 审计记录的数据集开始，我们可以使用以下方法总结高级攻击活动

只有 16 个节点的图。网络分析师可以使用所提供的 HSG 相对轻松地快速推断出攻击的总体情况（范围和程度）。

评估。我们根据 TC 程序生成的数据评估了 Holmes。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（在未受损内核的威胁模型下）。 Holmes 对九种现实 APT 攻击场景的评估，以及在为期两周的现场实验中将其作为实时入侵检测工具运行，表明 Holmes 能够清楚地区分攻击和良性场景，并能够发现网络攻击具有高精度和召回率的攻击（第 4.6 节）。

# 一个运行的例子

在本节中，我们将展示一个贯穿本章的运行示例来说明我们的方法。此示例代表红队作为 TC 计划的一部分执行的攻击。在这次攻击中，一个易受攻击的 Nginx Web 服务器在 FreeBSD 系统上运行。它的操作（系统调用）被记录在系统审计日志中。根据这些审计数据，我们构建了一个出处图，其片段如图 6 所示。该图中的节点表示系统实体，例如进程（表示为矩形）、文件（椭圆形）、网络连接（菱形）、内存对象（五边形）和用户（星星）。边对应于系统调用，并面向信息流和/或因果关系的方向。请注意，我们的出处图已使用参考文献 [36] 中描述的（优化的）节点版本控制技术呈现为非循环的。

执行

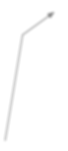
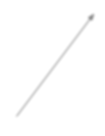
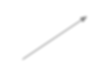
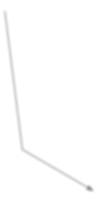
执行

执行

执行

执行

图 6. 运行示例的出处图。



不受信任的外部地址

在 出去

不受信任的外部地址

在

可信地址

**良性活动**

恩金克斯 恩金克斯

/var/log/nginxaccess.log

sshd

sshd

在

执行

执行管道 管道

sshd

/usr/lib/ libbsm.so.3

重击

php-fpm

php-fpm

/etc/ spwd.db

/lib/libz/so.6

/usr/local/www/ nginx/index.php

可信地址

在

不受信任的外部地址

其他行为

/etc/本地时间

在 出去 /lib/

libc.so.7

/libexec/ ld-elf.so.1

恩金克斯

执行

恩金克斯

恩金克斯

/var/log/nginxaccess.log

记忆

/etc/ nsswitch

.conf

猫 执行

重击

执行

执行管道 管道

mprotect\_exec 执行

/var/log/ auth.log

php-fpm

php-fpm

/usr/local/www/ nginx/index.php

/dev/tty

须藤 nginx

-s 重新加载

管道

系统日志

杀

不受信任的外部地址

U1

/usr/log/nginx-F1 根

主机名 P6

恩金克斯

**攻击者活动**

S1 输入

输出 S2

错误日志

设置

执行 执行

恩金克斯 P1

金士 P2

金鑫 P3

金丝 P4

金银 P7

输出 S3

mprotect\_exec

M1

执行

管道

管道

/etc/组 F2

写

取消链接

读

/etc/影子 F4

读

记忆

不受信任的外部地址

/usr/bin/ 前列腺素

/etc/passwd F3

/usr/local/www/ /usr/local/www/ nginx/stats.html nginx/stats.html

前列腺素

/usr/bin/

哇哇P5

恩金克斯

sshd

F5

攻击者的目标是从系统中窃取敏感信息。攻击者的活动如图 6 底部所示，包括以下步骤：

* + - 最初的妥协。攻击者在侦听端口 80 的套接字 (S1) 上发送恶意负载。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 通讯。受感染的 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。
    - 权限提升。攻击者利用现有漏洞将 Nginx 的权限提升到 root (U1)。
    - 内部侦察。接下来，攻击者发出诸如 whoami (P5) 和

主机名 (P6)。红队使用这些命令来模拟对机密/专有数据的访问。攻击者还读取用户名和密码哈希（F2、F3、F4）并将所有这些信息写入临时文件。

* + - 渗出。接下来，攻击者将包含收集到的信息的文件传输到她/他的机器（S3）。
    - 清理。在攻击的最后一步，攻击者删除临时文件 (F5) 以清理所有攻击残余。

此示例说明了以下描述的许多关键挑战：

隐形攻击。这种攻击在系统上留下的足迹最小。攻击的第一步，对 Nginx 服务器的初步入侵，在主内存中执行，不会留下任何可见的痕迹，例如下载的文件。此外，payload 在现有的 Nginx 进程中运行。检测这种隐形攻击非常具有挑战性，因为攻击者的活动与正常的系统操作无缝融合。

大海捞针。即使是单个主机也可以每天产生数千万个事件。除了极小部分（通常远低于 0.01%）外，所有这些都对应于良性

活动。 （图 6 的上半部分显示了审计日志中的一小部分良性活动。）如果没有很高的误报率，很难检测到此类罕见事件。更重要的是，从呈现给分析师的攻击摘要中过滤掉这些良性事件非常具有挑战性。

实时检测。我们设想将 Holmes 与网络响应系统结合使用，因此有必要在几秒钟内检测和总结正在进行的活动。实时检测对 Holmes 中使用的技术提出了额外的挑战和限制。

为了克服这些挑战，请注意，尽管无缝融入良性背景活动，但有两个因素与攻击有关。首先，攻击步骤实现了与某些 APT 阶段相对应的能力。其次，攻击活动是通过信息流连接起来的。在下一节中，我们将基于这两个关键观察来描述 Holmes 方法。

# 方法概述

我们的方法背后的核心观点是，尽管不同 APT 之间的具体攻击步骤可能有很大差异，但高级 APT 行为通常符合第 4 节中介绍的相同杀伤链（图 1）。我们对数百份 APT 报告的分析来自

1. 表明大多数 APT 由这些步骤的一个子集组成，如果不是全部的话。更重要的是，我们观察到这些步骤需要因果联系，这种联系是攻击正在展开的主要迹象。

请注意，每个 APT 步骤的具体表现可能会有所不同，例如，最初的妥协可能会以驱动下载或用户执行的恶意文件的鱼叉式网络钓鱼攻击的形式执行。无论如何，APT 步骤本身代表了攻击者意图的高级抽象，因此即使攻击者使用的操作策略因 APT 而异，它们也必须表现出来。此外，它们之间必然存在信息流或因果关系，因为 APT 步骤在逻辑上相互依赖，例如，渗透依赖于内部侦察以收集敏感数据。

因此，研究问题是我们是否可以基于我们的检测

* + APT 最重要的高级行为步骤，以及
  + 这些步骤之间的信息流依赖关系。

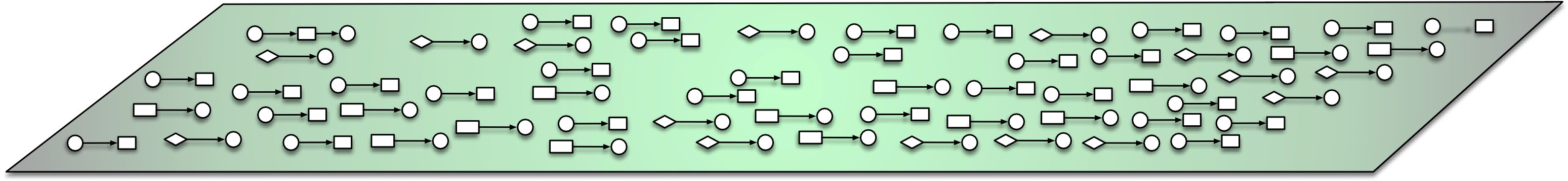
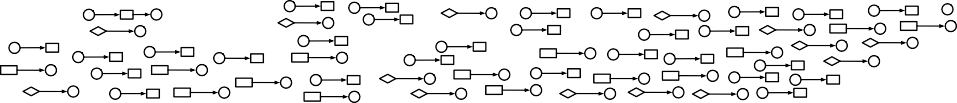
回答这个问题的一个主要挑战是低级审计数据与攻击者目标、意图和能力的高级杀伤链视图之间的巨大语义差距。

弥合语义鸿沟。为了弥合低级系统调用视图和高级杀伤链视图之间的语义鸿沟，我们构建了一个中间层，如图 7 所示。到该中间层的映射基于 MITRE 的 ATT&CK 框架 [10]，它描述了近 200 种行为模式，定义为在野外观察到的战术、技术和程序 (TTP)。

每个 TTP 都定义了一种可能的方式来实现特定的高级功能。例如，可以使用 11 个不同的 TTP 实现在受感染 Linux 系统中的持久性能力，每个 TTP 代表 ATT&CK 框架中可能的较低级别操作序列，例如安装 rootkit、修改引导脚本等上。这些低

**最初的妥协**

*不受信任的读取*



**APT阶段**

**建立立足点**

**内部侦察**

**权限提升**

**TTP 和 HSG**

*重击执行*

*rc.common C&C 通讯*

*敏感阅读*

*须藤执行*

*不受信任的执行者*

*可执行内存*

*敏感通讯*

*切换用户*

**审核日志**

图 7. Holmes 方法：从审计记录到高级 APT 阶段

级别操作更接近审计日志的抽象级别，因此可以根据出处图中的节点和边来描述 TTP。

技术挑战。实现图 7 中总结的方法的主要技术挑战是：

* + 低级事件流与 TTP 的有效匹配，
  + 检测攻击步骤之间的相关性，以及
  + 减少误报。

我们通过多项设计创新来解决这些挑战。为了有效匹配，我们使用审计日志的表示形式作为主存中的有向出处图（第 4.4 节），从而实现有效匹配。该图还对存在于系统实体（如进程和文件）之间的信息流依赖关系进行编码。 TTP 被指定为模式

利用这些依赖关系。例如，为了匹配维护持久性 TTP，从匹配初始妥协 TTP 到维护持久性 TTP 的进程必须存在信息流依赖性。

为了检测攻击步骤之间的相关性，我们构建了一个高级场景图（HSG）作为起源图的抽象。 HSG 中的每个节点代表一个匹配的 TTP，而边代表那些匹配的 TTP 之间的信息流和因果关系。 HSG 在图 7 的中间层中以粗体显示节点和边。 （我们请读者参考图 8 中运行示例的 HSG。）要确定 HSG 中节点之间的边，请使用 TTP 和 APT 阶段之间的先决条件-结果模式。

为了减少误报（即不代表攻击的 HSG）的数量，我们使用了以下组合：（a）学习可能产生误报 TTP 的良性模式，以及（b）为节点和路径分配权重的启发式算法。图表基于其严重性，以便可以对 HSG 进行排名，并将排名最高的 HSG 呈现给分析师。

总之，APT 的高级阶段使用一套通用的策略来操作，这些策略可以从审计数据中观察到。这些观察结果提供了一些恶意活动可能正在展开的证据。那么，福尔摩斯的工作就是收集证据并推断它们之间的相关性，并使用这些相关性来绘制整个攻击活动。

# 系统设计

与大多数依赖操作系统审计数据的先前工作 [25, 38-40] 一样，我们将操作系统内核和审计引擎视为可信计算库 (TCB) 的一部分。换句话说，对操作系统内核、审计系统及其产生的日志的攻击超出了我们的威胁模型的范围。我们还假设系统一开始是良性的，因此初始攻击必须源自企业外部，使用远程网络访问、可移动存储等手段。

# 数据收集和表示

Holmes 依赖于从可能运行不同操作系统 (OS) 的多个主机检索的审计日志构建的出处图。1如第 3 章所述，我们使用高度紧凑的出处图表示，平均而言，审计日志中每个事件需要少于 5 字节的主内存。此外，我们的出处图的实体是版本化的。如果该边更改了节点的现有依赖项（即祖先节点集），则在添加传入边之前创建节点的新版本。版本控制支持优化，可以在不改变取证分析结果的情况下删除审计日志中的大部分事件 [36]。而且，这个版本化的图是非循环的，可以简化很多图算法。这种表示可以在很长一段时间内实时消耗事件和图形构建。 正是在这个出处图上，我们的分析查询符合我们的 TTP 规范的行为。

1Holmes 的设计使得可以从各种 IDS/IPS 中获取额外的输入，例如事件和警报，但我们不会进一步讨论系统的这方面。

# TTP 规范

TTP 规范提供了低级审计事件和高级 APT 步骤之间的映射。因此，它们是我们方法的核心组成部分。在本小节中，我们将描述 TTP 设计中的三个关键选择，以实现高效和精确的攻击检测。

回想一下，在我们的设计中，TTP 代表具体审计日志和高级 APT 步骤之间的中间抽象层。具体来说，我们依靠两种主要技术将审计日志数据提升到这个中间层：（a）以源图的形式表示与安全相关的事件的操作系统中立表示，以及（b）使用所涉及实体之间的信息流依赖关系在 TTP 中。总之，这些技术可以实现恶意行为的高级规范，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、恶意软件的名称、创建的中间文件以及用于创建它们的程序等。在这方面，我们基于信息流的 TTP 规范方法比使用 IDS 文献中的误用规范 [41, 42] 更通用。 使用信息流依赖性对于检测通过使用良性系统进程来执行其目标来隐藏其活动的隐秘 APT 至关重要。

除了指定 TTP 的步骤之外，我们还需要捕获它的先决条件。先决条件不仅有助于减少误报，还有助于理解 TTP 在 APT 活动的更大背景下的作用。在我们的 TTP 规范中，先决条件采用 APT 阶段之间的因果关系和信息流的形式。

最后，TTP 匹配需要高效，并且必须不需要昂贵的技术，例如回溯。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，并对所涉及的主体和客体附加先决条件。

表 XII 显示了 TTP 规则规范的示例，其他规则出现在第 4.5 节中。在表 XII 中，第一列表示 APT 阶段，第二列表示关联的 TTP 名称和 TTP 中涉及的实体。第三列指定与 TTP 关联的事件系列。为了便于说明，该系列中包含的一些特定事件显示在第四列中，但请注意它们不是 TTP 规则的一部分。 （事件类定义一次，并在所有 TTP 规则中重复使用。）

表十二

示例 TTP。在严重性列中，L=Low，M=Moderate，H=High，C=Critical。实体类型由字符表示：P=Process, F=File, S=Socket, M=Memory, U=User。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **APT阶段** | **TTP** | **事件系列** | **活动** | **严重性** | **先决条件** |
| *初始妥协(P )* | *不受信任的读取（S，P）* | 读 | 文件读取 （视窗），  读/预读/读/预读 (Linux,BSD) | L | *S.ip ∈/*  {可信 IP 地址} |
| *使内存执行（P，M）* | 保护 | 虚拟分配 (Windows), mprotect (Linux,BSD) | M | $PROT EXEC$ *∈ M.flags*  *∧ ∃ 不受信任的读取（？，P）：*  *路径因子（P，P）<=*  *路径阈值* |
| *立足点(P)* | *壳牌执行（F，P）* | 执行 | ProcessStart （Windows）， execve/fexecve （Linux，BSD） | M | *F.路径∈*  {命令行实用程序}  *∧ ∃ 初始妥协（P）：*  *路径因子（P，P）<=*  *路径阈值* |

第五列代表与每个 TTP 相关的严重性级别。我们使用此严重性级别对系统发出的警报进行排名，优先处理最严重的警报。我们目前

严重性级别的分配基于 US-CERT 和 DHS 与 MITRE 合作定义的通用攻击模式枚举和分类 (CAPEC) 列表

[43] 但可以根据特定企业的需要进行定制。我们还提供了另一种自定义机制，每个严重性级别都可以映射到分析师指定的权重，该权重反映了部署上下文中不同 APT 阶段的相对重要性。

最后一列指定要匹配的 TTP 规则的先决条件。先决条件可以指定匹配 TTP 参数的条件，例如，第一行 Untrusted Read TTP 的套接字参数 S。先决条件还可以包含先前匹配的 TTP 及其参数的条件。例如，先决条件列

Make Mem Exec(P, M ) TTP 包含条件∃ Untrusted Read(?, P*!*)。这个前提

仅当先前已为进程 P*!*匹配了不受信任的读取 TTP，并且两个 TTP 中涉及的进程具有小于指定阈值的路径因子（定义如下）时，才满足。

先决条件可以捕获两个 TTP 中涉及的实体之间的关系，例如进程上的父子关系或文件之间的信息流。他们还可以捕获两个 TTP 共享一个共同父项的条件。使用先决条件，我们能够修剪许多误报，即类似于 TTP 的良性活动。

# 恒山建设

图 8 说明了运行示例的 HSG。该图的节点表示匹配的 TTP，并在图中用椭圆表示。在每个椭圆内，我们用灰色表示匹配的出处图实体。出于说明目的，我们还包括

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、中或高）。图的边缘表示不同 TTP 之间的先决条件。完成两个实体之间路径的虚线表示先决条件。例如，Make Mem Exec TTP 作为先决条件具有不受信任的读取 TTP，由两个节点之间的边缘表示。

HSG 的构建主要由先决条件驱动：如果满足所有先决条件，则匹配 TTP 并将其添加到 HSG。这个因素可以随时减少 HSG 中的 TTP 数量，从而可以在不影响实时性能的情况下进行复杂的分析。

# 避免虚假依赖

通过虚假依赖，我们指的是对攻击者活动的无趣和/或不相关的依赖。例如，在图 6 中，进程 nginx (P2) 写入文件

/usr/log/nginx-error.log，cat 进程稍后会读取该文件。但是，即使 cat 和日志文件之间存在依赖关系，但 cat 与攻击无关，并且通过 ssh 独立调用。更一般地，考虑任何消耗攻击活动产生的次要工件的进程，例如，一个日志轮换系统，它复制一个包含攻击者进程产生的部分条目的日志文件。此类进程虽然代表良性背景活动，但将在来源图中标记为依赖于攻击者的进程。如果不及时修剪这些虚假的依赖关系，则可能会出现依赖关系爆炸，从而极大地增加 HSG 的大小。因此，呈现给分析师的最终结果可能充满了良性活动，这可能导致

S1 P1 P2 S2

*C*

M1 P1

**APT阶段：特权**

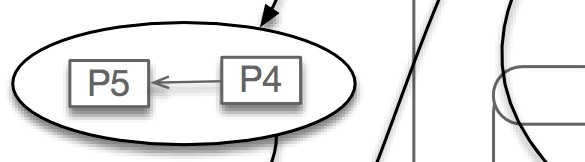
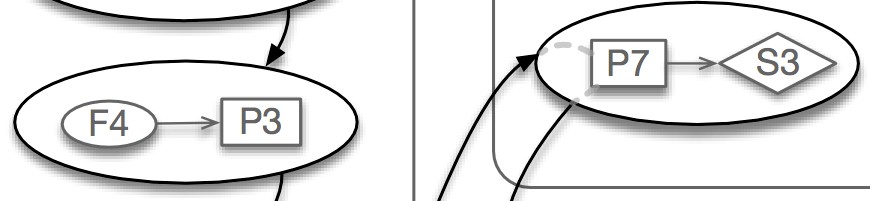
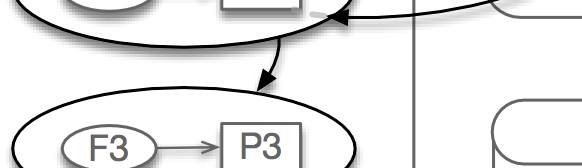
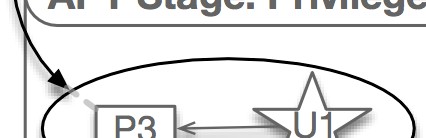
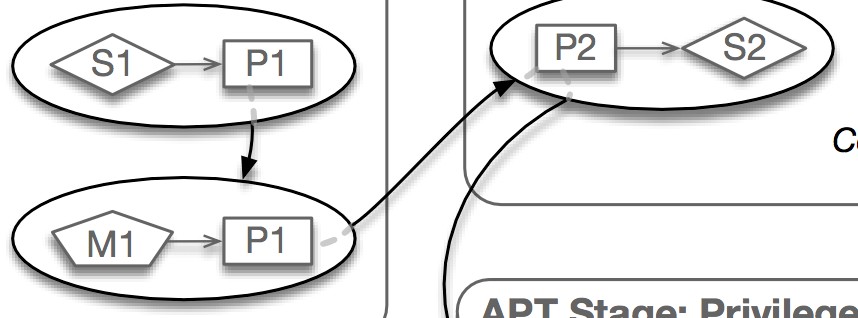
P3 U1

F2 P3

F3 P3

P7 S3

F4 P3



**APT 阶段：初始妥协**

*L*

*不受信任的读取*

**APT阶段：建立立足点**

*L*

*M*

*C&C通讯*

*使内存执行*

**升级**

**APT阶段：内部侦察**

*H*

*M* *开关 SU*

*敏感阅读*

*M*

*敏感阅读*

*M*

*敏感阅读*

**APT阶段：渗透**

*H*

*敏感泄漏*

*敏感命令*

*H*

*敏感命令*

*H*

**APT 阶段：清理**

*M*

*敏感的*

*温度RM*

图 8. 运行示例的高级场景图。

P5 P4

P6 P4 P7 F5

分析员错过嵌入在大图中的关键攻击步骤。为此，我们优先考虑

强依赖于弱依赖，尽可能地修剪掉后者。

直观地说，如果 P*d*是 P*a*的后代进程，我们可以说进程 P*d*对进程 P*a*具有很强的依赖性。类似地，如果 P*a*或其后代进程写入此文件/套接字，则文件或套接字对进程 P*a*具有很强的依赖性。更一般地，在出处图中考虑两个实体和它们之间的路径，该路径指示它们之间的信息。确定此流代表强信息流还是弱信息流等同于确定流中的实体是否共享受损的祖先。如果它们共享受感染的祖先，它们就是攻击者活动的一部分，并且它们之间存在很强的依赖关系，必须优先考虑。否则，我们认为依赖性很弱，并在我们的分析中不再强调它。

为了将上述讨论推广到可能存在多个受损进程的情况，我们引入以下概念，即信息流路径 f 上所有进程的祖先覆盖 AC(f)：

∀p ∈ f ∃a ∈ AC(f ) a = p 或者 a 是 p 的祖先

注意 f 中的非进程节点不影响上述定义。最小祖先覆盖，AC*分钟*f ) 是最小尺寸的祖先覆盖。直观地说，AC*分钟*f) 表示攻击者必须妥协的最小祖先数量（即漏洞利用次数）才能完全控制信息流路径 f 例如，再次考虑来自

受攻击者控制的 nginx 进程到 cat 进程。由于这两个进程没有共同的祖先，因此它们之间路径的最小祖先覆盖的大小等于 2。因此，为了控制 cat 进程，攻击者必须为 cat 开发额外的漏洞利用。这就需要攻击者首先找到cat中的漏洞，然后创建相应的exploit，最后将这个exploit写入日志文件。通过选择大小为 1 的祖先覆盖，我们捕捉到这样一个事实，即这种涉及 cat 的攻击比攻击活动由 nginx 及其后代执行的攻击的可能性要小得多。

我们现在可以定义前面讨论 TTP 中提到的路径因子 (N1, N2的概念。直观地说，它捕获了攻击者对从 N1到 N2的流量的控制程度。基于上述使用最小祖先覆盖作为依赖强度度量的讨论，我们将路径因子定义如下。考虑从 N1到 N2的所有信息流路径 f1, ..., f*n*，并让 m*i*是 f*i*的最小祖先覆盖大小。那么，路径因子(N1, N2) 就是 m1, 中的最小值。 . . , m*n*。

请注意，如果进程 N2是 N1的子进程，则在 N1到 N2之间存在一条只有一条边的路径。由于 N1是 N2的祖先，因此该路径的最小祖先覆盖的大小为 1。相比之下，从 nginx 到 cat 的（唯一）路径具有大小为 2 的最小祖先覆盖，因此路径因子（nginx，cat）= 2。

我们在 4.5 节描述了路径因子的有效计算。根据我们的经验，通过优先考虑受攻击者影响的流，路径因子的使用大大减轻了依赖性爆炸。

# 降噪

分析审计日志以进行攻击检测和取证的挑战之一是噪声的存在，即与 TTP 规则匹配的良性事件。浏览器、Web 服务器和 SSH 守护程序等长期存在的进程会不时触发 TTP 匹配。为了减少这些误报，我们结合了基于训练数据的降噪规则。我们利用两个概念：（1）良性先决条件匹配和（2）良性数据流量。

基于良性先决条件的降噪。对于每个进程，我们的系统都会学习系统在良性环境中运行时频繁触发的先决条件。在运行时，当触发的 TTP 的先决条件与训练期间遇到的先决条件匹配时，我们会忽略匹配。

基于数据流量的降噪。基于良性先决条件的过滤可能会导致误报：恶意事件可能会被忽视，因为它与学习阶段观察到的行为相匹配。例如，即使没有任何攻击，nginx 在启动阶段也会读取 /etc/passwd。但是，如果我们将所有对 /etc/passwd 的 nginx 访问列入白名单，那么受感染的 nginx 服务器的后续读取将不会被注意到。

为了解决这个问题，我们加强了学习，以合并大量的信息流，以传输的字节为单位。例如，可以从文件 /etc/passwd 流向 nginx 的信息量等于该文件的大小，因为 nginx 只读取该文件一次。因此，如果观察到大量字节从 /etc/passwd 流向 nginx，则该流可能是攻击的一部分。为了确定信息量的截止点，我们在良性环境中观察了一段时间内的进程文件和进程套接字对。

# 信号相关与检测

给定一组 HSG，我们如何以高置信度区分构成攻击的 HSG？我们通过为每个 HSG 分配严重性分数来应对这一挑战。该分配在下面进一步描述的两个步骤中进行。

威胁元组。首先，我们通过与相应 HSG 关联的抽象威胁元组来表示攻击者在活动中的进展。特别是，对于每个 HSG，威胁元组是

7 元组 (S1, S2, S3, ..., S7) 其中每个 S*i*对应于 APT 阶段的严重性级别

HSG 指数 i。我们根据对野外 APT 的广泛调查选择了 7 元组 [5]，但其他选择也是可能的。

由于属于某个 APT 阶段的不同 TTP 可能具有不同的严重级别，因此通常有多个候选者可供选择。在这些候选者中选择最高严重级别是很自然的。例如，与图 8 的 HSG 关联的威胁元组

是 (M, L, H, H, -, H, M)。该元组包含 6 个条目，因为其匹配的 TTP 属于 6

不同的APT阶段。条目根据杀伤链中 APT 阶段的顺序进行排序。例如，元组的第一个条目是 M，因为图中属于 Initial Reconnaissance 的最严重的 TTP 的严重性为 M。

HSG 排名和优先级。为了对 HSG 进行排名，我们首先将威胁元组转换为数值。特别是，我们首先将威胁元组的每个元素映射到基于通用漏洞评分系统 (CVSS) 中包含的转换表（表 XIII）的数值，CVSS 是通过跨领域安全专业人员协作创建的供应商中立行业标准商业、非商业和学术部门 [44]。替代得分

企业可以根据其感知到的威胁和过去的威胁历史来做出选择。

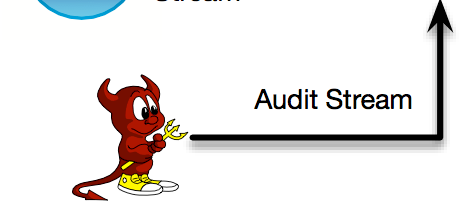
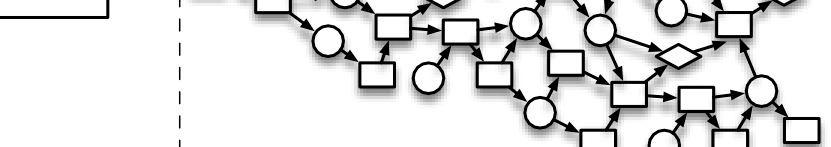
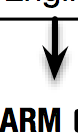
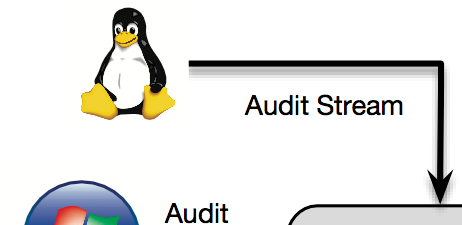
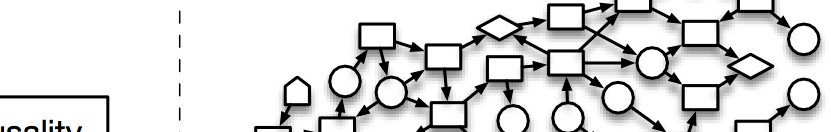
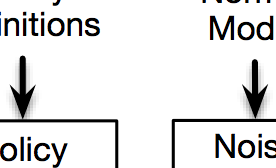
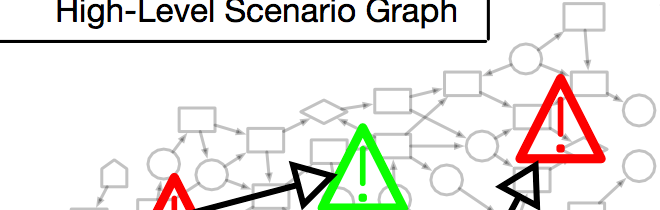
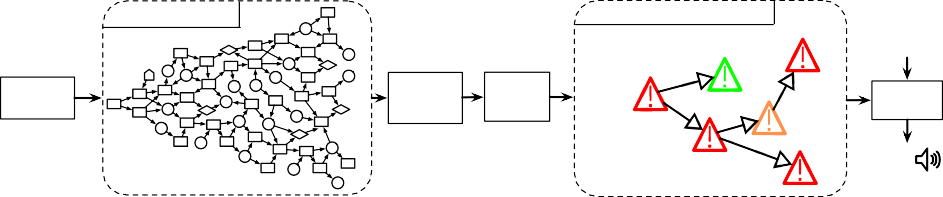
表十三

NIST 严重程度评定量表

|  |  |  |
| --- | --- | --- |
| **定性水平** | **定量范围** | **四舍五入的平均值** |
| 低的 | 0.1 - 3.9 | 2.0 |
| 中等的 | 4.0 - 6.9 | 6.0 |
| 高的 | 7.0 - 8.9 | 8.0 |
| 批判的 | 9.0 - 10.0 | 10.0 |

接下来，我们将 7 个 APT 阶段的数字分数组合成一个整体分数。我们用于计算此分数的公式在设计时考虑了两个主要标准：(1) 灵活性和定制化，以及 (2) APT 步骤的相关性反映在随着步骤展开而放大的分数。为了解决这些标准，我们将权重与转换后的威胁元组中的每个条目相关联，并计算威胁元组的加权乘积作为分数。这些权重可由系统管理员配置，它们可用于将特定阶段的检测优先于其他阶段。

使用训练集，我们进行了几次实验，并使用其他方案比较了结果，例如加权和、指数和和几何和。对于每个方程，我们测量了良性子图分数和攻击子图之间的平均边距



出处图

政策德

高级场景图

审计流

正常模型

权重/阈值

审计

流处理器

因果关系

P

引擎

e

离子 ne

**A**

图 9. Holmes 架构。

定义

奥利西 噪音

探测

溪流 匹配 过滤 恩吉

追踪器 引擎

**报警器**

审计流

归一化后的分数，发现加权产品的结果最好。因此，我们使用以下标准来标记 APT 攻击：

*n*

它

(S*i*)*无线*≥ τ(4.1)

*i*=1

这里，n是APT阶段的数量，w*i*和S*i*分别表示阶段i的权重和严重性，τ是检测阈值。如果在第 i 阶段没有发生 TTP，我们设置 S*i*= 1。

# 执行

来源图构建的流消耗。图 9 显示了 Holmes 的架构。为了实现平台独立性，来自不同操作系统的审计记录被标准化为通用数据表示 (CDR)，并为各种系统实体提供共享抽象。为了简化审计数据处理，将基于 CDR 的审计记录发布到流处理服务器 (Kafka)，并通过从流服务器消费进行实时分析和检测。我们使用我们的 Sleuth 系统 [1] 进行流消费、因果关系跟踪和来源图构建，因此我们不会在此处详细描述这些步骤。

策略匹配引擎和HSG构建。策略匹配引擎将 TTP 规则规范作为输入并在出处图上运行。 Holmes 的当前实现中使用的一组具有代表性的 TTP 规则规范如表 XIV 所示。为了匹配 TTP，在构建出处图时，策略匹配引擎会迭代规则表中的每个规则及其先决条件。该任务的一个特别具有挑战性的部分是检查每个 TTP 的先决条件关于先前匹配的 TTP 和路径因子。事实上，先前匹配的 TTP 可能位于图的较远区域，并且路径因子值可能取决于必须遍历的长路径。我们注意到，先前关于攻击取证的工作 [1, 9, 25, 40] 的一种常见做法是从 TTP 匹配点进行反向跟踪以到达初始妥协点。很遗憾， 这是一种在实时环境中计算成本很高的策略，因为出处图可能包含数百万个事件。

为了在不回溯的情况下解决这一挑战，我们使用增量匹配方法存储先前计算和匹配的结果，并沿图传播指向这些结果的指针。当某个特定的 TTP（可能在其他 TTP 中作为先决条件出现）匹配时，我们在 HSG 中创建相应的节点以及指向该节点的指针。接下来将指针传播到所有依赖于匹配 TTP 的实体的低级实体。

类似地计算路径因子。特别地，给定一个匹配的 TTP 表示为 HSG 中的一个节点，路径因子值被递增地计算出源图的节点，这些节点依赖于匹配的 TTP 的实体。假设 N1为

生成与 TTP 匹配的事件的过程，最初将路径因子 (N1, N1) 分配为 1。随后，当边 (N1, N2被添加到源图时, 如果 N2是一个非进程节点，或者如果它是一个至少有一个与 N1共同祖先的进程，则路径因子(N1, N2将为 1。否则，路径因子值增加 1。如果信息流发生从 N2到 N3，而 N2和 N3都已经具有来自 N1的依赖流，则构造新版本的N3，并将路径因子(N1, N3*新的*)设置为两个流计算的路径因子中的最小值。请注意，在基于此版本控制系统构建的非循环来源图中，路径因子（N1，N2一旦设置就永远不会改变。最后，当遇到一个TTP事件对应的事件时，如果可用，我们可以立即重用指向先决条件 TTP 和预先计算的路径因子的指针。

两层（起源图和 HSG）的这种基于指针的相关性的预期瓶颈是随着起源图随着时间的增长而增加的空间开销和复杂性。我们的操作观察是，通常大量实体指向同一组 TTP；这种现象不是随机的，实际上是进程树中指针从父进程传播到其所有后代的结果。事实上，随着分析的进行，很少会添加新的指针。一般来说，关键的实现洞察力是维护一个将出处图的实体映射到 HSG 的 TTP 的中间对象。因此，出处图中的每个实体只有一个指向中间映射器的指针，而映射器对象包含一组指针。

噪声过滤和检测引擎。噪声过滤引擎识别良性 TTP 匹配，以便将它们从 HSG 中排除。它将正常行为作为输入

表 XIV：代表性 TTP。事件族表示 Windows、Linux 和 FreeBSD 中的一组相应事件。在严重性列中，L=Low，M=Moderate，H=High，C=Critical。实体类型由字符表示：P=Process, F=File, S=Socket, M=Memory, U=User。

:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **APT阶段** | **TTP** | **事件**  **家庭** | **严重性** | **先决条件** |
| *初始妥协(P )* | *不受信任的读取（S，P）* | 读 | L | S.ip ∈/ {可信 IP 地址} |
| *使内存执行（P，M）* | 保护 | M | $PROT EXEC$ *∈ M.flags*  *∧ ∃ 不受信任的读取（？，P）：*  *路径因数（P， P）* |
| *使文件执行（P，F）* | 频道模组 | H | $PROT ∈ F.模式的 EXEC$  *∧ ∃ 不受信任的读取（？，P）：*  *路径因数（P， F ）*  *∧ ∃ 不受信任的读取（？，P）：*  *路径因数（P， P）* |
| *不受信任的文件执行（F，P）* | 执行 | C | *∃ 不受信任的读取（？， P） ： 路径因子（P， F ）* |
| *立足点(P)* | *壳牌执行（F，P）* | 执行 | M | F.path ∈ {命令行实用程序}  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *CnC*(*P, S*) | 发送 | L | *S.ip ∈/*  *{可信 IP 地址} ∧ ∃ 初始妥协 (P) 路径因子 (P, P ) <= 路径 thres* |
| *权限提升(P)* | *须藤执行（F，P）* | 执行 | H | *F.路径∈*  *{超级用户工具} ∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *开关 SU (U, P)* | 塞图伊德 | H | *U.id ∈*  *{超级用户组} ∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *内部侦察(P)* | *敏感读(F, P)* | 读 | M | F.path ∈ {敏感文件}  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *敏感指令(P, P )* | 叉子 | H | P.name ∈ {敏感命令}  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |

*下页继续*

*表十四——接上一页*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **APT阶段** | **TTP** | **事件**  **家庭** | **严重性** | **先决条件** |
| *横向移动(P )* | *发送内部（P，S）* | 发送 | M | S.ip ∈ {内部 IP 范围}  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *完成任务(P)* | *敏感泄漏(P, S)* | 发送 | H | S.ip ∈/ {可信 IP 地址}  *∧ ∃ 内部侦察（P）：*  *路径因数（P， P）*  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *破坏系统(F, P)* | 写/取消链接 | C | F.path ∈ {系统关键文件}  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *清理轨道(P)* | *清除日志(P, F)* | 取消链接 | H | *F.path ∈ {*Log Files*}∧ ∃ Initial Compromise*(*P* ) :  *路径因数（P， P）* |
| *敏感温度 RM (P, F)* | 取消链接 | M | *∃ 内部侦察（P）：*  *路径因数（P， F ）*  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |
| *不受信任的文件 RM (P, F)* | 取消链接 | M | *∃ 初始妥协（P）：*  *路径因数（P， F ）*  *∧ ∃ 初始妥协（P）：*  *路径因数（P， P）* |

在良性运行中学习的模型。此模型包含在良性运行中匹配的 TTP 的映射以及在这些运行中从系统对象读取或写入系统对象的字节数阈值。当策略匹配引擎匹配一个新的 TTP 时，在这个模型中搜索该 TTP 的实体和先决条件。如果模型中存在包含所有先决条件和匹配事件（具有相同实体名称）的条目，则根据良性阈值检查传输的字节总数。如果总字节数

转移低于良性阈值，则过滤掉匹配的TTP对应的节点；否则，与其对应的节点将被添加到 HSG。最后，检测引擎计算不同 HSG 的加权和，并在该值超过检测阈值时发出警报。

# 实验评价

我们的实验评估是针对 TC 计划组织的红队与蓝队对抗性交战进行的。我们首先在我们事先可用的数据集上评估了 Holmes（第 4.6.1、4.6.2、4.6.3、4.6.4 节）。使用此评估，我们在 4.6.5 节计算 Holmes 的最佳阈值，并在 4.6.6 节测量其性能。最后，在第 4.6.7 节中，我们探讨了 Holmes 作为现实世界实时检测系统在我们事先不知道红队何时或如何进行攻击的环境中的适用性。在我们的现场实验之后，该数据集已在公共领域 [28] 发布，以激发该领域的进一步研究。

# 数据集

攻击。我们用于评估的数据集总结在表 XV 中。此表显示了来自三个操作系统平台的 7 台主机的九个 APT 场景。每个平台都有三个场景。总的来说，这些流涵盖了 20 天的审计日志。流 5 和 7 各包含两个独立的 APT 攻击，而其余流各包含一个 APT 攻击。

简而言之，红队攻击场景中的对抗性目标和活动涵盖了备受瞩目的 APT 活动。其中包括典型的 APT 活动，例如浏览器引发的

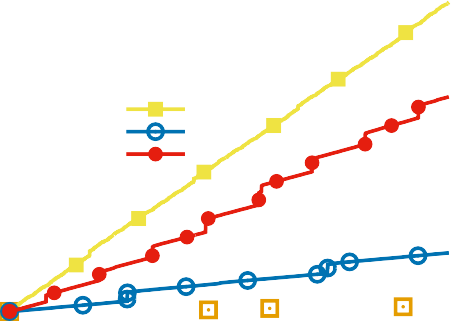
表十五

数据集。流 5 和 7 包含发生在同一主机上的两个独立攻击向量。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **流号** | **期间** | **平台** | **方案编号** | **方案名称** | **攻击面** |
| 1 | 0d1h17m | Ubuntu 14.04（64 位） | 1 | 路过下载 | 火狐 42.0 |
| 2 | 2d5h8m | Ubuntu 12.04（64 位） | 2 | 木马 | 火狐 20.0 |
| 3 | 1d7h25m | Ubuntu 12.04（64 位） | 3 | 木马 | 火狐 20.0 |
| 4 | 0d1h39m | Windows 7 专业版（64 位） | 4 | 间谍软件 | 火狐 44.0 |
| 5 | 5d5h17m | Windows 7 专业版（64 位） | 5.1 | 永恒之蓝 | 脆弱的中小企业 |
| 5.2 | 鼠 | 火狐 44.0 |
| 6 | 2d5h17m | FreeBSD 11.0（64 位） | 6 | 网络外壳 | 后门 Nginx |
| 7 | 8d7h15m | FreeBSD 11.0（64 位） | 7.1 | 鼠 | 后门 Nginx |
| 7.2 | 密码劫持 | 后门 Nginx |

偷偷摸摸的初始妥协、后门注入、特权升级、内部侦察、敏感资产的泄露以及攻击足迹的清理。在这些攻击中，红队使用复杂的攻击向量（例如反射加载、web-shell 功能和内存模块加载）来评估 Holmes 对 APT 攻击的鲁棒性。

10



流 1

流 2

流 3

流 4

流 5

流 6

流 7

9

发表记录（百万）

8

7

6

5

4

3

2

1

0

0 20 40 60 80 100

跟踪持续时间 (%)

1400

1200

1000

主题大小 (MB)

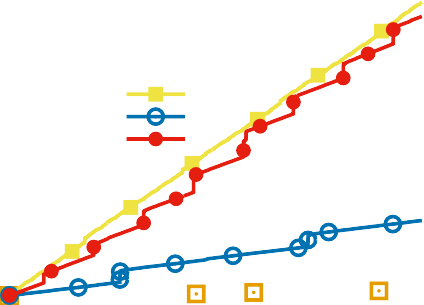
800

600

400

200

0



流 1

流 2

流 3

流 4

流 5

流 6

流 7

0 20 40 60 80 100

跟踪持续时间 (%)

图 10.（左）：已发布记录的数量（百万）与跟踪持续时间的百分比。（右）：主题大小 (MB) 与跟踪持续时间的百分比。

良性数据。为了构建降噪模型，我们使用了从目标主机流式传输的 4 天的良性审计数据。这些程序包括浏览器（例如 Firefox）、Web 服务器（例如 Nginx）和各种守护进程（例如 postfix、syslogd）。

# 设置

Holmes 订阅了 7 个 Kafka 主题（每个主机一个主题）以使用审计流并实时执行分析和检测。图 10（左）显示了与审计跟踪持续时间增长百分比有关的已发布记录数（以百万计）。请注意，Holmes 的消耗速度与 Kafka 服务器的发布速度一样快。图 10（右）显示了发布到每个 Kafka 主题的记录大小的增量增长。

我们为 Holmes 配置了表 XIV 中提到的 TTP，并为 TTP 的先决条件设置了路径 thres = 3，为 APT 阶段 i 设置了权重 = (10 + i)/10，这考虑到了后期 APT 阶段的权重稍高。

# 结果简而言之

表 XVI 总结了九种攻击场景的检测。第二列显示了检测过程中匹配的每个 HSG 的威胁元组，第三列显示了相应的威胁分数。第四列显示了在执行攻击场景的机器的所有良性场景中得分最高的。这些良性场景可能包含相应攻击场景中的确切程序。

分配给良性 HSG 的最高分数是 338（场景 3），分配给攻击 HSG 的最低分数是 608（场景 5.2），这与没有伤害的不完全攻击有关

对系统进行。这表明 Holmes 将攻击和良性场景分成了两个不相交的集群，并在它们之间做出了明确的区分。

表十六

分配给攻击场景的分数。L = 低，M = 中，H = 高，C = 严重。

**注意：**对于每个场景，数据集中的最高良性分数是分配给主机审计日志收集期间流式传输的良性后台活动的最高威胁分数（攻击前、攻击并行和攻击后）。

|  |  |  |  |
| --- | --- | --- | --- |
| **方案编号** | **威胁元组** | **威胁评分** | **数据集中的最高良性分数** |
| 1 | *(C, M, −, H, −, H, M )* | 1163881 | 61 |
| 2 | *(C, M, −, H, −, H, −)* | 55342 | 226 |
| 3 | *(C, M, −, H, −, H, M )* | 1163881 | 338 |
| 4 | *(C, M, −, H, −, −, M )* | 41780 | 5 |
| 5.1 | *(C, L, −, M, −, H, H)* | 339504 | 104 |
| 5.2 | *(C, L, −, −, −, −, M )* | 608 |
| 6 | *(L, L, H, M, −, H, −)* | 25162 | 137 |
| 7.1 | *(C, L, H, H, −, H, M )* | 4649220 | 133 |
| 7.2 | *(M, L, H, H, −, H, M )* | 2650614 |

学习降噪规则和路径因子的效果如图 11 所示。该图显示了在分析所有七个流后构建的所有良性和攻击 HSG 的威胁得分。这些分数显示在三种不同的设置下：默认启用学习和路径因子计算、不学习和没有路径因子。从图中可以明显看出，在学习和路径因子的作用下，攻击 HSG 与良性 HSG 之间存在更大的差距。如果没有学习或路径因素，我们会注意到噪声的增加，这会导致误报或误报。默认框的第 10 个百分位数、第一个四分位数和中位数都在该框的底线上（分数 = 2.1）。这个

表示超过 50% 的威胁评分为 2.1，这是许多 HSG 仅有一个低严重性 Untrusted Read TTP 的结果。

108

107

106

威胁评分

105

104

103

102

101

100

良性攻击

默认 不学习 没有路径因子

图 11. 学习和路径因子对降噪的影响。方框从第一个四分位数覆盖到第三个四分位数，中间的条表示中位数，胡须从第 10 个百分位数延伸到第 90 个百分位数。

# 攻击场景

我们现在描述 Holmes 检测到的另一个攻击场景。出于篇幅原因，我们在附录 B 中包含了其余场景的详细信息和相关图。我们注意到场景 7.2 在第 4.2 节中讨论，其部分来源图和 HSG 显示在图 6 和图8，分别。

场景 1：路过式下载。在这种攻击场景中（参见图 12），用户使用易受攻击的 Firefox 浏览器访问恶意网站。结果，一个名为 net 的文件被删除并在受害者的主机上执行。该文件在执行后会连接到 C&C 服务器，并向攻击者提供反向 shell。然后，攻击者启动一个 shell 提示符并执行诸如 hostname、whoami、ifconfig、netstat 和 uname 之类的命令。最后，恶意可执行文件将信息泄露到 C&C 服务器的 IP 地址，然后攻击者将删除的恶意文件删除。

从图 12 可以看出，在 Initial Compromise APT 阶段，执行了一个不受信任的文件，该文件匹配一个具有严重严重级别的 TTP。最后的威胁元组

对于所有 APT 阶段，该图看起来像 (C, M, −, H, −, H, M)（参见表 XVI）。最后，

转换后的量化值为 (10, 6, 1, 8, 1, 8, 6)，这导致威胁分数等于

1163881.

# 寻找最佳阈值

为了确定最佳阈值，我们通过改变阈值来测量精度和召回率，如图 13 所示。F 分数，精度和召回率的调和平均值，在区间 [338.25, 608.26] 处最大，即范围从良性子图的最高分到攻击子图的最低分。因此，通过在此范围内选择任何阈值，Holmes 可以清楚地区分测试数据集中的攻击子图和良性子图，准确率和召回率等于 1。

为了找到最佳值，我们首先将威胁分数转换为线性比例，方法是获取它们的

第 n 个根，其中 n 等于L7

*i*=1

w*i*。转换后的值显示平均贡献

*L*

A 收视率 B D 发送 A

*H*

C 修改B D 执行 E

777

*C*

C 执行 D **APT阶段：英特**

*H*

E 叉子 F

*H*

**C**

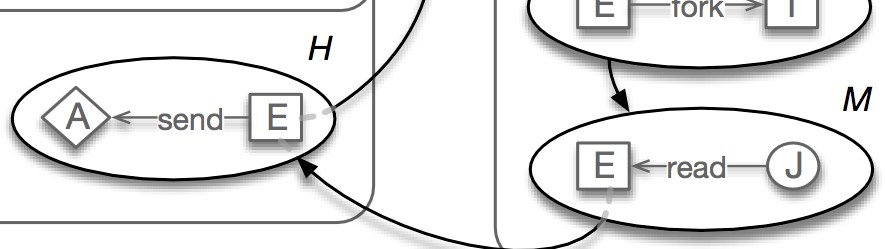
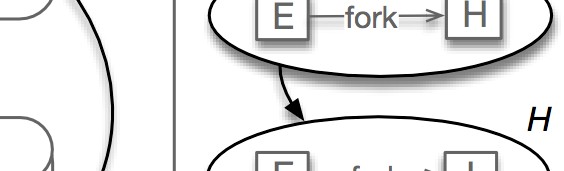
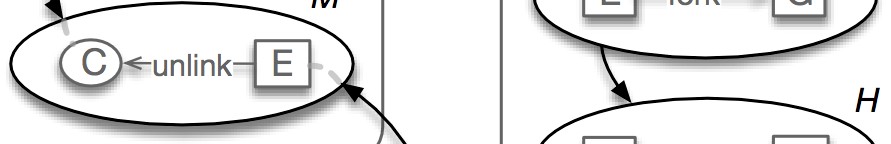
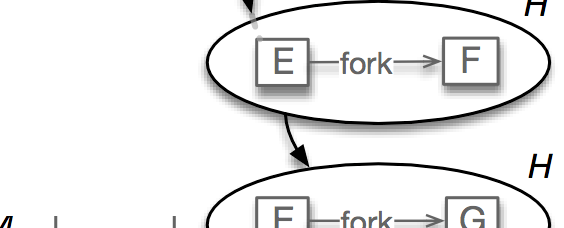
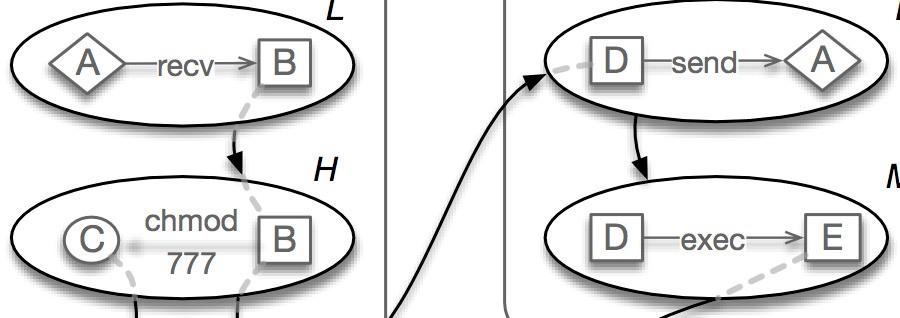
*M* E 叉子 G

C 取消链接 E

*H*

E 叉子 H

*H*



**APT 阶段：初始妥协**

*不受信任的读取*

**APT阶段：建立立足点**

*L*

*C&C 通讯*

*M*

*使文件执行* *壳*

*执行*

*不受信任的文件执行*

**最终侦察**

*敏感命令*

**APT阶段：** **瘦身**

*不受信任的文件 Rm*

*敏感命令*

*敏感命令*

**APT阶段：渗透**

*敏感命令*

*敏感泄漏*

*敏感阅读*

图 12. 场景 1 的 HSG（路过下载）。符号：A= 不受信任的外部地址；B=火狐；C=恶意丢弃的文件（网络）；D = RAT过程；E=重击；F = 哇哦；G=无名；我=网络统计；J=公司机密.txt；

E 叉子 I

*H*

A 发送 E *M*

E 读 J

每个 APT 步骤对总体威胁评分的影响，它是一个在 [1,10] 范围内的值。由于到目前为止我们测试的所有数据集都属于单个主机，因此我们排除了横向移动步骤的权重

1

精确

回忆 F 分数

0.9

0.8

0.7

0.6

速度

0.5

0.4

0.3

0.2

0.1

0

100 101 102 103 104 105 106 107

临界点

图 13. 通过改变阈值进行攻击检测的精度、召回率和 F 分数。

(w5)，这导致 n = 8.3。得到第 n 个根后，最大 F-score 的区间将变为 [2.01, 2.16]。最后，我们将此范围的中间值 (2.09) 视为在良性环境中允许每个 APT 步骤对整体威胁评分做出贡献的平均严重性。

# 表现

图形大小。图 14 显示了数千条边（左）和 HSG 边数（右）的出处图增长趋势的比较。在边中测量的图大小比为 1875:1，即在从出处图映射到 HSG 的过程中实现了 1875 倍的减少。

3000

省。图形大小（1000 条边）

2500

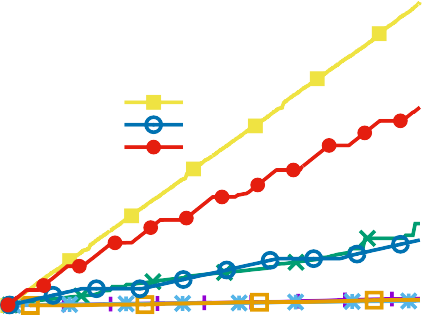
2000

1500

1000

500

0



流 1

流 2

流 3

流 4

流 5

流 6

流 7

0 10 20 30 40 50 60 70 80 90 100

1600

1400

1200

HSG 尺寸（边缘）

1000

800

600

400

200

0

0 10 20 30 40



流 1

流 2

流 3

流 4

流 5

流 6

流 7

50 60 70

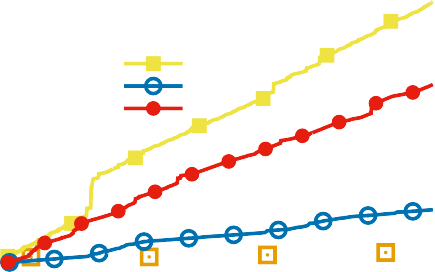
80 90 100

消费记录 (%) 消费记录 (%)

图 14.（左）：来源图表增长与消费记录。 （右）：HSG 增长与消费记录。

内存使用。 Holmes 在 8 核 CPU 上进行了测试，每个 CPU 的速度为 2.5GHz，内存为 150GB。图 15（左）显示了 Holmes 的内存消耗与审计记录的数量。由于我们的系统对内存中的审计记录进行操作，因此它显示出内存消耗几乎呈线性增长。图 15（右）显示了 Holmes 可以支持的主机数量（关于内存消耗）以及对拥有数百台主机的企业的可扩展性的推断。很明显，随着主机数量的增加，我们可以将完整出处图保存在内存中的持续时间会减少。请注意，x 轴和 y 轴都是 log-2 比例。

700



流 1

流 2

流 3

流 4

流 5

流 6

流 7

600

500

内存 (MB)

400

300

200

100

0

10 20 30 40 50 60 70 80 90 100

消费记录 (%)

4096

1024

流 1

流 2

流 3

流 4

流 5

流 6

流 7

天数

256

64

16

4

1

1

4 16 64 256

主机数量

1024

图 15.（左）：内存占用 (MB) 与消耗的记录百分比。 （右）：就内存消耗而言，Holmes 可以处理的天数与推断的主机数

运行。虽然 Holmes 在记录实时可用时使用并分析来自 Kafka 服务器的审计记录，以对其性能进行压力测试，但我们假设所有审计记录一次都可用。然后，我们测量了消耗记录、构建出处图、构建 HSG 和检测 APT 的 CPU 时间。我们将“CPU 利用率”定义为所需 CPU 时间与场景总持续时间的比率。在图 16 中，条形图显示了每个场景的 CPU 利用率，线条显示了如果 CPU 是限制因素，Holmes 可以支持多少主机（与场景相当的审计跟踪持续时间）的推断。这张图显示，我们的单个 CPU 可以支持拥有数百台主机的企业。

0.0045

0.004

0.0035

CPU 使用率

0.003

0.0025

0.002

0.0015

0.001

0.0005

0

2000

1500

外推主机数

1000

500

0

Stream 7

Stream 6

Stream 5

Stream 4

Stream 3

Stream 2

Stream 1

图 16. CPU 利用率和推断的主机可以处理的数量

Holmes 关于 CPU 时间。

# 现场实验

为了探索 Holmes 如何应对嵌入在主要良性事件流中的攻击，我们将其评估为实时检测系统。该实验历时 2 周，在此期间，Holmes 实时收集和分析了运行 Windows、Linux 或 BSD 的多个系统的审计日志。在这个实验中，一个企业模拟了一个安全关键的服务，例如一个 web 服务器、电子邮件服务器、SSH 服务器和一个 SMB 服务器，用于提供对文件的共享访问。与之前的数据集类似，在此实验期间进行了大量的正常活动，红队进行了一系列

的攻击。但是，这一次，我们预先配置了所有参数，并且事先不知道红队计划的攻击。 此外，我们有跨主机内部

连接性，这使得 APT 第 5 阶段（横向移动）成为攻击者可能采取的行动。对此

最后，我们将检测阈值设置为 2.097

*i*=1

*无线*= 2.099.8= 1378。图 17 显示了

Holmes 在本实验中构建的攻击和良性 HSG 的累积分布函数。请注意，有一些点代表了良性 HSG 的威胁评分，它们已经绕过了阈值。我们在下面将它们解释为误报，然后讨论一些潜在的误报情况。

1



攻击良性阈值

0.8

0.6

CDF

0.4

0.2

0

100 101 102 103 104 105 106

威胁评分

图 17. 攻击与良性 HSG 的累积分布函数

误报。由于系统管理员建立的 SSH 连接，我们注意到了一些误报。这些连接来自不受信任的 IP 地址，随后，Holmes 汇总了系统管理员通过 SSH 连接发出的所有操作的严重性评分。在某些情况下，威胁评分会绕过我们的阈值。解决方案是为执行身份验证的服务器（如 ssh）定义自定义标记策略，以便此类服务器的子节点不会被标记为不受信任 [1]。

为了进一步评估我们的系统是否存在误报，我们还在另外两周的良性活动期对其进行了评估。在此期间，进行了一系列不同的正常活动（包括通过软件包管理器进行软件更新和升级），并且 Holmes 没有产生误报。

根据我们的结果，考虑到它给企业带来的好处，我们声称 Holmes 的误报率是可以接受的。安全分析师可以手动检查发出的警报并消除错误构建的 HSG。

假阴性。虽然我们在实验过程中没有观察到任何假阴性，但我们在这里讨论了福尔摩斯可能错过的潜在场景。

*TTP 之间的隐含因果关系：*对于避免系统调用的信息流，Holmes 无法直接看到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以可见性展开，Holmes 仍将部分重建攻击。

*多个入口点：*作为一种主动规避技术，攻击者可能会利用多个入口点导致子图分离。福尔摩斯跟踪每一个入口点，直到

我们的检测阈值得到满足，并且当它们之间存在信息流时，将来自不相交子图的 TTP 关联起来。然而，可能需要一些额外的分析来完全关联来自不同入口点并且之间没有信息流的攻击步骤。

# 4.7 概括

我们介绍了 Holmes，这是一个实时 APT 检测系统，它关联了可能用于执行每个 APT 阶段的策略、技术和程序。 Holmes 生成一个高级图，实时总结攻击者的步骤。我们针对九种现实世界的 APT 威胁评估 Holmes，并将其部署为实时入侵检测工具。结果表明，Holmes 以高精度和低误报率成功检测了 APT 活动。

**第 5 章**

POIROT：**将攻击行为与网络威胁搜寻的内核审计记录保持一致**

*本章包括 [3] 中发表的材料的摘录和数字。*

# 介绍

一般来说，企业内部的威胁追踪存在几个挑战：

* + - *大规模搜索：*为了保持低调，攻击者通常会在很长一段时间（数周，或者在某些情况下，数月）内执行攻击步骤。因此，有必要设计一种方法，即使它们是在很长一段时间内进行的，也可以将相关的 IOC 链接在一起。为此，系统应该能够在数百万个日志事件中进行搜索（其中 99.9% 通常对应于良性活动）。
    - 与威胁相关的实体的强大识别和链接：威胁搜寻在识别攻击活动是否影响系统时必须是可靠的，即使攻击者可能已经改变了文件哈希和 IP 地址等工件以逃避检测。因此，稳健的方法不应该仅仅孤立地寻找匹配的 IOC，而是发现整个威胁场景，这对于攻击者来说更难变异。
    - *高效匹配：*为了让网络分析师及时了解威胁事件并做出反应，该方法必须有效地进行搜索并且不会产生太多

97

误报，以便及时启动适当的网络响应操作。

通常，有关 APT 活动中使用的恶意软件的知识发布在网络威胁情报 (CTI) 报告中，并以多种形式呈现，例如自然语言、结构化和半结构化形式。为了促进 IOC 形式的 CTI 的顺利交换并能够表征对抗性技术、策略和程序 (TTP)，安全社区采用了开放标准，例如 OpenIOC [45]、STIX [46] 和 MISP [47 ]。为了更好地概述攻击，这些标准通常包含描述性关系，显示指标或可观察物如何相互关联[48]。

然而，目前绝大多数威胁搜寻方法仅在网络威胁的碎片化视图中运行 [11, 12]，例如签名（例如，工件的散列）、可疑文件/进程名称和 IP 地址（域名），或者通过使用诸如时间戳之类的启发式方法来关联可疑事件 [49]。这些方法很有用，但也有局限性，例如（i）缺乏精确性来揭示威胁如何展开的完整画面，尤其是在很长一段时间（几周，或在某些情况下，几个月）内，（ii）容易受到错误信号的影响当攻击者使用看起来合法的名称（如 Windows 中的 svchost）使他们的攻击与良性系统活动无法区分时，以及 (iii) 依赖低级签名，这使得攻击者更新或重新利用它们时无效 [50,51]他们的工具或更改他们的签名（IP 地址或哈希值）以逃避检测。为了克服这些限制并建立一个强大的检测系统， 必须考虑 IOC 之间的相关性。事实上，IOC 工件之间的关系包含了关于

在受损系统内部进行攻击，这与攻击者的目标相关，因此更难改变 [52, 53]。

本章将 CTI 报告和 IOC 描述中的威胁搜寻问题形式化，开发了一种严格的方法来获得表明攻击活动成功可能性的置信度分数，并描述了一个名为 Poirot 的系统来实现这种方法。简而言之，给定基于图的 IOC 及其之间的关系表示 APT 的整体行为，我们称之为查询图，我们的方法有效地在更大的出处图中找到了该查询图的嵌入，这包含长期内核审计日志的表示。只要系统的内核没有受到损害，内核审计日志就不会受到未经授权的篡改，并且可靠地包含系统实体之间的关系（例如，进程、文件、套接字等），与其替代方案（例如防火墙、网络监控、和文件访问日志）提供部分信息。 我们假设为了维护内核审计日志的完整性，将实时内核审计存储在单独且安全的日志服务器上用作防止日志篡改的预防措施。更准确地说，我们将威胁追踪定义为一个图模式匹配 (GPM) 问题，用于搜索与查询图中描述的系统实体之间的因果依赖关系或信息流。为了对旨在影响匹配的规避攻击（例如，模仿攻击 [54, 55]）具有鲁棒性，我们根据攻击者产生的成本对流进行优先级排序。 鉴于图匹配问题 [56] 的 NP 完全性，我们提出了一种近似函数和一种新颖的相似性度量来评估

查询和出处图之间的对齐。

我们使用三个不同的数据集测试 Poirot 的有效性和效率，特别是 TC 程序 [28] 执行的红队/蓝队对抗性活动、公开可用的真实世界事件报告以及普通用户生成的无攻击活动。此外，我们在受控环境中模拟了来自真实场景的几种攻击，并将 Poirot 与当前用于进行威胁追踪的其他工具进行了比较。我们证明 Poirot 优于这些工具。我们为 Linux、FreeBSD 和 Windows 实现了不同的内核日志解析器，我们的评估结果表明，Poirot 可以在包含数百万个节点的图中搜索并在几分钟内查明攻击。

本章组织如下： Poirot 的整体架构出现在 Section

5.2.在 5.3 节中，我们提供了图对齐算法的正式细节。第 5.4 节讨论了评估，我们在第 5.5 节中得出结论。

# 方法概述

图 18 显示了我们方法的高级视图。接下来我们将简要概述 Poirot 的组件，更详细的讨论归入第 5.3 节。

# 来源图构建

Poirot 目前支持使用来自 Microsoft Windows、Linux 和 FreeBSD 的内核审计日志，并在内存中构建出处图，类似于第 3 章中描述的系统。为了支持在该图上的高效搜索，我们利用了其他方法，例如快速散列技术和用于将进程/文件名映射到唯一节点 ID 的反向索引。

错误

斯蒂克斯

内核审计日志

国际奥委会关系

攻击行为

（查询图，G）

系统行为

q

（来源图，G）

p

图形对齐 结盟

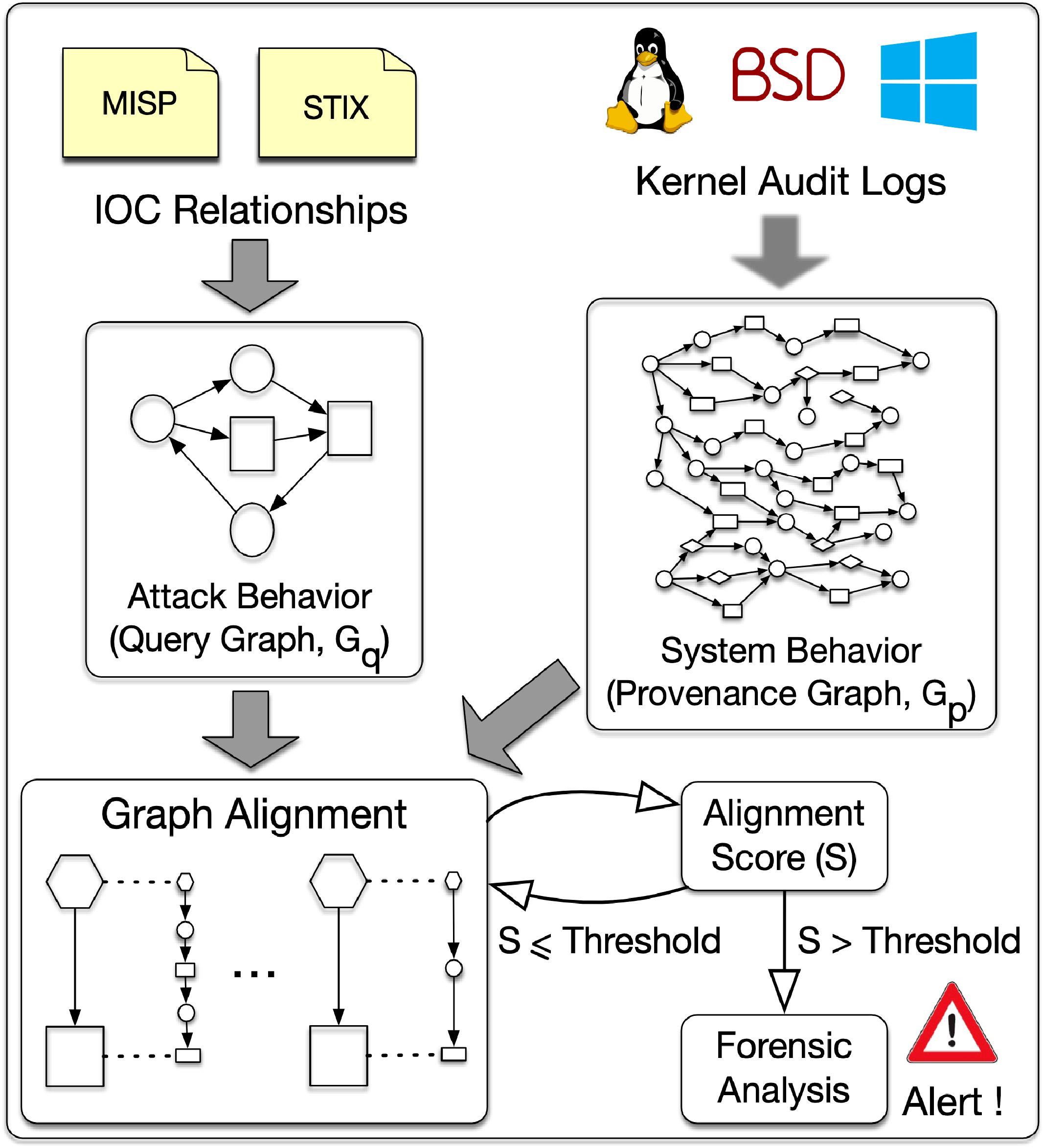
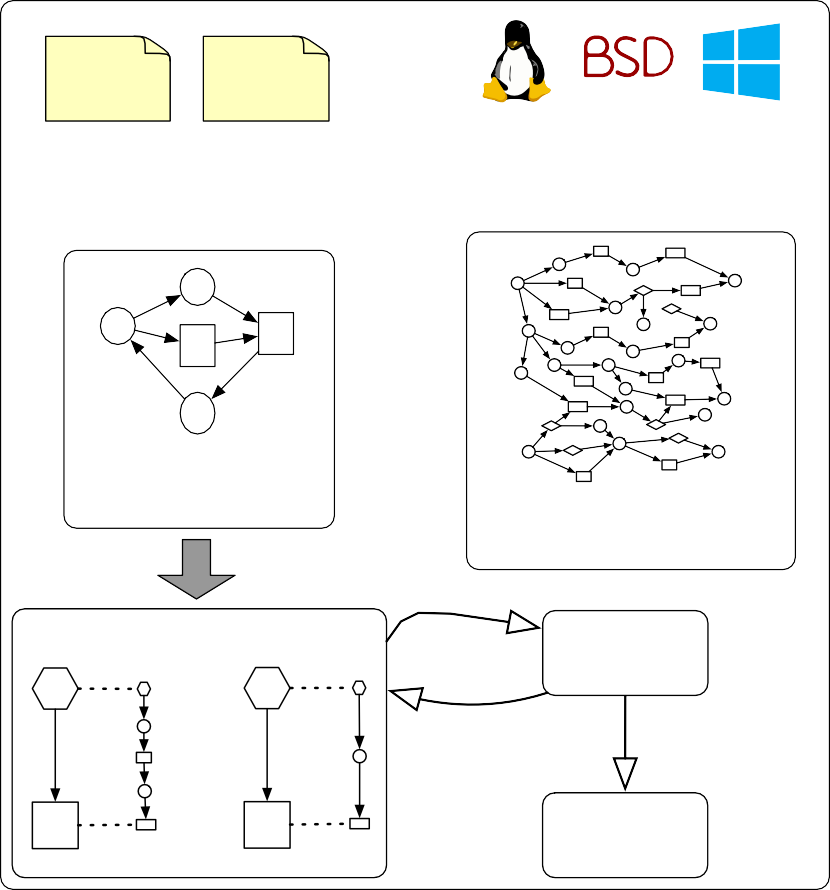


图 18. Poirot 方法概述。

分数 (S)

S < 阈值 S > 阈值

…

法医

分析

警报 ！

# 查询图构建

我们从与已知攻击相关的 CTI 报告中提取 IOC 以及它们之间的关系。这些报告出现在安全博客、行业威胁情报报告、网络威胁地下论坛以及公共和私人威胁情报源中。除自然语言外，攻击通常以结构化和半结构化的形式描述

标准格式也是如此。这些格式包括 OpenIOC [45]、STIX [46]、MISP [47] 等。本质上，这些交换格式用于描述攻击的重点、观察到的 IOC 以及它们之间的关系。例如，使用 OpenIOC，恶意软件样本的行为可以描述为工件列表，例如它打开的文件和加载的 DLL [57]。这些标准描述通常由安全操作员手动创建 [58, 59]。此外，还构建了自动化工具来自动从自然语言中提取 IOC，并补充人类操作员的工作 [60-62]。这些工具可用于执行初始特征提取以生成查询图，然后由安全专家手动改进。我们认为手动细化是查询图构建的重要组成部分，因为自动化方法通常会产生噪音并降低查询图的质量。

我们还将 CTI 报告中出现的行为建模为带标签、类型化和有向图，我们称之为查询图 (G*q*)。如果存在标准格式的描述，则查询图的创建可以很容易地自动化并由人工进一步细化。特别是，报告中出现的实体（例如，进程、文件）被转换为节点，而关系被转换为有向边[63]。查询图的节点和边可以进一步与附加信息相关联，例如标签（或名称）、类型（例如，进程、文件、套接字、管道等）和其他注释（例如，哈希值、创建时间等）取决于分析师可能认为匹配所需的信息。在当前的 Poirot 实现中，我们使用名称和类型来指定查询图中的节点和出处图中的节点之间的显式映射。



A

E

1. 执行 4.发送B
2. 写 3. 写

C D

图 19.代理狗恶意软件的查询图。 A=∗.%exe%, B=∗, C=%APPDATA%\∗, D=%HKCU%\Software\Microsoft\Windows\CurrentVersion\ Run\∗, E=%External IP address%。

作为查询图构建的示例，请考虑以下关于我们评估中使用的有关 ViceDog 恶意软件的报告 [64] 的摘录。

之上 执行， 8aba4b5184072f2a50cbc5ecfe326701 写 “28542CC0.dll” 至 这个

地点：

“C:\文档 和 设置\全部 用户\应用程序 数据\28542CC0.dll”。

在 命令 至 维持 坚持， 这 原来的 恶意软件 添加 这个 登记处 钥匙：

“%HKCU%\Software\Microsoft\Windows\CurrentVersion\Run\28542CC0”。

然后连接到韩国的主机 (180.150.228.102)。

恶意软件

摘录提到了几个执行它们的动作和实体，并且很容易被安全分析师转换成图表。例如，第一句话清楚地表示了一个写入文件的进程（在执行时，恶意软件将文件写入一个位置）。我们指出，此摘录中的详细程度在大多数 CTI 报告中都很常见，并且可以由合格的网络分析师转换为可靠的查询图。特别是，表达主语动作的动词通常可以很容易地映射到读/写

从/到磁盘或网络以及进程之间的交互（例如，浏览器下载文件，进程产生另一个进程，用户单击鱼叉式网络钓鱼链接等）。

图 19 显示了对应于上述摘录的查询图。椭圆形、菱形、矩形和五边形分别代表进程、套接字、文件和注册表项。

在图 19 中，节点 B 表示恶意软件进程或进程组（我们使用 \* 表示

它可以有任何名称），节点 A 代表恶意软件的图像文件，而节点 C、D 和 E 分别代表一个丢弃的文件、一个注册表和一个 Internet 位置。我们在这一点上强调，所构建的查询图仅包含有关特定实体之间的信息流的信息，因为它们出现在报告中（进程、文件、IP 地址等），并不旨在成为所有恶意软件的精确子图。攻击期间实际出现的实体。从某种意义上说，查询图是对实际攻击图的总结。在我们的实验中，我们获得的查询图通常很小，包含 10-40 个节点和多达 150 个边。

# 图形对齐

最后，我们将威胁搜寻建模为确定攻击的查询图 G*q*是否在出处图 G*p*中“表现”自身。我们将此问题称为图对齐问题。

在这一点上，我们注意到 G*q*表示实体之间的几个高级流（进程到文件等）。相比之下，G*p*表达了系统的完整低级活动。因此，G*q*中的一条边可能对应于 G*p*中由多条边组成的路径。例如，如果 G*q*表示受感染的浏览器正在写入系统文件，在 G*p*中，这可能

对应于代表 Firefox 进程的节点派生新进程的路径，其中只有一个最终写入系统文件。通常，攻击者可能会在他们的活动中添加噪音以逃避检测，从而创建这种对应关系。因此，我们需要一种图对齐技术，可以将 G*q*中的单个边与 G*p*中的路径匹配。这一要求在我们的算法设计中至关重要。

在图论文献中，存在多个版本的图匹配问题。在精确匹配中，嵌入在更大图 G*p*中的子图必须与 G*q*[65] 同构。相比之下，在图模式匹配（GPM）问题中，放宽了精确匹配的一些限制以提取更多有用的子图。然而，在一般情况下，这两个问题都是 NPcomplete [56]。尽管存在大量致力于 GPM 的工作 [66-72]，但许多工作存在局限性，使其无法部署在威胁追踪领域。具体来说，它们 (i) 不是为每个节点分配了标签和类型的有向图而设计的，(ii) 不能扩展到数百万个节点，或者 (iii) 旨在彻底对齐查询图中的所有节点或边。此外，这些方法不适用于威胁搜寻的背景， 考虑到试图利用潜在匹配标准的知识保持隐身的躲避对手。由于这些考虑，我们设计了一种新颖的图形模式匹配技术来解决这些限制。

在图 18 中，图形节点以不同的形状表示以对不同的节点类型进行建模，例如文件、进程和套​​接字，但是为简洁起见，省略了标签。特别是，Poirot 首先找到所有可能的候选对齐集合 i : j 其中 i 和 j 代表

节点分别位于 V (G*q*) 和 V (G*p*) 中。然后，从找到匹配的可能性最高的对齐开始，称为种子节点，我们扩展搜索以找到更多的节点对齐。种子节点在图 18 中由六边形表示，而两个图中的匹配节点由虚线连接。为了找到与 CTI 关系中表示的攻击相对应的对齐方式，沿着更有可能受到攻击者影响的路径扩展搜索。为了估计这种可能性，我们设计了一个名为影响分数的新指标。使用这个指标可以让我们在很大程度上从搜索中排除不相关的路径，并有效地缓解依赖爆炸问题。先前的工作还提出了基于计算为长度 [70, 71] 或成本 [1] 的分数来对流进行优先级排序的方法。然而，它们可能会被攻击 [54, 55] 击败，其中攻击者经常改变他们的方式来逃避检测技术。例如， 基于邻近度的图匹配方法 [70,71] 可能很容易被攻击者规避，他们知道底层系统和匹配方法，可能会生成一长串 fork 命令来影响基于邻近度的图匹配的精度。相比之下，我们的分数定义明确考虑了潜在攻击者的影响。特别是，我们增加了攻击者逃避我们检测的成本，方法是根据攻击者产生流的努力来确定流的优先级。我们对对齐的搜索使用了这样的优先流，并在第 5.3 节中进行了描述。

在找到对齐 G*q*:: G*p*后，计算一个分数，表示 G*q*与 G*p*的对齐子图之间的相似性。当分数高于阈值时，Poirot 会发出警报，宣布发生了攻击，并将对齐节点的报告提交给系统分析人员以进行进一步的取证分析。否则，波洛从

表十七

符号。

|  |  |
| --- | --- |
| **符号** | **描述** |
| *i* : *k* | 节点对齐。节点 i 与节点 k 对齐（i 和 k 在两个不同的图中）。 |
| *i j* | 流动。从节点 i 开始到节点 j 结束的路径。 |
| *i*标签  ***−−*** *→ j* | 从节点 i 到节点 j 的具有特定标签的边。 |
| *Gq* :: *Gp* | 图形对齐。一组节点对齐 i : k 其中 i 是 Gq 的节点， k 是 Gp 的节点。 |
| *V* (*G*) | 图 G 中所有顶点的集合。 |
| *E*(*G*) | 图 G 中所有边的集合。 |
| *F* (*G*) | 图 G 中所有流 i j 的集合，使得 i /= j。 |

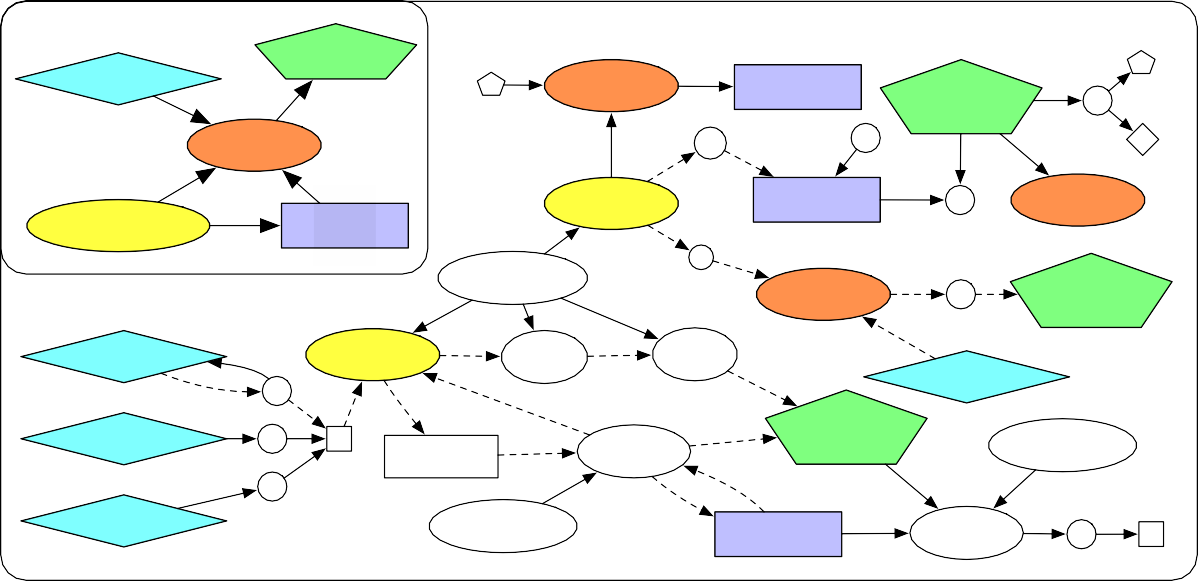
下一个种子节点候选。在 G*p*中找到攻击子图后，Poirot 生成一份报告，其中包含对齐的节点、它们之间的信息流以及相应的时间戳。在企业环境中，这种视觉紧凑且语义丰富的报告为网络分析师提供了可操作的情报，以计划和执行网络威胁响应。我们将在 5.3 节中讨论我们方法的细节。

# 算法

在本节中，我们将讨论 G*q*和 G*p*之间对齐的主要方法，方法是 (a) 定义对齐度量来衡量图对齐的正确程度，以及 (b) 设计基于尽力而为的相似性搜索关于特定领域的特征。

# 对齐指标

我们引入了一些符号（在表 XVII 中），其中我们定义了两种对齐方式，即两个不同图中的两个节点之间的节点对齐，以及作为一组节点对齐的图对齐。通常，当两个节点 i 和 j 表示同一实体时，它们处于节点对齐状态，例如，表示 CTI 报告中提到的常用浏览器的节点（图 20 的查询图 G*q*中的节点 %browser%）和一个节点代表



Gq:

外部 IP

%注册表%

阀芯v1

命令文件

%注册表%

\使用权

阀芯

8

8

%浏览器%

\*。exe

火狐1

tmp.exe

阀芯v2

7

Gp:

发射器1

7

阀芯v3

9

6

9

%注册表%

\跑

240.1.1.1

火狐2

1

2

爪哇

2

1

爪哇2

2

240.2.1.1

1

1

3

4

240.1.1.2

5

tmp.doc

3

字1

3

4

%注册表%

\火狐

发射器

3

5

240.1.1.3

发射器2

字.exe

字2



240.1.1.1

1

**(Gq::Gp)1**

火狐2

4

3

2

5

%注册表%

\火狐

字.exe



240.2.1.1

6

**(Gq::Gp)2**

7

火狐1

阀芯v3

9

8

%注册表%

\跑

tmp.exe

图 20. 简化的出处图 (G*p*、查询图 (G*q*和两个示例图对齐 (G*q*:: G*p*。节点类型以不同的形状显示，每个节点的可能对齐方式以相同的颜色显示。边缘上的数字仅用于说明可能的路径/流程，没有其他含义。

出处图中的 Firefox 进程。我们注意到，一般来说，节点对齐关系是从 V(G*q*到 V(G*p*的多对多关系，其中 V(G*q*和 V(G*p*) 分别是 G*q*和 G*p*的顶点集。因此，给定一个查询图 G*q*，可能有

G*q*和 G*p*的许多子图之间的大量图对齐。另外需要指出的是，这些图对齐中的每一个都可以对应于 G*p*的不同子图。这些子图中的每一个都包含与 G*q*的节点对齐的节点；但是，它们可能包含这些节点之间的不同路径。在这些子图中，我们有兴趣找到与图 G*q*最匹配的子图。

基于这些定义，问题是在一组候选图对齐中找到最佳的图对齐。为了说明这个问题，考虑查询和出处图 G*q*和 G*p*，以及图 20 中两个可能的对齐图，其中节点形状表示实体类型（例如，进程、文件、套接字）和边表示节点之间的信息流（例如，读、写、IPC）和因果依赖关系（例如，分叉、克隆）。 G*p*边缘上显示的数字不是出处图的一部分，但在我们的讨论中用于识别单个路径。此外，由这两个与 G*q*的图对齐确定的 G*p*的子图由 G*p*中的虚线边表示。 G*p*中的每个流和 G*q*中的相应边都用相同的数字标记。因此，问题是要决定在许多比对中哪一个是最佳候选。直观地说，对于这个特定的数字，对齐 (G*q*:: G*p*2比 (G*q*:: G*p*1更接近 G*q*， 主要是因为它的对齐节点数高于(G*q*:: G*p*)1，最重要的是它的流与查询图G*q*的边有更好的对应关系.

# 影响力分数

在将上述直觉形式化之前，我们必须引入一个路径评分函数，我们称之为影响评分，它为两个节点之间的给定流分配一个数字。

该分数将有助于定义图形对齐的“优点”。在实践中，影响分数表示攻击者可以产生流的可能性。为了显示

这个概念，请考虑图 20 中图形 G*p*中的两个节点 firefox2和 %registry%\firefox。

从 firefox2到 %registry%\firefox 存在两个流，一个由标记的边表示

编号为 2（并通过节点 java1和 java2），另一个由标记为 3、3 和 5 的边表示（并通过节点 tmp.doc 和 word1）。假设 firefox2处于攻击者的控制之下，攻击者更有可能执行第一个流而不是第二个流。事实上，为了执行第二个流程，攻击者必须控制进程启动器2或 word1以及 firefox2。由于 launcher2或 word1在进程树中与 firefox2没有共同的祖先，因此这种接管将不得不涉及对 launcher2或 word1的额外利用，这比只需执行第一个流程，其中所有进程共享一个共同的祖先启动器1。我们指出，这种可能性不取决于流的长度，而不是关于该流中的进程数量以及这些进程在进程树中共享的不同祖先的数量。事实上，一个人可以 想象一个长链的分叉进程，但是它们都在攻击者的控制之下，因为它们在进程树中都有一个共同的祖先，即链的第一个进程。野外存在的另一种可能的攻击场景涉及从单个受感染进程远程加载代码，其中所有具有恶意功能的代码都加载到主内存中，并且同一进程（例如，firefox）代表攻击者执行所有操作。虽然这种技术不会在文件系统上留下任何痕迹

并且可能会避开一些检测工具，Poirot 将能够检测到这种攻击。事实上，影响力分数保持微乎其微的不变。

另一个需要注意的重要一点是，这种衡量攻击者潜在影响的概念对于攻击者的规避方法非常稳健。攻击者可能用来增加噪音并试图逃避检测的每一项活动都可能具有相同的共同祖先，即攻击的初始妥协点，除非攻击者支付更高的成本来执行更多不同的妥协。因此，这样的努力在改变路径的影响分数方面是无效的。

基于这些观察，我们定义了节点 i 和节点之间的影响分数 Γ*i,j*

节点 j 如下：

Γ*i,j* =

1

*和 j AC分钟（i --➔ j）*

****0 否则

****max ∃i --➔ j |交流电

*分钟*

(*i* --➔ *j*) *≤ AC*

*thr*

(5.1)

在上面的等式中，我们使用了 4.4.4 中定义的最小祖先覆盖 (AC*分钟*i --➔ j)) 的概念，它表示攻击者必须进行的不同的、独立的妥协的最小数量才能能够生成流 i --➔ j。该值捕获了攻击者对流的控制程度，并根据流中存在的进程的最小共同祖先数进行计算。例如，如果存在从节点 i 到节点 j 的流，并且该流中涉及的所有进程在进程树中都有一个共同的祖先，则攻击者只需破坏该共同的祖先进程即可启动该流，并且因此 AC*分钟*i --➔ j) 等于 1。注意如果一个节点 i 代表一个进程

并且节点 j 是 i 的子进程，那么 AC*分钟*(i --➔ j) 将等于 1，因为 i 是 j 的父进程。如果共同祖先的数量大于一个（例如，路径 firefox2→ tmp.doc → word1→ %registry%\firefox 中有两个祖先），攻击者必须至少妥协（不相关的）独立处理；因此攻击者更难构建这样的流。例如，攻击者控制流 firefox2→ tmp.doc → word 1 →

%registry%\firefox ), (s)他需要同时控制启动器1和启动器2;因此 AC*分钟*等于 2。

我们还合理地假设攻击者破坏具有不同漏洞的少数进程是不切实际的。在绝大多数记录在案的 APT 中，攻击者通常只有一个入口点或极少数入口点进入系统，例如鱼叉式网络钓鱼电子邮件或浏览器上的路过式下载攻击。基于对大量关于 APT [5] 的白皮书的审查，我们已经确认这也是正确的。一旦攻击者进行了初步妥协，他们就不太可能投入额外的资源来发现和利用额外的入口点。因此，我们可以在 AC*分钟*i --➔ j) 值上设置一个限制 AC*thr*，并合理地假设两个节点之间的任何流 AC*分钟*i --➔ j) 大于AC*thr*不能由攻击者发起。虽然 AC*分钟*i --➔ j) 的值表示攻击者控制特定路径的难度， 影响分数表示攻击者控制该路径的难易程度，它被定义为与 AC*分钟*(i --➔ j) 成反比的值。如果有超过

两个节点 i 和 j 之间的一次流量 AC*分钟*小于或等于 AC*thr*，影响分数

1

将是最大值

*AC分钟（i --➔ j）*

在所有这些流量上。因此，如果存在一条路径

AC*分钟*i --➔ j) 从 i 到 j 小于或等于 AC*thr*，影响分数将与 4.4.4 中定义的路径因子相反；否则，影响分数将等于 0。鉴于，当存在 AC*分钟*(i --➔ j) 等于 1 且最小的流时，Γ*i,j*的值最大（等于 1） (等于 0) 当没有 AC*分钟*i --➔ j) 小于或等于 AC*thr*的流量时。

# 对齐分数

我们现在准备定义一个度量，它指定图形对齐 G*q*:: G*p*的分数。基于影响分数的概念，我们定义了评分函数 S(G*q*:: G*p*)，表示对齐 G*q*:: G*p*的分数如下：

*S(Gq*

1

:: G*p*) =

Γ*k,l*

*| i* : *k* & *j* : *l* (5.2)

*|F (Gq)|(i j)∈F (G )*

*q*

在公式 5.2 中，节点 i 和 j 是 V (G*q*的成员，节点 k 和 l 是 V (G*p*) 的成员。流 i --➔ j 是在 G*q*上定义的流。特别是，该公式首先计算所有节点对 (k, l) 之间的影响分数之和，其中至少有一条来自 k 的路径

到图 G*p*中的 l，使得 k 与 i 对齐并且 l 与 j 对齐。这笔款项是下一个

通过将其除以该总和可能的最大值进行归一化。事实上，|F(G*q*|是 G*q*中的流数。由于两个节点之间的影响分数的最大值等于1，则流数自动表示影响分数之和的最大值。

由式5.2，直观上看，S(G*q*:: G*p*的值越大，节点对齐的次数越多，G*q*中的流与G*p*中的流的相似度也越大）， 哪个是

可能受到潜在攻击者的影响。特别是，S(G*q*:: G*p*的值介于 0 和 1 之间。当 S(G*q*:: G*p*= 0 时，要么找不到正确的节点对齐方式在 V (G*q*) 中，或者在 G*p*中的对齐节点之间没有出现与 G*q*相似的流。相反，当 S(G*q*:: G*p*= 1 时，G*q*中的所有节点都与 G*p*中的节点对齐，并且 G*q*中存在的所有流也出现在 G*p*中对齐的节点之间，并且它们的影响分数都等于 1，即它们很可能在攻击者的控制之下。

最后，当对齐分数 S(G*q*::G*p*) 绕过预定阈值时

(τ)，我们发出警报。要确定此阈值的最佳值，请回忆 AC*thr*是我们假设攻击者愿意独立利用的不同入口点进程的最大数量。因此，假设攻击者能够影响任何信息

*影响力得分为*1*的流量*

*AC*

*thr*

或更高。另一方面，S(G*q*:: G*p*) 是

所有影响力分数。因此，我们将阈值 τ 定义如下：

*S(Gq :: Gp) ≥ τ* (5.3)

1

*τ* =

*乙酰胆碱*

(5.4)

如果 S(G*q*:: G*p*) 绕过 τ ，我们声明匹配并发出警报。

# 尽力相似性搜索

在定义了对齐分数之后，我们描述了我们的程序来搜索最大化该分数的对齐。特别是，给定查询图 G*q*，我们需要搜索一个非常大的出处图 G*p*，以根据公式 5.2 找到具有最高对齐分数的对齐 G*q*:: G*p*。

这样做的第一个挑战是 G*p*的大小，它可以达到数百万个节点和边缘。因此，存储 G*p*的所有节点对之间的影响分数是不切实际的。我们需要按需执行图遍历，以找到节点之间的影响分数，甚至找出两个节点之间是否存在路径。此外，我们假设所有分析都是在 G*p*的静态快照上完成的，并且从搜索开始到终止，其节点或边缘没有发生任何变化。

我们的搜索算法由以下四个步骤组成，其中重复步骤 2-4，直到找到得分高于阈值 τ 的对齐（公式 5.4）。

步骤 1. 查找所有候选节点对齐：我们首先在 G*p*的节点中搜索，以找到 G*q*中每个节点的候选对齐。这些候选对齐是根据查询图节点上的名称、类型和注释来选择的。例如，在 G*q*和 G*p*中出现的具有相同标签（例如，Firefox）的相同类型的节点（例如，两个进程节点）可能会形成候选对齐，其标签与正则表达式匹配的节点（例如，文件系统路径和文件名）等。用户还可以手动注释出处图中的节点，并明确指定与查询图中的节点的对齐方式。一般而言，G*q*中的一个节点在 G*p*中可能有任意数量的可能对齐，包括 0。请注意，在这第一步中，我们没有足够的关于路径和流的信息，并且正在孤立地查看节点.在图 20 中，候选节点对齐由具有相同颜色的节点对表示。

步骤 2. 选择种子节点：为了找到足够好的对齐 G*q*:: G*p*，我们需要通过执行图来探索步骤 1 中找到的候选对齐之间的连接

G*p*上的遍历。然而，由于 G*p*的结构和大尺寸，从 G*p*中随机对齐的节点开始一组图遍历可能会导致代价高昂且徒劳的搜索。为了确定一个好的起点，一个关键的观察是攻击活动通常只包含 G*p*的一小部分，而良性活动通常会重复多次。因此，特定于攻击的工件比良性活动的工件更有可能具有更少的对齐。基于这一观察，我们按照与每个节点相关的候选对齐数量的递增顺序对 G*p*的节点进行排序。我们首先选择对齐最少的种子节点。例如，对于图 20 中的示例，种子节点将是 %browser%，因为它具有最少数量的候选节点对齐。如果有相同数量的候选对齐的种子节点，我们随机选择其中一个。

步骤 3. 扩展搜索：在这一步中，从步骤 2 选择的种子节点开始，我们遍历 G*p*中与其对齐的所有节点，并启动一组图遍历，向前或向后，以找到找出我们是否可以到达步骤 1 中找到的其他对齐节点。例如，在选择节点 %browser% 作为种子节点后，我们从与 % 对齐的 G*p*中的节点开始一系列正向和反向图遍历browser%，即firefox1和firefox2。理论上，这些图遍历可能非常昂贵，因为图的大小以及候选对齐节点的数量，这些节点可以位于图中的任何位置。然而，在实践中，一旦种子节点和该路径中最后一个节点之间的影响分数达到 0，我们就可以停止沿路径扩展搜索。例如，假设我们决定图 20 中的 AC*thr*等于 2 . 然后，沿着搜索

*路径 (firefox*2*→ tmp.doc → word*1*→ %registry%\firefox → word*2*) 不会扩展超过*

节点 word2，因为 firefox2和沿该路径的任何节点之间的 AC*分钟*在 word2处变得大于 2，因此影响分数变为 0。注意还有一个额外的路径

通过 %registry%\firefox 从 firefox2到 word2并沿着这条路径，firefox2之间的 AC*分钟*

并且 word2仍然是 2。因此，由于这条路径，搜索将继续经过 word2。在图遍历中使用影响分数作为上限可以显着降低搜索复杂度，并能够快速探索图 G*p*。

根据查询图 G*q*的形状，可能需要多个前向/后向跟踪循环来访问所有节点（例如，如果我们在示例中选择 %browser% 作为种子节点，则节点 240.2.1.1 在G*p*在 firefox1或 firefox2) 开始只有一次向前或向后遍历是无法到达的。在这种情况下，我们从与未访问节点相邻但在先前遍历中已访问的节点开始重复向后和向前遍历（例如，我们示例中的节点 spoolsv3）。我们重复这个过程，直到我们覆盖查询图 G*q*的所有节点。

步骤 4. 图形对齐选择：此步骤负责产生最终结果或从步骤 2 开始另一个搜索迭代，以防找不到结果。特别是，在执行后向/前向遍历之后，我们为 G*q*中的每个节点识别 G*p*中的候选节点子集。例如，对于我们的示例，我们发现 node%browser% 有候选 firefox1和 firefox2，节点 External IP 有候选对齐 240.1.1.1、240.1.1.2、240.1.1.3 和 240.2.1.1 ， 等等。然而，这些候选节点可以形成的可能的候选图对齐的数量可能非常大。如果 G*q*中的每个节点 i 有 n*i*个候选对齐，那么可能的图对齐的数量是

等于rr .人力资源管理厅n*i*。例如，在我们的示例中，我们可以有 216 个可能的图形对齐方式（2 × 3 × 3 × 3 × 4）。在这一步中，我们搜索使对齐分数最大化的图对齐（公式 5.2）。

*i*

一种简单的方法是计算所有可能的图形对齐的对齐分数的蛮力方法。但是，这种方法效率非常低，并且没有充分利用领域知识。为了有效地执行此搜索，我们设计了一个程序，该程序基于一个近似函数迭代地为 G*q*中的每个节点选择最佳候选者，该近似函数测量每个对齐对最终对齐分数的最大贡献。

特别是，从 G*q*中的一个种子节点开始，我们选择 G*p*中对对齐分数贡献最大的节点并修复这个节点对齐（我们在下一段中讨论选择函数）。例如，在我们的示例中，从种子节点 %Browser% 开始，我们使用节点 firefox1修复对齐。从这个固定的节点对齐中，我们沿着 G*q*中的边来修复连接到种子节点的其他节点的对齐。为这些节点中的每一个选择的特定节点对齐是对对齐分数的贡献最大化的节点对齐。例如，从节点 %Browser%（与 firefox1对齐），我们可以继续

node ∗.exe 并修复其与 cmd.exe、tmp.exe 和 Word.exe 中的一个节点的对齐，

使得对对齐分数的贡献最大化。

选择功能。选择函数背后的关键直觉是选择和修复许多节点对齐中的一个，是估计每个对齐对最终对齐分数的贡献程度，并选择贡献最大的一个。对于 G*p*中给定的候选对齐节点 k，此贡献计算为 max-

该节点与 G*p*中所有其他候选节点 l 之间的最大影响分数：1) 可从 k 到达或具有到 k 的路径，以及 2) G*q*中其对应的对齐节点 j 具有流从/到 G*q*中与节点 k 对应的节点。例如，在我们的示例中考虑节点 %Browser% 和两个候选对齐节点 firefox1和 firefox2。为了确定 firefox1的贡献，我们测量每个流 (%Browser% --➔

*∗.exe, %Browser% --➔ spoolsv, %Browser% --➔ %registry% ) 从/到 Gq中的 %Browser%，*

firefox1和与 \*.exe、spoolsv 对齐的候选节点之间的最大影响分数，

和%registry%，分别。换句话说，我们计算 firefox1和 \*.exe 的每个节点对齐候选之间的最大影响分数，最大影响分数

在 firefox1和 spoolsv 的每个节点对齐候选之间，以及最大 in-

firefox1和 %registry% 的每个节点对齐候选之间的 fluence score。这三个最大值中的每一个都分别为 ∗.exe、spoolsv 和 %registry% 提供了对每个可能的未来对齐（尚未修复）的对齐分数的最大贡献。接下来，我们将这三个最大值相加，以获得 firefox1对对齐分数的最大贡献。我们对 firefox2重复相同的过程，最后选择贡献值最高的对齐。该贡献由以下公式正式计算，该公式近似于 A(i:k) 节点对齐 i:k 的贡献。

*A*(*i* : *k*)

= 1*{j:l} × Γk,l* + (1 − 1*{j:l}*) ×m∈

最大限度

*候选人(j)*

(Γ*k,m*)

(5.5)

*j*:(*i*---+*j*)*∈F* (*Gq* )

+ 1{j:l} *× Γ*l,k + (1 − 1{j:l}) ×m∈

最大限度

*候选人(j)*

(Γ*m,k*)

*j:*(j---+i​​)∈F (Gq )

其中 1*A*是指示函数，如果 A 中表示的对齐方式是固定的，则为 1，否则为 0。换句话说，如果节点 j 和 l 之间的对齐已经固定，则 l*{j:l}*等于 1，否则，如果节点 j 尚未与任何节点对齐，则 1*{j:l}*等于 0。注意 1*{j:l}*和(1 − 1*{j:l}*)是互斥的，在任何时候，只有一个等于1，另一个等于0。

我们注意到，第一次求和是对来自节点 i 的流出流执行的，而第二次求和是对传入节点 i 的流执行的。在每个求和中，第一项表示固定对齐，而第二项表示尚未固定的潜在对齐中的最大值，如上所述。

最后，对于每个节点 i 具有由步骤 3 生成的一组 K 候选对齐，选择函数，它固定 i 的对齐如下：

参数最大 A(i : k) (5.6)

*k∈K*

等式 5.5 和 5.6 背后的直觉是，一旦节点对齐固定，该节点的其他可能对齐将被算法的未来步骤忽略，并且与该对齐相关的最大影响分数的计算减少到

表查找而不是对候选节点对齐的迭代。特别是，搜索以蛮力搜索开始，但随着越来越多的节点对齐被固定，通过重用存储在表中的先前搜索的结果，搜索变得更快。使用公式 公式 5.5 和公式 5.6 可以显着加快确定正确图形对齐的速度。虽然从理论上讲，这代表了一种贪婪的方法，可能并不总是能带来最好的结果，但在实践中，我们发现它非常有效。

最后，在修复所有节点对齐后，对齐分数的计算如公式 5.2 所示。如果分数低于阈值，则再次执行步骤 2-4。我们在 5.4 节中的评估结果表明，攻击图通常在前几次迭代中找到。

# 评估

我们在三个不同的实验中评估了 Poirot 的有效性和可靠性。在第一个实验中，我们使用了一组 TC 程序红队与蓝队对抗性参与场景，这些场景设置在模拟企业网络的隔离网络中。在第二个实验中，我们在自然语言描述在互联网上公开可用的现实世界事件上进一步测试 Poirot。为了重现公开威胁报告中描述的攻击，我们在受控环境中获取并执行了它们的二进制样本，并收集了我们从中构建出处图的内核审计日志。在第三个实验中，我们评估了 Poirot 在无攻击数据集中对错误信号的鲁棒性。

在所有实验中，我们将 AC*thr*的值设置为 3（因此阈值为1）。该选择在第 5.4.3 节中得到验证。然而，我们注意到，根据对系统保护机制的信心，可以将 Poirot 配置为更高或更低的值

3

或者网络分析人员愿意花在检查警报上的努力。事实上，AC*thr*的值会影响误报和潜在误报的数量。较高的 AC*thr*会增加误报的数量，而较低的 AC*thr*会减少它。另一方面，较高的 AC*thr*值可能会检测到具有多个初始入口点的复杂攻击，而较小的值可能会错过它们。在找到与绕过阈值的分数对齐后，我们手动分析了所有匹配的攻击子图，以确认它们正确地查明了查询图中存在的实际攻击。

# 对 TC 数据集的评估

表十八

查询图的特征。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **设想** | **受试者 ∈ |V (Gq )|** | **对象 ∈ |V (Gq )|** | ***|****E*(*Gq* )***|*** | ***|****F* (*Gq* )***|*** |
| BSD-1 | 4 | 9 | 19 | 81 |
| BSD-2 | 1 | 7 | 10 | 32 |
| BSD-3 | 3 | 18 | 34 | 159 |
| BSD-4 | 2 | 8 | 13 | 43 |
| 赢一 | 13 | 8 | 26 | 149 |
| 双赢2 | 1 | 13 | 19 | 94 |
| Linux-1 | 2 | 9 | 19 | 62 |
| Linux-2 | 5 | 12 | 24 | 112 |
| Linux-3 | 2 | 8 | 22 | 48 |
| Linux-4 | 4 | 11 | 22 | 96 |

该实验是在 TC 计划发布的数据集上进行的，该数据集是在 2018 年 4 月红队与蓝队对抗期间生成的 [28]。在参与中，设置了不同的服务，包括 Web 服务器、SSH 服务器、电子邮件服务器和 SMB 服务器。模拟了大量的良性活动，包括系统

管理任务、浏览许多网站、下载、编译和安装多种工具。红队依靠威胁描述来执行这些攻击。我们获得了这些威胁描述，并使用它们为每个场景提取了一个查询图（总结见表 XVIII）。

总的来说，我们在十个攻击场景中评估了 Poirot，其中四个在 BSD 上，两个在 Windows 上，四个在 Linux 上。由于篇幅限制，我们无法展示所有的查询图；但是，它们的特性在表 XVIII 中进行了描述，其中主题表示进程，对象表示文件、内存对象和套接字。BSD-1-4 涉及对运行 Nginx 后门版本的 FreeBSD 11.0（64 位）网络服务器进行的攻击。Win-1&2 涉及在运行 Windows 7 Pro（64 位）的主机上进行的攻击。Win-1 场景包含带有恶意 Excel 宏附件的网络钓鱼电子邮件，而 Win-2 场景包含对 Firefox 浏览器易受攻击版本的利用。Linux1&2 和 Linux3&4 分别涉及对运行 Ubuntu 12.04（64 位）和 Ubuntu 14.04（64 位）的主机进行的攻击。Linux1&3 包含内存浏览器漏洞，而 Linux2&4 涉及使用恶意浏览器扩展的用户。

对齐分数。如第 5.3.2 节所述，Poirot 从候选节点较少的种子节点开始迭代地重复节点对齐过程。图 21 显示了 G*q*的每个节点的候选对齐节点的数量。G*q*的大部分节点在G*p*中的候选节点少于10个，同时也有一些节点有数千个候选节点。这些出现数千次的节点通常是无处不在的进程和文件，通常由良性活动（例如 Firefox 或 Thunderbird）访问。我们提醒读者，我们的

1 1

BSD-1 BSD-2 BSD-3 BSD-4

赢 1 赢 2

|V(Gq)|的百分比(CDF)

|V(Gq)|的百分比(CDF)

0.8 0.8

0.6 0.6

0.4

0.2

0

1 10 100 1000

候选人人数

1

Linux-1 Linux-2 Linux-3 Linux-4

|V(Gq)|的百分比(CDF)

0.4

0.2

0

1 10 100 1000

候选人人数

0.8

0.6

0.4

0.2

0

1 10 100 1000

候选人人数

图 21. |G*p*| 中候选者数量的累积分布函数 (CDF)对于 |G*q*| 的每个节点。从左到右：BSD、Windows 和 Linux 场景。

首先从对齐较少的节点中选择种子节点。在每次迭代中，构造一个对齐，并将其对齐分数与阈值进行比较，阈值设置为1

3

表 XIX 显示了在将查询图与相应的出处图对齐后，Poirot 对每个 TC 场景的匹配结果。我们停止搜索后

超过阈值的第一个对齐。表 XIX 的第二和第三列显示了第 5.3.2 节中介绍的步骤 2-4 的迭代次数以及绕过阈值的第一次对齐获得的实际分数。在 10 个场景中的 9 个场景中，在第一次迭代中发现了绕过阈值的对齐。在一种情况下，G*q*的精确匹配可以在 G*p*中找到（参见 BSD-4）。

表 XIX 的第四列显示了通过将我们的搜索算法的步骤 2-4 迭代 20 次构建的 20 个对齐中的最大对齐分数，而最后一列显示了导致最大值的最早迭代次数。可以看出，平均而言，我们的搜索很快收敛到一个完美的解决方案。在 10 个场景中的 7 个场景中，最大对齐分数在第一次迭代中计算，而在其他 3 个场景中，最大对齐分数在第四次或第五次迭代中计算。后者是由于攻击报告和红队的攻击实施之间存在细微差异，导致查询图和出处图之间的信息流和因果依赖关系略有不同。例如，在图 22 中，我们展示了查询图及其

表十九

Poirot 的图形对齐分数。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **设想** | **最早的迭代**  **绕过阈值** | **最早的分数**  **绕过阈值** | **20分最高分**  **迭代** | **最早的迭代**  **结果最高分数** |
| BSD-1 | 1 | 0.45 | 0.64 | 5 |
| BSD-2 | 1 | 0.81 | 0.81 | 1 |
| BSD-3 | 1 | 0.89 | 0.89 | 1 |
| BSD-4 | 1 | 1 | 1 | 1 |
| 赢一 | 1 | 0.63 | 0.63 | 1 |
| 双赢2 | 1 | 0.47 | 0.63 | 4 |
| Linux-1 | 1 | 0.58 | 0.58 | 1 |
| Linux-2 | 2 | 0.55 | 0.71 | 5 |
| Linux-3 | 1 | 0.54 | 0.54 | 1 |
| Linux-4 | 1 | 0.87 | 0.87 | 1 |

|  |  |  |
| --- | --- | --- |
|  | H | |
| /tmp/memtrace.so | |  |

图 22. 场景查询图：Linux-2（顶部）及其检测到的对齐（底部）。



A

发送

%浏览器%

.\*

发送

收视率

发送

收视率

\*/主机/pass\_mgr

写

C

\*gtcache\*

收视率

B

%外部 IP 地址%

叉子

23.194.99.39

145.199.103.57

E

L

%外部 IP

地址％

F

/etc/passwd

发送

收视率

/proc/\*/stat

读

读

读

/etc/组

执行

读

K

\*邮件\*

G

叉子

\*简介\*

.\*

收视率

执行

写写

\*gtcache\*

写

D

发送

M

%内网IP

地址％

B

61.130.69.232

火狐

A

L

149.52.198.23

E

F

/etc/passwd

/proc/30691/统计

K

/var/log/邮件

/bin/sh -c ./gtcache &&gt;/dev/null &

D

G

M

/首页/管理员/个人资料

128.55.12.10

J

/var/log/邮件

I

/var/log/wdev

/tmp/memtrace.so

H

/etc/firefox/native-messaging-hosts/gtcache

C

/var/log/wdev

I

/var/log/邮件

J

Linux-2 场景的对齐子图。在这种情况下，攻击者通过恶意密码管理器浏览器扩展利用 Firefox，将可执行文件植入磁盘。然后，攻击者运行丢弃的可执行文件以泄露一些机密信息，并对目标网络上的已知主机执行端口扫描。我们用相同的字母标签标记每个图中的对齐节点。查询图上的某些节点未与出处图中的任何节点对齐。这会将图形对齐的分数降低到小于

1.虽然G*q*与G*p*中的一个子图有很大的重叠，但有些节点没有对齐，有些信息流和因果依赖没有出现在出处图中。然而，这些节点的百分比很小。只要报告主要与实际的攻击活动相匹配，我们的方法就不会受此影响。

# 公共攻击评估

在本节中，我们将描述 Poirot 对现实世界恶意软件家族执行的攻击的评估，并将其有效性与其他类似工具的有效性进行比较。我们在表 XX 中显示了该评估的结果。这些恶意软件家族的名称、我们用来描述其行为的 CTI 报告以及报告发布的年份显示在前三列中。

突变检测评估。如前所述，攻击者的一种常见做法是变异恶意软件以逃避检测或为其添加更多功能。因此，CTI 报告可能会描述系统中实际存在的不同版本的恶意软件的行为，对于威胁搜寻工具来说，能够检测恶意软件样本的不同突变至关重要。为此，我们执行了几个真实世界的恶意软件系列，包括

表二十

恶意软件报告。在检测结果中，B=Behavior，PE=PE-Sieve，F=File Name，H=Hash，P=Process Name，R=Registry，S=Windows Security Event。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **恶意软件**  **姓名** | **报告**  **资源** | **年** | **报告**  **样品** | **已分析**  **恶意软件 MD5** | **样本**  **关系** | **孤立**  **国际 奥委会** |  | **检测** | **n 结果** | **s** |
| **红线** | **洛基** | **斯普伦克** | 波洛 |
| 新泽西州 | 菲德利斯  [73] | 2013 | 30 | 2013385034e5c8df-  bbe47958fd821ca0 | 不同的 | 153 | F+H | F+H | P | B  （分数=0.86） |
| 副狗 | 火眼  [64] | 2013 | 8 | 8 ABA 4b 51884072 F2 f2a-  50cbc5ecfe326701 | 子集 | 21 | F*×*2  +H+R | F*×*2  +H | P+R | B  （分数=0.71） |
| 乌罗布罗斯 | 格达塔  [74] | 2014 | 4 | 51e7e58a1e654b6e-  586fe36e10c67a73 | 子集 | 26 | F+H | F+H | R | B  （分数=0.76） |
| 卡巴纳克 | 卡巴斯基  [75] | 2015 | 109 | 1e47e12d11580e93-  5878b0ed78d2294f | 不同的 | 230 | - | 体育 | S | B  （分数=0.68） |
| 尘土飞扬 | 清空  [76] | 2016 | 79 | - 0756357497  41ed6a6d4403b395 | 不同的 | 250 | - | - | - | B  （分数=1.00） |
| 海洋莲花 | 场景 [77] | 2018 | 9 | d592b06f9d112c86-  50091166c19ea05a | 子集 | 117 | F+R | F+聚乙烯 | P+R | B  （分数=0.65） |
| 鹰眼 | 福蒂内特  [78] | 2019 | 3 | 666a200148559e4a-  83fabb7a1bf655ac | 不同的 | 3 | - | 体育 | - | B  （分数=0.62） |

在受控环境中，同一恶意软件的不同变异版本。表 XX 的第四列显示了属于相应 CTI 报告中提到的家族的具有不同哈希值的恶意软件样本的数量。我们注意到，报告仅描述了少数样本的行为。表 XX 的第五列显示了我们选择的样本的哈希值，而第六列显示了我们选择的样本与 CTI 报告所基于的样本之间的关系。例如，DeputyDog、Uroburos 和 OceanLotus 的报告涵盖了一组不同样本执行的不同活动，我们选择的样本就是其中之一。我们已将所有这些活动汇总到一个查询图中。对于其他测试用例，我们执行的样本与报告所基于的样本不同，这可以被认为是检测到变异的恶意软件。 njRAT 和 DustySky 明确提到了他们分析的样本， 这与我们选择的不同。 Carbanak 报告提到了 109 个样本，我们从中随机选择了一个。最后，样本

表 XXI：图 23 中查询图的节点标签及其对齐方式。

|  |  |  |  |
| --- | --- | --- | --- |
| **恶意软件** | **节点** | **标签** | **对齐节点标签** |
| 卡巴纳克 | A | %邮件申请% | 雷鸟 |
| B | \*.%exe% | 邀请函 |
| C | *∗* | 邀请 |
| D | %system32%*\*svchost | C:*\*Windows*\*SysWOW64*\*svchost.exe:WofCompressedData |
| E | 服务主机 | 服务主机 |
| F | *∗*Sys$ | 没有任何 |
| G | % common appdata % \ Mozilla \ \*。%exe% | C:*\*ProgramData*\*Mozilla*\*BwgWXFhfbVpfWgJfBg.bin |
| H | [香港中文大学]\软件\微软\Windows\  当前版本 \Internet 设置 | [hkcu]\ software \ microsoft \ windows \ current version(软件\ Microsoft \ windows \当前版本)  \互联网设置 |
| I | %AppData%*\*Mozilla*\*Firefox*\∗\*prefs.js | C:\Users\测试用户\AppData\Roaming\Mozilla\Firefox\  配置文件\ddl1t72n.default\prefs.js |
| J | %外部 IP 地址% | 没有任何 |
| 乌罗布罗斯 | A | *∗* | 合同.exe |
| B | % appdata % \ Microsoft \ cred proves . TLB | C:\Users\test 用户\AppData\Roaming\Microsoft\credprov.tlb |
| C | %APPDATA%*\*Microsoft*\*shdocvw.tlb | C:\Users\test 用户\AppData\Roaming\Microsoft\shdocvw.tlb |
| D | 运行dll32 | 运行dll32 |
| E | [HKCU]*\*Software*\*Classes*\*CLSID*\*42aedc87-  2188-41fd-b9a3-0c966feabec1*\* | [HKCU]*\*Software*\*Classes*\*CLSID*\*42aedc87-2188-41fd-b9a3-  0c966feabec1*\* |
| F | \*\winview.ocx | 没有任何 |
| G | \*\mskfp32.ocx | 没有任何 |
| H | \*\msvcrtd.tlb | 没有任何 |
| I | % appdata % \ Microsoft \ oleo 32 . dll | C:\Users\测试用户\AppData\Roaming\Microsoft\oleaut32.dll |
| J | % appdata % \ Microsoft \ oleo 32 . TLB | C:\Users\测试用户\AppData\Roaming\Microsoft\oleaut32.tlb |
| K | % appdata % \ Microsoft \ libdcodec . dll | C:\Users\测试用户\AppData\Roaming\Microsoft\libadcodec.dll |
| L | % appdata % \ Microsoft \ libdcodec . TLB | C:\Users\测试用户\AppData\Roaming\Microsoft\libadcodec.tlb |
| 尘土飞扬 | A | \*.%exe% | 新闻.docx.exe |
| B | *∗* | 消息 |
| C | %微软字% | C:*\*Program Files*\*Microsoft Office*\*Office12*\*Winword.exe |
| D | \*\vboxmrxnp.dll | C:*\*WINDOWS*\*vboxmrxnp.dlls |
| E | \*\vmbusres.dll | C:*\*WINDOWS*\*vmbusres.dlls |
| F | \*\vmGuestlib.dll | C:*\*WINDOWS*\*SysWOW64*\*vmGuestLib.dll |
| G | %TEMP%\\*.%exe% | C:\Users\测试用户\AppData\Local\Temp  *\*1371372533114561232114361100131187183149253.exe |
| H | *∗* | 1371372533114561232114361100131187183149253 |
| I | %TEMP%\时间 | C:\Users\测试用户\AppData\Local\Temp\temps |
|  | A | \*.%exe% | gui gui saigontel.exe 的详细信息 |

*下页继续*

*表二十一——接上一页*

|  |  |  |  |
| --- | --- | --- | --- |
| **恶意软件** | **节点** | **标签** | **对齐节点标签** |
| 海洋莲花 | B | *∗* | Chi tiet don 当eu nai gui saigontel |
| C | %温度%\\* | C:\Users\test 用户\AppData\Local\Temp\tmp.docx |
| D | %temp%*\*[0-9].tmp.%exe% | 没有任何 |
| E | %微软字% | C:*\*Program Files*\*Microsoft Office*\*Office12*\*Winword.exe |
| F | \*\rastlsc.%exe% | C:*\*Program Files (x86)*\*Symantec*\*Officewordtask*\*rastlsc.exe |
| G | \*\rastls.dll | C:*\*Program Files (x86)*\*Symantec*\*Officewordtask*\*rastls.dll |
| H | \*\*\（Sylog.bin.OUTLFLTR.DAT） | C:\Program 文件 (x86)\Symantec\  Officewordtask\OUTLFLTR。数据；数字录音带 |
| I | 拉斯克 | 拉斯克 |
| J | \软件\类\ AppX \* | 没有任何 |
| K | \*\HTTPProv.dll | 没有任何 |
| L | 软件\类\CLSID\{E3517E26-  8E93-458D-A6DF-8030BC80528B*}* | 软件\类\CLSID\{E3517E26-8E93-458D-A6DF-  8030BC80528B*}* |
| M | %外部 IP 地址% | 没有任何 |
| 新泽西州 | A | *∗* | 授权 |
| B | \*.exe.config | C:\Users\test 用户\Desktop\Authorization.exe.config |
| C | \*.tmp | C:\Users\测试用户\AppData\Roaming\ja33kk.exe.tmp |
| D | c:\ WINDOWS \ Prefetch \ \*。EXE-∫。脉波频率（Pulse Frequency的缩写） | c:\ Windows \预取\授权。EXE-69AD75AA.pf |
| E | %APPDATA%\\* | C:\Users\测试用户\AppData\Roaming\ja33kk.exe |
| F | *∗* | 生涩的 |
| G | c:\ WINDOWS \ Prefetch \ \*。EXE-∫。脉波频率（Pulse Frequency的缩写） | C:\Windows\Prefetch\JA33KK。EXE-7FA5E873.pf |
| H | %USER PROFILE%\开始菜单\程序\启动\∗ | C:*\*Users*\*test user*\*AppData*\*Roaming*\*Microsoft*\*Windows*\* Start Menu*\*Programs*\*Startup*\*-  9758a8dfbe15a00f55a11c8306f80da1.exe |
| I | 网盘 | 网盘 |
| J | c:\ WINDOWS \预取\NETSH。EXE-∫。脉波频率（Pulse Frequency的缩写） | c:\ Windows \预取\NETSH。EXE-CD959116.pf |
| K | [香港中文大学]\软件\微软\Windows\  当前版本\运行\ | [HKCU]\软件\微软\ Windows \当前版本\运行\ |
| L | [HKLM]\软件\微软\Windows\  当前版本\运行\ | [HKLM]\软件\微软\ Windows \当前版本\运行\ |
| M | [HKLM]*\*SYSTEM*\*CurrentControlSet*\* Services*\*SharedAccess*\*Parameters*\* FirewallPolicy*\*StandardProfile*\*Authorized-  应用程序\列表\APPDATA\ | 没有任何 |
| N | %外部 IP 地址% | 没有任何 |
|  | A | \*.%压缩% | 形式发票 20190423072201 pdf.bin.zip |
| B | %未达人% | WinRAR.exe文件 |

*下页继续*

*表二十一——接上一页*

|  |  |  |  |
| --- | --- | --- | --- |
| **恶意软件** | **节点** | **标签** | **对齐节点标签** |
| 鹰眼 | C | \*.%exe% | c:\ Users \ test user \ Desktop \形式发票20190423072201 pdf.bin |
| D | *∗* | 形式发票 20190423072201 pdf |
| E | 注册组合 | 注册组合 |
| F | vbc | vbc (PID1 ) |
| G | vbc | vbc (PID2 ) |
| H1 | \*歌剧\* | C:\Users\测试用户\AppData\Roaming\Opera\Opera7\  配置文件\wand.dat |
| H2 | \*铬\* | C:\Users\测试用户\AppData\Local\Google\Chrome\User  数据\默认\登录数据 |
| H3 | \*铬\* | C:*\*Users*\*test user*\*AppData*\*Local*\*Chromium*\*User Data |
| H4 | ∗铬 SxS∗ | C:*\*Users*\*test user*\*AppData*\*Local*\*Google*\*Chrome SxS*\*User Data |
| H5 | \*雷鸟\* | C:\Users\测试用户\AppData\Roaming\Thunderbird\Profiles |
| H6 | \*海猴\* | C:\Users\测试用户\AppData\Roaming\Mozilla\SeaMonkey\ Profiles |
| H7 | \*太阳鸟\* | 没有任何 |
| H8 | *\*IE\** | C:\Users\测试用户\AppData\Local\Microsoft\Windows\  历史\历史.IE5 |
| H9 | \*野生动物园\* | 没有任何 |
| H10 | \*火狐\* | C:\Users\测试用户\AppData\Roaming\Mozilla\Firefox\profiles.ini |
| H11 | \*Yandex\* | C:\Users\测试用户\AppData\Local\Yandex\YandexBrowser\  用户数据\默认\登录数据 |
| H12 | \*维瓦尔第\* | C:\Users\测试用户\AppData\Local\Vivaldi\User  数据\默认\登录数据 |
| I1 | \*雅虎\* | [HKLM]\软件\雅虎\寻呼机 |
| I2 | ∗群组邮件∗ | 没有任何 |
| I3 | \*雷鸟\* | C:\Users\测试用户\AppData\AppData\Roaming\  雷鸟\配置文件 |
| I4 | MSNMessenger | [HKLM]\软件\微软\MSNMessenger |
| I5 | ∗Windows 邮件∗ | C:\Users\test 用户\AppData\Local\Microsoft\Windows Mail |
| I6 | \*增加邮件\* | [HKLM]\软件\ WOW6432Node节点\增量邮件\标识 |
| I7 | \*展望\* | [HKLM]\软件\微软\ Office \ 16.0 \ Outlook \配置文件 |
| I8 | \*尤朵拉\* | [HKLM]\软件\高通\ Eudora \命令行 |
| J | %temp%\\*.tmp | C:\Users\测试用户\AppData\Local\Temp\tmp8FC3.tmp |
| K | %temp%\\*.tmp | C:\Users\测试用户\AppData\Local\Temp\tmp8BAB.tmp |
| L | http[s]:\ \ whatismyipaddress . com \ \* | 没有任何 |

*下页继续*

*表二十一——接上一页*

|  |  |  |  |
| --- | --- | --- | --- |
| **恶意软件** | **节点** | **标签** | **对齐节点标签** |
|  | M | %外部 IP 地址% | 没有任何 |
| 副狗（图 19） | A | \*.%exe% | C:\用户\测试用户\桌面\img20130823.jpg.exe |
| B | *∗* | img20130823 |
| C | %APPDATA%\\* | C:*\*ProgramData*\*28542CC0.dll |
| D | [香港中文大学]\软件\微软\Windows\  当前版本\运行\ | 没有任何 |
| E | %外部 IP 地址% | 180.150.228.102 |



E

I

E

F

C

G

7：写

5：加载

10：取消链接6：写

2：写

8：写

9：写

H

D  ~~3：叉子~~ A

4：加载

11：取消链接

1：写

15：取消链接

12：取消链接

13：取消链接

I

B

14：取消链接

J

L

K

乌罗布罗斯

3.阅读 2. 叉子 7. 写 9.执行

C G

10. 写

6.阅读

B ~~8. 叉子~~  H

1.执行

4.阅读

F

5. 阅读

D

尘土飞扬

I

J

9. 阅读

D

C ~~3. 写~~

4.执行

~~6. 叉子~~ F

E

10. 发送

5.取消链接

7. 写

B

A ~~1.写~~

2.执行

8. 阅读

H

G

卡巴纳克

A



G

C

D

B

2.写3. 写

H

7. 写

E

1. 阅读 8. 写

4. 写

A ~~9. 叉子~~  I

6.执行

5. 叉子

11. 写

14. 发送

10. 写

F

13. 写

12. 写

N

K

J

L

M

L

M

鹰眼

34. 阅读

12

H

J

31. 取消链接

33.写

1. 取消链接
2. 写

E

K

I8

29. 阅读

30. 阅读

18. 打开

27. 写

26. 打开

5. 叉子

4. 叉子

6. 叉子

28. 写

H2

F

D

G

1. 打开
2. 打开

20. 打开

I2

H1

1. 写
2. 执行

19. 打开

B

C

1. 阅读

A

I1

新泽西州

17. 阅读

16. 写

M

F

D

10. 执行

6.写

C

L

9. 叉子

15. 写

I

5.执行

14. 加载

11.加载

8. 写

H

K

1.执行

4. 叉子

B

1. 写
2. 写

13. 阅读

12. 阅读

7. 写

E

A

J

海洋莲花

G

图 23. 从 CTI 报告中提取的 Carbanak、DustySky、Uroburos、OceanLotus、njRAT 和 HawkEye 恶意软件的查询图。

HawkEye 恶意软件是从外部来源中挑选出来的，不在报告中提到的样本之列。

与现有工具的比较。我们将 Poirot 与其他三个工具的结果进行比较，即 RedLine [12]、Loki [11] 和 Splunk [15]。这些工具的输入是从我们提取查询图的同一份报告中提取的，并包含不同类型的 IOC，例如哈希值、进程名称、文件名和注册表。我们已将这些 IOC 转换为每个工具可接受的格式（例如，RedLine 接受 OpenIOC 格式的输入 [45]）。提交给 Redline、Loki 和 Splunk 的 IOC 的数量显示在第 7 列中，而提交给 Poirot 的查询图显示在图 23 中。这些查询图的详细说明展示了它们是如何构建的，可以在附录 C 中找到. 查询图中的节点标签与其实际名称之间的对应关系在表 XXI 的第二和第三列中表示，而 Poirot 产生的比对在最后一列中显示。

如提取的查询图所示，基于信息流和因果依赖关系的 Poirot 搜索方法的设计使我们能够包含良性节点（DustySky 中的节点 C、D、E 和 F）或攻击查询图中具有完全相同名称的良性实体（Carbanak 中的节点 E）的节点。但是，这些实体名称无法在其他测试工具中定义为 IOC，因为这会导致许多误报。由于 Redline、Loki 和 Splunk 孤立地查找每个 IOC，他们希望 IOC 作为输入，无论它们与其他实体的连接如何，这些输入始终是恶意的。为此，我们在干净的系统中对每个孤立的 IOC 进行初步搜索，并确保我们只提取了 IOC

在干净的系统中无法匹配。因此，对于一些测试用例，比如 HawkEye，虽然行为图很丰富，但是除了一些可以定义的哈希值之外，没有那么多孤立的 IOC。这凸显了 IOC 之间的依赖关系的重要性，它是 Poirot 搜索算法的基础，是其他工具不考虑的。

检测结果。表 XX 的最后四列包含检测结果，显示了每个工具如何检测测试样本。关键字 B、F、H、P 和 R 分别代表基于行为、文件名、哈希值、进程名和注册表的检测。此外，一些经过测试的工具还具有其他方法来检测异常、注入或其他安全事件。其中，我们遇到了一些来自 Windows Security Mitigation 和 PE-Sieve [79] 的警报，它们分别由关键字 S 和 PE 表示。而对于 Poirot，显示了一个分数，显示了每个查询的整体对齐的好坏

图表中，对于其他工具，×N 表示已超过一个时的命中数

命中特定类型的 IOC。

如表 XX 所示，对于所有的测试用例，Poirot 找到了一个绕过的对齐方式

1的阈值

3

.运行搜索算法后，在大多数情况下，Poirot

只为查询图中的一部分实体找到了节点对齐，除了 DustySky，Poirot 为每个实体找到了节点对齐。对齐节点之间出现的信息流和因果依赖关系通常与查询图相同，但有一些例外。例如，与它在 njRAT 的查询图中的显示方式相比，节点 A 生成了大部分动作，在我们的实验中，节点 F 生成了大部分动作，例如对节点 C、G、K 的写入事件， L，以及节点 I 的分叉事件。然而，

由于查询图中存在从节点 A 到节点 F 的路径，因此 Poirot 能够找到这些对齐并测量出较高的对齐分数。

所有其他工具也检测到 njRAT、DeputyDog、Uroburous 和 OceanLotus 的样本，因为这些样本使用威胁情报中可用的唯一名称或哈希值，并且可能归因于这些恶意软件。对于其他三个测试用例，由于恶意软件突变、每次运行中使用随机名称（HawkEye 查询图中的节点 J 和 K）以及使用合法库或其相似名称等不同原因，都无法检测到任何孤立的 IOC。尽管如此，Splunk 还是发现了一个与 Carbanak 样本相关的 ETW 事件，该事件是在 Windows 安全缓解服务阻止 svchost 生成动态代码时生成的。 Loki 的 PE-Sieve 还检测到一些代码植入尝试，这些尝试导致发出一些警告信号而不是警报。 PE-Sieve 会检测对进程所做的所有修改，即使它们不一定是恶意的。由于此类修改在许多良性场景中经常发生， PE-Sieve 检测被视为需要进一步调查的警告信号。

结论。我们的分析结果表明，当样本是编写报告所依据的样本的子集时，其他工具通常表现良好。这种情况类似于没有突变的情况，因此，存在可以用作恶意软件签名的唯一哈希值或名称。例如，DeputyDog 样本会丢弃许多具有唯一名称和哈希值的文件，这些文件不会出现在良性系统中，并且发现它们是这种恶意软件的强烈迹象。但是，它的查询图（图 19）不是很丰富，而且 Poirot 无法将修改后的注册表（节点 D）与其余对齐的节点相关联。虽然

计算出的分数仍然高于阈值，但当恶意软件使用强大到足以单独指示攻击的知名 IOC 时，其他工具的性能可能会更好。

相反，当所选样品与报告分析的样本不同时，这与恶意软件发生的情况相似，其他工具通常无法找到攻击。在这种情况下，Poirot 有更好的机会检测到攻击，因为行为通常在突变中保持不变。

# 对良性数据集的评估

为了对 Poirot 的误报进行压力测试，我们使用了作为 TC 计划中对抗性参与的一部分生成的良性数据集和我们监控了一个月的四台机器（客户端、SSH 服务器、邮件服务器和 Web 服务器）。总的来说，这些数据集包含价值超过 7 个月的良性审计记录和数十亿条 Windows、Linux 和 FreeBSD 上的审计记录。在此期间，多个用户使用这些系统并执行了典型的无攻击操作，包括网页浏览、安装安全更新（包括内核更新）、病毒扫描、备份和软件卸载。

收集日志后，我们运行 Poirot 构建出处图，然后搜索我们从 TC 报告和公共恶意软件报告中提取的所有查询图。我们从每个查询图每个出处图的不同种子节点选择开始尝试多达 20 次迭代，并选择最高分。请注意，尽管这些日志不受攻击，但它们与我们的查询图共享许多节点和事件，例如机密文件、关键系统文件、文件编辑工具或与 Web 浏览/托管相关的进程，以及

1

最优值

精确

回忆 F 分数

0.8

0.6

速度

0.4

0.2

0

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

临界点

图 24. 选择最佳阈值。

电子邮件客户端，所有这些都是在良性数据收集期间访问的。然而，即使在偶然出现类似流量的情况下，影响分数也会删掉其中的许多流量。因此，Poirot 在所有良性数据集中计算的图形对齐分数最多等于 0.16，远低于阈值。

验证阈值。阈值的选择对于避免错误信号至关重要。阈值太低可能导致过早匹配（误报），而阈值太高可能导致丢失合理匹配（误报）。因此，在选择最佳阈值时存在折衷。为了确定最佳阈值，我们使用不同的阈值测量了 F-score，如图 24 所示。该分析是基于在 Poirot 搜索算法的 20 次迭代中计算出的最高对齐分数，针对我们的所有攻击和良性场景进行了评价。如图所示，最高

当阈值在区间[0.17, 0.54]时达到F-score值，这是正确找到所有攻击子图的范围，对良性数据集不发出警报。这个区间的中间值，即 0.35，最大化了攻击和良性分数之间的差距，选择这个值作为最佳阈值可以最小化分类错误。因此，我们

将 AC*thr*设置为 3，导致1=1接近最优值。

*ACthr* 3

### 5.4.4 效率

我们使用的不同工具的开销和搜索时间如表 XXII 所示。 Redline 和 Loki 是离线工具，用于搜索攻击者在系统上留下的工件，而 Splunk 和 Poirot 是在线工具，基于运行时收集的系统事件进行搜索。因此，由于审计日志收集，Redline 和 Loki 没有运行时开销。由于日志收集而导致的 Splunk 和 Poirot 的运行时开销使用 Apache 基准测试 [80] 测量 Web 服务器响应能力，JetStream [81] 测量浏览器执行时间，HDTune [82] 测量繁重的硬盘驱动器事务。如表 XXII 所示，这两种工具的运行时开销都可以忽略不计，而运行时

#### 表二十二

与相关系统的效率比较。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **检测**  **方法** | **类型** |  | **运行时开销** |  | **搜索**  **时间（分钟）** |
| **阿帕奇 [80]** | **喷射流 [81]** | **高清调谐 [82]** |
| 红线 | 离线 | - | - | - | 124 |
| 洛基 | 离线 | - | - | - | 215 |
| 斯普伦克 | 在线的 | 3.70% | 2.94% | 4.37% | ¡ 1 |
| 波洛 | 在线的 | 0.82% | 1.86% | 0.64% | ¡ 1 |

Splunk 可以通过将其设置为分布式设置并将数据索引任务卸载到另一台服务器来进一步改进。

表 XXII 的最后一列显示了搜索每个工具的 IOC 所花费的时间。离线工具的搜索时间很大程度上取决于正在运行的进程数和占用的磁盘空间量，在我们的例子中为 500 GB。另一方面，在线方法的搜索时间很大程度上取决于日志的大小、类型和日志所代表的活动数量。由于我们对真实世界恶意软件样本的实验是在没有许多后台良性活动和 Internet 连接的受控环境中运行的，因此 Splunk 和 Poirot 都花费不到一分钟的时间来搜索表 XX 中提到的所有 IOC。在下文中，我们对 Poirot 在 TC 场景中的效率进行了深入分析，这些场景总体包含超过一个月的日志数据，包含攻击和良性活动的组合。分析是在 8 核 CPU 上完成的，每个 CPU 的速度为 2.5GHz，内存为 150GB。

审计日志消耗。在表 XXIII 中，第二列显示了磁盘上日志的初始大小，第三列表示消耗所有审计日志事件所需的时间

#### 表二十三

日志、消费和搜索时间的统计。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **设想** | **磁盘大小**  **（未压缩）** | **消耗**  **时间** | **占据**  **记忆** | **日志**  **期间** | **子∈**  ***|****V* (*Gp* )***|*** | **obj** *∈*  ***|****V* (*Gp* )***|*** | ***|****E*(*Gp* )***|*** | **搜索**  **时间（秒）** |
| BSD-1 | 3022 兆字节 | 0h-34m-59s | 867 兆字节 | 03d-18h-01m | 110.66 K | 1.48 M | 7.53 M | 3.28 |
| BSD-2 | 4808 兆字节 | 0h-58m-05s | 1240 兆字节 | 05d-01h-15m | 213.10 K | 2.25 M | 12.66 M | 0.04 |
| BSD-3  BSD-4 | 1828 兆字节 | 0h-21m-31s | 638 兆字节 | 02d-00h-59m | 84.39 K | 897.63 K | 4.65 M | 26.09 |
| 1.47 |
| 赢一  双赢2 | 54.57 千兆字节 | 4h-58m-30s | 3790 兆字节 | 08d-13h-35m | 1.04 M | 2.38 M | 70.82 M | 125.26 |
| 46.02 |
| Linux-1  Linux-2 | 9436 兆字节 | 1h-26m-37s | 4444 兆字节 | 03d-04h-20m | 324.68 K | 30.33 M | 51.98 M | 1279.32 |
| 1170.86 |
| Linux-3 | 131.1 千兆字节 | 2h-30m-37s | 21.2 千兆字节 | 10d-15h-52m | 374.71 K | 5.32 M | 69.89 M | 385.16 |
| Linux-4 | 4952 兆字节 | 0h-04m-00s | 1095 兆字节 | 00d-07h-13m | 35.81 K | 859.03 K | 13.06 M | 20.72 |

从磁盘用于在内存中构建出处图。该时间以挂钟时间来衡量，并根据每个审计日志的大小和在每个平台（BSD、Windows、Linux）中生成的审计日志的结构而有所不同。第四列显示每个出处图的总内存消耗。比较磁盘与内存的大小，我们注意到我们通过基于 [1] 的紧凑内存来源图表示的平均压缩率为 1:4 (25%)。但是，如果内存是一个问题，仍然可以使用该领域提出的其他技术来实现更好的压缩 [36,83,84]。第五列显示了收集日志的持续时间，而第 6、7 和 8 列分别显示了从日志构建的出处图中的主题（即进程）、对象和事件的总数。我们注意到查询图平均比这些场景的出处图小 209K 倍。尽管如此， Poirot 仍然能够非常快速地找到 G*q*在 G*p*中的精确嵌入，如最后一列所示。我们注意到一些场景被加入（例如，Win-1&2），因为它们是在同一台机器上同时执行的。

图分析。在表 XXIII 的最后一列中，我们展示了 Poirot 搜索算法的图形分析运行时间。这些时间是从提交搜索查询的那一刻开始计算的，直到我们在 G*p*中找到一个相似的图，其对齐分数超过了阈值。因此，对于 Linux-2，时间包括两次迭代的时间之和。主要瓶颈在于图搜索扩展（第 3 步），波洛花在图搜索上的时间取决于几个因素。显然，查询图和出处图的大小都与运行时间成正比。但是，我们注意到 G*q*中的节点名称和

这个图形的形状有更显着的影响。特别是，当存在具有许多候选对齐的节点时，多次反转方向的机会更高，并且运行时间相应增加。

### 5.5 概括

Poirot 将网络威胁搜寻制定为图形模式匹配问题，以可靠地检测已知的网络攻击。 Poirot 基于一种有效的对齐算法，在内核审计记录的出处图中找到表示威胁行为的图的嵌入。我们评估 Poirot 在现实世界的网络攻击和由专业红队在三个操作系统平台上进行的十种攻击场景，以及数千万条审计记录。 Poirot 在几分钟内就成功检测到所有攻击，信心十足，没有错误信号。

# 第 6 章

**相关工作**

*本章包括 [1-3] 中发表的材料的摘录和数字。*

在本章中，我们将讨论与本文提出的方法密切相关的工作。具体来说，我们涵盖了起源图、入侵检测、警报关联、查询处理系统、行为发现和图模式匹配等领域的相关研究。

### 出处图

已经建立了几个日志记录和出处跟踪系统来监控系统的活动 [38, 39, 85-89] 并构建出处图。其中，Backtracker [8, 9] 是最早使用依赖图追溯入侵根源的作品之一。这些图表是通过关联日志系统收集的事件并确定系统实体之间的因果关系而构建的，以帮助在检测到攻击后进行取证分析。

Sleuth 在两个重要方面改进了 Backtracker 的技术。首先，Backtracker 旨在在取证环境中运行，而我们的分析和数据表示技术旨在实现实时检测。撇开硬件比较不谈，我们注意到 Bactracker 需要 3 小时来分析 24 小时内的审计数据，而 Sleuth 能够在不到 3 分钟的时间内处理 358 小时的日志。其次，Backtracker 依赖于外部工具生成的警报，因此它的取证搜索和剪枝

143

无法利用产生这些警报的原因。相比之下，我们的分析程序利用了我们原则性的基于标签的检测方法的结果，因此本质上更精确。例如，如果攻击故意写入一个众所周知的日志文件，Backtracker 的搜索启发式可能会从最终图表中删除该日志文件，而我们基于标签的分析将防止该节点被修剪掉。

为了更有效地使用出处图，几种方法引入了压缩、汇总和日志缩减技术 [36,83,84,90] 以区分有价值的事件和无信息的事件，从而减少存储大小。将流程划分为更小的单元是在出处图中添加更多粒度的方法之一，为此，研究人员使用了不同的方法，例如动态二进制分析 [25、40]、源代码注释 [91] 或建模基于推理 [4, 92,93]。此外，为了更精确的跟踪，提出了记录和回放[94、95]和并行执行方法[96]。与我们提出的方法类似，最近的一些研究利用出处图来实现不同的目标，例如及时的因果关系分析 [97]、警报分类 [98] 和零日攻击路径识别 [99]。

虽然大多数上述系统都在系统调用级别运行，但其他几个系统以更精细的粒度跟踪信息流[23-25]。他们通常使用应用程序（例如，使用 Pin [26]）来跟踪通过程序的信息流。这种细粒度的污染可以提供更精确的出处信息，但代价是更高的开销。细粒度的污点跟踪不仅可以减轻低级语言中的内存损坏漏洞 [29]，而且还可以覆盖更广泛的漏洞

高级和脚本语言的程序 [30]。它还被证明是一种非常强大的恶意软件分析和补丁自动生成技术。最后，已经表明，使用细粒度的 taint [100] 可以显着增强 IDS 技术的能力。

### 入侵检测

使用日志的离线入侵检测已经研究了很长时间[101-103]，并且考虑了不同的日志来源。 BotHunter [104] 分析网络流量以检测恶意软件感染。歌剧等。 [105] 利用 DNS 或 Web 代理日志来检测企业中的早期感染。 Disclosure [106] 从 NetFlow 日志中提取统计特征来检测僵尸网络 C&C 通道。 DNS 日志也被广泛使用 [107, 108] 来检测恶意域。

使用系统调用监控和/或审计日志的基于主机的 IDS 已通过大量研究工作进行了调查 [109-112]。这些方法分为三类：（1）基于误用 [41, 42]，检测与已知攻击相关的行为； (2) 基于异常的 [103, 110, 113-119]，它学习一个良性行为模型并检测与它的偏差； (3) 基于规范的[120, 121]，根据专家指定的策略检测攻击。虽然第一类的技术无法处理未知攻击，但第二类的技术会产生许多误报。从表面上看，在 Holmes 中使用 TTP 可以看作是误用检测的一个例子。然而，我们的方法超越了经典的误用检测 [41, 42]，在匹配的 TTP 模式中涉及的实体之间存在信息流依赖关系时，使用匹配的先决条件 - 后果模式。

### 警报相关性

网络 IDS 经常产生无数警报。警报关联分析警报之间的关系，帮助用户应对泛滥。通常一起使用的主要方法是对相似的警报进行聚类，对警报进行优先级排序，并确定警报之间的因果关系[122-126]。 Hercule [49] 使用社区发现技术来关联可能分散在多个日志中的攻击步骤。此外，行业使用类似的方法来构建 SIEM [15-17]，以基于来自不同数据源的日志进行警报关联和执行。这些方法依赖于在用户空间中运行的第三方应用程序生成的日志。此外，基于警报时间戳等统计特征的警报关联无助于精确检测多阶段 APT 攻击，因为它们通常持续很长时间。

与这些方法相比，Holmes 建立在各种攻击步骤之间存在的信息流之上，以达到警报关联的目的。在这种情况下使用内核审计数据首先是在 [127] 中进行的。然而，与 Holmes 不同的是，这项工作纯粹是基于滥用的，其重点是利用事件之间的相关性来检测 IDS 遗漏的攻击步骤。 Holmes 使用相同的内核审计数据，但采用不同的方法，基于构建具有低内存占用的主内存依赖图，然后根据 TTP 的高级规范推导 HSG 以引发警报，最后关联警报基于它们之间的信息流。警报相关性的另一项工作依赖于警报的及时接近性[128]。相比之下，福尔摩斯 依赖于信息流和因果关系来关联警报，因此能够检测到步骤执行非常缓慢的攻击。

### 查询处理系统

先前的工作结合了新颖的优化技术、图索引和查询处理方法 [129-131] 以支持及时的攻击调查。 SAQL [132] 是一个异常查询引擎，它查询指定的异常以识别系统事件中的异常行为。 AIQL [133] 可以用作取证查询系统，它具有特定领域的语言，用于从历史审计日志中调查攻击活动。帕斯奎尔等人。 [134] 提出了一个名为 CamQuery 的查询框架，它支持对来源图的实时分析，以解决数据丢失预防、入侵检测和法规遵从等问题。舒等人。 [135] 还提出了一种人工辅助查询系统，为威胁猎人配备了一套强大的新工具。这些作品与 Poirot 正交，可以作为实现我们搜索算法的基础。

### 行为发现

在先前的工作中已经研究了提取诸如信息流和因果依赖关系之类的恶意行为并将它们作为稳健指标进行搜索。克里斯多雷斯库等人。 [136] 提出了一种从恶意软件样本的动态跟踪中挖掘恶意软件行为的方法。同样，Kolbitsch 等人。 [52] 使用符号执行自动生成恶意软件的行为模型。他们将此行为表示为图形，并在未知程序的运行时行为中搜索它。相反，波洛不依赖符号表达，而是在整个系统上寻找相关性和信息流。 TGMiner [53] 是一种从训练审计日志中挖掘判别图模式并在测试数据中搜索它们存在的方法。这项工作的重点是查询公式化，而不是

模式查询处理，作者使用子序列匹配解决方案 [65] 进行搜索，这与我们的图形模式匹配方法不同。

### 图模式匹配

图形模式匹配（GPM）已被证明在各种应用中很有用[66]。 GPM 可以定义为一个大图内的搜索问题，其中包含在小查询图中连续指定的类似连接的子图。这个问题在一般情况下是 NP 完全的 [56]。范等。人。 [67] 提出了一种多项式时间方法，假设模式中的每个连接只能映射到具有预定义跳数的路径。其他工作 [68,69] 通过在向量空间中使用一系列连接函数来解决这个问题。 NeMa [70] 是一种基于节点邻近度的基于邻域的子图匹配技术。相比之下，G-Ray 和后来的 Mage [71, 72] 考虑了查询图的形状和边缘属性，并且更类似于我们的方法，其中相似的信息流和因果依赖关系起着至关重要的作用。然而，这些方法基于随机游走， 这对于生成虚假事件（如 5.3.1 中所述）的攻击者（具有威胁搜寻方法的知识）是不可靠的。虽然我们的图对齐概念与这些作品相似，但 Poirot 分析的图特征提出了新的挑战，例如标记、定向、键入、数百万个节点，以及在对抗环境中构建。此外，许多相关工作正在寻找一个子图，该子图包含查询图的每个节点和每条边的准确对齐，并且无法在某些节点或边可能没有对齐的设置中运行。因此，我们开发了一种新的尽力匹配技术来应对这些挑战。

# 第七章

**结论和未来的工作**

系统事件之间的因果联系对网络安全任务的推理起着至关重要的作用。在本文中，我们提出了利用内核审计日志之间的因果联系来促进网络安全任务的有效方法，例如 APT 检测、场景重建和威胁追踪。所提出的系统利用存储在主存储器中的平台中立的图形表示，它是通过处理内核审计日志的序列来构建的。

在第 3 章中，我们介绍了一种称为 Sleuth 的方法和系统，用于实时检测攻击并从内核审计日志中重建攻击。 Sleuth 使用丰富的基于标签的策略框架，使其分析既高效又精确。我们在来自独立红队攻击的 3 个主要操作系统的大型数据集上评估了 Sleuth，有效地重建了所有攻击，几乎没有错误。

在第 4 章中，我们介绍了 Holmes，这是一个实时 APT 检测系统，它关联了可能用于执行每个 APT 阶段的策略、技术和程序。 Holmes 生成一个高级图表，实时总结攻击者的步骤。我们针对九种现实世界的 APT 威胁评估 Holmes，并将其部署为实时入侵检测工具。结果表明，Holmes 以高精度和低误报率成功检测了 APT 活动。

149

第 5 章将网络威胁搜寻表述为一个图形模式匹配问题，以可靠地检测已知的网络攻击。所提出的系统称为 Poirot，它基于一种有效的对齐算法，以在内核审计记录的出处图中找到表示威胁行为的图的嵌入。我们评估 Poirot 在现实世界的网络攻击和由专业红队在三个操作系统平台上进行的十种攻击场景，以及数千万条审计记录。 Poirot 在几分钟内就成功检测到所有攻击，信心十足，没有错误信号。

综上所述，通过在本文中提出上述系统，我们证明了从内核审计日志中推断出的低级因果信息可用于实现稳健可靠的威胁检测方法，从而有效地查明威胁并揭示攻击的高级图景。通过生成攻击步骤的紧凑视觉图。虽然 Sleuth、Holmes 和 Poirot 分别专注于场景重建、APT 检测和网络威胁搜索，但也有其他与安全相关的任务可以利用内核审计日志中的语义连接来增强。例如，内核审计日志被证明可用于加快取证分析 [97]、警报分类 [98] 和零日攻击路径识别 [99]。除了这些最近的工作之外，我们认为未来还有其他目标可以从这种类型的分析中受益：

* 横向移动检测：攻击者的一种常见做法是逐步在网络中的不同机器上移动以访问高价值资产。在这些攻击中，攻击者入侵一台机器并尝试横向入侵其他机器以建立立足点或访问更敏感的数据。检测横向移动-

对于当前最先进的系统来说，评论是一个巨大的挑战，因为它看起来像是来自内部机器的合法通信。

* 相似性搜索：当我们检测到一台机器上的危害时，很可能一个组织中或跨多个组织的多台机器也受到同一事件的危害。从检测到的实例中提取攻击模式并搜索类似实例有助于增强安全性并改进当前的检测系统。
* 假设测试：安全管理员经常在他们的日常任务中提出不同的假设，并有兴趣验证它们。例如，系统管理员注意到 Desktop 上有一个可疑的可执行文件，并且在后台存在许多正在运行的 svchost 进程。假设可能是 svchost 进程与出现在桌面上的可疑文件相关。内核审计日志可以成为验证一个实体是否影响另一个实体或某些实体之间是否存在任何关联的重要来源。

在本论文中，我们迈出了利用内核审计日志设计高效威胁检测系统的第一步。尽管如此，该研究领域仍有许多挑战和开放性问题需要解决。一个主要挑战是将提议的方法扩展到拥有数千台机器的企业，其中带宽、存储和处理限制比比皆是。在这种情况下，网络防御者通常会得到一个部分收集的出处图，其中可能无法通过执行向后/向前遍历来找到实体之间的连接。作为未来的工作，我们可以结合一些特征

实体（例如它们的名称之间的相似性）以及向后/向前跟踪结果，以在部分收集的出处图上建立相关性。

**附录**

153

**附录 A**

# SLEUTH 的其他攻击讨论

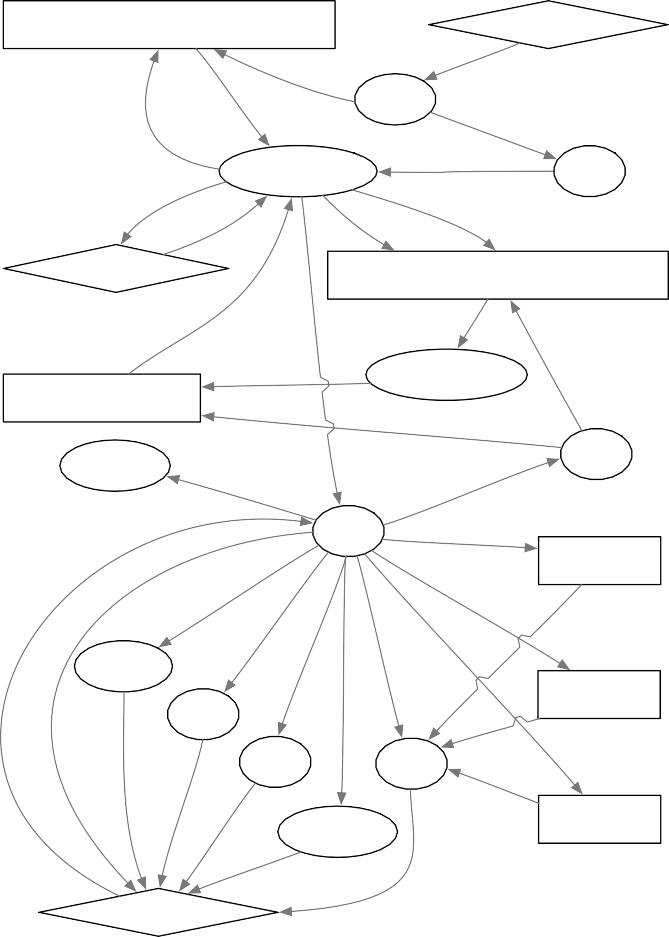
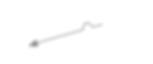
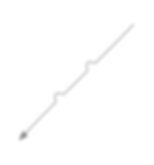
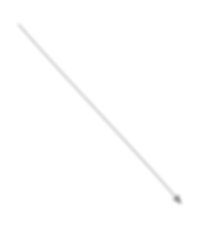
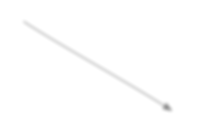
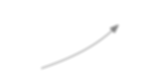
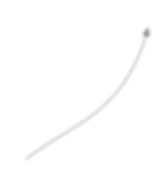
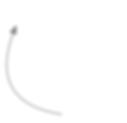
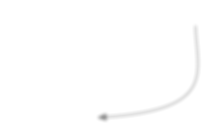
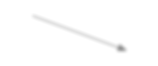
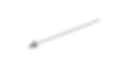
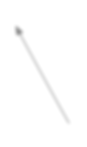
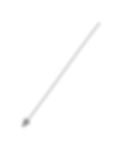
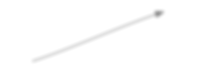
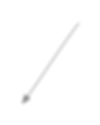
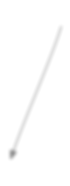
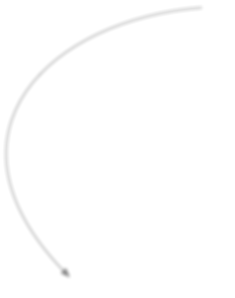
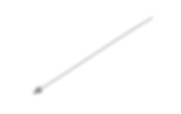
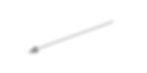
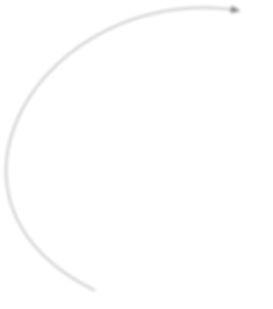
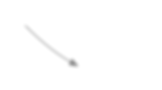
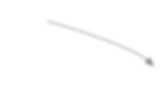
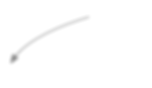
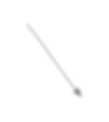
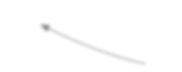
在本节中，我们提供了重建第 3.6.4 节中未讨论的攻击活动的图表。具体来说，我们讨论攻击 L-1、F-1、F-2、W-1 和 L-3。

攻击 L-1。在此攻击中（图 25），利用 firefox 通过 shell 删除并执行文件 mozillanightly。 mozillanightly 进程首先下载并执行 mozillaautoup，然后启动一个 shell，它产生几个其他进程。接下来，文件 netrecon.log 中收集的信息被泄露并删除该文件。

攻击 F-1。在此攻击中（图 26），利用 nginx 服务器通过 shell 删除和执行文件释放器。在执行时，dropper 进程会派生一个 shell，该 shell 会产生多个进程，这些进程写入文件并读取和写入敏感文件。此外，dropper 与攻击者的 IP 进行通信。我们在图中报告了与参与后进行的恢复和管理相关的图表，如第 3.6.5 节所述。攻击 F-2。这次攻击的开始（图 27）与 F-1 类似。但是，在执行时，dropper 进程会下载三个名为 recon、sysman 和 mailman 的文件。

稍后，这些文件被执行和使用，用于泄露从系统收集的数据。

攻击 W-1。在这次攻击中（图 28），firefox 被利用两次来删除并执行一个文件 mozillanightly。第一个 mozillanightly 进程下载并执行文件 photosnap.exe，该文件会截取受害者的屏幕截图并将其保存为 png 文件。



/home/User1/traffic\_gen/mozillanightly

129.55.12.167:8000

*1.接收*

*36. rm*

*2.写*

*5.执行*

火狐

*3. 叉子*

mozillanightly *4. 叉子*

嘘

1. *发送*
2. *收到*

129.55.12.167:443

*7.chmod是什么*

*6.写*

/home/User1/traffic\_gen/mozillaautoup

*10. 阅读*

*8.执行*

*13. 叉子*

/tmp/netrecon.log

*9. 写*

Mozilla自动升级

*35. rm*

*34. rm*

OpenSSL(OpenSSL)

rm

*25. 叉子*

*33.执行*

嘘

*26. 写*

/etc/sudoers

*28. 写*

我是谁

*16.执行*

*18. 执行*

*20. 执行*

*27. 阅读*

*30. 写*

-什么

*24. 执行*

*22. 执行*

猫

/etc/影子

*29. 阅读*

*17. 发送*

*14. 发送*

*19. 发送*

你

*31. 阅读*

/etc/passwd

*15. 收到*

*21. 发送*

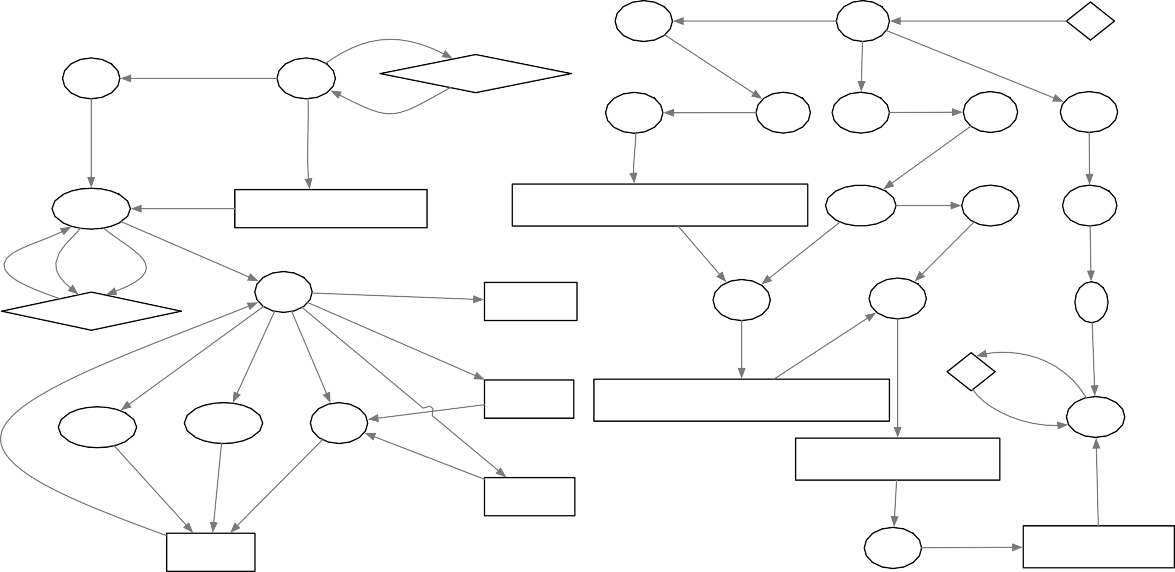
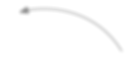
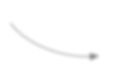
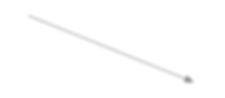
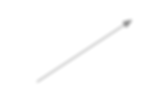
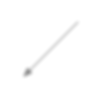
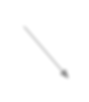
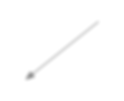
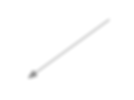
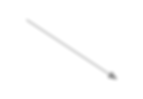
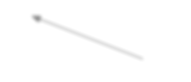
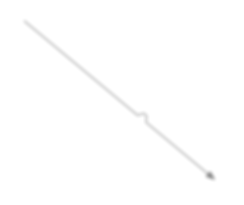
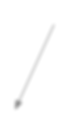
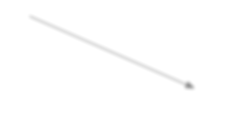
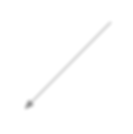
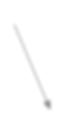
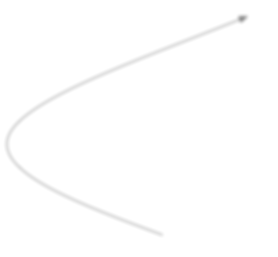
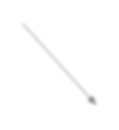
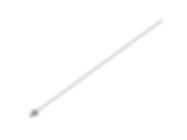
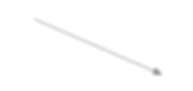
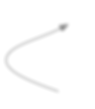
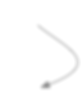
主机名

*23. 发送 32. 发送*

129.55.12.167:4430

图 25. 从活动 L-1 重建的场景图。

随后，jpeg 文件被 mozillanightly 泄露。第二个 mozillanightly 进程下载并执行两个文件： 1) burnout.bat，读取后用于向 cmd.exe 发出命令以收集有关系统的数据； 2) mnsend.exe，由cmd.exe执行，将之前收集的数据泄露出去。



重击

*23. 叉子*

sshd

*22. 收到*

???

*2.发送*

*24. 叉子*

*27. 叉子*

嘘

*4. 叉子*

恩金克斯

129.55.12.167:8000

*39.叉子*

*1.接收*

vi *25. 叉子* 须藤 *bash 28. 叉子* 须藤

重击

*5. 叉子*

*3. 写*

*26. 写*

*29. 叉子*

*40. 叉子*

滴管 *6.执行* /var/tmp/ginx/client\_body\_temp/滴管

/usr/ports/www/nginx/work/nginx-1.10.1/

src/http/ngx\_http\_request\_body.c

制作

*33. 叉子* cc

须藤

*8. 接收 7.发送*

*10. 发送*

*9. 叉子*

*31. 地图 30. 叉子*

*34. 叉子*

*41. 叉子*

嘘

*17. 写*

129.55.12.167:443

/etc/sudoers

cc

ld个

嘘

*15. 写*

*12. 叉子*

*32. 写 35. 阅读36。写*

*45. 发送 42. 分叉*

*10. 叉子*

1. *叉子 16. 写*
2. *读*

猫

???

/etc/passwd

*14. 阅读*

/usr/ports/www/nginx/work/nginx-1.10.1/

objs/src/http/ngx\_http\_request\_body.o

*44. 收到* 恩金克斯

我是谁

OpenSSL(OpenSSL)

*20. 阅读*

/usr/ports/www/nginx/work/

engine x-1 . 10 . 1/对象/engine x \_ 191

*11. 写*

*13. 写*

*21. 写*

*43.执行*

/etc/影子

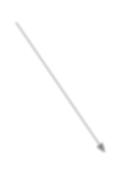
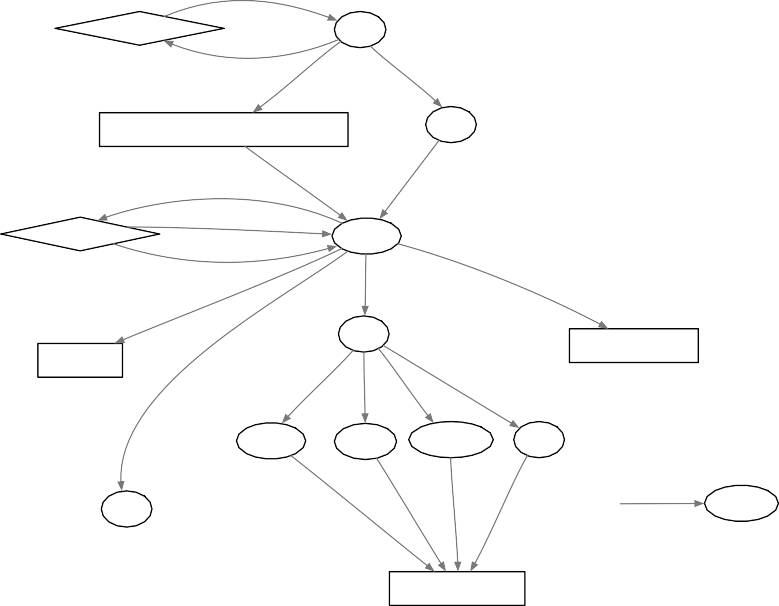
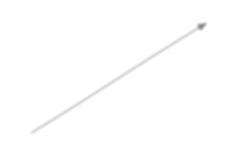
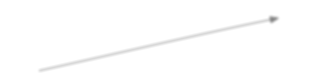
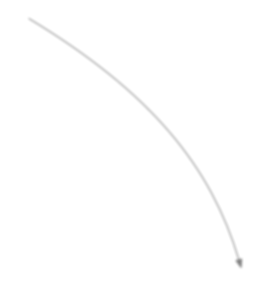
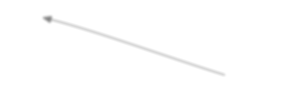
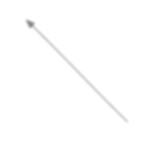
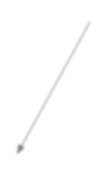
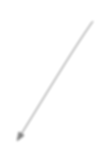
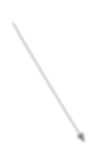
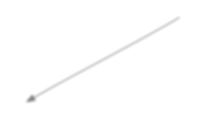
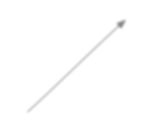
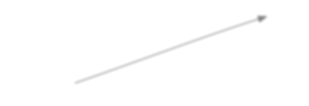
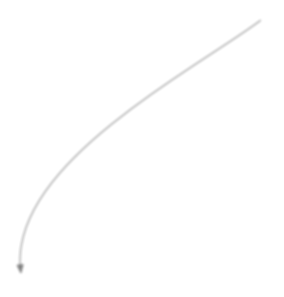
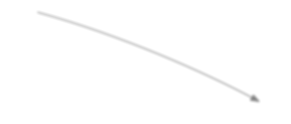
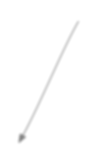
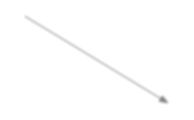
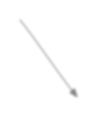
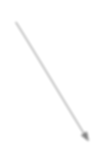
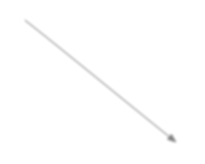
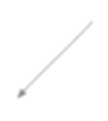
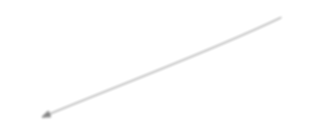
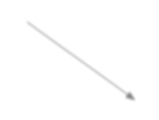
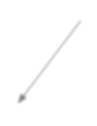
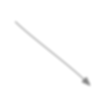
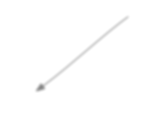
*37. 地图*

<未知> 缺少审计数据！

缔约方会议 *38. 写* /usr/local/sbin/nginx

图 26. 从战役 F-1 重建的情景图。

攻击 L-3。在这次攻击中（图 29），文件 dropbearLINUX.tar 被下载并解压。接下来，执行程序dropbearkey创建三个密钥，由程序dropbear读取，随后执行渗透。



*1.接收*

129.55.12.167:8000

恩金克斯

1. *发送*
2. *写*

*4. 叉子*

/var/tmp/nginx/client\_body\_temp/dropper

嘘

*9.执行*

*7. 写*

*6.执行*

我不通过

*5. 叉子*

*12. 发送*

*8. 叉子*

10. 写

129.55.12.167:443

*13. 收到*

滴管

11. 阅读

*29. 收到*

*14. 叉子*

*28. 写*

*27. 写*

嘘

/tmp/mailer/邮递员

/tmp/系统管理员

*30. 叉子*

129.55.12.167:6666

*15. 叉子*

*17. 叉子*

*25. 叉子*

*23. 叉子*

*19. 叉子*

*35. 叉子*

129.55.12.167:2525

*32.执行*

*37.执行*

网络统计

我是谁

无名

主机名

-什么

*39. 发送*

*34. 发送*

18. 写

*22. 写*

系统管理员

*31. 叉子*

嘘

*16. 写*

20. 写

*21. 写*

*24. 写*

*26. 写*

嘘 *36. 叉子*

邮递员

*33. 阅读*

*38. 阅读*

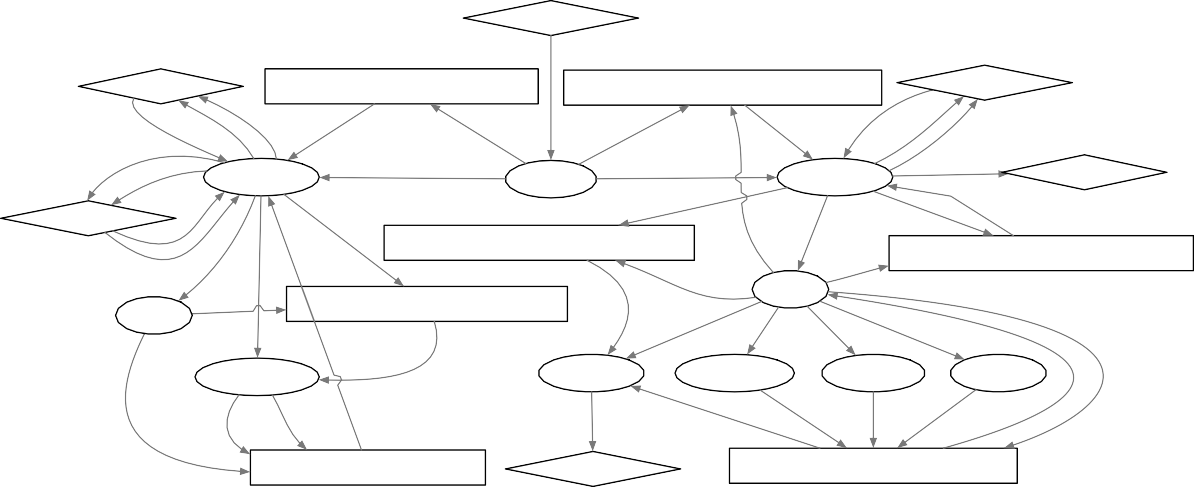
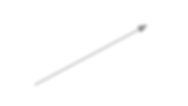
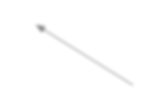
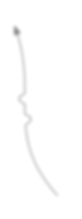
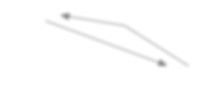
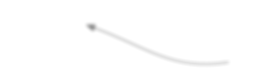
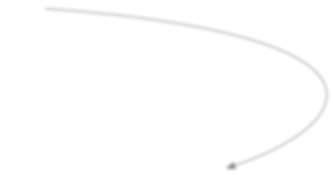
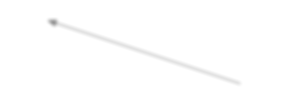
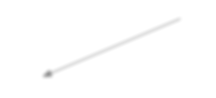
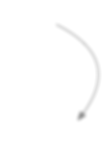
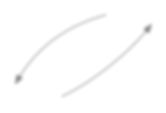
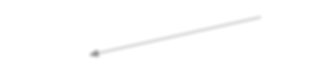
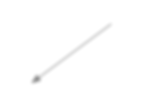
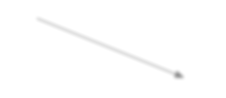
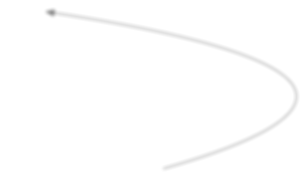
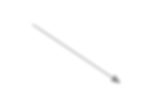
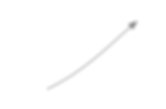
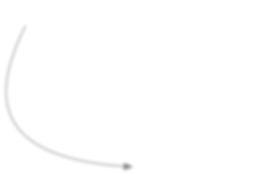
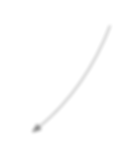
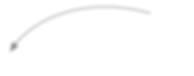
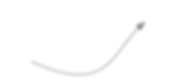
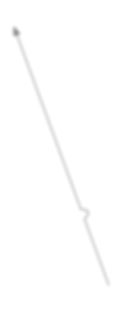
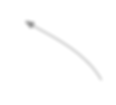
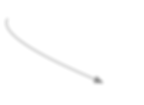
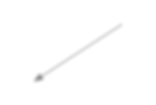
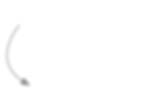
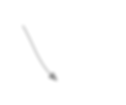
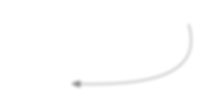
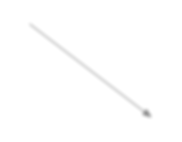
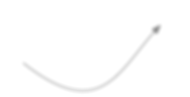
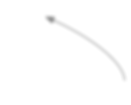
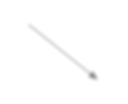
/tmp/mailer/mailer.log

/tmp/syslog.dat

/tmp/netrecon.log

/tmp/netrecon

图 27. 从战役 F-2 重建的场景图。



129.55.12.167:8000

*1.接收*

129.55.12.167:443

C:\\用户\\用户1\\下载\

\mozillanightly

C:\\Users\\User1\\Downloads\\firefox\

\mozillanightly

129.55.12.167:4430

*15. 接收 14. 发送*

*5.发送 4.执行* 2.写 21. 写 *23. 执行 47. 接收*

*16. 发送*

*6.发送*

mozillanightly

*3. 叉子*

火狐

*22. 叉子*

mozillanightly

*27. 发送*

*51.发送*

*24. 发送*

*26. 阅读*

*25. 写*

129.55.12.167:443

129.55.12.167:4430

*17. 收到*

*7. 收到*

*18. 叉子*

*36. 写*

*8. 写*

C:\\Users\\User1\\Downloads\\firefox\

\mnsend.exe

*49. rm*

*28. 叉子*

C:\\Users\\User1\\Downloads\\firefox\

\倦怠.bat

*48. rm*

*9. 叉子*

*46. rm* 命令文件

命令文件

*19. rm*

C:\\用户\\用户1\\下载\

\photosnap.exe

*38.执行*

*37. 前叉 29. 前叉 31. 前叉 33. 前叉*

照片快照程序

*10.执行*

邮件发送程序

主机名.exe

哇.exe

网络统计工具

*50 分钟*

*13. 阅读*

*20. rm*

*12. chmod 11. 写*

*40. 发送*

*39. 阅读*

*30. 写 32. 写 34. 写*

*35. 阅读*

C:\\用户\\用户1\\下载\

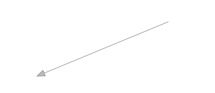
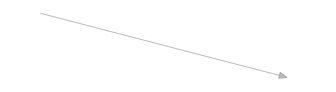
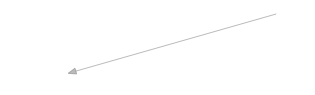
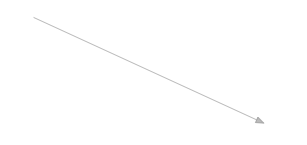
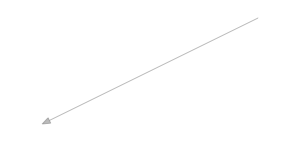
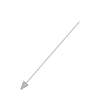
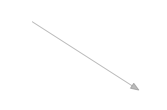
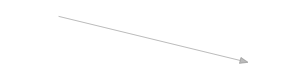
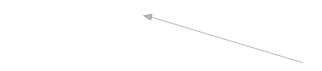
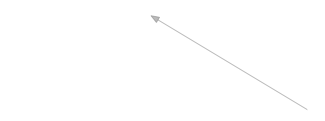
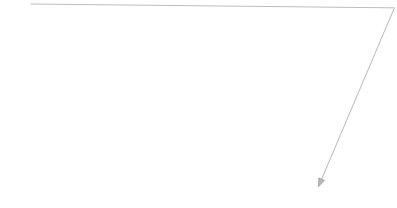
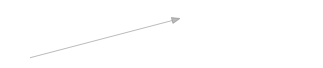
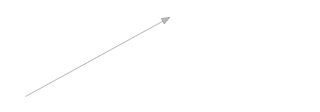
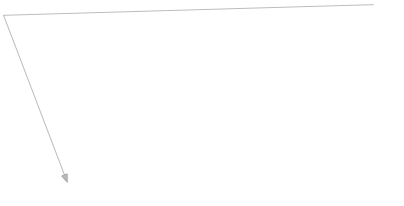
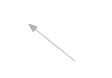
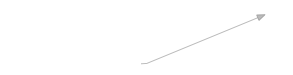
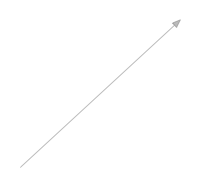
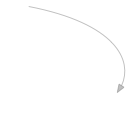
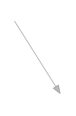
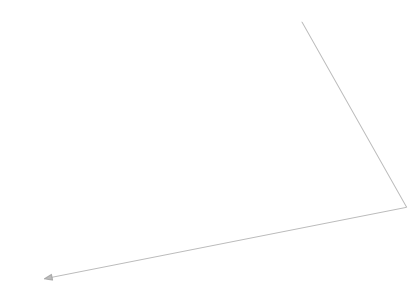
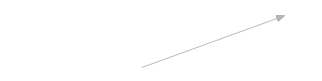
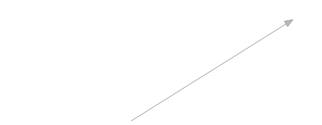
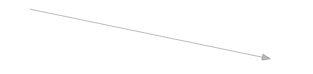
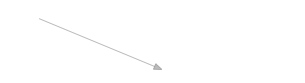
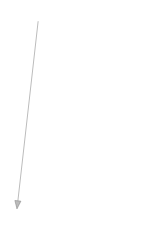
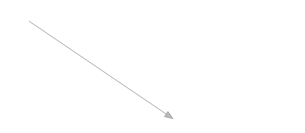
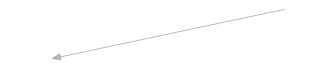
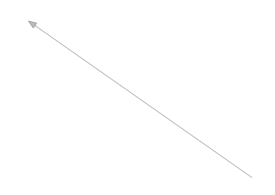
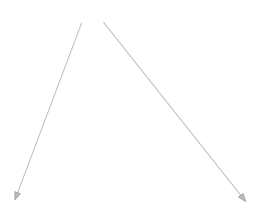
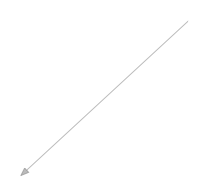
\pic.png

129.55.12.167:7770

C:\\Users\\User1\\Downloads\\firefox\

\4662.log

图 28. 从战役 W-1 重建的场景图。



8.chmod是什么

7. 写

需要

4.阅读

5. 写

6. chmod

2.写

3. 叉子

10.执行

scp

1. 叉子

15. 执行

38. 叉子

须藤

42. 叉子

9. 叉子

14.叉子

小熊钥匙

小熊

1. 写
2. 写
3. 写

/etc/dropbear/dropbear\_ecdsa\_host\_key

1. 读
2. 读
3. 读

/etc/dropbear/dropbear\_dss\_host\_key

1. 写
2. 写
3. 写

43. rm

/etc/dropbear/dropbear\_rsa\_host\_key

撕碎

1. rm
2. rm

rm

25. 阅读

24. 阅读

19. 叉子

23. 阅读

26. 阅读

22. 阅读

1. 读
2. 读
3. 写

小熊

1. 读
2. 读

30. 写

1. 写
2. 写

/proc/sys/内核/

ngroups\_max/运行/utmp

1. 发送 35. 发送 37.发送
2. 发送 36. 发送

128.55.12.167:38509

128.55.12.167:40246

128.55.12.167:39490

128.55.12.167:39335 128.55.12.167:38510

/var/dropbear\_latest/dropbearLINUX/dropbear

/运行/utmp

/var/log/lastlog

/dev/ptmx

/var/log/wtmp

/etc/本地时间

/proc/loadavg

/etc/nsswitch

/etc/shells

/proc/中断

/%/timer\_list

/etc/影子

/proc/vmstat

/var/dropbear\_latest/dropbearLINUX.tar

/var/dropbear\_latest/dropbearLINUX/dropbearkey

图 29. 从 L-3 战役重建的场景图。

159

## 附录 B

**福尔摩斯攻击情景**

场景 2：特洛伊木马。这种攻击场景（图 30）从用户下载恶意文件开始。然后用户执行该文件。执行导致与攻击者机器的 C&C 通信通道。然后，攻击者启动一个 shell 并执行一些信息收集命令，例如 hostname、whoami、ifconfig、netstat 和 uname。最后，攻击者窃取了一些秘密文件。请注意，这种攻击场景类似于前面讨论的路过下载场景，只是最初的攻击是通过用户下载的程序进行的。此场景的检测结果的另一个重要见解是，它缺少与攻击的 C&C 通信（连接）和最终清理（取消链接）活动相关的重要事件。即使有如此不完整的数据，Holmes 也能够将其标记为 APT，因为威胁分数超过了阈值。

场景 3：木马。在这种攻击中（图 31），用户被说服通过 Firefox 下载恶意木马程序（文本编辑器）。接下来，用户将可执行文件移动到另一个目录，更改其名称（tedit），最后执行它。执行后会创建一个 C&C 通道，并向攻击者提供一个反向 shell。攻击者启动 shell 提示并执行信息收集命令，如主机名、whoami、ifconfig 和 netstat。然后攻击者部署另一个恶意文件，泄露信息，最后清理他的足迹。此场景与 Trojan-1 不同，因为它具有远程部署新恶意可执行文件的附加活动。

160

E H

**老的** *M*

E I

*L*

A D

## 附录 B（续）

*H*

*L*

E F

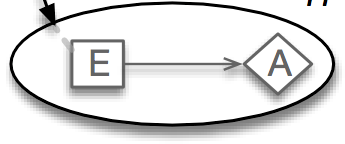
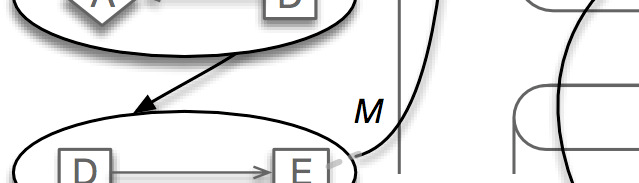
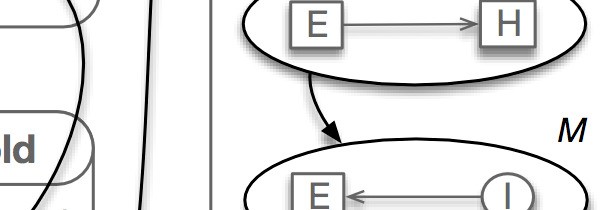
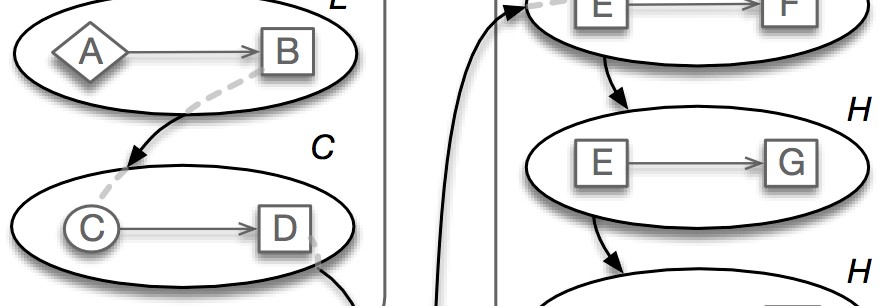
A B

*H*

*C* E G

C D

*H*



**APT阶段：内部侦察**

**APT 阶段：初始妥协**

叉子

*不受信任的读取*

收视率

*敏感命令*

叉子

*不受信任的文件执行*

*敏感命令*

执行

叉子

*敏感命令*

**APT阶段：建立立足点**

读

*敏感阅读*

*C&C 通讯*

发送

*壳牌执行*

执行

**APT阶段：渗透**

*H*

发送

*敏感泄漏*

图 30. 场景 2 的 HSG。符号：A= 不受信任的外部地址； B=火狐； C=木马文件（diff）； D=已执行的木马进程； E= /bin/破折号； F = ifconfig; G=主机名； H=网络统计；我=密码.txt；

*M*

D E

E A

场景 4：间谍软件。这种攻击（图 32）从红队攻击 Firefox 开始。然后，受害者主机上的用户加载了一个被劫持的远程 URL。接下来，执行来自 URL 的 shellcode 以连接到 C&C 服务器，从该服务器下载恶意二进制文件，将其写入磁盘并执行。恶意二进制文件的执行会导致 C&C 通信的反向 shell 通道。然后攻击者运行了shell命令，

导致新的 cmd.exe 进程和与 C&C 服务器的新连接。操作员运行侦察命令（主机名、whoami、ipconfig、netstat、uname）。攻击者随后窃取了 password.txt 文件，然后将其删除。最后，恶意二进制文件会删除一个批处理文件，删除攻击足迹，包括恶意二进制文件本身。

场景 5.1：永恒之蓝。此 APT 利用 Windows 中易受攻击的 SMB [137] 服务。在这种情况下（参见图 33），Meterpreter [138] 与最近实施的 Eternal Blue 漏洞利用和双脉冲星反射加载功能一起使用。攻击者利用了目标端口 445 上的监听 SMB 服务。然后在目标上下载并执行一个 shellcode。 shellcode 将进程注入到 lsass.exe 进程中。然后lsass.exe启动rundll32.exe，它连接到C&C服务器并下载并执行Meterpreter。接下来，Meterpreter 泄露了一个敏感文件并清除了 Windows 事件日志。

场景 5.2：大鼠。在此攻击中（图 34），Firefox 导航到恶意网站并被利用。然后，将远程访问木马 (RAT) 上传到受害者的机器并执行。执行后，与 C&C 服务器发生连接，恶意 RAT 被删除。这种攻击场景是不完整的，并没有造成任何伤害。

场景 6：Web-Shell。本次攻击的假设（图 35）是 Nginx Web 服务器存在漏洞，可让攻击者访问在服务器上运行任意命令（类似于 Shellshock 漏洞）。结果，攻击者泄露了敏感文件。这里的重要见解是，通过捕获持续攻击的足够强的 APT 信号

通过 TTP 匹配，Holmes 准确地标记了 APT，即使缺少关键的 APT 步骤（在这种情况下是初始妥协）。

场景 7.1：老鼠。在设置期间安装了易受攻击的 Nginx 服务器。攻击者通过抛出恶意 shell 代码来利用 Nginx 服务器。 Nginx 运行恶意 shell 代码，导致下载和执行恶意 RAT。接下来，RAT 连接到 C&C 服务器并向远程攻击者授予管理权限。攻击者远程执行一些命令。然后它会部署一些恶意 Python 脚本并泄露信息。这种攻击的 HSG 如图 36 所示。

##### APT 阶段：初始妥协

*不受信任的读取*

*不受信任的文件执行*

*执行*

C 执行 D

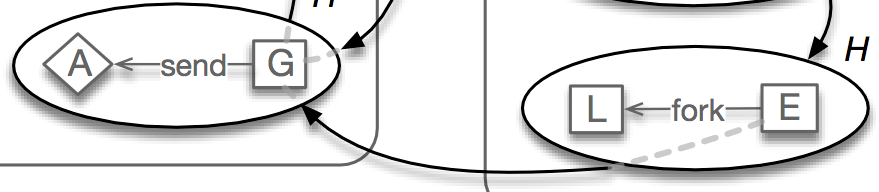
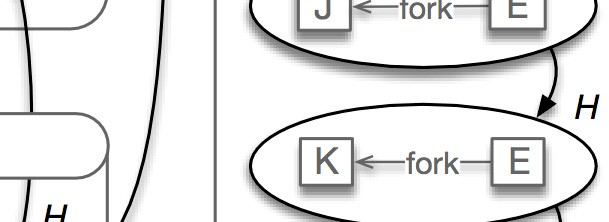
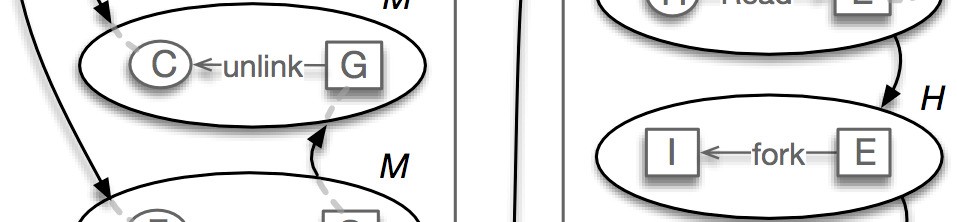
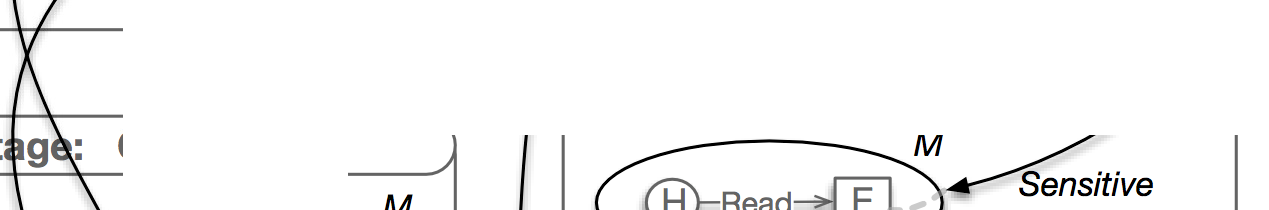
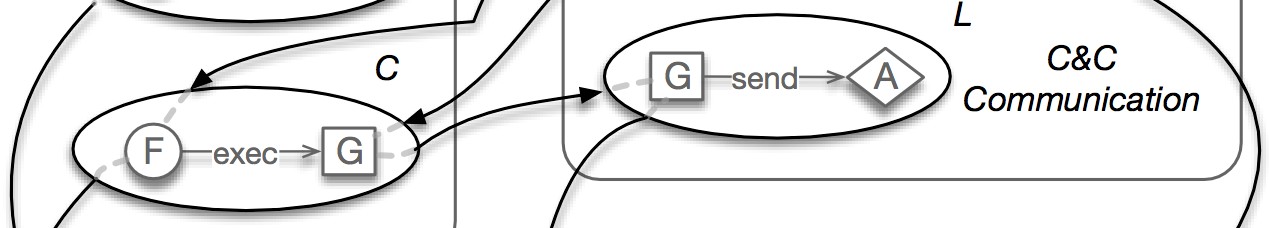
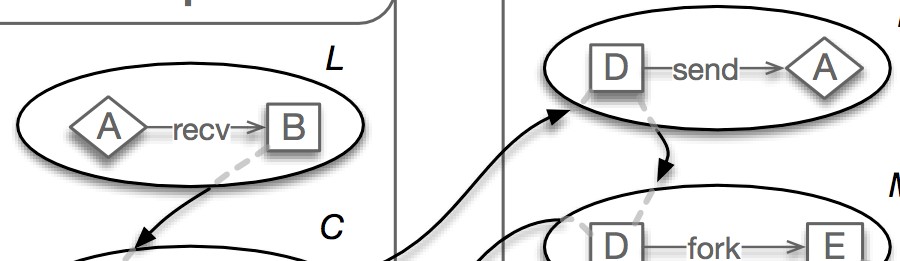
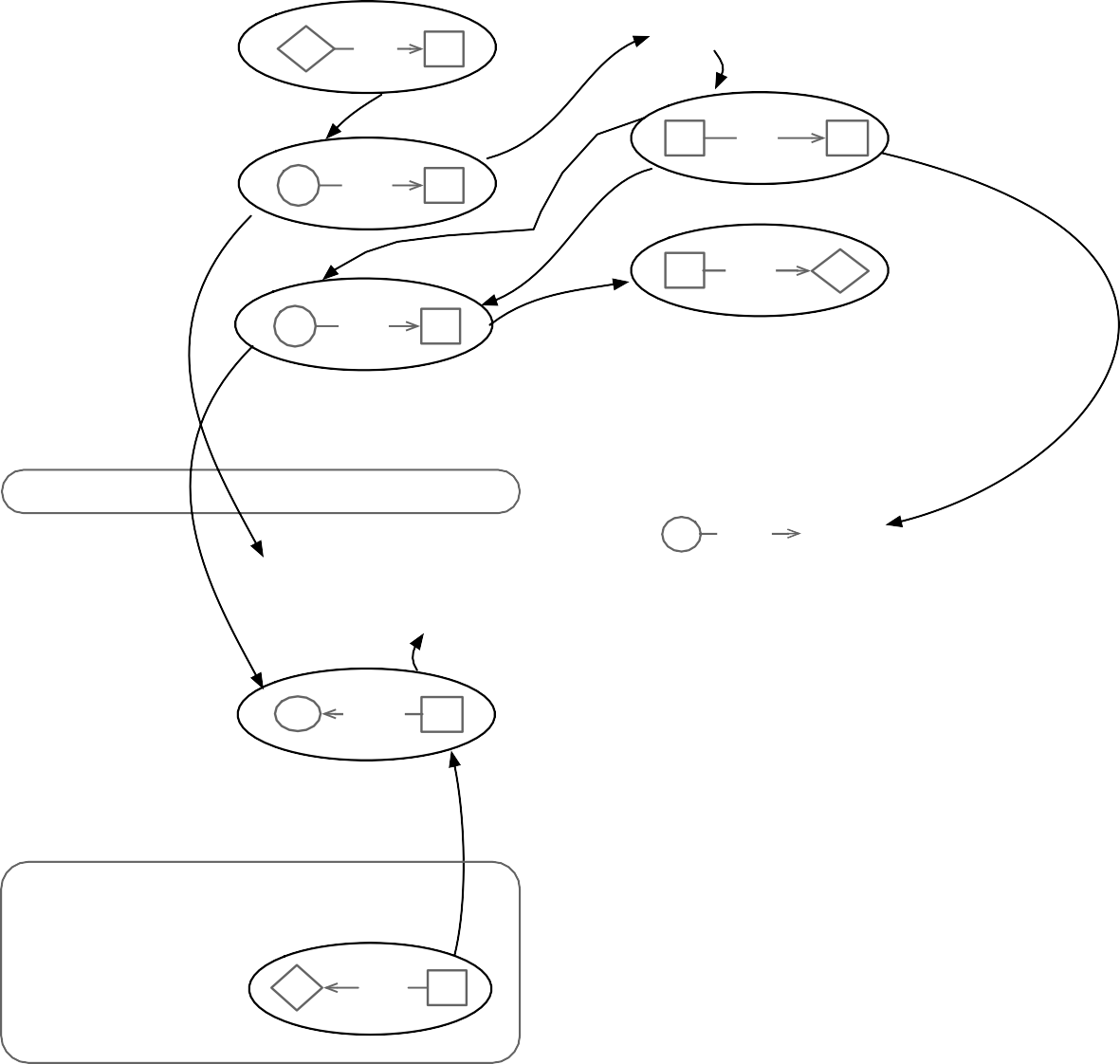
*L*

*C* G 发送 A *C&C通讯*

F 执行 G

##### APT阶段：建立立足点

*L*



*C&C 通讯*

*L* D 发送 A

A 收视率 B

*C* D 叉子 E

*M*

*壳*

*不受信任的文件执行*

##### APT 清理

**内部侦察**

**日：**

*不受信任的文件 Rm*

C 取消链接 G

*H*

*M* I 叉子 E

*不受信任的文件 Rm*

F 取消链接 G

*H*

##### APT阶段：国际

*读*

*M*

*M* H读 E *敏感的*

*敏感命令*

*敏感命令*

J 叉子 E

*H*

K 叉子 E

*H*

##### APT阶段：渗透

*敏感泄漏*

A 发送 G *H*

L 叉子 E

*敏感命令*

*敏感命令*

图 31. 场景 3 的 HSG。符号：A= 不受信任的外部地址； B=火狐； C=木马文件（tedit）； D=已执行的木马进程； E= /bin/破折号； F=恶意可执行文件（py）； G = 执行的恶意进程； H=密码.txt；我=哇哦； J = ifconfig; K=网络统计； L=无名；

##### APT 阶段：初始妥协

*不受信任的读取*

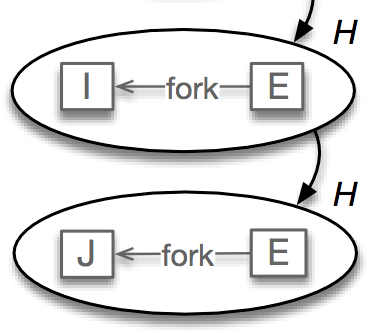
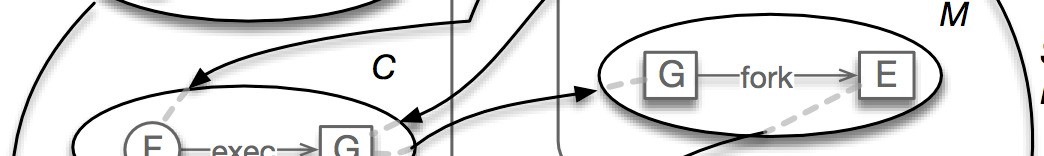
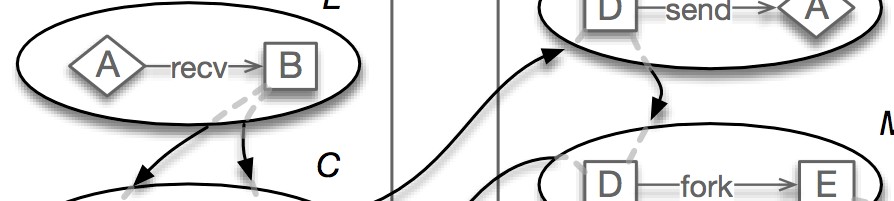
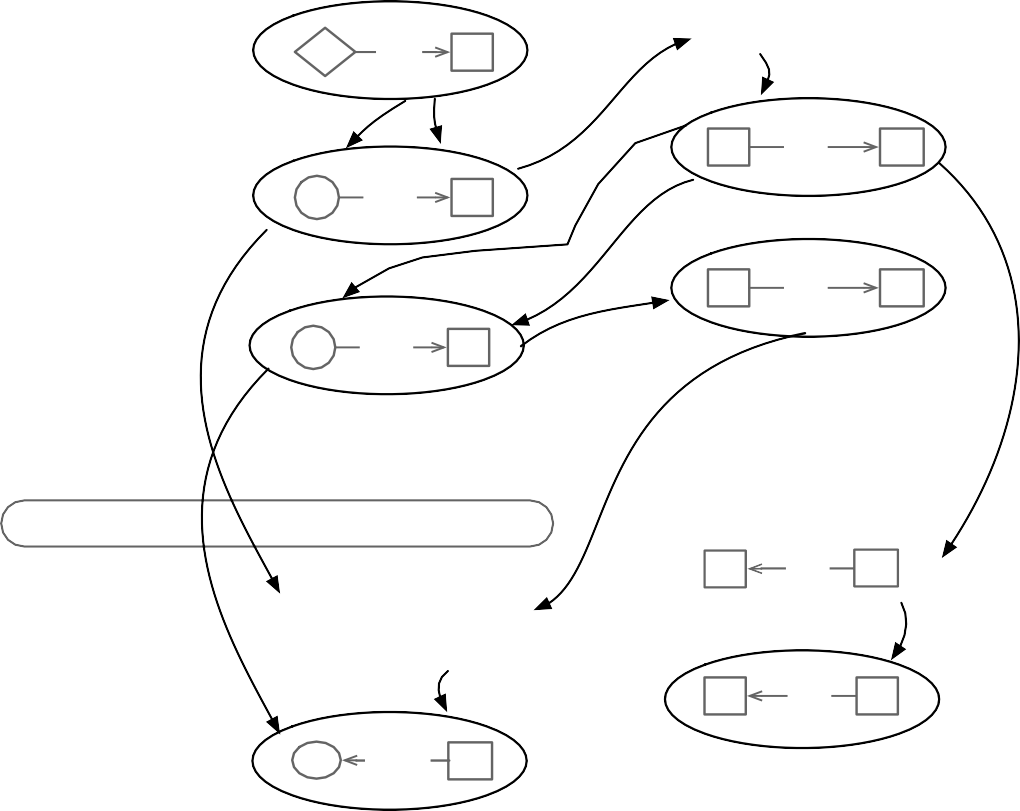
*不受信任的文件执行*

C 执行 D

*不受信任的文件执行*

##### APT阶段：建立立足点

*L*



*C&C 通讯*

*L*

D 发送 A

A 收视率 B

*C* D 叉子 E

*M*

*壳牌执行*

*壳牌执行*

*M*

*C* G 叉子 E

F 执行 G

##### APT 清理

**日：**

*M*

*不受信任的文件 Rm*

C E

取消链接

*M*

*不受信任的文件 Rm*

F 取消链接 E

##### 年龄：在 经济

*敏感命令*

**外部R**

**APT街**

*H*

H 叉子 E *S*

*敏感命令*

*H*

I 叉子 E

*敏感命令*

*H*

J 叉子 E

图 32. 场景 4 的 HSG。符号：A= 不受信任的外部地址； B=火狐.exe; C=恶意删除文件（procman.exe）； D= 执行的恶意软件进程； E=cmd.exe； F=恶意批处理文件（burnout.bat）； G=已执行的批处理； H=主机名；我=哇哦； J=ipconfig；

*L*

A 收视率 B B 发送 A

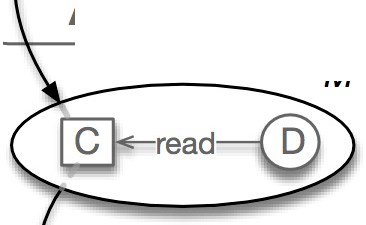
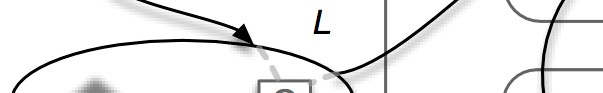
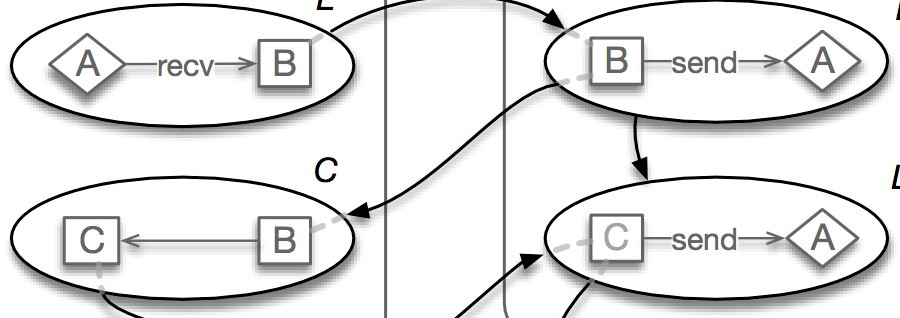
*C*

C B C 发送 A

*L*

收视率 C

C 读 D



**APT 阶段：初始妥协**

*不受信任的读取*

**APT阶段：建立立足点**

*L*

*C&C 通讯*

*L*

*不受信任的注入* 注入

*C&C 通讯*

*不受信任的读取*

A

**APT阶段：内部侦察**

*M*

*秘密阅读*

**APT 阶段：清理**

*H*

*清除日志*

**APT阶段：渗透**

*H*

*秘密泄露*

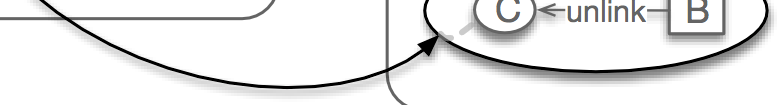
图 33. 场景 5.1 的 HSG（永恒之蓝）。符号：A= 不受信任的外部地址； B=lsass.exe； C=运行dll32.exe； D=密码.txt； E=Winevt 日志；

E 取消链接 C

C 发送 A

D 发送 A

A 收视率 B



**APT 阶段：初始妥协**

*L*

*不受信任的读取*

**APT阶段：建立立足点**

*L*

*C&C 通讯*

*不受信任的文件执行*

**T阶段：清理**

*M*

*不受信任的文件 Rm*

图 34. 场景 5.2 的 HSG。符号：A= 不受信任的外部地址； B=火狐.exe; C=恶意删除文件（spd.exe）； D= 执行的恶意软件进程；

*C*

**美联社**

C 执行 D

C 取消链接 B

*L*

A B

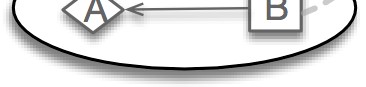
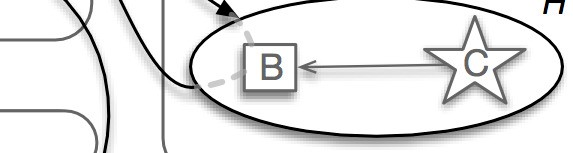
B A

*M*

D B **APT阶段**

*H*

B C



**APT 阶段：初始妥协**

*不受信任的读取*

收视率

**APT阶段：内部侦察**

**APT阶段：建立立足点**

*L*

发送

*C&C 通讯*

*敏感阅读*

读

**: 权限提升**

setUid(设置Uid)

*开关 SU*

**APT阶段：渗透**

*敏感泄漏*

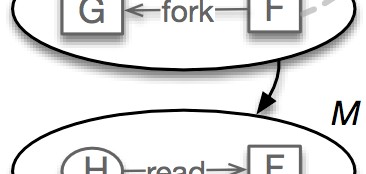
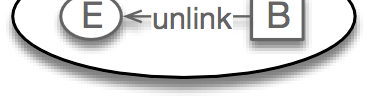
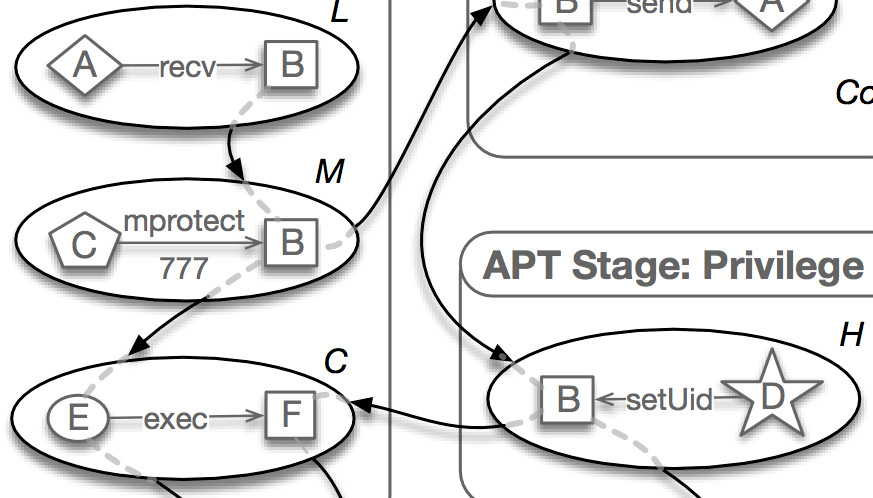
发送

图 35. 场景 6 的 HSG。符号：A= 不受信任的外部地址； B=Nginx； C=根用户ID； D=密码.txt；

*H*

A B

## APT阶段：建立立足点



**APT 阶段：初始妥协**

*L*

B A

*不受信任的读取*

*使内存执行*

*C&C通讯*

## 升级

*L* 发送

A 收视率 B

*C*

*M*

保护

C 777 B **APT阶段：特权**

*H*

*C*

E 执行 F B setUid(设置Uid) D

*不受信任的执行者*

*开关 SU*

## APT阶段：实习生

*H*

*敏感命令*

G 叉子 F

*M*

H 读 F

*敏感阅读*

## ： 清理

*不受信任的文件 Rm*

**APT阶段**

**侦察** *M*

E 取消链接 B

## APT阶段：渗透

*H*

F 发送 A

*敏感泄漏*

图 36. 场景 7.1 的 HSG。符号：A= 不受信任的外部地址； B=Nginx； C=内存； D=根用户ID； E=恶意丢弃的文件（py）； F = 执行的恶意软件进程； G=无名； H= /etc/影子；

169

#### 附录 C

**POIROT CTI 描述**

在本节中，我们提供了每个恶意软件的简要历史记录以及我们用来构建查询图的相应报告中的陈述摘要。

njRAT。 njRAT 是一种公开可用的远程访问木马 (RAT)，它使攻击者可以完全控制受害系统。尽管 njRAT 的源代码是公开的，但利用 njRAT 的攻击主要针对位于或专注于中东地区的政府、电信和能源部门的组织。执行恶意软件时，它会尝试从扩展名为“.exe.config”（边缘 1）的文件中读取其配置。 njRAT 恶意软件将记录的击键存储在“.tmp”文件（边缘 2）中，并写入“.pf”文件（边缘 3）。为了获得持久性，njRAT 恶意软件会创建一些自身的副本（边缘 4 和 8）。执行后（边缘 5 和 6），其中一个副本写入“.pf”文件（边缘 7）。 njRAT 恶意软件还会启动位于（边缘 9）的 netsh 进程，这会导致创建另一个“.pf”文件（边缘 10）。最后，恶意软件将一些注册表值（边缘 11-13）和信标设置到位于 217.66.231.245（边缘 14）的 C&C 服务器。

副狗。根据 FireEye 的报告，DeputyDog 指的是一种似乎以日本组织为目标的恶意软件。我们从该恶意软件的报告中提取的查询图如图 19 所示，并在 5.2 中进行了描述。

乌鲁布罗斯。 Uroburos、ComRAT、Snake、Turla 和 Agent.BTZ 都指的是一个 rootkit 家族，该家族对美国军用计算机的最严重破坏负责。这

恶意软件首先删除两个 Microsoft Windows 动态库（边缘 1 和 2）并调用 rundll32.exe（边缘 3）来安装这些库（边缘 4 和 5）。然后，要在启动过程中启动，恶意软件会创建一个注册表项（边缘 6）。该恶意软件创建三个日志文件（边缘 7-9）并删除一组文件（边缘 10-14）。

卡巴纳克。 Carbanak 是一个远程后门，用于提供对受感染机器的远程访问。攻击者的主要动机似乎是经济利益，这导致累计损失高达 10 亿美元 [75]。入侵最初是使用看似合法的银行通信的鱼叉式网络钓鱼电子邮件（优势 1）。漏洞利用后，Carbanak 将自身复制到名为“svchost.exe”的“%system32%”（边缘 2-4）并删除由漏洞利用有效负载创建的原始文件（边缘 5）。为了访问自动运行权限，恶意软件会创建一个名称格式为“<ServiceName>Sys”的新服务，其中 ServiceName 是随机选择的任何现有服务（边缘 6）。 Carbanak 创建一个具有随机名称和 .bin 扩展名的文件，其中存储要执行的命令（边缘 7）。然后，恶意软件从注册表项（edge 8）和 Mozilla Firefox 配置文件（edge 9）获取代理配置。最后， Carbanak 与其 C&C 服务器（edge 10）进行通信。

尘土飞扬的天空。 DustySky 是一种多阶段恶意软件，其主要目标是出于政治目的收集情报。恶意软件样本伪装成 Microsoft Word 文件，一旦执行（边缘 1），就会打开阿拉伯语言的诱饵 Microsoft Word 文档（边缘 2 和 3），同时恶意软件在后台执行情报收集。对于 VM evasion，dropper 检查是否存在一些 DLL 文件，特别是 vboxmrxnp.dll 和 vmbusres.dll 表示存在 VirtualBox（边缘 4 和 5）和 vmGuestlib.dll 表示存在

VMware（边缘 6）。 DustySky Core 下降到 %TEMP%（边缘 7&8&9），并且击键

日志保存到 %TEMP%\temps（边缘 10）。

海莲花。 OceanLotus，也称为APT32，被认为是一个针对东南亚国家的越南APT组织。执行此恶意软件后（边缘 1），诱饵文档和橡皮擦应用程序被丢弃（边缘 2 和 3），诱饵文档在 Microsoft Word 中午餐（边缘 4 和 5）。然后，可执行文件解密其资源并丢弃合法 Symantec Network Access Control 应用程序 (edge 6)、加密后门 (edge 7) 和恶意 DLL 文件 (edge 8) 的副本。默认情况下，经过签名且合法的 Symantec 应用程序会将所有库加载到同一文件夹中。在这种情况下，执行后（边缘 9 和 10），此应用程序会加载已放置在同一目录（边缘 11）中的恶意 DLL 文件。然后它读取后门文件（edge 12），从而访问注册表（edge 13），加载 HTTPProv.dll 库（edge 14），并创建注册表项（edge 15）。最后，恶意软件连接到它的母舰（边缘 16 和 17）。

鹰眼。 HawkEye 是一种恶意软件即服务凭证窃取恶意软件，是 APT 活动中的流行工具。该恶意软件的新变种使用进程空心将其代码注入合法签名的 .NET 框架可执行文件，并附带许多复杂的函数来逃避检测。这个新变种通常以压缩文件的形式交付，在解压缩（边缘 1 和 2）和执行（边缘 3）之后，它会产生一个子进程（边缘 4），称为 RegAsm，它是来自 Microsoft .Net 框架的程序集注册工具. HawkEye 将 PE 文件提取到其内存中，然后将其注入 RegAsm 进程。休眠 10 秒后，RegAsm 进程生成两个名为 vbc 的子进程，它们都来自

.Net 框架也是如此（边缘 5 和 6）。其中一个过程收集浏览器的凭据，而另一个过程专注于电子邮件和即时消息 (IM) 应用程序。我们添加了一个节点，类型为文件或注册表，对应于报告中提到的每个浏览器（边缘 7-18）或电子邮件/IM（边缘 19-26）应用程序的名称。请注意，这些应用程序可能会将攻击者感兴趣的一些机密信息存储到文件或注册表中，这就是我们没有将搜索限制为仅文件或注册表的原因。收集的凭据

定期保存到 %temp% 目录（边缘 27 和 28）中的 ∗.tmp 文件中，而过了一会儿，

RegAsm 进程将这些 tmp 文件的全部数据读入其内存（边缘 29 和 30）并立即删除它们（边缘 31 和 32）。最后，RegAsm 查找机器的公共 IP

从“http[s]:\\whatismyipaddress.com\”网络服务（边缘 33 和 34），然后渗出

收集信息到攻击者的电子邮件地址（edge 35）。

需要注意的是，在 HawkEye 的查询图中，有一些标签和类型完全相同的节点，例如 F&G 或 J&K。但是，这些节点会根据它们与其他实体的依赖关系与不同的节点对齐。例如，节点 F 与浏览器应用程序交互，而节点 G 与电子邮件/IM 应用程序交互。此外，浏览器或邮件应用程序节点的对齐与它们在系统上的安装无关。许多这些应用程序没有安装在测试机器上，但是当恶意软件试图检查这些应用程序是否安装在系统上时，它会启动一个被 Poirot 检测到的 OPEN 事件。

173

**附录 D**

### 版权许可声明

DocuSign 信封 ID：0BE9773A-0101-4F69-B24E-04E6C1D9E1C8

11925; 11928; 11931; **参考文件同意书**



## USENIX Security '17：第 26 届 USENIX 安全研讨会

**2017 年 8 月 16 日至 18 日，加拿大不列颠哥伦比亚省温哥华**

**出版授权和许可**

USENIX 协会（“USENIX”）的一项主要任务是为新知识的创造和传播提供服务。为了促进这一过程，USENIX 允许作者保留其作品的版权所有权，只要求 USENIX 被授予出版该作品的权利。因此，双方同意如下：

1. 授予出版权。您特此授予 USENIX 协会（“USENIX”）发表和重新发表临时标题为：

**论文标题：**

SLEUTH：从 COTS 审计数据重构实时攻击场景

**请在此处填写标题**

（“文件”）以印刷形式、电子形式和视听格式（“权利”）。 USENIX 的权利是非排他性的、全球性的、永久的和不可撤销的，但也是不可再许可和不可转让的。您在此授予 USENIX 的权利之一是成为论文的第一出版商的权利。 USENIX 最初计划在以下出版物中发布该文章：第 26 届 USENIX 安全研讨会 (USENIX Security ’17)（“原始出版物”）的论文集，并将文件永久挂载在 USENIX 网站上显示，

但也可以通过媒体、现场直播以及 USENIX 选择的任何其他电子或视听格式传播论文以及论文的口头陈述。

1. 发布。您保留本文中的所有权利，但须遵守上文第 1 节中授予 USENIX 的非专有权利。
2. 通知请求。 USENIX 将在原始出版物的第一页包含以下通知：“本作品的版权由作者保留。”
3. 作者随后出版。您必须在随后发布的论文或论文的修订版本中包含 USENIX 原始发布的通知。
4. 保证。您保证：
   1. 您是论文的唯一作者和所有者以及与之相关的权利，或者，如果论文是共同作者的作品，则论文的所有所有者都已签署并提交了本文件的副本。如果论文归您的雇主所有，您保证您已获得签订本协议并授予 USENIX 此处所述权利的许可。
   2. USENIX 无需任何其他方的许可即可行使此处授予的权利。
   3. 本文不包含任何诽谤或其他非法内容。
   4. 本论文不侵犯他人权利。
   5. 论文及其内容为您原创。
   6. 该论文目前未提交在其他地方发表，以前也没有在其他地方发表过。
   7. 如果论文包含来自其他受版权保护的材料的重要摘录，则已获得这些材料的所有版权所有者的书面许可，此类许可的副本附于本文档，并且在论文本身中已给出适当的信用和归属。
5. 撤回访问权。 USENIX 保留随时以任何理由将其控制下的内容和出版物从公众访问中删除的权利。有时，USENIX 会收到通知，声称其网站、其自己的出版物或其发布的论文和会议记录中的内容可能侵犯第三方权利或可能违反某些法律或法规。 USENIX 通常不对这些指控的实质采取立场。但是，当这种情况发生时，USENIX 将通知作者这些指控，并可能从公众访问中删除已识别的内容或出版物。 USENIX 可在其认为适当时自行决定恢复该材料。 USENIX 也可能将包含指控的通知发送给 Lumen (lumendatabase.org)。

**USENIX 行为准则准则**

演讲者对其演讲的内容负责，但 USENIX 要求演讲者意识到潜在的冒犯行为、语言或图像，并考虑是否有必要传达他们的信息。如果他们决定加入，USENIX 会要求他们在演讲开始时警告听众，并让他们有机会离开房间以避免看到或听到材料。

您在下方的签名授予 USENIX 发布的权利，同时也表明您已阅读 USENIX 活动的行为准则和演讲者指南，网址为[www.usenix.org/conferences/coc.](http://www.usenix.org/conferences/coc)

2017 年 6 月 28 日 I 09:09 PDT

日期：

作者签名

纳希德侯赛因

作者的印刷体姓名

USENIX 授权签名

16680;

**结业证书**

信封 ID：0BE9773A01014F69B24E04E6C1D9E1C8 状态：已完成 主题：请 DocuSign：USENIX Security '17 审稿同意书

源信封：

文档页数：1 签名：1 信封发起人：

补充文件页数：0 首字母：0 同意书证书页数：1

自动导航：已启用

EnvelopeId 标记：已启用

时区：(UTC-08:00) 太平洋时间（美国和加拿大）

付款：0 2560 9th St, Ste 215

加利福尼亚州伯克利 94710[同意@usenix.org](mailto:consent@usenix.org)

IP地址：130.245.137.80

**记录跟踪**

状态：原始

2017 年 6 月 28 日上午 9:05:26

持有人：同意书

[同意@usenix.org](mailto:consent@usenix.org)

地点：DocuSign

**签名者事件 签名 时间戳**

纳希德侯赛因[mdnhossain@cs.stonybrook.edu](mailto:mdnhossain@cs.stonybrook.edu)

安全级别：电子邮件、帐户身份验证（无）



使用 IP 地址：130.245.137.80

已发送：2017 年 6 月 28 日上午 9:05:26 已查看：2017 年 6 月 28 日上午 9:06:11 签名：2017 年 6 月 28 日上午 9:09:20

**电子记录和签名披露：**

不通过 DocuSign 提供

**亲自签名者活动 签名 时间戳**

**编辑交付活动 地位 时间戳**

**代理交付事件 地位 时间戳**

**中介交付事件 地位 时间戳**

**认证交付活动 地位 时间戳**

**复本活动 地位 时间戳**

**公证人事件 签名 时间戳**

**信封摘要事件 地位 时间戳**

已发送信封 散列/加密 2017 年 6 月 28 日上午 9:05:26

认证交付 安全检查 2017 年 6 月 28 日上午 9:06:12

签署完成 安全检查 2017 年 6 月 28 日上午 9:09:20

完全的 安全检查 2017 年 6 月 28 日上午 9:09:20

**付款事件 地位 时间戳**

**同意发表**

# 计算机科学讲义

**书名或会议名称： 。1 4.t 。在 t.erna i.o.na l..on .fe re n.ce .on .I.nf.o.r.m .tion .S y.s.te .m s..**

**卷编辑器名称：R .K .Sh y.am .as u .n.da r,。V .in .od .G a.na .at.h.y,。a .d .Tr.ent .J.ae g e.r..........**

**投稿标题：P .r.oP a .tro l.:.A .ttack .I.nve s ti.gat.io .n..ia .Ex.tra c te .H .i .h-.v e .l 。塔 s k.s.......**

**作者全名：.Sa d .eg h .M .o m .e .i .M .il.aje.rd.i, Bir .anu .E .s.he te,.R.ig.el。G.jo.m.em.o,。V.en k.t.V**

###### 通讯作者姓名、单位地址和电子邮件：

.a d e.噢，天啊 .M .唵 .e n .i.M .这.a.是 .三 .i,. " U .尼夫.呃.西耶 .的. 伊琳.瓦斯. a t.C .我知道了 .去，. .200 .W . .H .里.儿子 .S t,. . . . . . . . . . .

美国伊利诺伊州芝加哥 60607",[smomen 2@uic . edu欢迎收看](mailto:smomen2@uic.edu)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

当作者不止一个人时，除非另有说明，否则本协议中使用的“作者”一词将集体适用。

出版商打算以 Springer 的印记出版作品。该作品可能会在丛书中出版

**计算机科学讲义（LNCS、LNAI 或 LNBI）。**

###### § 1 授予的权利

作者特此授予 Springer Nature Switzerland AG, Gewerbestrasse 11, 6330 Cham, Switzerland（以下称为出版商）独家、唯一、永久、全球、可转让、可再许可和无限制的复制、出版、分发、传输的权利、向公众提供或以其他方式交流、翻译、公开表演、存档、存储、出租或出借和出售贡献或其部分单独或与任何语言的其他作品一起，在所有修订和版本（包括软封面、书籍俱乐部和收藏版、选集、预印本、重印或按订单打印、缩微胶卷、听力图和录像），以所有形式和表达媒体，包括电子形式（包括离线和在线使用、推送或拉取技术、在数据库中使用）和数据网络（例如互联网），用于在任何和所有固定或便携式最终用户设备上显示、打印和存储，例如文本阅读器、音频、 视频或交互式设备，以及用于多媒体或交互式版本以及在数据网络或搜索引擎中显示或传输投稿或其部分，以及在与作品密切相关的社交媒体帐户上发布投稿），全部、部分或节略形式，在每种情况下都如现在所知或将来发展，包括授予进一步的有时限或永久权利的权利。出版商尤其有权允许他人使用个别插图、表格或文字引用，并可将投稿用于广告目的。出于在电子表格中使用的目的，出版商可以将投稿调整为相应的使用形式并包括链接（例如框架或内联链接）或以其他方式将其与其他作品结合和/或删除与提供的其他作品的链接或组合贡献。为避免疑义， 无论投稿和/或作品本身是否构成适用版权法下的数据库，本合同的所有条款均适用。

投稿的著作权归出版商所有。作者已声明他/她在所有版本和版本的作品及其部分中被确定为本贡献的创作者的权利，以所有形式和媒体出版。出版商可以以自己的名义或以作者的名义采取任何必要的措施来保护根据本协议授予的权利免受第三方的侵犯。根据世界版权公约 (UCC) 的规定，它将在作品的所有版本中插入版权声明。

双方承认，美国政府官员或雇员为履行其公务而准备的文稿在美国可能没有版权主张。如果

投稿是根据美国政府合同执行的，但作者不是美国政府雇员，出版商授予美国政府复制全部或部分投稿的免版税许可，并授权其他人为美国政府这样做目的。如果投稿由官方（即英联邦国家的立宪君主）或任何官方政府部门编制或出版，或在其指导或控制下出版，则投稿的版权应在与作者达成任何协议的情况下属于皇冠。如果作者是官员或雇员

美国政府或官​​方的，将在签名页上提及此状态。

16.01.2018 10:38

2

###### §2 作者保留的权利

除法律允许的用途外，作者保留将投稿内容传达给其他研究同事、以手稿形式与他们分享投稿、执行或展示投稿或将内容用于非商业用途的权利内部和教育目的，前提是根据任何印刷或电子材料中的当前引用标准引用出版物的原始来源。作者保留在仅由作者自己的作品组成的任何集合中免费重新发布投稿的权利，但须确保出版商的出版得到适当的注明，并逐字重复相关版权声明。作者可以在他/她自己的网站和/或作者所在部门或学院的存储库上自行归档他/她的贡献的作者创建版本。 作者也可以应资助者的要求或出于法律义务将此版本存放在其资助者或资助者的指定存储库中。他/她不得使用发布者平台上发布的发布者PDF版本进行自存档或存入。此外，作者只能发布他/她自己的版本，前提是对原始出版来源给予确认，并在出版商网站上插入指向已发表文章的链接。必须通过在以下句子中插入文章的 DOI 号来提供链接：“最终认证版本可在 https://doi.org/[插入 DOI] 在线获得。” DOI（数字对象标识符）可以在已发表论文的第一页底部找到。

在 ArXiv/CoRR 和 HAL 等非商业预印本服务器上发布的投稿的早期版本可以保留在这些服务器上和/或可以使用作者接受的版本进行更新。最终发布的版本（pdf 或 html/xml 格式）不能用于此目的。最终出版物需要致谢，并且必须通过在以下句子中插入文章的 DOI 号，将链接插入到出版商网站上已发布的投稿：“最终经过验证的出版物可在线获取 https:// doi.org/[插入 DOI]”。

作者保留将他/她的贡献用于他/她进一步的科学事业的权利，方法是将最终发表的论文包含在他/她的论文或博士论文中，前提是对原始发表来源的确认。作者还保留在无需支付费用且无需通知出版商的情况下使用部分投稿（例如插图）的权利，以便包含在未来的作品中。如果遵守以下原则，作者可以将其会议论文的扩展版本作为期刊文章发表：a) 扩展版本包括至少 30% 的新材料，b) 引用了原始出版物，并且 c) 它包括一个明确的关于增量的声明（例如，新结果、更好的材料描述等）。

###### §3 保证

作者同意，应出版商的要求，执行所有文件并执行出版商合理要求的所有事情，以授予出版商根据本协议授予的所有权利。作者保证投稿是原创的，但在版权所有者许可下可能包含的受版权保护作品（包括插图、表格、动画和文本引用）的摘录除外，在这种情况下，作者需要获得书面在必要的范围内获得许可，并在手稿中注明摘录的确切来源。还要求作者存储已签署的许可表格，并在需要时将其提供给出版商。

作者保证，作者有权根据第 1 条“授予的权利”授予权利，作者未将此类权利转让给第三方，投稿尚未全部或部分出版，投稿不包含诽谤或诽谤性陈述，并且不侵犯他人的任何版权、商标、专利、法定权利或专有权利，包括通过许可获得的权利；并且该作者将赔偿出版商因任何索赔（如果属实将构成作者违反本协议中任何作者的陈述或保证）可能导致出版商承担的任何成本、开支或损害赔偿。

作者同意修改投稿以删除在任何时候发现的任何潜在的淫秽、诽谤、诽谤、恶意虚假或其他非法部分。任何此类删除或更改均不影响作者在本协议中提供的保证和赔偿。

###### § 4 投稿和出版的交付

作者同意在商定的日期向负责的卷编编辑（对于会议，通常是其中一位项目主席）交付根据出版商的作者说明创建的手稿。出版商将自行承担费用和风险对投稿的复制和分发。提交由通讯作者签署的出版同意书后，出版商将不接受作者身份或所列作者顺序的更改。

16.01.2018 10:38

3

###### § 5 书籍作者折扣

只要作者和出版商之间有合同安排，作者有权以 40% 的折扣购买出版商出版的作品或其他书籍供他/她个人使用（如果直接从出版商处订购），并且受适用的图书价格规定的约束。

不允许转售此类副本。

###### § 6 适用法律和管辖权

如果作者和出版商之间在本协议的含义或双方的权利和责任方面出现任何分歧，双方应进行善意讨论，以寻求双方都满意的争议解决方案。本协议受瑞士法律管辖并按其解释。瑞士楚格法院具有专属管辖权。

通讯作者代表任何和所有共同作者签署并承担发布此材料的责任。

###### 通讯作者签名： 日期：

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .1.0. /.0.8. /.2. 0. .1.8. . . . . . . . . . . . . . . .

我是美国政府的雇员，在可转让的范围内转让权利（适用《美国法典》第 17 条第 105 条）我是官方的雇员，投稿的版权属于官方

□

□

*仅限内部使用：*

法人实体编号：1128 Springer Nature Switzerland AG Springer-C-CTP-01/2018

16.01.2018 10:38

# IEEE 版权许可表格

**用于 IEEE 计算机协会的实验性“延迟开放访问”模型**

#### 作品名称（以下简称“作品”）：

HOLMES：通过可疑信息流的关联进行实时 APT 检测

### 作者：

Sadegh Momeni Milajerdi、Rigel Gjomemo、Birhanu Eshete、R. Sekar、Venkat Venkatakrishnan

#### IEEE 出版物标题：2019 年 IEEE 安全和隐私研讨会 (SP)

下列签署人特此授予电气和电子工程师协会（“IEEE”）在十二（ 12) 自本协议签署之日起几个月，并将论文纳入 IEEE Xplore 数字图书馆。在此排他性期限结束后，IEEE 应永久保留以前在非排他性基础上授予的所有权利。 IEEE 还应在独占期结束后通过 IEEE 的数字图书馆免费提供该作品。

作者声明并保证作品是原创的，并且他/她/他们是作品的作者。在要求的范围内，作者声明并保证他们已从任何第三方知识产权所有者处获得所有必要的许可。作者进一步声明他们有权制定和执行本协议。

IEEE 承认作者保留作品的所有权，包括：

1. 作者或其雇主保留作品中描述的任何过程、程序或制造物品的所有所有权。
2. 对于根据美国政府合同或授权执行的工作，IEEE 承认美国政府拥有复制全部或部分工作的免版税许可，并授权他人这样做，仅用于美国政府的官方目的，如果合同/赠款如此要求。

作者特此授予 IEEE 在全球范围内使用他/她的姓名、图片、肖像、声音和传记信息作为包含该作品的产品的广告、分发和销售的一部分的无限的、全球性的、不可撤销的权利，并从任何基于隐私权或公开权的索赔。

作者同意赔偿 IEEE 并使其免受因违反上述任何保证而可能产生的任何损害或费用。

如果上述作品未被 IEEE 接受和出版，或在被 IEEE 接受之前被作者撤回，则本协议无效。

萨迪克·莫明尼·米兰迪尔

作者

09.25.2018

日期

### 联合作者

合着的作品，由所有合着者署名，或由其中一名作者代他人署名。如果一名或多名作者是美国政府雇员，但至少有一名作者不是，则该非政府作者应在此表格上签名。

# 条款和条件以及保留的权利

**一般的**

* 1. 在资助机构的资助下进行工作的作者可以自由地履行该资助机构的任何存款任务。

**开放获取文章的作者在线使用**

* 1. 课堂或内部培训使用。明确允许作者在作者的个人网站或作者所在机构或公司的服务器上发布与作者的教学、培训或工作职责有关的他/她自己的 IEEE 版权文章的最终、已发表版本的任何部分，前提是适当的版权、信用和重复使用通知出现在张贴材料的显眼位置。允许使用的示例包括讲座材料、课程包、电子储备、会议演示或内部培训课程。

**作者信息**

**作者责任：** IEEE 在世界各地分发其技术出版物，并希望确保提交给其出版物的材料能够正确地提供给这些出版物的读者。作者必须确保其作品符合 IEEE PSPB 操作手册第 8.2.1 节中所述的要求，包括关于原创性、作者身份、作者责任和作者不当行为的规定。有关 IEEE 出版政策的更多信息，请访问[http://www.ieee.org/publications\_standards/publications/rights/authorrightsresponsibilities.html.](http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html) IEEE PSPB 操作手册第 8.2.1.B12 节特别建议作者：“确定披露其材料是否需要其他方的事先同意，如果需要，获得它。”还建议作者注意 IEEE PSPB 操作手册第 8.1.1B 节：“IEEE 发表的作品中的陈述和意见是作者的表达。”

作者/雇主权利：如果您在受雇范围内受雇并就某个主题准备作品，则作品的版权属于您的雇主作为雇佣作品。在这种情况下，IEEE 假定当您签署本表格时，您已获得雇主的授权，并且您的雇主已同意发表权的陈述和保证以及本表格的所有其他条款和条件。如果您未获得此类授权和同意，您的雇主的授权代表应以作者身份签署本表格。内容分发：IEEE 通过硬拷贝、缩微胶片、缩微胶卷和电子媒体等多种方式在全球分发其技术出版物。它还提取并可能翻译其出版物和其中包含的文章，以包含在各种纲要、集体作品、数据库和类似出版物中。

在标准 IEEE 版权政策允许的情况下，作者可以在其个人和/或机构网站上发布已接受的作品版本。本许可到期后，作者可以在此类网站上发布已发布的作品版本，IEEE 开发的任何交互式版本除外。作者不受限制制作自己的、独立衍生的互动作品。

**关于使用开放存取文章的读者信息**

**重复使用：**第三方用户可以出于个人使用或学术目的查看、打印、复制和下载内容。用户还可以发布链接到已发布的 IEEE 版本的作品，包括数字对象标识符 (DOI)，用于他们自己的非商业用途。所有第三方用户必须提供 IEEE 出版物标题和作品作者姓名的完整署名。以其他方式复制，或由第三方分发/展示作品，需要获得 IEEE 的许可。

***IEEE 的保留权利：IEEE 保留在本许可期间（自 IEEE Xplore 数字图书馆中首次发布日期起 12 个月）限制衍生作品的生产和对作品的商业利用的权利。如果第三方在此期间寻求基于作品制作衍生作品或对作品进行商业利用，他们必须获得 IEEE 的许可。***

***文本和数据挖掘/翻译权利：允许对作品进行非商业翻译以及文本和数据挖掘。但是，在许可期间，如果第三方出于商业目的寻求从事此类活动，则必须获得 IEEE 的明确书面许可。任何翻译都必须显着地链接到 IEEE 发布的版本，并显着显示以下内容：“这是 IEEE 出版物的非官方翻译。 IEEE 并未认可此翻译的内容。”***

**此表格必须随附提交作者的手稿。**

**有关提交表格或手稿的问题必须发送给出版物的编辑。**

**请将有关 IEEE 版权政策的所有问题发送至：**

**IEEE 知识产权局，**[**版权所有@ieee.org,**](mailto:copyrights@ieee.org)**+1-732-562-3966**

#### ACM 版权和音频/视频发布

**工作名称：**POIROT：将攻击行为与内核审计记录对齐以进行网络威胁搜寻

**提交编号：fv607**

作者/演讲者：Sadegh M. Milajerdi (UIC)； Birhanu Eshete（密歇根大学迪尔伯恩分校）； Rigel Gjomemo (UIC); V.N.文卡塔克里希南（UIC）；

**材料类型：全纸**

**出版物和/或会议名称：** CCS '19：2019 ACM SIGSAC 计算机和通信安全会议论文集

#### 版权转让、保留权利和许可使用 38889;

\* 您的版权转让以您同意以下条款为条件。

作品以及与作品一起提交以供审查和出版的任何补充文件的版权，例如扩展证明、PowerPoint 大纲或可能超过印刷页数限制的附录，（包括但不限于将作品全部或部分以现在或以后已知的任何和所有形式的媒体发布）特此转让给 ACM（用于政府工作，在可转让的范围内），自本协议签署之日起生效，但前提是该作品已被 ACM 接受出版。

保留权利和允许使用

1. 作者未授予 ACM 的所有权利和许可均保留给所有者，包括所有其他专有权利，例如专利或商标权。
2. 此外，尽管所有者已授予 ACM 专有权利，但所有者应有权执行以下操作：
3. 在作者编写或编辑的任何未来作品中免费重复使用作品的任何部分，包括任何和所有媒体中的书籍、讲座和演示文稿。
4. 创建一个由作者全资拥有的“重大修订”
5. 将作品的接受版本发布在 (1) 作者的主页、(2) 所有者的机构存储库、(3) 任何由资助该作品所依据的研究的机构合法授权的存储库，以及 (4) 任何非- 不复制 ACM 目录的商业存储库或聚合，即其链接模式基本上不复制 ACM 受版权保护的卷或问题。非商业存储库在此被理解为由非营利组织拥有的存储库，这些存储库不收取访问已存文章的费用，也不出售广告或以其他方式从提供文章中获利。
6. 在 (1) 作者的主页或 (2) 所有者的机构存储库上发布“Author-Izer”链接，允许免费下载 ACM 数字图书馆中的记录版本；
7. 在 ACM 同行评审流程开始之前，将提交给 ACM 的作品版本（“提交的版本”或任何早期版本）发布到非同行评审的服务器；
8. 在内部向所有者的员工免费分发最终发布的记录版本（如果适用）；
9. 为课堂和个人使用免费分发已发布的记录版本；
10. 将作品捆绑在所有者的任何软件发行版中；和
11. 使用任何独立于作品的辅助材料。 (x) 如果您的论文在被撤回之前被撤回

发表在 ACM 数字图书馆中，版权归作者所有。

在准备使用 ACM TeX 模板提交论文时，权利和许可信息以及书目条带必须出现在第一页的左下方。

新的 ACM Consolidated TeX 模板版本 1.3 及更高版本会根据系统根据您的权限管理选择和此特定会议生成的代码片段自动为您创建和定位这些文本块。

注意：对于使用 ACM Microsoft Word 主文章模板和出版工作流程的作者，ACM 出版系统 (TAPS) 将为您将权利声明添加到您的论文中。请与您的会议联系人联系以获取有关提交源文件进行处理的信息。

*请在 \begin{document} 之前复制并粘贴 \setcopyright{acmcopyright} 并将以下代码片段复制并粘贴到您的 TeX 文件中 \begin{document} 和 \maketitle 之间，在 CCS 代码之后或之前。*

\版权年{2019}

\acm年{2019}

\acmConference[CCS '19]{2019 ACM SIGSAC 计算机和通信安全会议}{2019 年 11 月 11--15 日}{英国伦敦}

\acmBooktitle{2019 年 ACM SIGSAC 计算机和通信安全会议 (CCS '19)，2019 年 11 月 11--15 日，英国伦敦}

\ accompromise { 15.00 }

\ acmdoi { 10.1145/3319353363217 }

\ acmisbn { 978-1-453-6747-9/19/11 }

*如果您使用 ACM Microsoft Word 模板，或仍在使用旧版本的 ACM TeX 模板，或当前版本的 ACM SIGCHI、SIGGRAPH 或 SIGPLAN TeX 模板，您必须将以下文本块复制并粘贴到您的文档中根据您使用的模板提供的说明：*

允许为个人或课堂使用制作全部或部分作品的数字或硬拷贝，但不收取任何费用，前提是拷贝不是为了营利或商业利益而制作或分发的，并且拷贝带有本通知和首页上的完整引文.必须尊重 ACM 以外的其他人拥有的本作品组件的版权。允许以信用摘录。要以其他方式复制或重新发布、在服务器上发布或重新分发到列表，需要事先获得特定许可和/或收费。请求权限[权限@acm.org。](mailto:Permissions@acm.org)

CCS '19，2019 年 11 月 11 日至 15 日，英国伦敦

© 2019 计算机协会。 ACM ISBN 978-1-4503-6747-9/19/11…$15.00

https://doi.org/10.1145/3319535.3363217

*注意：确保将您文章的 DOI 作为 bibstrip 数据的一部分； DOI 将在 ACM 数字图书馆发布后不久注册并生效。准备好相机副本后，请将源文件和 PDF 发送给您的活动联系人进行处理。*

A. 同意转让。我在此声明并保证我是唯一所有者（或授权



版权所有者的代理人），但下文第 III 节详述的第三方材料除外。我已获得作品中包含的任何第三方材料的许可。

B. 政府工作宣言。我是我国国民政府的雇员



并且我的政府声称对该作品拥有权利，或者它不具有版权（政府作品仅在美国被归类为公共领域）

是否有任何国家政府的合著者、雇员或承包商？ 是的 不



# 会议录播权限

\* 您的音频/视频发布以您同意以下条款为条件。

我在此允许 ACM 以任何和所有形式为会议和/或出版物包括我的姓名、肖像、演示文稿和评论。

我进一步授权 ACM 录制和/或转录和复制我的演示文稿，作为 ACM 数字图书馆的一部分，并以 CD-ROM、DVD、网络广播的形式作为 ACM 产品的一部分以全部或部分形式进行销售、USB 设备、流媒体视频或现在或以后已知的任何其他媒体格式。

我了解，未经我的直接同意，我的演示文稿不会作为独立产品单独出售。因此，我授予 ACM 使用我的图像、声音、声明、肖像和我的姓名以及我提交的与会议和/或出版物有关的任何传记材料的权利，无论是摘录还是完整地用于分发上述以及任何相关的广告或展览。

您是否同意上述音频/视频发布？ 是的 不



# 辅助材料

有辅助材料吗？ 是的 不



# 第三方材料

如果我的演示文稿或辅助材料中使用的任何材料包含

第三方个人或组织（包括受版权保护的音乐或电影摘录或不属于我的任何东西），我了解我有责任确保获得任何必要的许可和/或许可以进行印刷和/或数字出版，并引用或附上它们以下。

我们/我没有使用第三方材料。



我们/我使用了第三方材料并拥有必要的权限。

# 艺术图像

如果您的论文包含为除本论文以外的任何目的而创建的图像，并且您或您的雇主要求版权，您必须填写第五部分并确保在论文中的每个此类图像中包含版权声明。

我们/我没有任何艺术形象。我们/我有任何艺术形象。



# 陈述、保证和契约

下列签署人特此声明、保证和承诺如下：

* 1. 所有者是作品所有者的唯一所有者或授权代理人；
  2. 下列签署人被授权签订本协议并将本许可中包含的权利授予 ACM；
  3. 作品为原创作品，不侵犯任何第三方权利；已获得在范围和期限上与授予 ACM 的权利一致的使用第三方材料的所有许可，已向 ACM 提供此类许可的副本，并且提交给 ACM 的作品清楚准确地表明对所有者的信用任何此类第三方材料（包括任何适用的版权声明），或将被修改以表明此类信用；
  4. 除了在未经同行评审的服务器上发布非正式帖子外，该作品尚未发布，并且所有者承诺尽最大努力将 ACM DOI 指针放在任何此类先前发布的帖子上；
  5. 辅助材料（如果有）不包含恶意代码、病毒、特洛伊木马或其他旨在允许未经授权访问或禁用、擦除或以其他方式损害任何计算机系统或软件的软件例程或硬件组件；和
  6. 艺术图像（如有）在提交的版本中清楚准确地注明（包括任何适用的版权声明）。

我同意声明、保证和契约



日期：09/16/2019 发送至[smomen2@uic.edu](mailto:smomen2@uic.edu) 在 14:**09:04**

# 被引文献

* + 1. Hossain, M. N.、Milajerdi, S. M.、Wang, J.、Eshete, B.、Gjomemo, R.、Sekar, R.、Stoller, S. 和 Venkatakrishnan, V.：SLEUTH：从 COTS 审计中重建实时攻击场景数据。在第 26 届 USENIX 安全研讨会（USENIX Security

17)，第 487-504 页，温哥华，不列颠哥伦比亚省，2017 年。USENIX 协会。

* + 1. Miajerdi, S. M.、Gjomemo, R.、Eshete, B.、Sekar, R. 和 Venkatakrishnan, V.：HOLMES：通过可疑信息流相关性进行实时 APT 检测。在 IEEE 安全和隐私研讨会论文集上。IEEE，2019。
    2. Miajerdi, S. M.、Eshete, B.、Gjomemo, R. 和 Venkatakrishnan, V.：Poirot：将攻击行为与网络威胁追踪的内核审计记录对齐。在 2019 年 ACM SIGSAC 计算机和通信安全会议记录中，第 1813-1830 页。ACM，2019 年。
    3. M. Miajerdi, S.、Eshete, B.、Gjomemo, R. 和 Venkatakrishnan, V.：Propatrol：通过提取的高级任务进行攻击调查。在信息系统安全国际会议上。斯普林格，2018。
    4. Westcott, D. 和 Bandla, K.：APT 笔记。<https://github.com/aptnotes/data>, 2018.
    5. Albanesius, C.：目标忽略数据泄露警告标志。[http://www.pcmag.com/](http://www.pcmag.com/article2/0%2C2817%2C2454977%2C00.asp) [文章2/0,2817,2454977,00.asp](http://www.pcmag.com/article2/0%2C2817%2C2454977%2C00.asp), 2014. [在线;2017 年 2 月 16 日访问]。
    6. Mlot, S.：Neiman Marcus 黑客引发了近 6 万条警报。[http://www.pcmag.com/](http://www.pcmag.com/article2/0%2C2817%2C2453873%2C00.asp) [文章2/0,2817,2453873,00.asp](http://www.pcmag.com/article2/0%2C2817%2C2453873%2C00.asp), 2014. [在线;2017 年 2 月 16 日访问]。
    7. King, S. T. 和 Chen, P. M.：回溯入侵。在 SOSP 中。ACM，2003 年。
    8. King, S. T.、Mao, Z. M.、Lucchetti, D. G. 和 Chen, P. M.：通过多主机因果关系丰富入侵警报。在 NDSS，2005 年。
    9. 对抗性战术、技术和常识。[https://attack.mitre.org/](https://attack.mitre.org/wiki/Main_Page) [维基/Main\_Page](https://attack.mitre.org/wiki/Main_Page).

185

* + 1. 系统， N.: LOKI，免费的 IOC 扫描仪 - Nextron Systems。 [https://www.](https://www.nextron-systems.com/loki/) [nextron-systems.com/loki/](https://www.nextron-systems.com/loki/), 2017.
    2. 火眼： 红线。 <https://www.fireeye.com/services/freeware/redline.html>, 2018。访问时间：2019-04-23。
    3. MANDIANT：揭露中国的网络间谍单位之一。[https://www.fireeye.](https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf) [com/content/dam/fire eye-www/services/pdf/man diant-AP t1-report . pdf](https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf).访问时间：2016-11-10。
    4. 菲尔德斯，J.： Stuxnet 病毒的目标和传播方式已揭晓。[https://www.bbc.com/news/](https://www.bbc.com/news/technology-12465688) [技术-12465688](https://www.bbc.com/news/technology-12465688), 2011.
    5. Splunk：SIEM、AIOps、应用程序管理、日志管理、机器学习和合规性。<https://www.splunk.com/>, 2019.
    6. Logrhythm，安全情报公司。<https://logrhythm.com/>.
    7. IBM QRadar SIEM。<https://www.ibm.com/us-en/marketplace/ibm-qradar-siem>.
    8. Sekar, R.：MAPRLE 内部技术报告。技术报告，石溪大学，2015 年。
    9. Qin, X. 和 Lee, W.：信息安全警报数据的统计因果分析。在 RAID 中。斯普林格，2003 年。
    10. Ning, P. 和 Xu, D.：从入侵警报中学习攻击策略。在 CCS 中。 ACM，2003 年。
    11. Wang, W.：面向图的网络取证分析方法。 2010 年。
    12. Noble, C. C. 和 Cook, D. J.：基于图形的异常检测。在第九届 ACM SIGKDD 知识发现和数据挖掘国际会议论文集中，2003 年。
    13. Kemerlis, V. P.、Portokalidis, G.、Jee, K. 和 Keromytis, A. D.：Libdft：商品系统的实用动态数据流跟踪。 SIGPLAN 不，2012。
    14. Arzt, S.、Rasthofer, S.、Fritz, C.、Bodden, E.、Bartel, A.、Klein, J.、Le Traon, Y.、Octeau, D. 和 McDaniel, P.：Flowdroid：精确android 应用程序的上下文、流、字段、对象敏感和生命周期感知污点分析。 SIGPLAN 不是，2014 年。
    15. Lee, K. H.、Zhang, X. 和 Xu, D.：基于二进制执行分区的高精度攻击来源。在 NDSS，2013 年。
    16. Luk, C.-K.、Cohn, R.、Muth, R.、Patil, H. 和 Klauser 等人，A.：Pin：使用动态仪器构建定制的程序分析工具。在 PLDI，2005 年。
    17. 透明计算。<https://www.darpa.mil/program/transparent-computing>.
    18. Keromytis, A. D.：透明计算参与 3 数据发布。[https://github.](https://github.com/darpa-i2o/Transparent-Computing) [com/darpa-i2o/透明计算](https://github.com/darpa-i2o/Transparent-Computing), 2018.
    19. Newsome, J. 和 Song, D.：用于自动检测、分析和签名生成商品软件漏洞利用的动态污点分析。 2005 年。
    20. Xu, W.、Bhatkar, S. 和 Sekar, R.：污染增强的策略执行：击败各种攻击的实用方法。在 USENIX 安全，2006 年。
    21. McColl, R. C.、Ediger, D.、Poovey, J.、Campbell, D. 和 Bader, D. A.：开源图形数据库的性能评估。在第一次关于分析应用程序的并行编程研讨会的论文集中。 ACM，2014 年。
    22. Neo4j 图数据库。<https://neo4j.com/>.
    23. Titan图形数据库。<http://titan.thinkaurelius.com/>.
    24. Ediger, D.、McColl, R.、Riedy, J. 和 Bader, D. A.：Stinger：流图的高性能数据结构。在高性能极限计算 (HPEC) 中。 IEEE，2012。
    25. Network-x 图数据库。<https://networkx.github.io/>.
    26. Hossain, M. N.、Wang, J.、Sekar, R. 和 Stoller, S.：用于可扩展取证分析的依赖保留数据压缩。在 USENIX 安全研讨会上。 USENIX 协会，2018 年。
    27. 情报驱动的计算机网络防御通过分析对手活动和入侵杀伤链来提供信息。 [http://www.](http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf) [lockheedmartin.com/content/dam/lockheed/data/corporate/documents/](http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf)

[LM 白皮书英特尔驱动型国防.pdf](http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf).访问时间：2016-11-10。

* + 1. Gehani, A. 和 Tariq, D.：Spade：支持分布式环境中的出处审计。在第 13 届国际中间件会议论文集上。斯普林格，2012。
    2. Bates, A.、Tian, D. J.、Butler, K. R. 和 Moyer, T.：Linux 内核值得信赖的全系统出处。在 USENIX 安全，2015 年。
    3. Ma, S.、Zhang, X. 和 Xu, D.：ProTracer：通过在记录和污染之间交替进行实际的来源追踪。在 NDSS，2016 年。
    4. Kumar, S.：计算机入侵的分类和检测。博士论文，博士论文，普渡大学，1995。
    5. Porras, P. A. 和 Kemmerer, R. A.：渗透状态转换分析：基于规则的入侵检测方法。在计算机安全应用会议上，1992 年。

诉讼程序。，第八届年度，第 220-229 页。 IEEE，1992 年。

* + 1. CAPEC：常见攻击模式枚举和分类。[https://capec.](https://capec.mitre.org/index.html) [mitre.org/index.html](https://capec.mitre.org/index.html).访问时间：2018-02-27。
    2. 通用漏洞评分系统 v3.0：规范文档。[https://www.first.](https://www.first.org/cvss/specification-document) [org/cvss/规范文件](https://www.first.org/cvss/specification-document).
    3. FireEye：打开 IOC。[https://openioc.org](https://openioc.org/), 2018.
    4. Mitre：结构化威胁信息表达 (STIX)。[https://stixproject.](https://stixproject.github.io/) [github.io](https://stixproject.github.io/), 2018.
    5. MISP：MISP - 开源威胁情报平台和威胁信息共享的开放标准。<https://www.misp-project.org/>, 2019。访问时间：2019-04-23。
    6. Iklody, M. A.：MISP 对象中的默认关系类型。[https://github.com/MISP/](https://github.com/MISP/misp-objects/blob/master/relationships/definition.json) [misp-objects/blob/master/relations/definition.json](https://github.com/MISP/misp-objects/blob/master/relationships/definition.json), 2019。访问时间：2019-04-23。
    7. Pei, K.、Gu, Z.、Saltaformaggio, B.、Ma, S.、Wang, F.、Zhang, Z.、Si, L.、Zhang, X. 和 Xu, D.：Hercule：攻击故事通过相关日志图上的社区发现进行重建。在第 32 届计算机安全应用年会论文集上，第 583-595 页。ACM，2016 年。
    8. 赛门铁克：Buckeye：间谍机构在影子经纪人泄漏之前使用了 Equation Group 工具。[https://www.symantec.com/blogs/threat-intelligence/](https://www.symantec.com/blogs/threat-intelligence/buckeye-windows-zero-day-exploit) [buckeyewindows 零日漏洞利用](https://www.symantec.com/blogs/threat-intelligence/buckeye-windows-zero-day-exploit), 2019.
    9. 纽约时报：中国间谍如何获得 NSA 的黑客工具，并利用它们进行攻击。[https://www.nytimes.com/2019/05/06/us/politics/](https://www.nytimes.com/2019/05/06/us/politics/china-hacking-cyber.html) [中国黑客网络.html](https://www.nytimes.com/2019/05/06/us/politics/china-hacking-cyber.html), 2019.
    10. Kolbitsch, C.、Comparetti, P. M.、Kruegel, C.、Kirda, E.、Zhou, X.-y. 和 Wang, X.：有效且高效的终端主机恶意软件检测。在 USENIX 安全研讨会，第 4 卷，第 351-366 页，2009 年。
    11. Zong, B.、Xiao, X.、Li, Z.、Wu, Z.、Qian, Z.、Yan, X.、Singh, A. K. 和 Jiang, G.：系统生成的时间图中的行为查询发现。VLDB 捐赠基金会议记录，9(4):240–251, 2015。
    12. Wagner, D. 和 Soto, P.：对基于主机的入侵检测系统的模拟攻击。在第 9 届 ACM 计算机和通信安全会议记录中，第 255-264 页。ACM，2002 年。
    13. Parampalli, C.、Sekar, R. 和 Johnson, R.：针对强大的系统调用监视器的实用模仿攻击。在信息、计算机和通信安全方面。ACM，2008 年。
    14. De Nardo, L.、Ranzato, F. 和 Tapparo, F.：子图相似性问题。IEEE Transactions on Knowledge and Data Engineering, 21(5):748–749, 2009。
    15. 火眼： OpenIOC 系列：使用妥协指标 (IOC) 进行调查 - 第 I 部分。[https://www.fireeye.com/blog/threat-research/2013/12/](https://www.fireeye.com/blog/threat-research/2013/12/openioc-series-investigating-indicators-compromise-iocs.html) [openioc-系列调查指标-妥协-iocs.html](https://www.fireeye.com/blog/threat-research/2013/12/openioc-series-investigating-indicators-compromise-iocs.html), 2013.
    16. 团队，M.S.：向 STIX 报告转换的 APT1。[https://stix.mitre.org/language/](https://stix.mitre.org/language/version1.0.1/samples/README.txt) [版本1.0.1/示例/自述文件.txt](https://stix.mitre.org/language/version1.0.1/samples/README.txt), 2013。访问时间：2019-04-23。
    17. 团队，M.S.：FireEye Poison Evy 向 STIX 报告转换。[https://stix.mitre.](https://stix.mitre.org/language/version1.0.1/samples/README-fireeye.txt) [org/language/version1.0.1/samples/README-fireeye.txt](https://stix.mitre.org/language/version1.0.1/samples/README-fireeye.txt), 2013。访问时间：2019-04-23。
    18. Zhu, Z. 和 Dumitras, T.：Chainsmith：通过挖掘威胁情报报告自动学习恶意活动的语义。2018 年 IEEE 欧洲安全和隐私研讨会 (EuroS&P)，第 458-472 页。IEEE，2018 年。
    19. Liao, X.、Yuan, K.、Wang, X.、Li, Z.、Xing, L. 和 Beyah, R.：参与 ioc 游戏：走向开源网络威胁情报的自动发现和分析。在 2016 年 ACM SIGSAC 计算机和通信安全会议记录中，第 755-766 页。ACM，2016 年。
    20. Husari, G.、Al-Shaer, E.、Ahmed, M.、Chu, B. 和 Niu, X.：Ttpdrill：从 cti 源的非结构化文本中自动准确地提取威胁行动。在第 33 届年度计算机安全应用会议论文集上，第 103-115 页。ACM，2017 年。
    21. STIX：STIX 可视化。[https://oasis-open.github.io/cti-documentation/](https://oasis-open.github.io/cti-documentation/stix/gettingstarted.html#stix-visualization) [stix/getting started . html # stix-可视化](https://oasis-open.github.io/cti-documentation/stix/gettingstarted.html#stix-visualization), 2019。访问时间：2019-05-15。
    22. Moran, F. N. 和 Villeneuve, N.： 副狗行动：ZeroDay (CVE-2013-3893) 对日本目标的攻击。[https://www.fireeye.com/blog/threat-research/2013/09/](https://www.fireeye.com/blog/threat-research/2013/09/operation-deputydog-zero-day-cve-2013-3893-attack-against-japanese-targets.html)

[操作-副狗-零日-cve-2013-3893-攻击-针对日本-](https://www.fireeye.com/blog/threat-research/2013/09/operation-deputydog-zero-day-cve-2013-3893-attack-against-japanese-targets.html)目标。[html](https://www.fireeye.com/blog/threat-research/2013/09/operation-deputydog-zero-day-cve-2013-3893-attack-against-japanese-targets.html), 2013。访问时间：2019-04-19。

* + 1. Zong, B.、Raghavendra, R.、Srivatsa, M.、Yan, X.、Singh, A. K. 和 Lee, K.-W.：通过子图匹配放置云服务。2014 年 IEEE 第 30 届国际会议

数据工程会议，第 832-843 页。IEEE，2014。

* + 1. Gallagher, B.：匹配结构和语义：关于基于图形的模式匹配的调查。AAAI FS，2006 年 6：45-53。
    2. Fan, W., Li, J., Ma, S., Tang, N., Wu, Y. 和 Wu, Y.：图形模式匹配：从难以处理到多项式时间。VLDB 捐赠基金会议记录，3(1-2):264–275, 2010。
    3. Cheng, J.、Yu, J. X.、Ding, B.、Philip, S. Y. 和 Wang, H.：快速图形模式匹配。2008 年 IEEE 第 24 届国际数据工程会议，第 913-922 页。IEEE，2008 年。
    4. Zou, L.、Chen, L. 和 O¨zsu, M. T.：距离连接：大型图形数据库中的模式匹配查询。VLDB 捐赠基金会议记录，2(1):886–897, 2009。
    5. Khan, A.、Wu, Y.、Aggarwal, C. C. 和 Yan, X.：Nema：具有标签相似性的快速图搜索。在 VLDB 捐赠基金会议记录中，第 6 卷，第 181-192 页。VLDB 捐赠基金，2013 年。
    6. Tong, H.、Faloutsos, C.、Gallagher, B. 和 Eliassi-Rad, T.：大型属性图中的快速尽力模式匹配。在第 13 届 ACM SIGKDD 国际

知识发现和数据挖掘会议（KDD 2007），第 737-746 页。ACM，2007。

* + 1. Pienta, R.、Tamersoy, A.、Tong, H. 和 Chau, D. H.：Mage：在属性丰富的图中匹配近似模式。 2014 年 IEEE 大数据国际会议（大数据），第 585-590 页。 IEEE，2014。
    2. 解决方案，G. D. F. C.： njRAT 被发现。[https://app.box.com/s/](https://app.box.com/s/vdg51zbfvap52w60zj0is3l1dmyya0n4) [vdg51zbfvap52w60zj0is3l1dmyya0n4](https://app.box.com/s/vdg51zbfvap52w60zj0is3l1dmyya0n4), 2013。访问时间：2019-04-19。
    3. 博客，G.D.： Uroburos 案例：确定了新的复杂 RAT。 [https://www.gdatasoftware.com/blog/2014/11/](https://www.gdatasoftware.com/blog/2014/11/23937-the-uroburos-case-new-sophisticated-rat-identified) [23937-尿脓-病例-新-复杂-大鼠-鉴定](https://www.gdatasoftware.com/blog/2014/11/23937-the-uroburos-case-new-sophisticated-rat-identified), 2013.

访问时间：2019-04-19。

* + 1. (伟大的), K. L. G. R. A. T.：Carbanak APT：大银行抢劫案。[https://media.kasperskycontenthub.com/wp-content/uploads/sites/](https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak_APT_eng.pdf) [Carbanak\_APT\_eng.pdf/03/08064518/43/2018](https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak_APT_eng.pdf), 2015。访问时间：2019-04-19。
    2. C. C. S. 团队：DustySky 行动。[https://www.clearskysec.com/wp-content/](https://www.clearskysec.com/wp-content/uploads/2016/01/Operation%20DustySky_TLP_WHITE.pdf) [上传/2016/01/Operation%20DustySky\_TLP\_WHITE.pdf](https://www.clearskysec.com/wp-content/uploads/2016/01/Operation%20DustySky_TLP_WHITE.pdf), 2016。访问时间：2019-04-19。
    3. 由 ESET, W.: 海莲花： 老技术，新后门。 [https://www.](https://www.welivesecurity.com/wp-content/uploads/2018/03/ESET_OceanLotus.pdf) [welivesecurity.com/wp-content/uploads/2018/03/ESET\_OceanLotus.pdf](https://www.welivesecurity.com/wp-content/uploads/2018/03/ESET_OceanLotus.pdf), 2018。访问时间：2019-08-12。
    4. 由 FortiGuard Labs, T. A. 提供：新 HawkEye 变体的分析。[https://www.fortinet.](https://www.fortinet.com/blog/threat-research/hawkeye-malware-analysis.html) [com/blog/threat-research/hawkeye-malware-analysis.html](https://www.fortinet.com/blog/threat-research/hawkeye-malware-analysis.html), 2019。访问时间：2019-08-12。
    5. hasherezade：PE-Sieve：扫描给定的进程。识别并转储各种潜在的恶意植入程序（替换/注入的 pes、shellcode、钩子、内存补丁）。<https://github.com/hasherezade/pe-sieve>, 2018.
    6. Apache: ab - Apache HTTP 服务器基准测试工具。[https://httpd.apache.org/](https://httpd.apache.org/docs/2.4/programs/ab.html) [docs/2.4/programs/ab.html](https://httpd.apache.org/docs/2.4/programs/ab.html), 2019。访问时间：2019-08-27。
    7. 工作台：Jetstream2。<https://browserbench.org/JetStream/index.html>, 2019。访问时间：2019-08-27。
    8. EFD：高清调谐。[https://www.hdtune.com](https://www.hdtune.com/), 2019。访问时间：2019-08-27。
    9. Lee, K. H.、Zhang, X. 和 Xu, D.： Loggc：垃圾收集审计日志。在 2013 年 ACM SIGSAC 计算机与通信安全会议记录中，第 1005-1016 页。 ACM，2013 年。
    10. Xu, Z.、Wu, Z.、Li, Z.、Jee, K.、Rhee, J.、Xiao, X.、Xu, F.、Wang, H. 和 Jiang, G.：高保真数据缩减用于大数据安全依赖性分析。在 2016 年 ACM SIGSAC 计算机和通信安全会议记录中，第 504-516 页。 ACM，2016 年。
    11. Muniswamy-Reddy, K.-K., Holland, D. A., Braun, U. 和 Seltzer, M. I.：Provenanceaware 存储系统。在 2006 年 USENIX 年度技术会议上。
    12. Goel, A.、Po, K.、Farhadi, K.、Li, Z. 和 de Lara, E.：泰瑟枪入侵恢复系统。 SIGOPS 操作。系统。修订版，2005 年。
    13. Goel, A.、Feng, W. C.、Maier, D.、Feng, W. C. 和 Walpole, J.：Forensix：强大的高性能重建系统。在第 25 届 IEEE 分布式计算系统研讨会国际会议上，2005 年。
    14. Braun, U.、Garfinkel, S.、Holland, D. A.、Muniswamy-Reddy, K.-K. 和 Seltzer, M. I.：自动来源收集问题。在国际出处和注释研讨会。斯普林格，2006 年。
    15. Pohly, D. J.、McLaughlin, S.、McDaniel, P. 和 Butler, K.：Hi-fi：收集高保真全系统出处。在ACSAC。 ACM，2012 年。
    16. Hassan, W.U.、Lemay, M.、Aguse, N.、Bates, A. 和 Moyer, T.：通过对出处图的语法推断实现可扩展的集群审计。在网络和

分布式系统安全研讨会，2018 年。

* + 1. Ma, S.、Zhai, J.、Wang, F.、Lee, K. H.、Zhang, X. 和 Xu, D.：Mpi：具有语义感知执行分区的多视角攻击调查。在第 26 届 {USENIX} 安全研讨会（{USENIX} 安全 17），第 1111-1128 页，2017 年。
    2. Ma, S.、Lee, K. H.、Kim, C. H.、Rhee, J.、Zhang, X. 和 Xu, D.：用于 Windows 的准确、低成本和无工具安全审计日志记录。在 31 日的诉讼中

年度计算机安全应用会议，ACSAC 2015，第 401-410 页，纽约，纽约，美国，2015。ACM。

* + 1. Kwon，Y.，Wang，F.，Wang，W.，Lee，K.H.，Lee，W.-C.，Ma，S.，Zhang，X.，Xu，D.，Jha，S.，Ciocarlie，G ., et al.: Mci: 用于攻击调查的审计日志中基于建模的因果推理。在过程中。第 25 届网络和分布式系统安全研讨会 (NDSS’18)，2018 年。
    2. Ji, Y.、Lee, S.、Downing, E.、Wang, W.、Fazzini, M.、Kim, T.、Orso, A. 和 Lee, W.：雨：按需精炼攻击调查进程间信息流跟踪。在 2017 年 ACM SIGSAC 计算机会议论文集中

和通信安全，第 377-390 页。 ACM，2017 年。

* + 1. Ji, Y.、Lee, S.、Fazzini, M.、Allen, J.、Downing, E.、Kim, T.、Orso, A. 和 Lee, W.：以高效的方式启用可优化的跨主机攻击调查数据流标记和跟踪。在第 27 届 {USENIX} 安全研讨会（{USENIX} 安全 18）中，第 1705-1722 页，2018 年。
    2. Kwon, Y.、Kim, D.、Sumner, W. N.、Kim, K.、Saltaformaggio, B.、Zhang, X. 和 Xu, D.：Ldx：轻量级双重执行的因果关系推断。 ACM SIGOPS 操作

系统评论，50（2）：503–515，2016。

* + 1. Liu, Y.、Zhang, M.、Li, D.、Jee, K.、Li, Z.、Wu, Z.、Rhee, J. 和 Mittal, P.：对企业安全进行及时的因果关系分析。在 2018 年网络和分布式系统安全研讨会上。
    2. Hassan, W. U.、Guo, S.、Li, D.、Chen, Z.、Jee, K.、Li, Z. 和 Bates, A.：Nodoze：通过自动出处分类来对抗威胁警报疲劳。在 NDSS，2019 年。
    3. Sun, X.、Dai, J.、Liu, P.、Singhal, A. 和 Yen, J.：使用贝叶斯网络对零日攻击路径进行概率识别。 IEEE信息汇刊

取证与安全，13(10):2506–2521, 2018。

* + 1. Cavallaro, L. 和 Sekar, R.：污染增强异常检测。在信息系统

安全。斯普林格，2011。

* + 1. Denning, D. E.：入侵检测模型。 IEEE 软件工程汇刊，1987 年。
    2. Lunt, T. F.、Tamaru, A. 和 Gillham, F.：实时入侵检测专家系统

（IDES）。 SRI 国际。计算机科学实验室，1992 年。

* + 1. Forrest, S.、Hofmeyr, S.、Somayaji, A.、Longstaff, T. 等人：unix 进程的自我意识。在标准普尔。 IEEE，1996 年。
    2. Gu, G.、Porras, P. A.、Yegneswaran, V.、Fong, M. W. 和 Lee, W.：Bothunter：通过 ids 驱动的对话相关性检测恶意软件感染。在 USENIX 安全研讨会，第 7 卷，第 1-16 页，2007 年。
    3. Oprea, A., Li, Z., Yen, T.-F., Chin, S. H. 和 Alrwais, S.：通过挖掘大规模日志数据检测早期企业感染。在可靠系统和

Networks (DSN)，2015 年第 45 届年度 IEEE/IFIP 国际会议，第 45-56 页。 IEEE，2015。

* + 1. Bilge, L.、Balzarotti, D.、Robertson, W.、Kirda, E. 和 Kruegel, C.：披露：通过大规模网络流分析检测僵尸网络命令和控制服务器。在第 28 届计算机安全应用年度会议记录中

会议，第 129-138 页。 ACM，2012 年。

* + 1. Antonakakis, M.、Perdisci, R.、Lee, W.、Vasiloglou, N. 和 Dagon, D.：检测上层 dns 层次结构的恶意软件域。在 USENIX 安全研讨会，第 11 卷，第 1-16 页，2011 年。
    2. Antonakakis, M.、Perdisci, R.、Nadji, Y.、Vasiloglou, N.、Abu-Nimeh, S.、Lee, W. 和 Dagon, D.：从一次性流量到机器人：检测基于 dga 的恶意软件。在 USENIX 安全研讨会，第 12 卷，2012 年。
    3. Warrender, C.、Forrest, S. 和 Pearlmutter, B.：使用系统调用检测入侵：替代数据模型。在标准普尔。 IEEE，1999 年。
    4. Sekar, R.、Bendre, M.、Dhurjati, D. 和 Bollineni, P.：一种基于自动机的快速检测异常程序行为的方法。在标准普尔。 IEEE，2001 年。
    5. Wagner, D. 和 Dean, D.：通过静态分析进行入侵检测。在标准普尔。 IEEE，2001 年。
    6. Kruegel, C.、Valeur, F. 和 Vigna, G.：入侵检测和关联：挑战和解决方案。施普林格科学与商业媒体，2005 年。
    7. Feng, H. H.、Kolesnikov, O. M.、Fogla, P.、Lee, W. 和 Gong, W.：使用调用堆栈信息进行异常检测。在标准普尔。 IEEE，2003 年。
    8. Lee, W.、Stolfo, S. J. 和 Mok, K. W.：用于构建入侵检测模型的数据挖掘框架。在标准普尔。 IEEE，1999 年。
    9. Gao, D.、Reiter, M. K. 和 Song, D.：用于异常检测的执行图灰盒提取。在 CCS 中。 ACM，2004 年。
    10. Kruegel, C. 和 Vigna, G.：基于 Web 的攻击的异常检测。在 CCS 中。 ACM，2003 年。
    11. Manzoor, E.、Milajerdi, S. M. 和 Akoglu, L.：流式异构图中的快速内存高效异常检测。在第 22 届 ACM SIGKDD 知识发现和数据挖掘国际会议论文集中，第 1035-1044 页。ACM，2016。
    12. Berlin, K.、Slater, D. 和 Saxe, J.：使用 Windows 审计日志进行恶意行为检测。在第 8 届 ACM 人工智能与安全研讨会论文集上，2015 年。
    13. Shu, X.、Yao, D. 和 Ramakrishnan, N.：挖掘隐藏在极长执行路径中的隐形程序攻击。在 CCS 中。 ACM，2015 年。
    14. Ko, C.、Ruschitzka, M. 和 Levitt, K.：分布式系统中安全关键程序的执行监控：基于规范的方法。在标准普尔。 IEEE，1997 年。
    15. Uppuluri, P. 和 Sekar, R.：基于规范的入侵检测经验。在 RAID 中。斯普林格，2001 年。
    16. Debar, H. 和 Wespi, A.：入侵检测警报的聚合和关联。在 RAID 中。斯普林格，2001 年。
    17. Ning, P. 和 Xu, D.：从入侵警报中学习攻击策略。在 CCS 中。 ACM，2003 年。
    18. Qin, X. 和 Lee, W.：信息安全警报数据的统计因果分析。在 RAID 中。斯普林格，2003 年。
    19. Noel, S.、Robertson, E. 和 Jajodia, S.：关联入侵事件并通过攻击图距离构建攻击场景。在ACSAC。 IEEE，2004 年。
    20. Wang, W. 和 Daniels, T. E.：基于图形的网络取证分析方法。信息和系统安全交易 (TISSEC)，2008 年。
    21. Zhai, Y.、Ning, P. 和 Xu, J.：集成 ids 警报相关性和 os-level 依赖跟踪。在情报和安全信息学国际会议上，第 272-284 页。斯普林格，2006 年。
    22. Kruegel, C.、Valeur, F. 和 Vigna, G.：入侵检测和关联：挑战和解决方案，第 14 卷。Springer 科学与商业媒体，2004 年。
    23. Wang, X.、Ding, X.、Tung, A. K.、Ying, S. 和 Jin, H.：一种有效的图索引方法。 2012 年 IEEE 第 28 届国际数据工程会议，第 210-221 页。 IEEE，2012。
    24. Giugno, R. 和 Shasha, D.：Graphgrep：一种快速通用的图形查询方法。在模式识别中，2002 年。诉讼。第 16 届国际会议，第 2 卷，第 112-115 页。 IEEE，2002 年。
    25. Sun, Z.、Wang, H.、Wang, H.、Shao, B. 和 Li, J.：十亿节点图上的高效子图匹配。 VLDB 捐赠基金会议记录，5(9):788–799，2012。
    26. Gao, P., Xiao, X., Li, D., Li, Z., Jee, K., Wu, Z., Kim, C. H., Kulkarni, S. R. 和 Mittal, P.：{SAQL}：流基于查询系统的实时异常系统行为检测。在第 27 届 {USENIX} 安全研讨会（{USENIX} 安全 18）中，第 639-656 页，2018 年。

* + 1. Gao, P.、Xiao, X.、Li, Z.、Xu, F.、Kulkarni, S. R. 和 Mittal, P.：{AIQL}：从系统监控数据中进行有效的攻击调查。 2018 年 {USENIX} 年度技术会议 ({USENIX}{ATC} 18)，第 113-126 页，2018 年。
    2. Pasquier, T.、Han, X.、Moyer, T.、Bates, A.、Hermant, O.、Eyers, D.、Bacon, J. 和 Seltzer, M.：整个系统的运行时分析证明- 金融。在 2018 年 ACM SIGSAC 计算机和通信安全会议记录中，CCS '18，第 1601-1616 页，纽约，纽约，美国，2018 年。ACM。
    3. 舒，x，阿劳若，f，沙莱斯，D. L，施托克林，M. P .，张，j .，黄，h .和饶，

J.R.：威胁情报计算。在 2018 年 ACM SIGSAC 会议记录中

计算机和通信安全会议，CCS '18，第 1883-1898 页，美国纽约州纽约市，2018 年。ACM。

* + 1. Christodorescu, M.、Jha, S. 和 Kruegel, C.：恶意行为的挖掘规范。在第六届欧洲软件工程联席会议论文集上

会议和关于软件工程基础的 ACM SIGSOFT 研讨会，第 5-14 页。 ACM，2007 年。

* + 1. [ms-smb2]：服务器消息块 (smb) 协议版本 2 和 3。 [https://msdn.](https://msdn.microsoft.com/en-us/library/cc246231.aspx) [microsoft.com/en-us/library/cc246231.aspx](https://msdn.microsoft.com/en-us/library/cc246231.aspx).
    2. 关于 这 元数据 计程器。 [https://www.offensive-security.com/](https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/) [元智能释放/约计长/](https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/).

# 生活

**姓名** 萨迪克·莫明尼·米兰迪尔

**教育** 博士，计算机科学，伊利诺伊大学芝加哥分校，伊利诺伊州芝加哥，2014-2020。

理学硕士，信息系统工程，谢里夫理工大学，德黑兰，伊朗，2010-2012。

理学士，计算机系统工程，设拉子科技大学，伊朗设拉子，2006-2010。

**出版物** Miajerdi SM、Eshete B、Gjomemo R、Venkatakrishnan VN。 “POIROT：将攻击行为与网络威胁搜寻的内核审计记录保持一致”。 ACM SIGSAC 计算机和通信安全 (CCS) 会议论文集。 ACM，2019 年 11 月。

米拉杰迪 SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN。 “福尔摩斯：通过可疑信息流关联的实时 APT 检测”。 IEEE 安全和隐私 (S&P) 研讨会论文集。 IEEE，2019 年 5 月。

Miajerdi SM、Eshete B、Gjomemo R、Venkatakrishnan VN。 “ProPatrol：通过提取的高级任务进行攻击调查”。信息系统安全国际会议。计算机科学讲义，第 11281 卷。Springer，2018 年 12 月。

Hossain MN、Milajerdi SM、Wang J、Eshete B、Gjomemo R、Sekar R、Stoller S、Venkatakrishnan VN。 “SLEUTH：从 COTS 审计数据重构实时攻击场景”。第 26 届 USENIX 安全研讨会论文集。 USENIX 安全，2017 年。

Manzoor E、Milajerdi SM、Akoglu L.“流式异构图中的快速内存高效异常检测”。第 22 届 ACM SIGKDD 知识发现和数据挖掘国际会议论文集。 ACM，2016 年。

Milajerdi SM, Kharrazi M.“Tor 匿名网络的复合度量路径选择技术”。系统与软件杂志。 2015 年。

198