

订阅DeepL Pro以编辑此演示文稿。  
访问[www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)，了解更多信息。

EXTRACTOR: 从威胁报告中提取攻击行为

摘要--网络威胁情报（CTI）报告中包含的攻击知识对于有效识别和快速应对网络威胁非常重要。然而，这些知识往往被嵌入到大量的文本中，因此难以有效利用。为了应对这一挑战，我们提出了一种名为EXTRACTOR的新方法和工具，可以从CTI报告中精确地自动提取简洁的攻击行为。EXTRACTOR对文本不做任何强硬的假设，能够从非结构化的文本中提取攻击行为作为来源图。我们使用来自不同来源的真实事件报告以及DARPA对抗行动的报告来评估EXTRACTOR，这些报告涉及Windows、Linux和FreeBSD等不同操作系统平台上的一些攻击活动。我们的评估结果表明，EXTRACTOR可以从CTI报告中提取简明的出处图，并表明这些图可以成功地被网络分析工具用于威胁猎取中。

1. 简介

网络威胁情报（CTI），通常在技术报告、白皮书、博客和新闻组中报告，是有关网络攻击的宝贵信息来源。这些报告用自然语言描述了攻击的许多方面，包括行动的顺序、对被攻击系统的影响以及破坏指标（IOC）。CTI报告中包含的知识对于网络操作和响应人员、系统管理员以及入侵检测和预防产品的供应商来说是至关重要的。

以前的研究[52]、[45]、[89]利用各种NLP技术自动提取CTI报告中以IOCs（即[52]、[89]）和威胁行动（即[45]）形式存在的知识。虽然这些工作为从CTI报告中自动提取威胁元素（IOCs和威胁行动）提供了一个良好的起点，但它们没有提取IOCs和威胁行动之间的关系，以提供攻击行为的全面视图。这种攻击行为的提取在威胁猎取活动中至关重要。特别是，提取攻击行为和攻击的全貌需要提取所涉及的实体（如文件和套接字）、行动（如系统调用）、它们之间的因果关系和时间依赖，以及实体之间的信息流。提取攻击行为需要一种能够从自然文本中理解 "谁对谁做了什么"、"何时 "和 "何处 "的方法。这项任务提出了几个挑战。

挑战1：言之有物。威胁报告中注入了大量不相关的文字；通常，报告中只有一小部分描述了攻击行为。例如，对恶意软件的地理来源的描述虽然有趣，但对描述系统中的恶意软件行为没有帮助。

挑战2.CTI文本的复杂性 以往方法的一个重要假设是，CTI报告的文本结构（a）相对简单[52]，或者（b）遵循特定的语法结构[45]，或者（c）假设在描述概念时有一些模式[88]，或者（d）在表述句子时考虑主语、动词和宾语的稳定语法关系[52]，[45]。虽然这些假设并不妨碍这些工作的目标，即孤立地提取IOC和威胁行动，但事实上，大多数CTI报告包含更复杂的特定领域背景（见第二节），这使得攻击行为的提取和因果推理更具挑战性。CTI报告在语法和语义上的复杂性、技术术语的普遍性以及这些报告中缺乏适当的标点符号[62]，很容易影响报告的解释和攻击行为的提取。

挑战3。关系提取。正如[52]和[45]所建议的，可以使用字符串匹配和分类器等方法提取IOC和威胁行动。然而，在保持整个报告中攻击的简明因果关系、时间和信息流的同时，提取大的画面要复杂得多，而且具有挑战性。事实上，准确解释技术报告中的复杂逻辑是已知的NLP中的一个公开问题[62]。

在本文中，我们介绍了EXTRACTOR1，它解决了这些挑战。EXTRACTOR的主要目标是以图的形式从技术报告中简洁地提取攻击行为的全貌。EXTRACTOR克服了第一个挑战，提出了一种新的文本总结方法，将攻击行为文本与其他文本区分开来。为了克服第二个挑战并优化整个系统的性能，EXTRACTOR使用了一套技术，将高度复杂的文本转化为更容易消费的形式。为了解决第三个挑战，EXTRACTOR使用了一种新颖的方法--语义角色标记（SRL），它允许我们通过推断 "谁对谁做了什么"、"何时 "和 "何地 "的事实来提取攻击行为和句子的主体、客体和行动（这些步骤的细节在第二和第三部分讨论）。最后，在最后一步中，SRL的结果以图的形式呈现，描述了攻击的步骤、人工制品、相关实体之间的因果信息流。

除了提取完整的攻击画面这一主要目标外，EXTRACTOR还遵循另外两个目标。

目标1：可操作的情报。我们希望自动构建我们称之为可操作的情报。我们希望从CTI报告中只提取可用于检测的信息，或威胁猎取。这意味着我们从文本中提取的攻击行为必须可以在系统审计日志中观察到，并且可以有效地用于威胁检测。这是每个从CTI报告中提取攻击信息的方法的一个重要目标。事实上，我们设想将EXTRACTOR的部署作为威胁狩猎行动的第一步。

目标2：准确处理大量的CTI报告。我们希望能处理大量的CTI报告、博客和来自威胁检测中心的攻击描述。完成这一目标将使分析员能够自动地从一个比目前可能的大得多的知识来源中挖掘。EXTRACTOR的应用。正如已经被广泛证明的那样，在检测威胁时，有一个简明的攻击行为描述比有一个简单的IOCs集合要好[59], [84], [39], [55], [50], [56]。EXTRACTOR能够从CTI报告中建立代表这种简明描述的图形表示，从而指导入侵检测和威胁猎取系统。EXTRACTOR的另一个设想的用途是，从不同组织中与同一攻击有关的各种CTI来源中提取信息。这是为了全面了解同一恶意行为者在不同情况下可能的行为。

EXTRACTOR通过几个重要的贡献大大超越了目前的技术水平。特别是，EXTRACTOR：1）大大扩展了可以处理的CTI报告的范围，2）比以前的研究（例如。[52], [45], [89]）；这包括对攻击的因果关系和时间顺序的提取，3）实现了提取攻击的人工制品之间的语义关系的新颖应用，使其能够获得一个更好的攻击图片，4）实现了几个文本简化和减少（或总结）的新颖应用，使其能够在不损失有用信息的情况下浓缩文本。

本文的组织结构如下。在第二节中，我们提供了一个更详细的问题描述和一些背景信息。在第三节中，我们详细描述了我们的方法。在第四节中，我们简要介绍了实施和使用的不同工具。第五节介绍了评估情况。第六节提供了讨论，而第七节和第八节分别包含了相关的工作和结论。

1. 问题和背景

A.问题描述

正如引言中提到的，本文的主要目标是从通用CTI报告中提取代表攻击行为的可操作图。我们所说的可操作性是指将提取的知识作为威胁猎取的信号这一重要目标。我们把这些图称为出处图。证据图是内核审计日志的常见表示方法[49], [48]。它们将系统中的事件（系统调用）表示为实体（进程、文件、套接字）之间的边。最近，在大量的研究中，证明图被成功地用于威胁检测和取证分析[41]、[59]、[84]、[39]、[55]、[50]、[56]。

图1显示了一个CTI报告中包含的文本的例子，其灵感来源于njRAT攻击[80]。这个例子将在本文中用来说明我们方法的不同方面。从该文本中提取的相应来源图的例子也显示在图1的右侧。可以注意到，出处图包含代表参与攻击的实体（进程、注册表键等）的节点和代表这些节点所执行的行动的边。此外，节点的名称是可以在审计日志中观察到的，连接节点的边代表也出现在审计日志中的系统调用（目标1：可操作情报）。此外，该图只包含与攻击行为相关的节点，没有其他信息（主要目标是完整的攻击画面和简洁性）。我们注意到，图1中的自然文本没有任何特定的结构（目标2：处理用自然语言编写的CTI报告）。

从用自然语言编写的CTI报告中提取简明和可操作的来源图有几个挑战。首先，我们需要将攻击行为文本与报告的其他部分区分开来。这意味着从自然文本中只提取描述攻击行为和可以在审计日志中观察到的那种关系，而过滤掉其余文本。因此，我们需要理解文本中提到的系统实体之间发生的关系和行动，以便将这些行动映射到系统调用，这些调用在来源图中被表示为边。第二，我们需要克服CTI文本的复杂性，这可能会影响我们的图提取。这意味着要解决自然语言写作中存在的不同类型的模糊性和复杂性。接下来我们将更详细地描述必须解决的挑战。

B.挑战

啰嗦性。一般来说，CTI报告可以是冗长的。包含有用信息的句子可能嵌套在与攻击无关的文本中，例如，介绍性细节。例如，在42页的DustySky报告[7]中，只有11句话描述了在审计日志中可以观察到的实际攻击行为。我们使用一种新颖的总结技术将有用的内容与无用的内容分开（见III-C）。CTI文本复杂度。网络安全领域使用的语言有一些特殊性，NLP工具/技术（为更多的通用领域开发）往往难以应对。这使得这些工具的使用具有挑战性。我们在下面列出其中的一些特殊性。

标点符号。许多CTI报告没有使用句尾字符'...，！，？'来划分句子的界限。这使得流行的NLP工具包，如Stanford[14]、NLTK[54]和spaCy[9]，很难理解CTI报告中真正的句子边界，导致文本中的长句子，每个长句子都包含几个短句子。例如，从对微软威胁报告中心的4020份威胁报告的观察中，我们注意到作者倾向于在一个句子中打包许多行动，因此使平均句子长度等于52个单词（有些例子每句长达313个单词）。相比之下，NLP工具通常被训练和设计的平均英语文本，每句大约包含14.4个单词[5]。

特定领域的词。表示网络安全领域对象的词可能与普通英语中使用的词有不同的含义和语境，而NLP工具是在普通英语中训练的。例如，IP地址、路径、进程名称、系统调用名称和许多其他术语经常被普通NLP工具误解。这一挑战必须由一个能带来领域洞察力的机制来应对，以便为这些术语分配含义。

省略号。这个术语指的是句子中的一个缺口。1）有一个缺失的主语，或2）有一个缺失的宾语[24]。这种结构在自然英语写作中并不常见[64]，但在CTI报告中却非常常见，因为攻击被描述为行动的序列。例如，在图1中的以下位置创建一个自己的副本代表了省略号主语的一个例子。

代词。代词在英语中是非常常用的[70]。忽略代词可能会导致它们作为节点出现在出处图中，而不是指代实体的位置。其他语言学结构。结构的复杂性和各种语言技术的使用，如拟态、名词化和列表（III-B），会使普通的NLP工具感到困惑。总的效果是，许多主语、动词和宾语被错误地分类，无法解决。

关系提取。克服前面的挑战可以帮助确定攻击描述中重要的正确实体。下一步是确定 "谁对谁做了什么"、"何时 "和 "何地"，或者换句话说，我们需要发现进程和系统对象之间的关系以及它们与审计事件的映射。目前与该任务相关的方法，如统计依赖分析器，在来自与自然英语文本不同的领域的句子上的性能下降是众所周知的[57], [58]。为了解决这个问题，我们需要一种更全面的方法，将句子的语义考虑在内，而不是仅仅依靠句子的句法结构（即纯粹使用依赖分析，如[52]、[45]、[89]所使用的）。正如我们将在下一节中详细说明的那样，为了解决这一挑战，我们使用了语义角色标签（SRL），这是一种能够检测句子中实体之间语义关系的处理模型。

在继续描述我们的方法之前，我们先简要介绍一下本文中所使用的NLP技术的背景。

C.NLP背景

语篇（POS）。POS标签为句子中的每个词分配一个句法角色（如名词、动词等）。然而，在某些情况下，POS模型可能无法正确标记单词。在图2中，形容词 following 和 downloaded 被错误地标记为动词（VBG/VBN）。

依赖性分析（DP）。DP为句子中的单词分配语法联系和依赖关系。DP标签的例子包括nsubj代表句子的主语，obj代表句子的宾语，等等。然而，如果句子的复杂性增加，DP可能无法准确地检测标签和单词之间的关系。此外，DP标签员可能无法分配正确的标签，特别是如果他们没有接受过包含技术语言的语境训练。常见的错误包括将过去分词形式标记为形容词，将动词标记为名词，等等。在我们目前的问题中，利用DP的另一个缺点是，它们产生的标签只指语法关系，如主语、宾语等。因此，它们不能帮助完成需要理解不同句子成分之间语义的任务。这些关系可能包括时间性（某事何时发生）、方式（某事如何发生）等。事实上，为了实现我们的目标，需要更深入的理解。

图2展示了由spaCy[9]（上）和Stanford[13]（下）驱动的POS标签和DP树的例子，其中不正确的标签如follow和download（如前面讨论的）导致DP关系错误地生成为主语和宾语。另一个例子是顶部句子中的动词Beacon，它被错误地标记为一个专有名词（即NNP）。更多关于POS和DP的信息可以在[27]和[22]找到。

语义角色标签（SRL）。SRL本质上是确定 "谁对谁做了什么"、"何时 "和 "何地"[67]。SRL是一种较新的NLP技术，它可以为句子中的短语和单词分配语义标签，每个标签都规定了每个短语或单词在句子中与句子的谓语或动词相关联的语义角色。在SRL中，分配给句子成分的标签被称为论据（用ARG表示）。一些参数的例子和相应的语义角色见表一。

1. 办法

简而言之，EXTRACTOR的运作方式是对文本进行不同轮次的转换，使其从高度复杂和潜在模糊的形式变成较简单的形式。这种简化的文本被进一步处理，以获得一个可以成功用于威胁检测的出处图。图3显示了EXTRACTOR的概况。EXTRACTOR有四个主要部分。1）归一化，2）解析，3）总结，和4）图形生成。归一化负责最初一轮的句子简化和转换为规范形式。解决这些句子中的歧义（这两个部分有助于解决CTI文本复杂性的挑战）。归纳删除与攻击行为没有严格关系的那部分文本，这些文本在日志中无法观察到。最后，图谱生成负责解决文本中事件的时间和因果顺序，并建立最终的出处图（该组件解决关系提取的挑战）。这些组件中的一些可以由一组包含与CTI语言有关的术语的字典来协助（依靠特定领域的概念字典是许多基于知识的NLP系统中的常见方法[76], [71], [78]）。特别是，EXTRACTOR使用两个字典。首先，我们的系统调用同义词字典，它包含代表系统调用的动词（例如，写，叉）和它们相应的同义词。这些同义词代表了可以在CTI报告中使用的可能动词，并且很可能指的是系统调用。第二，我们的CTI名词词典包含了CTI报告中常用的名词短语，以及同一概念的不同文本表述。前者包含87个代表系统调用的动词，而后者则包含CTI报告中超过1112个常见的名词短语。这两个词典在图3中都有描述，并将在第四节中进一步讨论。

A.归一化

为了解决CTI文本复杂性的挑战，并使EXTRACTOR使用的技术的准确性最大化，我们首先必须有一些规范的句子形式。我们通过规范化来实现这一点，规范化负责将长而复杂的句子分解为以规范形式出现的较短的句子，这更容易处理。直观地说，我们希望每个句子都能表达一个行动，这样行动的主语和宾语以及行动本身就更容易识别。规范化由标记化、同质化和转换组成。这些步骤分别进行句子边界的检测、词的同质化和被动到主动动词的转换。接下来我们将分别描述这些步骤。

符号化。正确地定义句子的边界是非常重要的，因为EXTRACTOR使用的几种技术都是在句子层面上操作的。然而，现有的句子标记器（例如NLTK[54]）在发现句子边界时，通常只考虑经典的标点符号（'...，！，?'），在CTI报告上表现不佳。事实上，在这个领域中，包含多个动作的长句子和非标准的句子分隔符的情况很普遍。例如，在微软的威胁报告中，平均句子长度几乎是普通英语句子的四倍。

为了解决这个问题，我们设计了一个专门用于CTI报告的强化标记器。特别是，除了传统的句子定界符外，我们的标记器还使用新的行、项目符号、列举的数字、以及标题和页眉，作为句子定界符，将长序列划分为较短的序列集。在将长句分割成较短的词语序列后，如果每个短序列满足以下情况之一，则被 "提升 "为一个句子；1）该序列以大写的主语开始，它包含构成一个完整句子所需的所有成分（主语、谓语、宾语），并且前面和后面的序列也构成完整的句子；2）该句子以系统调用字典中的动词开始，它包含构成一个完整句子所需的所有成分（除去主语），并且前面和后面的序列也构成完整句子。后一种情况代表了（CTI报告中）Ellipsis Subject的常见现象（见第二节B）。如果上述两种情况都不满足，我们就认为该句子是一个无法突破的完整句子。

作为这个程序的一个例子，请看图4，它说明了论文中描述的几种技术。在这个图中，跨越图4中第4-9行的长句首先被分割成较短的序列（图中每行一个）。接下来，每个序列都被一个POS标记器和DP标记，并检查它是否满足上述两个条件之一。在图中，第4行的序列满足第一种情况的条件，而第5-9行的序列则满足第二种情况的条件（省略号主体）。

标记器的结果是一组较短的句子，更有可能描述一个单一的行动。

同质化。CTI报告经常包含一些结构和同义词，这些结构和同义词会带来歧义并影响最终结果的质量。例如，C2、C&C和命令与控制是同一实体的不同表述，而像存储、保存这样的动词可能表示一个与写系统调用相对应的动作。同质化是指同一概念的多种文本表述被同一文本表述所取代的过程。

我们使用两个专门建立的字典对名词短语和动词进行同源化处理，将CTI报告中出现的不同行话和同义词映射到审计日志中可以观察到的实体和行动。例如，C2、C&C、命令和控制中的每一个都被映射到IP:.\*，这是一个代表IP地址的通配符。以同样的方式，我们将系统调用字典中与系统调用同义的动词翻译成该系统调用动词。

同质化大大降低了文本的异质性，并支持我们提供可操作情报的目标。我们决定，被选来代表一个同义词组的所有其他词的单个词，是一个极有可能作为一个系统实体，在日志或系统调用中可以观察到的。

转换。作为文本规范化的最后一步，EXTRACTOR将被动语态转换为主动语态。这种转换有助于发现系统主体（过程）和系统客体，并使因果关系的推断更加准确，这一点在III-D小节中讨论。

为了进行这种转换，我们首先使用POS和DP标签检测被动句子。这类句子主要由DP树中的特定和已知模式来代表。例如，考虑下载的文件被恶意软件删除这一句子。在DP树中，is被标记为辅助（和被动）动词，delete被标记为动词和DP树的头部，下载的文件是一个名词短语，是被动语态的主语（nsubjpass），by malware是宾语（obj）。请注意，在某些情况下，代理人并没有出现在被动句中，而是被暗示。例如，在图4的第10行，代理是恶意软件，但句子中没有提到它。利用这些模式，EXTRACTOR可以检测出被动句子，并区分出有明确代理的被动句子和有隐含代理的被动句子。在前一种情况下，它可以切换代理和主语，并将被动动词连接成主动动词。

这一步的最终结果是，长句被转化为主动形式的短句，可能每句表达一个动作。

B.决议

在规范化之后，解析将指向同一实体的隐式引用调和为实际的引用。这些隐性引用必须被明确化，原因有二。首先，隐性引用降低了后续步骤的准确性，并使最终的出处图变得模糊和不精确。其次，审计日志只包含明确命名的实体，每一种威胁猎取方法都无法将系统进程与代词和其他隐性参考匹配。关于这种语言结构的更透彻、更精彩的讨论可以在[81]、[65]、[83]找到。

省略主语解析（ESR）。正如第二节中所讨论的，省略号主语是一种语言结构，其中句子的主语不存在。这种结构在大量的CTI报告中都有，用于描述同一行为人（过程或攻击者）所进行的一系列行动--第五节介绍了这种现象在各种来源中的普及情况。省略的主体使最先进的NLP工具包感到困惑，从而导致叙事顺序和故事关系（行动的主体和客体）的丢失。图4中第5-9行描述的所有动作都是省略主体的例子。

为了解决这个问题，我们开发了一个Ellipsis Subject Resolver（ESR）模块。该模块利用POS和DP解析以及系统调用的字典。解决这个问题的第一步是检测缺失主语的句子。这一步使用POS和DP以及系统调用字典，正如在讨论标记器时所描述的那样（第III-A小节）。一旦检测到这种句子，ESR就会在当前句子之前的句子中出现的实体中建立一个候选主体列表。接下来，该模块根据候选者与缺失主语的句子的距离（以句子数计算），从列表中挑选出最可能的候选者。特别是，距离越近的候选人被选中的概率就越高。例如，在图4中，第5-9行的句子中缺少主语。ESR模块检测了前面的句子中的主语和其他对象，它选择了冒号前出现的代词it作为主语。

代词解析（PR）。代词解析是指代词被映射和替换到它们所指的前述实体的过程。在没有PR的情况下处理文档（构建出处图）会导致一个实体出现多个节点（即代词）。例如，在图4中，第3行和第5行的代词it和itself应该被替换为实际的主体Authorization.exe。

为了解决代词问题，我们采用了一个流行的核心词解析模型--NeuralCoref[10]。我们注意到，这个模型在解决CTI报告领域的代词方面效果最好，特别是在前面的ESR和Tokenization步骤之后。图4，第4、5、6行展示了已解决的代词（即it和itself）和它们相应的参考（Autorization.exe）。

实体解析（ER）。实体解析是指在同一句子中提到另一个实体的名词或动词短语被该实体所取代或作为多余的内容被删除的过程。一般来说，这是一项庞大的任务，然而，我们指出，我们只对提取可操作的信息感兴趣，因此，可以专注于只对可能出现在审计日志中的实体和行动执行ER。事实上，通过对大量CTI报告的初步观察，我们注意到这些实体和行动之间的冗余主要以三种不同的语言形式出现。

拟声词。隐喻是指用一个词或代词来指代句子中以前使用过的另一个词或短语，以避免重复。例如，在图4的第11行，以下文件指的是mscno.exe authorization.EXE-0AD199D6.pf。这种形式在CTI报告中很普遍，它被用来描述参与某种共同行动的实体的列表。名义化。这是一种辅助动词与名词一起使用的形式，以代替动词。例如，makes a modification代替modifies。这种形式经常被用于代表系统调用的动作。特别是，它在TrendMicro中出现了大约3524次，在微软博客中出现了1261次。另一种与系统调用有关的类似形式作为辅助动词出现在与系统调用有关的实际动词后面，例如，图4中的tries to open而不是opens。

为了解决这些情况，我们将POS标签和DP与CTI名词词典中包含的领域知识或在每种情况下出现的常见短语的语料库（例如，以下文件是一个常见的拟态）相结合。特别是，如果在文本中检测到上述三种形式之一，我们会检索该形式附近的其他词的DP和POS标签，并检查它们是否遵循特定的模式。特别是，对于拟声词，我们检查在出现拟声词的主句之后是否有一列名词短语，并以名词短语替换拟声词。对于名词化，我们检查语料库中出现的名词是否是前面一个辅助动词的宾语，并用其动词形式替换该名词（例如，makes the modification！modifies）。对于辅助动词，我们检测是否有一个不定式形式出现在可能代表系统调用的动词之前，并将整个短语替换为实际的动词（tries to open ! open）。

在解析步骤完成后，文本由具有明确主语、宾语和动词的句子组成。ER模块也在一定程度上减少了文本的数量。然而，主要的文本缩减步骤是在 "解析 "之后执行的，接下来将介绍。

C.文本归纳

为了减少言辞，获得可直接用于检测攻击的攻击行为的简明描述，必须删除大量的多余文字。理想情况下，只有那些描述可能在审计日志中观察到的行为的句子应该被保留下来。要做到这一点，EXTRACTOR必须了解哪些句子严格描述了攻击行为，哪些句子没有。以前的相关工作使用主题分类[52]，[45]来识别域外语境（例如，广告文本与技术文本）中与主题相关的语境。虽然这些方法成功地将不相关的内容（如广告）与技术内容分开，但它们还不够强大，无法将后者分成描述可观察到的攻击行为的行为内容和其他 "技术 "内容，后者作为介绍或背景描述。我们把这个问题称为句子的冗长性。图5中显示了一个句子冗长的例子。在该图中，报告的文本显示在顶部。左上角方框中的句子，由Productive标记，包含了对恶意软件实际行为的描述，这可以在审计日志中观察到，因此，这对检测是有用的。右上角由非生产性标记的句子包含对恶意软件的补充描述，但没有可以在审计日志中观察到的行动。尽管这两个文本部分在性质上是技术性的，而且是关于同一主题的，但我们只对有成效的文本感兴趣，并希望删除无成效的文本。

另一个需要解决的问题是我们所称的词的冗长性。特别是，在每个句子里面，通常会出现一些词语结构，如副词和形容词短语，这些词语对行为描述没有帮助，可以安全地删除（例如，然而，可以，还有在图中）。

为了处理这些问题，我们设计了一个两步法。这个方法如图5所示，由一个处理句子粗制滥造的BERT分类器和一个处理单词粗制滥造的BiLSTM网络组成。

句子的动词性。为了区分描述实际威胁行为的句子和不代表威胁行为的句子，我们需要超越主题分类，对文本有更深入的理解。直观地说，与其他句子相比，有成效的句子在主语和宾语之间表达了更多的 "直接 "联系。因此，为了对这些联系进行分类，文本的语言学模型必须建立一个关于单词上下文的细粒度表示。

目前，建立这种细粒度表示法的最佳模型之一是BERT（Bidirectional Encoder Representations from Transformer）[28]。与Word2Vec[1]和GloVe[2]的单词表征不同，BERT建立了单词的上下文表征，同时考虑到了单词之前的文本和单词之后的文本。换句话说，BERT考虑了每个单词的周围环境。此外，BERT还学习了子词的嵌入，即从词干中得出的一个词的子成分。这使得该模型能够更有效地处理词汇外的单词。一般来说，这种能力对技术性网络安全文件是有益的，因为这些文件可能包含在BERT的训练集中没有出现的复杂词汇。因此，BERT可以比其他方法更好地将句子分类为生产性和非生产性。特别是，我们将8,000个威胁句子标记为生产性和非生产性两类，并在这个集合上训练BERT。结果是有希望的，并在第五节中详细说明。

词的Verbosity。文本总结器的第二步是从它从BERT收到的生产性句子中删除不必要的词。它由两个阶段组成，一个是得出句子成分语义作用的BiLSTM网络，另一个是去词阶段。我们发现BiLSTM最适合这一目的，因为它可以处理技术文件中出现的长距离依赖关系。

在一个句子被BiLSTM网络处理后，其成分被标记为Agent、Patient和Action，以及其他类型的参数（例如，在图5中，However这个词被标记为ARGM-DIS，一个连接句子和前句的话语标记）。在下一个阶段，不必要的句子成分被删除。从理论上讲，这只能通过保留句子中的Agent、Action和Patient成分来实现。然而，在某些情况下，这种方法会删除重要的信息。例如，像恶意软件.exe何时被执行这样的句子可能被标记为ARGM-TMP（一个时间标记），并被天真的方法删除，导致攻击的一个重要部分被删除。为了使这个第二阶段更加精确，并且不删除可能包含像恶意软件.exe这样的重要对象的句子成分，我们使用系统实体提取器（SEE），这将在第III-D2节介绍。特别是，如果一个被标记为删除的句子组件不包含任何可由SEE组件的规则生成的实体，它将被删除。

文本归纳是EXTRACTOR的核心组成部分之一。它负责大大减少文本的复杂性和数量，同时保留最重要的描述可观察行为的句子。

D.图形提取

经过前面的步骤，得到的文本是这样一种形式：系统主体（如进程）、对象（如文件、套接字）和动作（如执行）是明确的、有序的，而且大部分多余的文本被消除了。在这最后一步，EXTRACTOR解决了关系提取的难题，从简化的文本中提取出一个出处图。

尽管这一步的文本非常简单，但将节点分配给主语和宾语、将边分配给动词的天真图式提取会产生含糊不清的大图。这是因为在同一个句子中可能表达了几个角色和主客体之间的关系。为了处理这个问题，我们使用语义角色标签（SRL）和一套规则来提取因果关系和信息流的方向。接下来将介绍这些。

1）语义角色标记（SRL）。如第二节所述，SRL是一种发现句子中的语义角色的技术。为了对SRL的力量有一个直观的概述，请考虑图6中的两个例子，一个是主动形式的，一个是被动形式的。SRL能够从每个句子中提取两个角色（用Raw SRL表示），并理解哪个名词是病人（也就是动作落在上面的人，用ARG1表示），哪个是代理人（携带动作的名词，用ARG0表示）。在我们的讨论中，一个SRL角色可以被认为是一个动作。因此，SRL能够正确地将句子中的每个成分与语义标签联系起来。

EXTRACTOR考虑所有与SRL检测到的动词相关的可能参数，作为潜在的攻击主体或对象，然后修剪掉那些不是系统实体的参数。在修剪过程中，我们使用SEE模块（见第III-D2节），它可以检测到可能的系统实体名称（例如，文件或进程名称、IP和注册表键）。特别是，SEE模块分析每个节点，并修剪掉整个节点或节点名称中不符合正则表达式或应用程序名称的部分。图6中修剪的结果用Pruned SRL表示。

动作到系统调用的转换。在SRL之后，EXTRACTOR对SRL模块检测到的动词角色进行了第二次同质化步骤。这一步对于纠正由于POS和DP标记器的低效率而导致的最终错误是必要的，这可能会使动词漏掉，并保持其原始形式（未翻译成系统调用）。在这一步骤之后，我们对那些由SRL创建的、不代表系统调用动作的角色进行剪裁。例如，在这第二个修剪步骤之后，图6上半部分中与动词bypass有关的第二个角色被修剪掉了。

2）图形生成器（GB）。我们方法的最后一步是根据SRL的输出构建图。图形生成器分两步操作。首先，它将具有相同文本的SRL论据合并到同一个节点中，并使用SEE剔除不属于系统实体的词。接下来，GB使用以下方法构建图。1）节点-边缘-节点三联体。对于每个句子，如果它至少有三个角色，包括一个动词角色（作为连接器的系统调用表示）和两个节点，GB就会生成边和节点对。2）边缘方向。EXTRACTOR通过使用与系统调用字典相关的边缘方向的小地图来确定边缘的方向。这在本节后面的因果推理段落中有更详细的讨论。

系统实体提取器（SEE）。我们开发了SEE模块，以从SRL生成的角色中提取代表系统实体的简明节点，并修剪掉不能构成可能的系统实体的徒劳的演讲部分。SEE使用超过32种不同的正则表达式和一个应用程序名称或知名进程的数据库来检测可能的系统实体名称（例如，文件或进程名称、IP、注册表键）。特别是，SEE模块分析每个名词短语，并剔除整个短语或与正则表达式或应用程序名称不匹配的部分。例如，在这句话中，恶意软件删除了regex.exe。，SEE将（ARG0）修剪成星号（\*--在查询处理系统中会被推断为任何）并将（ARG1）变成regex.exe。这一步对于系统实体的简明准确的名称是必不可少的，可以用来搜索审计日志以进行威胁检测。这个模块还定义了最终图中节点的形状，其中矩形、椭圆形、五边形和菱形分别代表文件、进程、注册表和套接字。

因果推断。这一步决定了图中边的正确方向，以代表节点之间的因果关系和信息流。为了推断这个方向，它使用系统调用与系统流动方向的映射。该映射包含与每个系统调用相关联的条目，即主体和客体之间的边的方向（例如，对于发送系统调用，流向是从主体到客体，而对于recv，流向是从客体到主体）。此外，这一步解决了可能出现在CTI报告中的否定动词。事实上，在随意推理中，我们使用SRL标签检测否定句，如果没有影响角色的条件句，就清除被否定的角色。例如，svchost.exe不创建explorer.exe将被清除，因为没有条件句影响这个句子。

最后一步的输出是一个出处图，它清楚地显示了参与攻击和受攻击影响的实体作为节点，以及连接它们的系统调用作为边。图1显示了与运行实例有关的这种图的例子。

1. 执行情况

在这一节中，我们简要介绍EXTRACTOR使用的一些额外的实现细节、工具和技术。

NLP工具箱。我们使用了各种最先进的NLP工具箱的组合来实现我们的方法。这些工具包括spaCy POS和DP标记器、NLTK和Stanford [9], [13], [54]。我们在标记化、同质化、解析和被动到主动的转换步骤中使用SpaCy来确定不同句子成分的POS和DP标签。特别是，我们使用了spaCy的大型预训练统计模型第2版[82]，因为该模型在处理CTI报告时表现优于其他统计模型。

符号化。我们的句子标记器是建立在NLTK句子标记器之上的。NLTK（Natural Language ToolKit）是一个通用的NLP工具包，包含几个库和针对英语语言开发的统计自然语言处理。我们选择NLTK是因为我们发现它比其他的（spaCy、Stanford）更好用，而且它在处理CTI领域的文本时更加一致。

文本归纳。我们使用了一个12个隐藏层的BERT[28]来辨别有成效的句子和无成效的句子。为了训练我们的模型，我们使用了8,000个标记的句子。为了理解词语在文本总结器中的作用，我们使用了一个深度BiLSTM模型的重新实施[40]。由于该模型没有经过微调以处理网络安全的句子，我们使用了3,000个手动标记的句子来训练该模型。

SRL。为了实现SRL，我们使用了[79]上描述的方法，由[34]部署，该方法在NLP领域越来越受欢迎。为了采用该系统并在网络安全领域获得更精确的输出，我们用2000个与我们注意到系统未能正确预测角色的领域有关的网络安全句子进一步重新训练该模型。为了进一步完善，我们评估并展示了我们的再训练的结果（见第五部分D）。

数据集和词典的构建。为了构建字典和我们的数据集，我们使用了从各种来源刮来的CTI报告库。我们使用不同的来源，即APT报告库[6]、微软威胁中心[8]、赛门铁克安全中心[16]、威胁百科全书[18]和病毒雷达[20]，以确保多样性和覆盖面。

对于我们的文本总结器，我们对一个平衡的数据集进行注释，该数据集由8,000个从不同来源抽样的句子组成，并对其进行两类注释：生产性句子和非生产性句子。总共有3,800个句子被注释为生产性句子，4,200条信息被注释为非生产性句子。我们将数据集分成4,800个句子用于训练，1,600个句子用于验证，1,600个句子用于测试。我们使用不同的数据集进行测试和评估。

我们以迭代的方式进行注释，三位主题专家参与了我们数据集的注释工作。我们要求每个注释者将收集到的数据注释为生产性和非生产性两类中的一类。然后，在与注释者的几次讨论中，我们讨论并澄清了攻击行为（即生产性）与其他（即非生产性）的概念，以确保对攻击行为的理解是准确的。按照先前的指南和研究（即[33]和[30]），注释任务以迭代的方式开始。在每一轮中，分配200条信息，并与每个注释者讨论不同意见。在每一轮讨论之后，通过Cohens kappa系数衡量，达到100%的注释者之间的一致（IAA）。在三轮初始注释之后，注释者被分配到剩余的7,400个句子，其中IAA为91%。最后一轮的分歧被讨论，标签由本文的一位作者最终确定。

将动词短语翻译成相应的系统调用的另一个解决方案是使用像WordNet[60]和Thesaurus[17]这样的工具（像[45]这样的研究人员以前就使用过这种方法）。然而，我们注意到，通过建立一个简单但有效的字典，我们可以取得更好的结果。为了建立我们的字典，与注释我们的数据集的过程类似，我们与一个由三位安全专家组成的团队以迭代的方式工作。在一年的时间里，这些成员参与了审查和注释从各种来源随机选择的3000份技术威胁报告。然后，以迭代的方式，对提取的短语及其相应的同义词进行了讨论和商定。同样，系统调用的字典来自于WordNet[60]和Thesaurus[17]。这些同义词已经被提取出来并经过多次讨论以保证质量。表二列出了这个词典。另外，附录中的表XIII代表了名词词典的例子。

1. 评估

为了评估EXTRACTOR，我们设计了三个实验，每个实验都在具有不同写作风格的CTI报告上进行。在第一个实验中（§V-A），EXTRACTOR从一组描述真实世界事件的公共CTI报告中生成图形。在第二个实验中（§V-B），EXTRACTOR从DARPA透明计算项目[19]评估中的攻击描述中建立图形。最后，在一个大规模的实验中（§V-C），EXTRACTOR处理了来自微软安全情报[8]的4100份非结构化CTI报告和来自TrendMicro[18]的11600份报告，以提取出处图，在V-C节中进一步讨论。

在前两个实验中，我们以两种不同的方式评估EXTRACTOR。(1) 我们衡量EXTRACTOR在使用报告中存在的基础事实捕捉所有相关攻击行为的能力。为此，我们报告精度、召回率和F1分数。(2) 为了证明EXTRACTOR在支持威胁猎取方面的有用性，我们使用威胁猎取系统POIROT[59]，并使用EXTRACTOR生成的图。最后，为了评估我们方法的可扩展性，我们进行了一个大规模的实验，这将在第五节C中讨论。

威胁猎取。为了评估由EXTRACTOR生成的图对威胁检测的作用，我们使用了POIROT系统[59]。该系统将代表攻击事件的小型出处图（称为查询图）作为输入，并在从受攻击系统的审计日志建立的更大的出处图中搜索该图的嵌入。POIROT中的查询图是由专家在阅读CTI报告后手动建立的，代表了这些报告中描述的攻击活动。在我们的评估中，我们使用相同的CTI报告，用EXTRACTOR自动构建图，并将这些图作为POIROT的查询图。通过这种方式，我们比较了由人类专家构建的图和由EXTRACTOR构建的图，以及这两种图对检测威胁的有用程度。我们定义了一个P(G1;G2)=S的操作，其中G1代表由EXTRACTOR构建的图，G2代表更大的出处图，代表被攻击系统的审计日志。接下来，我们使用POIROT在G2中搜索G1，并检索出相似度得分S。如果S大于POIROT的阈值（t \_ 0:3），那么G1就被成功定位在G2中，表明成功检测到了威胁。否则，没有检测到攻击。关于POIROT的更多细节，请参考[59]。

在所有的实验中，我们衡量EXTRACTOR的假阳性和假阴性边缘。所谓假阳性边，我们指的是包含在提取图中的边，这些边并不代表攻击活动。所谓假负边，我们指的是应该被包括在提取图中的边。我们指出，这些假阳性和假阴性边缘的概念仅指最终图中的节点和边缘的存在（或缺乏），而不是指使用该最终图对威胁的实际检测。事实上，许多检测工具可能能够使用一小部分节点和边作为IOCs。作为一个具体的例子，我们在本文中使用的工具POIROT采用了近似的图形匹配，使用具有不相干或缺失边缘的图形[59]，并且对一定程度的假阳性和假阴性边缘具有鲁棒性。

A.对公共CTI报告的评价。

在第一组实验中，我们使用公开的CTI报告来评估EXTRACTOR。为了进行比较，我们选择了POIROT[59]的作者所选择的相同报告。这个实验使我们能够：1）比较EXTRACTOR生成的图形和POIROT作者手动生成的图形；2）使用POIROT使用EXTRACTOR自动生成的图形进行威胁猎取，看是否成功检测到攻击。在这个实验中，审计日志包含良性活动产生的事件和在受控和隔离的环境中执行恶意软件实例和CTI报告中描述的相同攻击活动产生的事件，如POIROT[59]的作者所述。附录中的表XIV提供了关于每个恶意软件样本的额外细节。

表三表示EXTRACTOR图与人工图的特点，以及这些公共CTI报告中的威胁检测结果。第一列显示了恶意软件的名称，CTI报告的参考，以及提取的出处图的参考。接下来的四列显示了POIROT作者手动绘制的图形和EXTRACTOR自动生成的图形的节点（V（G））和边（E（G））的数量。可以看出，EXTRACTOR提取的节点和边的数量与手工绘制的相当。节点和边的数量不同的主要原因是：1）手工图中使用了通配符（例如，手工图中使用了C=\*. tmp，而不是[CWD]\. tmp和C:\Extracted\.tmp），2）被EXTRACTOR选中但在手工图中没有出现的节点和边（例如，2: exec - 10: exec），因为人类已经把这些细节抽象出来了。最后，第六列和第七列显示了威胁猎取的结果，这是通过在适当的良性活动中进行这些恶意软件攻击，并收集审计记录得到的。一个近似匹配算法[59]被用来匹配EXTRACTOR生成的图，该图是由被攻击系统的审计日志生成的更大的出处图。在所有情况下，我们的检测分数都超过了检测阈值（t \_ 0:3），攻击被成功检测到。总之，通过这个实验，我们可以得出结论，EXTRACTOR生成的图与人类生成的图在威胁检测中一样有用。

虽然使用EXTRACTOR的图形进行威胁检测的结果（表三）证实了EXTRACTOR捕捉相关攻击行为的能力，但为了进一步评估EXTRACTOR的性能，衡量其捕捉所有相关攻击行为的能力，我们报告了精度、召回率和F1得分（表四）。在这个评估中，我们使用报告本身作为基础事实，并检查图中捕获的活动是否在报告中存在。我们不使用POIROT中专家生成的图作为基础事实，因为这些图中有许多包含代表进程集的通配符节点。表四显示了EXTRACTOR的性能。如表四所示，EXTRACTOR成功地从报告中捕获了攻击行为（平均F-1得分93%）。然而，正如预期，由于语言的复杂性，EXTRACTOR产生了少量的假阳性和假阴性。有时这是由于倒置的边缘造成的，但更多的时候是由于EXTRACTOR没有完全解决一些歧义或没有检测到文本中的一些实体。例如，在Uroburos对credprov.tlb、load、explorer.exe和\*、fork/exec、winview.ocx是虚假的节点和边（见附录中的图10-(e)）。然而，这些并不影响威胁检测（表三）。

攻击描述。图7显示了EXTRACTOR从[80]中生成的图，恶意软件修改了几个注册表组件并写入了几个文件。图中分为左右两个子图，反映了报告的结构，它描述了恶意软件使用各种进程（authorization.exe，\*）进行的操作。另外，附录中的图10展示了从公共CTI报告中生成的其他图表，在[59]中讨论过，除了OceanLotus[31]，其中的攻击行为是用图而不是自然语言描述。

图10-(c)是使用EXTRACTOR从报告[61]中生成的图。该图展示了各种系统调用的执行情况与特定的系统实体和星号进程（矩形内的\*）。该图显示了几个重要的攻击者活动以及它们之间的联系。图中的不连贯性是由于作者的写作风格，即在不同的句子中用非常不同的名称来指代同一个实体。例如，两个节点\*和8aba4b5184072f2a50cbc5ecfe326701代表同一个实体，但它们在图中是分开的。

附录中的图10-(d)显示了使用EXTRACTOR从报告[7]中生成的图。由于文本的复杂性，EXTRACTOR产生了三个假阳性边16.exec、17.exec和18.exec。附录中的图10-(a)、10-(b)和10-(e)分别展示了Carbanak、HawkEye和Uroburos图。

在所有情况下，POIROT都能够检测到攻击，即使在存在假阳性边缘的情况下。

B.对DARPA透明计算数据集的评估。在这个实验中，我们利用DARPA透明计算活动数据集，从攻击的自然语言描述中自动生成攻击行为图。在这些活动中，红队对蓝队防守的基础设施进行了攻击。这些攻击是在四个系统上进行的，包括一个客户端、一个邮件服务器、一个网络服务器和一个SSH服务器，持续时间为几周。由EXTRACTOR处理的攻击的文本描述是由红队成员写的，作为演习的地面真相发布的一部分。这些报告比公开的CTI报告中的报告更短、更简明。除了文字描述外，它们还包含红队成员产生的攻击的图形表示。由EXTRACTOR生成的图形与这些图形进行了比较，作为地面真相。

表五描述了这个实验的结果。对于每一种攻击（在第一栏中命名），它显示了报告的大小（在第二栏中）以及攻击者和EXTRACTOR产生的手工图的大小。由于CTI报告的大小较短，而且简明扼要，所以人工图和EXTRACTOR的图之间的差异很小。第七列和第八列代表了威胁猎取的结果，其中POIROT[59]被用来检测EXTRACTOR生成的图在审计记录中产生的出处图，其中包括攻击和良性活动。在所有情况下，我们的检测分数都超过了检测阈值（t \_ 0:3），攻击被成功检测。

表六显示了EXTRACTOR的性能和它在DARPA报告上捕获所有相关攻击行为的能力。结果显示，与公共CTI报告相比，EXTRACTOR在DARPA CTI报告上的性能有所提高（表四）。这是由于DARPA报告的简单性，这导致产生较少的假阳性。与CTI报告类似，大多数假阴性是由于EXTRACTOR模型不能从句子中驱动关系。

为了进一步研究产生错误检测信号的可能性，我们在DARPA TC项目的审计日志的良性数据集上运行POIROT，并在输入中提供由EXTRACTOR提取的（攻击）图。该数据集包括来自不同操作系统的12GB良性审计日志，包括Windows、Linux、FreeBSD。我们使用POIROT进行的威胁检测没有出现错误信号。这个实验明确表明，EXTRACTOR的图足够简洁，不会在良性环境中引发错误的检测信号。

C.大规模的实验

为了评估我们的方法的可扩展性及其对其他写作方式的准确性，我们用EXTRACTOR处理了大量来自两个主要CTI来源的非结构化CTI报告，即微软安全情报[8]和TrendMicro威胁百科[18]。

这种评估的主要挑战是缺乏基础真相。虽然在前两个案例中，有图表可以比较，但这些CTI来源并没有提供这样的图表。然而，报告本身指出了一个克服这种挑战的方法，接下来会介绍。

这些来源的报告通常包含几个部分，包括威胁摘要、技术描述和解决方案，其中描述了攻击的概述、技术攻击的细节以及消除攻击所需的步骤。虽然第一部分提供了一些一般信息，如感染率和风险以及威胁的严重程度，但第二和第三部分提供了关于攻击和如何扭转其影响的宝贵技术见解。通常情况下，这些报告的最后两部分是相似的，但又是相互对立的。换句话说，攻击描述部分描述了破坏系统的步骤，包括创建和执行的文件、被破坏的进程等，而解决方案部分则详细说明了删除攻击的工件所需的步骤，即创建的相同文件和被破坏的进程。例如，解决方案中的一句话：从HKEY\CURRENT\USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run中删除<systemfolder>\sysformat.exe，概述了撤销威胁行为所需的行动，描述为在注册表键：<system folder>\sysformat.在攻击描述部分的注册表键：HKEY\CURRENT\USER\Software\Microsoft\ Windows\CurrentVersion\Run中添加注册表值：sysformat，数据为：<system folder>\sysformat.exe。作为另一个例子，解决方案部分的句子检查与10.13.13.1的开放连接映射到攻击描述部分的连接到命令和控制服务器10.13.13.1。在这一点上，我们注意到，在所有的报告中，解决方案部分并不具有这些特征。事实上，它往往是关于如何下载和执行一个补丁文件的说明，它完成了所有的清理和修补工作。然而，自动区分包含详细清理步骤的较大的解决方案部分和指示运行补丁文件的较小的解决方案部分是比较容易的，并过滤掉后者。[3]和[4]中的报告就是这种CTI报告的例子，而更多的例子可以在[18]和[8]中找到。

为了评估EXTRACTOR，在这个实验中，对于每个同时包含攻击描述和详细解决方案的报告，我们通过从报告的其他部分省略其他部分来建立与每个部分相关的来源图。然后，我们反转从解决方案中获得的图，并计算其与从攻击描述部分获得的图的相似度。为了衡量这两个图之间的相似性，我们使用最大共同子图（MCS）[75]，这是一个衡量较小的图在较大的图内的包含程度的指标。(我们在这个大规模的评估中使用这个指标，因为它比[59]中使用的对齐概念要简单得多）。)

表七显示了这个实验的结果。在这个表中，第二列显示了被评估报告的数量。第三列和第四列描述了最小和最大报告的长度。第三列和第四列显示文本总结前后的平均句子数。平均攻击描述和平均解决方案列显示了从技术细节和清除部分建立的节点和边的平均数量，最后，最后一列衡量了两个图之间的相似度。可以看出，提取的图形之间的平均相似度衡量，微软等于0.91，TrendMicro等于0.85。这意味着EXTRACTOR在大多数情况下都能正确地从文本中提取图形。

为了进一步提高这一实验的全面性，我们还进行了人工 "抽查"，即对随机选择的50份报告进行人工评估。然后在会议上对个别评估进行讨论并达成一致。假阴性和假阳性是由于未解决的复杂性而产生的，考虑到报告的复杂性，假阴性和假阳性仍然是最小的，并且与我们之前的结果一致。我们的精确度、召回率和F1分数为0.88、0.93、0.90。

D.细粒度的性能评估

在本小节中，我们对我们方法的一些最重要的步骤进行了更精细的评估。

文本总结。以前从CTI报告中自动提取知识的工作[52], [45]使用主题分类（TC）来辨别CTI报告中的广告等不相关内容。在EXTRACTOR中，我们使用了一种不同的文本总结方法，旨在实现精细化的总结。图8显示了这两种方法并列的情况。为了进行主题分类，与之前的方法[52]类似，我们对1500个CTI技术句子与1000个广告句子和关于作者的细节进行了支持向量机（SVM）分类器。我们使用10倍交叉验证对该模型进行了评估，取得了97%的精确度和99%的召回率。从图中可以看出，与主题分类相比，我们的文本总结方法大大减少了文本的大小。

最后，表八显示了使用最先进的方法去除句子Verbosity的结果，证明了BERT优于其他流行模型。

消减研究。为了证明每个EXTRACTOR模块对最终图形的贡献，我们进行了一项消融研究，以衡量在没有该模块的情况下产生的图形与基线（所有模块都处于激活状态）的相似性。

表九显示了我们消融研究的结果。每一列代表了EXTRACTOR在没有该组件(w/o)的情况下生成的图形的MCS相似度得分结果。本研究的基线是EXTRACTOR的整体性能，设定为1。每一列显示了在整体方法中省略任何特定组件时的性能损失。该表还显示了写作风格的多样性，以及每一种技术都很重要的事实（因为没有一列是全部为1的），显示了在各种报告场景中结合这些不同技术的必要性。我们的结果表明，EXTRACTOR中的所有不同模块都成功地在不同程度上对简明图的生成做出了贡献，这取决于文本风格。特别是，它们使EXTRACTOR能够成功地处理各种各样的写作风格。

此外，我们考察了规范化、解析和总结等步骤的影响，不执行这些步骤，而是从原始报告中建立图。图9显示，这种图的大小为数百条边，而通过完整的模块链得到的图的大小则小得多。

表十显示了第二节讨论的挑战在两个主要威胁报告网站[8]和[18]中的普遍性。每一列代表每个模块被调用的次数（我们避免添加标记器，因为它已经反映在表七中）。第二列显示了被分析的报告总数。第三列和第四列分别概述了同质化实例的平均数量和被动到主动的对话。最后，决议栏列出了ESR、PR和ER的结果。

表十一是SRL在再训练前后的表现。最后，表十二显示了SEE模块在挑选正确和有意义的节点方面的表现。为了评估我们的SEE的完整性，我们在[89]使用的1000份公共报告上运行了我们的SEE模块，并将SSE的结果与他们的结果作为基线进行比较。表十二列出了这个评估的结果。

1. 讨论和限制

假阳性和假阴性。如表四和表六所示，在EXTRACTOR生成的图中存在不相干的节点或边。正如第二节和第三节所讨论的，我们注意到，由于处理自然语言复杂性的一般问题，在提取中损失一些精度是不可避免的。因此，这些不相干的节点和边（或缺失的节点和边）在EXTRACTOR生成的图中对于威胁猎取问题的 "大局 "有什么意义？为了回答这个问题，我们注意到我们选择了一种使用近似匹配的威胁猎取方法[59]，这有利于我们成功识别威胁，尽管有不相干的信息。事实上，我们可以继续论证，这种近似技术是一般威胁猎取问题所需要的，因为在基于活动的出处图中，由于内在因素（如影响威胁二进制执行的随机性或时间等非确定性因素），威胁报告中描述的所有活动很可能不会在主机中表现出来。因此，如第五节所示，威胁猎取所固有的近似能够在NLP固有的小的精度损失下工作。

限制和未来的工作。EXTRACTOR的性能在行动描述跨越几个句子或一个段落的情况下可能会减弱，在这种情况下，根据威胁报告的撰写方式，主语或宾语的解析可能会面临挑战。关于这种挑战的例子，见第V-A节中关于图10-(c)的讨论。如果以备用报告的形式提供额外的信息来解决这些挑战，EXTRACTOR可以处理这个问题。另一种解决方式是在EXTRACTOR中促进机制，积极与人类操作者合作，解决这些实体。

我们基于图的方法的另一个限制是，它不适用于涉及计时、侧信道推理等的攻击。描述攻击行为的图是以审计日志为模型的，而审计日志所捕获的信息的粒度并不能使这些攻击被检测出来。然而，这种限制与其他涉及基于来源图的检测或威胁猎取的方法是共同的。同样，我们的方法只提取自然语言描述的攻击行为，不能解决以其他形式（如数字和图表）表示的行为。

EXTRACTOR的各种模块使用字典来提高整体性能。虽然我们在选择适当的词语时尽量做到完整，但可能会有一些报道，人们使用的词语不在词典中。因此，可能还有进一步改进词典的空间。使用命名实体识别（NER）系统也可以加强将名词短语翻译成统一的系统表示的方法。此外，未来的工作可以扩展EXTRACTOR，从非结构化的漏洞报告中推断出图形。这些图可以进一步用于检测应用程序中可能存在的漏洞。

1. 相关的工作

证据图分析。一些研究项目利用系统审计日志进行攻击重建和取证分析，以及威胁狩猎[42]、[36]、[37]、[59]、[53]、[73]。Hercule [69]通过比较从不同来源收集的日志，重建攻击阶段。Bilge等人[23]利用NetFlow日志来检测僵尸网络的C&C通道。Oprea等人[66]使用网络代理和DNS日志来识别企业网络中的感染情况。

King等人[49], [48]介绍了从内核审计日志中构建来源图的做法。一些研究在攻击检测和取证中使用了来源图[43]、[51]、[85]、[38]。Hossain等人[43]、Xu等人[85]和LogGC[51]提出了减少图的大小，同时保持取证分析所需的准确性的还原技术。EXTRACTOR可以作为这些方法的伴侣，提供一个清晰的攻击图片。

NLP和威胁信息提取。一些开放标准，如STIX[15]、MISP[72]和OpenIOC[11]，已经被提出来，以可互操作的方式交换关于IOC的知识。然而，与我们的方法不同，这些标准更侧重于交换IOC，而不是描述这些IOC是如何连接的以及攻击的行为方式（见[12]的例子）。使用威胁交换标准而不是仅仅依靠高级威胁数据的交换的公司可以从野外公开的知识中获益。另一方面，这些威胁交换标准的使用是有限的，因为各公司对分享其数据的兴趣不一样。此外，交换的数据不包含技术细节，如受影响的注册表、文件路径和应用程序名称，因为它们可能是公司私人信息的隐私泄漏。正如在许多情况下，组织的内部政策阻止与外部实体共享数据[77]，[26]。

VirusTotal图表[21]也与我们的工作不同，因为它只代表了攻击的高层视图，主要包括可能的威胁场景中涉及的哈希值、IP和域。另外，与我们的方法不同，VirusTotal的报告是基于对样本恶意软件的分析产生的，而EXTRACTOR通过访问公开的报告（可以包括VirusTotal），允许利用公共CTI，将原始报告转化为可操作的知识。

iACE[52]提出了一种图形挖掘方法，从安全文章中提取IOCs。ChainSmith[89]使用NLP从安全文章中提取IOCs，并进一步将其归类为活动阶段。TTPDrill[45]提出了一个本体论，有助于理解网络威胁的特征和规格。它使用NLP和信息检索（IR）来从报告中提取威胁行动。[46]的工作使用DP规则从报告中创建TTP链。与这些方法不同，EXTRACTOR专注于提取攻击行为，并以出处图的形式捕捉系统级的因果关系。

SemFuzz[86]通过从漏洞报告中提取的信息进行模糊处理。Feng等人[32]使用NLP从非结构化的漏洞报告中生成网络签名。他们将这些签名用于入侵检测和防火墙系统。Dong等人[29]使用命名实体识别和关系提取来提取软件名称和版本以及主要漏洞数据库之间的报告不一致。尽管在某种程度上是相关的，但EXTRACTOR的目标和技术与这些工作有本质上的不同。

Featuresmith[88]从安全文献中生成了一个用于检测安卓恶意软件的特征集。相比之下，EXTRACTOR旨在建立一个代表攻击的实际行为的出处图。Privee[90]利用机器学习来检索网络策略。[74]和[68]的工作使用NLP将应用描述与权限联系起来。[63]和[44]的工作确定了Android应用中用户的敏感输入。EKLAVYA[25]使用NLP从二进制代码中恢复函数签名。

1. 结论

EXTRACTOR从以自然语言编写的CTI报告中自动构建出处图。我们使用各种威胁报告和真实世界的攻击场景来评估EXTRACTOR。EXTRACTOR成功地提取了与安全专家手动绘制的图相匹配的图，这些图被成功用于威胁检测。

1. 鸣谢

我们要感谢审稿人对稿件提出的有益的审查意见和建议。特别要感谢我们的匿名牧者，他提供了许多有用的改进建议。这项工作得到了DARPA的SPAWAR（N6600118C4035）和NSF（CNS- 1918542）的支持。所表达的观点、意见和/或发现是作者的，不应该被解释为代表美国政府的官方观点或政策。