

# 基于 ATT&CK 的 APT 攻击语义规则构建

潘亚峰<sup>1</sup>, 周天阳<sup>1,2</sup>, 朱俊虎<sup>1,2</sup>, 曾子懿<sup>1,2</sup>

<sup>1</sup>信息工程大学 数学工程与先进计算国家重点实验室 郑州 中国 450001

<sup>2</sup>国家数字交换系统工程技术研究中心 郑州 中国 450001

**摘要** 从自然语言描述文本中提取网络攻击知识存在语义鸿沟, 导致 TTPs 威胁情报自动化利用低。为提高威胁情报自动分析效率, 设计并实现了基于 ATT&CK 的 APT 攻击语义规则。首先, 构建带标签的有向图语义规则模型, 对自然语言文本描述的攻击技术进行知识化描述; 其次, 定义语义规则, 阐释网络实体属性及其逻辑运算关系的形式化描述方法; 最后, 利用关键词组识别、知识抽取等自然语言处理技术, 从攻击技术文本中抽取形成 123 个 APT 攻击语义规则, 涵盖 ATT&CK 的 115 项技术和 12 种战术。利用模拟场景采集的 APT 攻击日志数据, 对语义规则进行验证, 实验结果表明, 语义规则检出率达到 93.1%, 并具备一定的攻击上下文信息还原能力, 可有效支撑威胁检测分析。

**关键词** 语义规则; APT 攻击; ATT&CK; 威胁情报; 自然语言处理  
中图分类号 TP393.0 DOI 号 10.19363/J.cnki.cn10-1380/tn.2021.05.05

## Construction of APT Attack Semantic Rules Based on ATT&CK

PAN Yafeng<sup>1</sup>, ZHOU Tianyang<sup>1,2</sup>, ZHU Junhu<sup>1,2</sup>, ZENG Ziyi<sup>1,2</sup>

<sup>1</sup> State Key Laboratory of Mathematical Engineering and Advanced Computing, Information & Engineering University, Zhengzhou 450001, China

<sup>2</sup> National Engineering Technology Research Center of the National Digital Switching System, Zhengzhou 450001, China

**Abstract** Aiming at the problem of semantic gap in the extraction of cyber attack knowledge from natural language description, which leads to low automatic utilization of tactics, techniques, and procedures (TTPs) threat intelligence, this paper designs and implements APT attacks semantic rules based on Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK). First, we construct a labeled directed graph semantic rule model to describe the attack technology of natural language description, and then define semantic rules to explain the formal description method of network entity attributes and their logical operation relation. Finally, using natural language processing techniques such as phrase recognition and knowledge extraction, 123 APT attack semantic rules were extracted from the attack technical text, covering 115 techniques and 12 tactics of ATT&CK. After experimenting on the APT attack audit data collected in the simulation scenario to verify the semantic rules, experimental results show that the detection rate of semantic rules reaches 93.1%, and has a certain ability to reconstruct the context information of the attack behavior, which can effectively contribute to threat detection and analysis.

**Key words** semantic rules; APT attack; ATT&CK; threat intelligence; natural language processing

## 1 引言

随着高级持续威胁(Advanced and Persistent Threat, APT)不断出现, 网络安全面临严峻挑战, 在此背景下, 以威胁情报驱动的新型网络安全防御机制应运而生。Gartner 公司将威胁情报定义为一种基于证据的知识, 为威胁响应提供包括相关场景、威胁指标、含义和可行建议等决策依据<sup>[1]</sup>。Bianco 提出金

字塔模型<sup>[2]</sup>, 将威胁情报数据划分为 6 个层次, 自下而上依次为哈希、IP 地址、域名、网络或主机特征、攻击工具、TTPs, 在模型中, 威胁情报收集难度逐层增加, 网络防御能力逐层增强。

威胁情报发展迅速, 但是对威胁情报数据的利用却严重不足。SANS 发布的 2020 年威胁情报调查报告<sup>[3]</sup>显示, 在对威胁情报的利用中, IOC(Indicators of Compromise)威胁情报底层数据始终占据主导地

通讯作者: 周天阳, 博士, 副教授, Email: aipteamzhouty@aliyun.com

本课题得到国家自然科学基金(No. 61502528)资助。

收稿日期: 2020-07-04; 修改日期: 2020-09-10; 定稿日期: 2021-03-05

位, 而对价值更高的 TTPs 威胁情报信息利用较少。IOC 是描述特定攻击活动中主机或网络行为特征的基本指示数据<sup>[4]</sup>, 可用于识别某种攻击, 但是由于缺乏对攻击技术等行为规律的关注, 攻击者很容易通过文件混淆、更换 IP 地址、域名等简单方式规避检测。TTPs 是直接反映 APT 攻击者行为的高层语义信息, 描述了攻击战术、技术和过程等行为规律本质。防御者可利用 TTPs 检测和描述 APT 攻击, 识别攻击者。攻击者对抗检测, 需要开发新技术、新工具、重新部署 IP、域名等基础攻击资源, 极大增加了攻击成本和难度。基于人工的 TTPs 信息分析与利用效率低、难以适应攻击快速发展, 因此急需加强对 TTPs 威胁情报的自动化分析与利用研究。但是, 当前 TTPs 主要以自然语言文本形式描述, 在文本的自动化处理分析中存在攻击语义鸿沟问题<sup>[5]</sup>。

语义规则可以消除上层 TTPs 和底层数据之间的语义鸿沟, 可用于从审计数据中自动化分析网络威胁。开展基于 ATT&CK 攻击技术的语义规则提取研究, 可以促进对 TTPs 威胁情报的自动化利用。本文首先介绍了研究背景, 其次, 构建语义规则模型, 描述攻击行为规律, 最后借鉴自然语言处理领域的知识抽取方法, 将 ATT&CK 的技术定义文本中的语义知识转化为可用于数据匹配的语义规则。实验证明, 构建的语义规则可从审计数据中检测网络攻击并还原网络环境、攻击目标、攻击过程等攻击上下文信息。

## 2 研究背景

### 2.1 威胁情报数据利用

学术界对威胁情报的研究主要关注 IOC 的提取和利用。文献[6]和文献[7]专注于自动化生成 IOC 威胁情报, 用于威胁检测分析。但是, 文献[8]和文献[9]研究发现, IOC 生命周期较短, 绝大部分的 IOC 只出现一次, 用 IOC 检测威胁效率较低。另外, IOC 缺乏攻击战术、技术等高层语义信息, 无法分析还原网络攻击上下文信息。文献[10]尝试利用 IOC 分析开源威胁情报对攻击演变的影响, 但是此分析只能还原出攻击者 IP、域名、攻击漏洞的变化, 依旧停留在较低的语义层次, 缺乏对攻击技术和行为规律的分析。文献[11]将 IOC 威胁情报构建成查询图(Query Graph), 通过消除攻击过程中的签名特征, 刻画特定攻击行为模式来检测攻击, 但是该方法只适用于特定的攻击活动, 未深入攻击技术行为规律本质, 当攻击者改变攻击工具、漏洞就会导致攻击行为和查询图产生较大差异, 由此产生漏报。

TTPs 威胁情报中包含详细的攻击上下文信息,

由于 TTPs 威胁情报一般都是报告形式的自然语言文本, 虽然包含丰富的攻击语义, 但是很难用于自动化威胁分析。要将威胁情报用于自动化威胁分析, 建立威胁情报标准是重要前提<sup>[9]</sup>。以 ATT&CK<sup>[12]</sup>为代表的攻击战术(Tactics)、技术(Techniques)知识框架, 是对高层攻击语义知识的标准化描述, 为实现自动化威胁分析提供基础。在 Mitre 公司构建 ATT&CK 的同时, 也提出了一套规范的威胁分析流程方法<sup>[13]</sup>, 但是主要依靠人工分析。人工分析虽然可以消除 TTPs 威胁情报和底层数据间的语义鸿沟, 但是分析效率是主要制约因素。除 ATT&CK 外, 互联网中还广泛存在着其他 TTPs 威胁情报, 网络中不断发生新的攻击事件也会产生新的 TTPs 威胁情报, 如果单纯依靠人工分析, 很难及时跟踪掌握最新攻击技术, 导致安全防御严重滞后网络攻击技术演进。因此急需加强 TTPs 威胁情报自动化利用研究。

将 TTPs 用于自动化威胁分析已经有若干工作。文献[14]根据 ATT&CK 技术语义信息, 针对收集恶意代码相关技术在实施过程中的指纹特征(例如特定的系统函数调用), 以此作为语义规则, 可自动化分析恶意代码行为的攻击技术语义; 文献[5]则对同一个攻击阶段下的攻击技术进一步总结归纳, 以函数形式描述 IP、文件、进程等在攻击实施中的特征, 以此来构建语义规则, 用来消除底层数据到上层攻击过程之间的语义差异。上述方法都是采用攻击指纹特征来构建特征规则, 未对攻击行为进行建模, 实质上仍然是传统的指纹特征匹配, 缺乏结合上下文信息的高层语义规则提取。从威胁情报的角度, 此类规则将金字塔模型上层 TTPs 威胁情报映射到了中低层次的威胁情报, 降低了威胁情报的价值。

### 2.2 ATT&CK 框架

ATT&CK 是在大量已知 APT 攻击事件基础上构建出来的, 从战术和技术两个维度对 APT 攻击进行总结归纳。战术是攻击的目的, 技术是实现战术所采取的具体方法, 在对技术的定义中, 包含了丰富的攻击上下文语义信息。例如, 图 1 展示了“Exploit Public-Facing Application”技术<sup>[15]</sup>的部分定义文本, 其中就完整地介绍了该技术的实施过程, 即使用软件、数据或命令, 利用系统或程序的脆弱点, 对目标系统实施攻击, 从而引发一些恶意行为, 随后对其涉及的漏洞、目标应用服务等又进行了解释说明。由此可见, 技术的定义文本中包含了该技术在实施过程的普遍规律, 既包含抽象的攻击流程, 又包含了实施过程中涉及的网络环境(例如 Internet-facing)、关键工具(例如 weakness)和目标服务(例如 SQL)实

例。但是, 此类以自然语言文本描述的语义知识只能通过分析人员学习理解后, 才能以知识经验的形式应用于威胁分析, 效率较低。因此, 需要对技术定义文本中的语义知识的形式化描述, 促进语义知识在威胁分析中的自动化应用。

The use of software, data, or commands to take advantage of a weakness in an Internet-facing computer system or program in order to cause unintended or unanticipated behavior. The weakness in the system can be a bug, a glitch, or a design vulnerability. These applications are often websites, but can include databases (like SQL), standard services (like SMB or SSH), and any other applications with Internet accessible open sockets, such as web servers and related services.

图 1 ATT&CK Exploit Public-Facing Application 技术部分定义文本<sup>[15]</sup>

Figure 1 Part definition text of ATT&CK Exploit Public-Facing Application technique<sup>[15]</sup>

3 语义规则模型

3.1 模型定义

**定义 1.**语义规则模型. 是一张带标签的有向图  $G=(V, E, L, A)$ 。其中  $V$  是图中的顶点, 表示攻击过程中的网络实体集合,  $E$  是图中边的集合, 描述网络实体之间的关系,  $L$  是网络实体数据类型标签集合,  $A$  是网络实体和标签的映射集合。

**定义 2.**网络实体. 网络中存在的客观事物, 包括概念和对象。概念是对具有相同特点或属性事物的抽象, 用符号  $\Psi$  表示, 对象是概念的具体实例, 用符号  $\Theta$  表示。

**定义 3.**网络实体关系. 表示网络实体之间的相互作用和联系, 用符号  $R$  表示。包括操作关系、从属关系、并列关系三种类型。




- 操作关系。表示网络实体对另一个实体的操作行为, 例如 software 对 weakness 的 execute (take advantage of)关系。操作关系表示攻击过程中的动作, 结合动作主体、客体就可以刻画攻击步骤。
- 从属关系。表示对网络实体划分, 是对实体的进一步具化, 用 isa 表示。例如 bug 是 weakness 的一种具体类别, 可以用 isa 关系表示, 其中 weakness 被称为 bug 的父实体; SQL 是 database 的对象也可以用 isa 表示, 其中, database 是 SQL 的父实体。
- 并列关系。表示对等的两个网络实体, 例如 SMB 和 SSH 的 or 关系, 并列关系的网络实体具有相同的父实体。并列关系不显性表示, 而是通过同一个父实体的 isa 关系表示。

表 1 语义规则模型关键元素

Table 1 Key elements of semantic rule model			
名称	符号	定义	描述
网络实体	$V=\{v_1, \dots\}$	$v \in \Psi \cup \Theta$	包括概念(例如 service)和对象(例如 SMB)
边	$E=\{e_1, \dots\}$	$e=\langle v_i, r, v_j \rangle$	网络实体之间操作关系或从属关系, $r \in R$
标签	$L$		网络实体数据类型的符号标识
标签映射	$A$	$V \leftarrow L$	网络实体和标签的映射

在模型中, 为了清晰地描述网络实体之间的依赖关系, 利用起源图<sup>[16-17]</sup>的数据表示方法, 对网络实体标签以及操作关系进一步规范, 如表 2 所示。将网络实体分为进程、文件、套接字三类, 分别对应 Process、File、Socket 三个标签, 用不同形状表示。在网络和主机中, 进程是关键的行为主体, 网络数据收发、文件读写执行、进程的启动和停止都是由进程发起。本文定义了以进程为主体, 进程、文件、套接字为客体的网络实体操作关系, 操作关系的边用蓝色线条表示, 用黑色边表示网络实体的从属关系。

表 2 网络实体标签及其关系<sup>[17]</sup>

Table 2 Network entity labels and relation <sup>[17]</sup>			
主体标签	客体标签	形状	关系
	Process		start, end
Process	File		file_read, file_write, execute
	Socket		ip_read, ip_write

语义规则图模型将自然语言描述的技术语义信息表示成有向图, 是对攻击技术的知识化表示。图 2 是“Exploit Public-Facing Application”技术根据模型生成的语义规则图, 在图中  $P1 \rightarrow F1 \rightarrow S1 \rightarrow P2 \rightarrow F2$  描述了技术的实施过程, 网络实体通过操作关系连接起来。在实际远程漏洞利用过程中, 常用的方法是利用漏洞攻击工具, 加载特定漏洞的攻击脚本, 通过网络向目标发送攻击载荷并在目标系统中执行恶意代码。  $F1 \rightarrow S1$  边表示漏洞利用程序通过网络向目标发送攻击载荷, 在  $F1$  和  $S1$  之间隐含了一个进程实体, 由于该进程并不影响整个攻击过程的完整性, 如果增加它反而会提升规则匹配的复杂度, 所以在此将该进程实体省略。当前数据仅限于 ATT&CK 攻击定义文本, 对抽取的网络实体的子实体无法做到完全列举, 因此, 如果进一步扩充文本数据(例如 APT 分析报告), 对网络实体的划分会更加具体, 对攻击技术的语义描述更加精确。

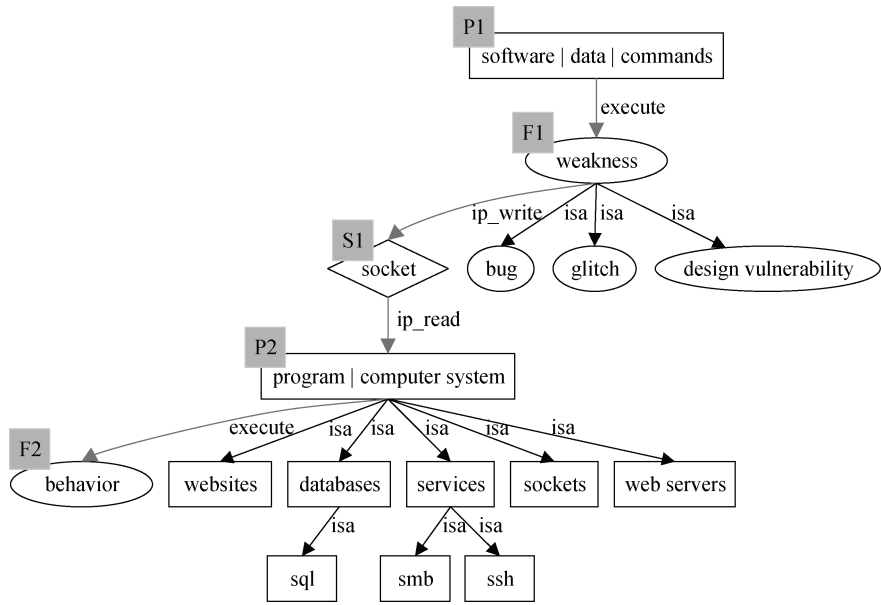


图 2 Exploit Public-Facing Application 语义规则图

Figure 2 Semantic rule graph of ATT&CK Exploit Public-Facing Application technique

3.2 规则定义

**定义 4.**语义规则. 是描述攻击行为模式规律的法则, 针对网络中可能出现的威胁, 从设备审计日志数据中还原攻击技术语义和上下文信息, 为威胁分析、评估、响应提供支撑。表示为  $R=H\cup Z$ , 其中  $H$  表示实体匹配规则集合,  $Z$  表示关系匹配规则集合。

**定义 5.**网络实体属性. 是网络实体用于相互区分的性质、特征、类型, 包括通用属性和特有属性。

语义规则是一个规则集合, 包括实体匹配规则和关系匹配规则两类。两类规则都是对网络实体属性的逻辑运算, 用于对数据中的网络实体及其关系的进行匹配, 还原攻击技术语义。在数据中, 实体之间通过属性进行区分, 通用属性是所有网络实体都具备的属性, 包括 *label* 和 *children*。*label* 属性实现了标签和网络实体的映射, 属性值可以继承; *children* 属性是网络实体的子实体的列表, 对象类型网络实体的 *children* 属性为空。例如, 图 2 中网络实体 P2 的 *children* 属性为  $\{[websites, databases, services, sockets, web servers], [sql, smb, ssh]\}$ 。进程、文件、套接字三类网络实体都有各自的属性, 如表 3 所示。

表 3 网络实体属性列表

Table 3 Attributes of network entities

网络实体	属性	说明
进程	<i>name, pid, ppid, gid, uid, cwd, cmd_line, local_addr</i>	<i>local_addr</i> 是进程绑定的 IP、端口
文件	<i>path, content</i>	路径和读写的内容
套接字	<i>local_addr, remote_addr, payload</i>	IP、端口、数据载荷

属性之间比较运算包括大于(>)、小于(<)、等于(=)、不等( $\neq$ )四种, 比较对象可以是两个实体属性, 也可以是单个实体属性和特定数值。例如, 两个进程实体  $p_x, p_y$ , 通过对两个进程属性进行比较运算, 即  $p_x.pid=p_y.ppid$ , 可以判断两进程之间是否是父子关系, 通过对单个进程属性和特定数值进行比较运算, 即  $p_x.local\_addr.port>10000$ , 可以判断进程是否绑定了非通用端口。属性之间的比较运算关系, 一方面可用于表示关系匹配规则, 另一方面, 可以增强语义规则的扩展性, 便于安全人员根据实际网络环境以及安全经验知识对语义规则进行定制, 使得语义规则更加适用于不同网络环境。

语义规则是面向受害端的, 由于攻击发起端的行为无法通过审计捕获, 所以在将语义规则图转化生成语义规则后, 会再经过人工确认, 将攻击发起端的行为特征转化到防御侧。表 4 是一个技术语义规则示例, 实体匹配规则是验证单个网络实体属性值是否满足一定规则条件, 关系匹配规则是对主客实体的操作关系进行匹配。如果数据既满足实体匹配规则又满足关系匹配规则, 则匹配成功。通过对数据进行匹配, 语义规则不仅可以自动发现数据中的攻击威胁, 还能还原攻击上下文信息, 辅助分析人员进行威胁分析。

4 语义规则构建

4.1 技术框架

ATT&CK 中技术是以文本形式描述的, 因此需



表 4 Exploit Public-Facing Application 技术语义规则  
Table 4 Semantic rule of ATT&CK Exploit Public-Facing Application technique

匹配项目	名称	规则
实体 匹配 规则	$s$	套接字 $s.label==Socket \wedge$ $s.addr\_info.ip \in \{Public\_IP\}$
	$p_1$	进程 1 $p_1.label==Process \wedge p_1 \in$ $P2.children$
	$f$	文件 $f.label==File \wedge f \in F2.children$
	$p_2$	进程 2 $p_2.label==Process \wedge$ $p_2.cmd\_line \in F1.children$
关系	$ip\_read(p_1, s)$	网络数据读取 $p_1.local\_addr==s.remote\_addr$
匹配	$execute(p_1, f)$	文件执行 $p_1.cmd\_line==f.path$
规则	$start(p_1, p_2)$	进程启动 $p_1.pid==p_2.ppid$

要从文本中抽取语义知识, 以此构建语义规则。从文本中抽取语义知识是自然语言处理领域的一项重要内容, 一般采用命名实体识别、关系抽取、属性抽取等方法<sup>[18]</sup>。由于本文的目的是构建语义规则, 只需描

述网络实体属性之间的逻辑运算关系, 而不是描述某一个网络实体的具体特性, 因此在进行知识抽取时, 不涉及属性抽取, 这是和自然语言处理领域的最主要区别。由于本文专注于 ATT&CK 的攻击技术定义文本, 文本数量较少, 描述比较规范, 再加上网络实体仅限定在进程、文件、套接字三类, 因此采用基于规则的知识抽取方法, 避免基于机器学习的知识抽取方法对构建语料库的要求。

语义规则构建技术框架如图 3 所示, 框架输入是 ATT&CK 技术定义中的自然语言文本, 输出是语义规则。在方法上, 主要包括文本数据预处理、知识抽取、语义规则构建三个阶段。预处理阶段是识别文本中的关键词组, 再对文本经过语法解析, 得到标注有词性和语法依赖关系的词汇集合; 知识抽取阶段是从预处理后的词汇集合中, 识别关键网络实体以及关系; 规则构建阶段是对网络实体进行标注, 建立网络实体和标签的映射, 构建语义规则图, 生成语义规则。

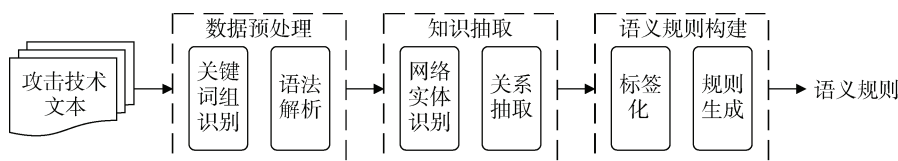


图 3 语义规则构建技术框架

Figure 3 Technology framework of semantic rules construction

## 4.2 数据预处理

数据预处理包括关键词组识别和语法解析。关键词组识别是根据领域词汇特点, 利用词汇之间的相关性, 识别文本中的词组, 避免分词导致的网络实体识别错误。本文采用逐点互信息 (Pointwise Mutual Information, PMI)<sup>[19]</sup>表示词汇之间的相关性。文本可表示为  $\{w_1, w_2, \dots, w_i, \dots, w_n\}$ , 其中  $w_i$  表示文本中的单词, PMI 定义如下:

$$PMI(w_i, w_{i+1}) = \log_2 \frac{p(w_i, w_{i+1})}{p(w_i)p(w_{i+1})} \quad (1)$$

其中,  $(w_i, w_{i+1})$  是两个连续单词组成的词组,  $p(w_i, w_{i+1})$  是词组出现的概率,  $p(w_i)$  是单词  $w_i$  出现的概率。当单词之间的相关性超过一定阈值  $\sigma$ , 则接受该词组。表示为:

$$IF PMI(w_i, w_{i+1}) > \sigma THEN accept (w_i, w_{i+1})$$

筛选出词组后, 对表示操作行为的动词词组进行词典替换, 简化文本的语法结构。例如, 动词词组 “take advantage of”, 其语义和单词 use 相同, 因此, 可以使用 use 进行替换, 生成新的文本。

语法解析是对文本进行分词, 标注单词词性以

及单词之间的依赖关系。本文采用 NLTK<sup>[20]</sup> 和 spaCy<sup>[21]</sup> 工具进行语法解析, 生成带词性标签和依赖关系的词汇集合。表示为:

$$voc = \{v'_1, v'_2, \dots, v'_m \mid v'_i = \langle w_x, tag_x, pos, w_y, tag_y, \rangle\} \quad (2)$$

其中,  $tag$  是单词的词性标签,  $pos$  表示单词  $w_x$  和单词  $w_y$  之间的语法依赖关系<sup>[22]</sup>。

## 4.3 知识抽取

知识抽取包括网络实体识别和关系抽取。通过分析网络实体词汇在文本中的语法特点, 本文构建了一套网络实体词汇语法规则, 用来抽取网络实体, 如表 5 所示。语法规则是根据英文文本的语法表达规范, 利用词汇词性和语法依赖关系构建的, 输入是预处理阶段生成的带词性标签和依赖关系的词汇集合, 输出是网络实体集合。

在网络实体识别过程中, 为了尽可能多的覆盖文本语义信息, 对文本语法规则定义较为宽泛, 但是这也造成了一定程度上的信息冗余。例如 adversary 就不属于语义规则模型定义三类网络实体, 因此仍需对抽取的网络实体进行进一步筛选。筛选过程在规则构建阶段进行。

表 5 知识抽取语法规则

Table 5 Grammar rules for knowledge extraction

语法规则	描述
$Entity \in \{ COMPOUND_{NOUN}, PROP_{NOUN}, NOUN \}, COMPOUND_{NOUN} \in \{ (w_1, \dots, w_n)   \langle w_i, compound \ w_n \rangle, w_i \in \{ NOUN, PROP_{NOUN} \} \}$	实体是名词、专有名词或复合名词, 复合名词是 compound 依赖关系的一组名词或专有名词
$\langle Subject, Predicate, Object \rangle \wedge \langle Subject, Object \in \{ Entity \} \}$	主谓宾结构, 主语和宾语是实体, 谓语动词表示实体之间的关系
$\langle Subject, LinkingVerb, Predicative \rangle \wedge \langle Subject, Predicative \in \{ Entity \} \}$	主系表结构, 主语和表语是实体, 主语是表语的父实体, 实体间是 isa 关系
$\langle Entity\_A \ SCONJ \ Entity\_B \rangle$	实体 A、B 是从属关系, A 是 B 的父实体, 实体间是 isa 关系
$\langle Entity\_A \ CONJ \ Entity\_B \rangle$	实体 A 和实体 B 是并列依赖, 实体 B 可继承实体 A 与 A 父实体的关系

(注: Entity 表示网络实体; COMPOUND<sub>NOUN</sub> 指复合名词; PROP<sub>NOUN</sub> 指专有名词; NOUN 指名词; compound 指单词复合依赖关系; Subject 指句子主语; Predicate 指句子谓语; Objects 指句子宾语; LinkingVerb 指句子系动词; Predicative 指句子表语; SCONJ 指单词从属依赖关系; CONJ 指单词并列依赖关系)

文本语法规则中也包含了网络实体关系。例如主谓宾句子就表示主语对谓语的动作, 如果主语、宾语都是网络实体, 那么谓语就可以用来表示实体之间的动作关系。类似的也可以从主系表结构、从属关系、并列依赖的句子中抽取实体之间的从属、并列关系。并列关系的网络实体相互继承父实体。因此, 可通过语法规则抽取网络实体关系。

#### 算法 1. 基于路径搜索的关系抽取算法.

输入: *voc* 词汇集合, *entityList* 网络实体集合, *maxNop* 最大跳数

输出: *result* 网络实体及其关系集合

FUNCTION

SearchRelation(*voc*, *entityList*, *maxNop*)

FOR EACH *entity* IN *entityList* DO

*nop*  $\leftarrow$  0

*p*  $\leftarrow$  NewPath(*entity.word*, *entity.tag*)

*newResult*  $\leftarrow$  Recur(*voc*, *entityList*, *maxNop*, *nop*, *p*)

*result*  $\leftarrow$  *result*  $\cup$  *newResult*

END FOR

RETURN *result*

END FUNCTION

FUNCTION

Recur(*voc*, *entityList*, *maxNop*, *nop*, *p*)

FOR EACH *w<sub>i</sub>*, *tag<sub>i</sub>*, *pos*, *w<sub>j</sub>*, *tag<sub>j</sub>* IN *voc* DO

*w<sub>first</sub>*, *tag<sub>first</sub>*  $\leftarrow$  GetFirstItem(*p*)

*w<sub>last</sub>*, *tag<sub>last</sub>*  $\leftarrow$  GetLastItem(*p*)

IF *w<sub>i</sub>* == *w<sub>last</sub>* THEN

IF *w<sub>j</sub>* IN *entityList* THEN

*nop*  $\leftarrow$  *nop* + 1

IF *nop*  $\geq$  *maxNop* THEN

*result*  $\leftarrow$  *result*  $\cup$  (*w<sub>first</sub>*, *w<sub>i</sub>*, *w<sub>j</sub>*)

ELSE

*p'*  $\leftarrow$  *path*  $\cup$  (*w<sub>j</sub>*, *tag<sub>j</sub>*, *pos*)

Recur(*voc*, *entityList*, *maxNop*, *nop*, *p'*)

END IF

ELSE IF *tag<sub>j</sub>* IS VERB OR *tag<sub>j</sub>* IS ADP THEN

*p'*  $\leftarrow$  *p*  $\cup$  (*w<sub>j</sub>*, *tag<sub>j</sub>*, *pos*)

Recur(*voc*, *entityList*, *maxNop*, *nop*, *p'*)

END IF

END IF

END FOR

RETURN *result*

END FUNCTION

由于自然语言语法灵活多样, 语法规则不能保证抽取文本中所有的网络实体关系。例如图 1 文本中的“the use of software”介词短语后跟不定式结构, 表示 software 和 weakness 之间的 use 关系, 该结构就不在表 5 文本语法规则中。为了解决这一问题, 本文利用路径搜索思想, 从某一实体出发, 搜索多次语法依赖关系后可达的网络实体, 实体之间关系用距离实体最近的动词(VERB)或介词(ADP)表示, 具体如算法 1 所示。设置 *maxNop*=2, 从上述介词短语加不定式的文本结构可以抽取到<use, of, software>和<use, use, weakness>两组关系, 通过名词 use 实体作为桥梁, 推出 software 和 weakness 之间的 use 关系。算法 1 是对语法规则的补充, 进一步抽取网络实体关系, 提升语义知识信息完整性。

#### 4.4 语义规则构建

规则构建包括标签化和规则生成。标签化是根

据语义规则定义对网络实体标签及其关系的定义, 对知识抽取的网络实体指定数据类型标签。本文采用基于语义相似度的实体标注方法, 将网络实体通过 word2vec<sup>[23]</sup>转化为向量矩阵, 计算待标注实体和已标注实体的向量 cos 相似度, 对网络实体进行标注。

语义规则构建就是将标记好的网络实体、关系进行组织,生成一张语义规则图,再将规则图转化为语义规则。在构建语义规则时,首先去除不属于进程、文件、套接字类型的网络实体及其连接关系,然后将网络实体、关系三元组进行合并,生成语义规则图。在这个过程中,用不同形状节点表示不同的实体标签,蓝色线条表示操作类关系,黑色线条表示从属、并列类关系。由于语义规则模型描述的是完整的攻击过程,未区分攻击发起端和受害端,再加上防御者数据采集方法的限制,因此需要对自动化生成的语义规则进一步人工确认。人工确认遵循以下原则:

- 最大化保留攻击发起端行为语义信息。攻击发起端的数据虽然无法直接采集,但是其行为最终会在受害主机中体现,因此,最大化的将攻击发起端的行为转化到防御主机侧,可增强语义规则对攻击行为的还原能力。例如表 4.1 将漏洞利用代码文件实体转化到受害主机的进程实体中。
- 最大化保证语义规则有效性。语义规则最终是用于对审计日志进行匹配,从中发现威胁并还原上下文信息,因此要使得语义规则描述的行为不能超出防御者数据采集能力范围。例如去除代码签名、自定义指控协议、多跳代理等超出审计日志数据范围的语义规则。

## 5 实验评估

### 5.1 语义规则匹配算法实现

从 255 条 ATT&CK 技术文本中,构建 123 个 APT 攻击语义规则,涵盖 ATT&CK 的 115 项技术和 12 种战术。表 6 的每一行列举了每一个战术对应的规则数量、规则所涵盖的技术数量(规则技术)以及战术对应的技术总数,最后计算了规则涵盖的技术占技术总数的比例。由于 ATT&CK 战术和技术不是一一对应的,因此在统计总数时,对重复项目只统计一次。

从表 6 中可以看出,语义规则并未覆盖 ATT&CK 的所有技术,每个战术对应的规则技术占该战术技术总数比例也存在差异。这是由于在语义规则构建过程中,依据最大化保证语义规则有效性原则,对超出审计数据采集范围的规则进行人工筛选导致的。因为攻击执行战术中的技术大多在受害主机上实施,通过审计即可记录攻击过程,而命令控制战术对应的攻击技术特征一般表现在网络流量中,通过主机审计已经无法记录。因此,在规则涵盖的技术占技术总数的比例中,攻击执行战术比例最高,而命令控制战术对应的比例最少。

表 6 语义规则统计表

Table 6 Statistics of semantic rules

战术名称	规则数量	规则技术	技术总数	比例(%)
初始接入	5	5	11	45.45
攻击执行	28	27	34	79.41
持续控守	25	24	62	38.71
权限提升	9	9	32	28.12
防御规避	35	33	69	47.83
凭据窃取	9	8	21	38.10
信息发现	21	17	23	73.91
横向扩展	6	6	18	33.33
信息收集	5	5	13	38.46
命令控制	4	4	22	18.18
数据窃取	3	3	9	33.33
攻击影响	8	8	16	50.00
<b>总数</b>	<b>123</b>	<b>115</b>	<b>255</b>	<b>45.10</b>

语义规则匹配算法是用语义规则对审计日志进行匹配,发现威胁并还原上下文信息,具体如算法 2 所示。输入为语义规则和日志数据起源图,通过输入路径最小和最大长度,控制路径搜索范围,通过匹配阈值控制和语义规则的匹配程度。语义规则包括实体匹配规则集合  $H$  和关系匹配规则集合  $Z$ 。规则匹配分值是匹配到的关系和规则中关系总数的比值,用来衡量规则和数据的匹配程度,计算如下:

$$score = \frac{count(matchedRelation)}{count(R.Z)} \quad (3)$$

其中,  $matchedRelation$  是与语义规则成功匹配的关系集合,  $R.Z$  是语义规则的关系规则集合,  $count$  表示计数运算。算法使用 python 语言实现。

**算法 2.** 语义规则匹配算法.

输入:  $R$  语义规则,  $G_p$  日志数据起源图,  $min$  路径最小长度,  $max$  路径最大长度,  $threshold$  匹配阈值  
输出:  $result$  匹配成功的路径

```

FUNCTION Match( $R, G_p, min, max, threshold$ )
  nodeList  $\leftarrow$  GetNodeList( $G_p$ )
  FOR EACH node IN nodeList DO
    pathList  $\leftarrow$  GetPathList( $node, min, max$ )
    FOR EACH path IN pathList DO
      isMatchedEntity  $\leftarrow$  MatchEntities( $R.H, path$ )
      IF isMatchedEntity IS True THEN
        score  $\leftarrow$  MatchRelation( $R.Z, path$ )
        IF score > threshold THEN
          result  $\leftarrow$  result  $\cup$  path
        END IF
      END IF
    END IF
  END IF

```

```

END FOR
END FOR
RETURN result
END FUNCTION

```

语义规则处理对象是起源图, 而非采集的审计日志数据, 一方面可避免不同日志数据格式对语义规则的影响, 另一方面也提升了语义规则对不同网络场景的适应能力。此外, 研究发现, ATT&CK 单个攻击技术实施过程一般只发生在单个受害主机, 很少有跨多主机的操作。因此可以针对网络场景中的不同设备分别构建起源图, 不仅可以缩小起源图中的路径搜索范围, 提升检测效率, 还可以降低网络节点规模的影响。综上所述, 对于语义规则而言, 即使被应用于不同网络环境, 仍然只是对单个主机构建的起源图进行规则匹配, 检测攻击行为并还原攻击技术语义。因此, 在具备一定审计数据采集能力条件下, 只要攻击者技术手法不超出 ATT&CK 定义的技术范围, 网络环境和攻击技术手法对语义规则检测能力不会产生明显影响。另外, 由于语义规则也具备一定的可扩展性, 通过语义规则属性比较运算, 安全人员可根据实际网络环境和安全经验知识对语义规则进行定制, 进一步提升语义规则对不同网络场景和攻击技术手法的适应能力。

## 5.2 实验环境和数据

由于语义规则涵盖的攻击技术较多, 难以在现网中收集到所有技术对应的攻击日志数据。本文通过模拟构建实验场景, 依据 APT 生命周期模型<sup>[24]</sup>模拟实施攻击过程, 利用 Audit<sup>[25]</sup>审计工具采集场景中的日志数据并作为实验数据。

实验场景由三台 VMWare Workstation 虚拟机组成: (1)Ubuntu 16.04.6 LTS 64-bit 虚拟机作为内网网关, 运行 OpenVPN 服务, 将内外网隔离; (2)Fedora 29 64-bit 虚拟机作为内网主机, 开启 SSH 服务; (3)Kali Linux 64-bit 虚拟机作为攻击机, 从外网对内网发起攻击。场景中的虚拟机器均使用 Intel® Core™ i3-7100 CPU @ 3.90GHz × 2 CPU, 4GB 内存。

在采集数据时, 利用 Audit 工具对内网网关和内网主机进行审计, 通过监控系统调用、关键文件、目录的方式, 重点记录进程、文件、套接字三种实体的日志数据。实验共分三个阶段模拟 APT 攻击过程。在第一阶段, 利用 CVE-2014-6271 漏洞<sup>[26]</sup>对内网网关实施攻击, 建立立足点, 搜集主机、服务等信息, 提升权限; 在第二阶段, 通过 VPN 接入内网, 向内网网关主机植入木马后门, 实现对内网网关的持续控守; 在第三阶段, 以内网网关为跳板, 对内网其他主机进行探测, 通过口令猜解接入内网主机, 实现

横向扩展和情报收集, 最终完成攻击任务, 消除攻击痕迹。整个攻击过程共使用了 ATT&CK 中的 29 项攻击技术。

利用 SPADE<sup>[27]</sup>工具对实验数据进行预处理, 将审计日志转化为起源图, 图中节点是监控的进程、文件、套接字信息, 边是实体之间的关系。实验数据信息如表 7 所示。

表 7 实验数据信息  
Table 7 Experimental data

项目	数值
日志数据大小	31.65MB
网络实体总数	32056 个
网络实体关系总数	73364 个

## 5.3 语义规则有效性测试

通过对实际数据进行匹配测试, 验证语义规则的有效性。主要依据检出率(Recall)和虚警率(False Alarm)两个指标来衡量。检出率是成功识别攻击技术占数据中真实攻击技术总数的比例, 虚警率是将正常行为判断为攻击行为的数量占算法检测出攻击行为总数的比例。从两个角度衡量语义规则的攻击行为检测能力。

在实验中, 设置匹配算法的路径最小长度为 5, 路径最大长度为 12, 匹配阈值为 0.7。实验结果显示, 算法从 29 项攻击技术中匹配出 27 项攻击技术, 在该场景中检出率达 93.1%, 具有较强的检测能力, 匹配结果如表 8 所示。其中, 列举了匹配成功的技术和匹配到的行为数量, 并且每一项匹配成功的技术列举了一个行为示例, 该示例是根据语义规则简化后的行为路径。在检测结果中, T1018 技术是攻击者通过脚本连续发送大量的 ping 数据包, 探测内网存活主机, 因此匹配到的行为数量最多; T1219 技术是由于植入的木马在运行时由于绑定端口失败, 从而不断尝试, 因此产生大量的异常行为; T1210 技术是在对内网进行扫描时, 产生了大量扫描探测行为。对于漏检的两项技术, T1169 由于审计工具没有捕获相应日志数据, 导致规则匹配失败; T1043 是由于语义规则 and 实际数据不一致导致匹配失败。

语义规则虚警率难量化衡量, 这是由于 ATT&CK 中定义的很多攻击技术都是操作系统提供的正常功能, 之所以将其定义为攻击, 是因为这些功能在一定条件下会被攻击者的滥用, 从而达到攻击目的。例如, 远程服务(T1021)、进程发现(T1057)、文件删除(T1107)等技术, 如果从技术本身来看, 都是正常的用户操作, 但是在 APT 攻击的上下文语境



表 8 语义规则匹配结果

Table 8 Experimental results of semantic rule matching

技术编号	技术名称	数量	示例
T1016	System Network Configuration Discovery	18	
T1018	Remote System Discovery	1016	
T1021	Remote Services	6	
T1033	System Owner or User Discovery	3	
T1046	Network Service Scanning	3	
T1049	System Network Connections Discovery	12	
T1057	Process Discovery	14	
T1059	Command-Line Interface	198	
T1065	Uncommonly Used Port	4	
T1069	Permission Groups Discovery-linux	2	
T1078	Valid Accounts	1	
T1081	Credentials in Files	9	
T1082	System Information Discovery	57	
T1083	File and Directory Discovery	3	
T1087	Account Discovery	11	
T1098	Account Manipulation	4	
T1105	Remote File Copy	14	
T1107	File Deletion	6	
T1133	External Remote Services	2	
T1166	Setuid and Setgid	4	
T1190	Exploit Public-Facing Application	4	
T1210	Exploitation of Remote Services	311	
T1219	Remote Access Tools	942	
T1222	File and Directory Permissions Modification	16	
T1485	Data Destruction	6	
T1518	Software Discovery	3	
T1531	Account Access Removal	3	
T1169	Sudo	0	-
T1043	Commonly Used Port	0	-

中, 这些技术可作为攻击者服务探测、信息收集、痕迹消除的手段。所以要判断检测结果是否属于攻击, 要结合上下文信息进行具体分析。语义规则匹配的自动化从审计数据中发现潜在攻击行为, 并为分析人员提供相关上下文信息, 辅助威胁分析研判。下面通过具体案例, 测试语义规则的上下文语义还原能力。

案例 1: 远程漏洞攻击

语义规则从数据中匹配发现攻击者远程漏洞攻击

行为, 并还原其上下文语义信息, 如图 4 所示, 其中 G1 是语义规则匹配结果, 即攻击者利用远程漏洞突破内网网关。除此之外, 语义规则还还原了漏洞利用成功后, 攻击者建立立足点、搜集信息、建立帐户实现持续控制等行为, 具体包括获取内网网关远程 shell 终端, 搜集获取内网网关设备、系统、用户、文件、网络信息, 并添加新用户实现持续化控制等。这些信息比较完整地刻画了攻击者在漏洞利用成功后地后续行为, 可为攻击过程分析、评估提供重要信息支撑。

## 案例 2: 内网扫描攻击

语义规则从数据中发现了两次内网扫描攻击, 都是采用远程系统发现技术, 如图 5 所示。其中, G2 是规则匹配结果, 即脚本通过 ping 对远程主机进行探测, 如果单纯从匹配结果看, 很难判断该行

为是否异常。但是经还原其完整上下文信息, 发现该脚本是对一个网段 254 个 IP 地址进行探测, 是个典型的内网扫描行为, 而且扫描脚本是通过远程 shell 终端启动的, 进一步确定规则匹配结果属于攻击。

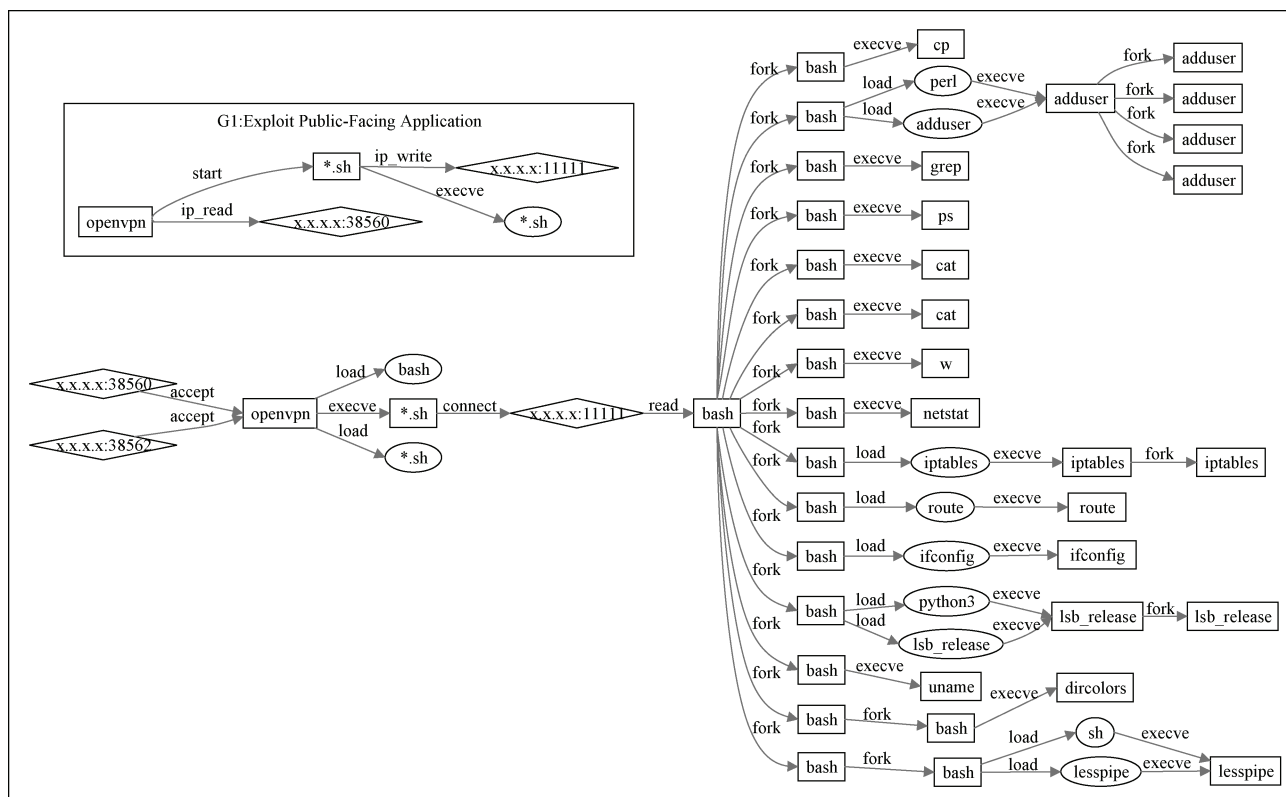


图 4 远程漏洞利用攻击上下文信息

Figure 4 Context information of remote exploit attack

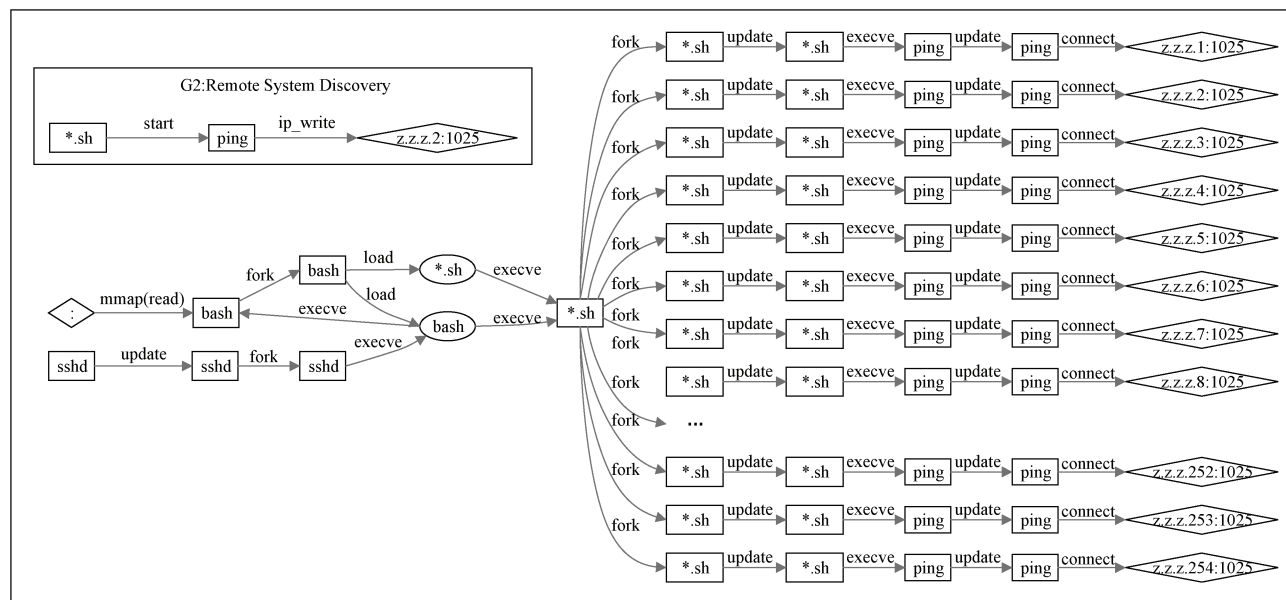


图 5 内网扫描攻击上下文信息

Figure 5 Context information of intranet scan

通过以上案例证明, 语义规则可通过上下文信息还原, 有效发现攻击行为, 降低虚警率。语义规则检测对象是日志数据起源图中的路径, 该路径反映了攻击实施过程在受该主机中所表现的行为序列, 不仅包含行为主体、客体, 还包含主客体之间的操作关系。而语义规则就是分别针对此行为主体、客体及其操作关系进行检测, 一旦匹配成功, 可以还原整个行为序列, 即攻击上下文信息。因此, 路径长度可决定行为序列包含信息量的多少, 直接影响语义规则对攻击上下文的还原能力。

5.4 语义规则匹配对比测试

本实验将语义规则和 IOC 规则进行对比, 通过检出率指标比较两种规则对攻击行为的检测能力。通过 IOC Editor<sup>[4]</sup>工具创建 IOC 规则, 对实验场景中的审计数据进行匹配, 发现攻击行为。IOC 规则关键信息如表 9 所示。其中, 主要包括攻击者使用的木马、扫描脚本、指控 IP 地址、URL 以及木马和攻击工具文件哈希等。由于 IOC 规则只能判断主机是否遭受攻击, 无法还原攻击对应的 TTPs 信息, 因此需通过人工分析, 按照 ATT&CK 攻击技术分类标准, 将 IOC 匹配结果和攻击技术进行映射, 从而计算出率。实验结果表明, IOC 规则共检测出 13 项技术, 检出率为 44.8%。具体如表 10 所示。

表 9 IOC 检测规则信息  
Table 9 Major information of IOC rules

项目	数值
URLs	http://x.x.x[.]x/http, http://x.x.x[.]x/netscan.sh
IP 地址	x.x.x[.]x
文件	/usr/bin/http, netscan[.]sh, contact_list[.]xls
MD5 哈希	2df731f803055bb8b455623feb4b3a09 c430c4da79ba6da3e1629c71c407928b

从对比结果可以看出, 对攻击的检测上, 语义规则明显优于 IOC 规则。IOC 规则主要基于攻击指纹特征, 用于直接从数据中识别攻击, 对检测准确性要求较高, 所以检测结果都是包含攻击指纹特征的行为, 但是对攻击者滥用系统正常功能而执行操作无法识别。而语义规则旨在为分析人员提供潜在攻击线索, 只要符合规则描述的行为规律, 都会被成功匹配, 因此匹配结果既包含远程漏洞利用等攻击行为, 又包含攻击者调用系统正常功能而执行的恶意操作, 对攻击行为检测更加全面。

实验中, IOC 规则检测攻击并还原语义耗时近 1 h, 而语义规则平均耗时约 205 s。在效率上, 语义规则也优于 IOC 规则。IOC 规则只能检测攻

表 10 IOC 检测规则和语义规则匹配结果对比  
Table 10 Comparison of experimental results  
between IOC rules and semantic rules

技术编号	IOC 规则	语义规则	技术编号	IOC 规则	语义规则
T1016		✓	T1098	✓	✓
T1018	✓	✓	T1105		✓
T1021	✓	✓	T1107		✓
T1033		✓	T1133		✓
T1046	✓	✓	T1166	✓	✓
T1049		✓	T1190	✓	✓
T1057	✓	✓	T1210		✓
T1059		✓	T1219	✓	✓
T1065	✓	✓	T1222		✓
T1069		✓	T1485		✓
T1078	✓	✓	T1518		✓
T1081		✓	T1531	✓	✓
T1082		✓	T1169		
T1083	✓	✓	T1043	✓	
T1087		✓			

(注: ✓表示规则可成功识别对应的技术)

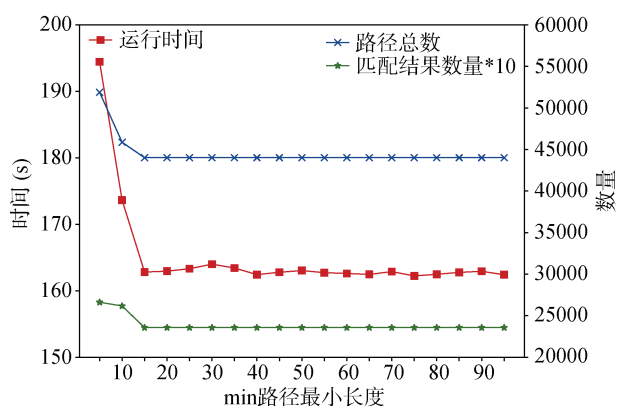
击, 却无法自动还原攻击语义, 因此基于 IOC 的攻击语义还原需要人工参与, 这种半自动化的方式制约了分析效率; 而语义规则可自动化从审计数据中发现攻击并还原语义信息, 避免了工参与, 分析效率更高。

5.5 语义规则匹配性能测试

通过测试语义规则匹配效率, 研究影响效率的主要参数。实验在一台工作站上进行, 操作系统为 Microsoft Windows 10 64 位专业版, CPU 为 Intel® Core™ i3-7100 CPU @ 3.90GHz, 内存为 16GB DDR4。在正常运行时, 设置路径最小长度为 4, 路径最大长度为 12, 匹配阈值为 0.7, 算法 10 次运行平均时间约为 205.05 s, 从 54252 条路径中匹配到 2662 个攻击行为, 还原生成 27 个攻击上下文信息。

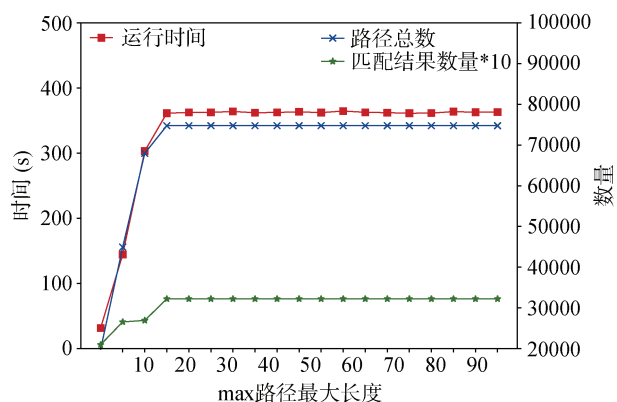
本实验采用控制变量法, 分别测试路径最小长度(min)、路径最大长度(max)、匹配阈值(threshold)三个参数对语义规则匹配效率的影响。每个参数都经过 10 轮测试, 取平均测试结果, 如图 6 所示。从图中可以看出, 路径最小长度和路径最大长度对算法运行时间效率有明显影响, 匹配阈值对性能影响变化不大。随着路径最小长度的增大, 算法运行时间开始下降, 当路径最小长度大于 15 时, 运行时间逐渐趋于稳定, 如图 6.1 所示; 随着路径最大长度的增加, 算法运行时间开始急剧增加, 当路径最大长度大于 20 时, 运行时间趋于稳定, 如图 6.2 所示; 当算

法匹配阈值不断增加时, 算法运行时间在小范围内略微变化, 但是总体趋于稳定, 如图 6.3 所示。



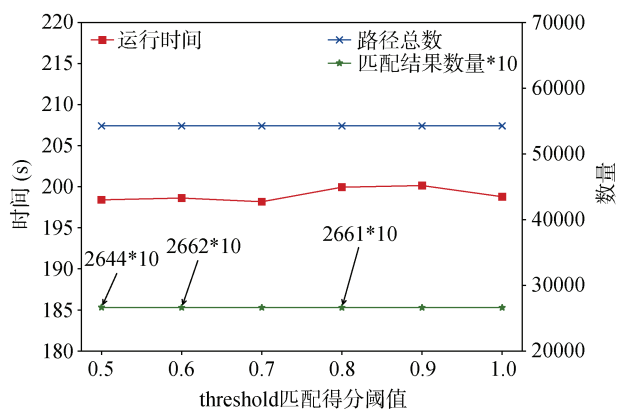
(a) 路径最小长度对性能影响(max=100, threshold=0.7)

(a) The impact of the minimum path length on performance (max=100, threshold=0.7)



(b) 路径最大长度对性能影响(min=4, threshold=0.7)

(b) The impact of the maximum path length on performance (min=4, threshold=0.7)



(c) 匹配阈值对性能影响(min=4, max=12)

(c) The impact of the threshold on performance (min=4, max=12)

图 6 性能测试结果

Figure 6 Experimental results of performance test

在测试算法运行时间同时, 对算法从输入数据中搜索的路径总数以及成功匹配到的结果总数进行了统计, 如图 6 所示(为了更好体现变化, 在图中对匹配结果数量放大了 10 倍)。在图中可以看出, 路径

总数、匹配结果数量都和运行时间成正相关, 并且受到路径最小长度和路径最大长度两个参数影响较大, 受到匹配阈值影响很小。这是由于在给定的输入日志数据不变的情况下, 路径最小长度和路径最大长度两个参数直接控制算法搜索的路径数量, 算法会对每一条路径进行语义规则匹配, 路径越多, 算法匹配耗时越长, 因此路径总数和运行时间呈正相关。但是随着路径长度的变化, 路径总数不是无限增大(或减小)的, 而是逐渐趋于稳定, 这是由输入日志数据起源图本身最大路径长度决定的。在路径最小长度和路径最大长度不变的情况下, 匹配阈值决定算法接受匹配结果的严格程度, 但是路径总数不会变化, 所以算法匹配时间总体不变, 只会影响匹配结果数量, 因此匹配阈值对算法运行效率影响较小。路径长度决定路径中包含的上下文信息量, 在保证一定检出率和语义还原能力的情况下, 选择适当的路径最小长度和路径最大长度, 可以有效优化算法运行效率。

## 6 结论

自然语言语义鸿沟是制约 TTPs 威胁情报自动化利用的重要因素, 本文设计并实现了一种基于 ATT&CK 的 APT 攻击语义规则, 将攻击技术文本中的语义知识抽取成可用对数据检测的规则, 从而实现底层数据到上层攻击技术语义知识的映射, 以此加强 TTPs 威胁情报在威胁分析中的自动化应用。实验结果表明, 语义规则检出率达到 93.1%, 具备一定的自动化攻击检测和上下文信息还原能力, 能够辅助分析人员较为快速全面地发现数据中的潜在攻击行为, 提升威胁分析效率。语义规则匹配结果是攻击行为以及攻击上下文信息, 只是 APT 生命周期不同阶段所表现的局部行为, 但是对完整生命周期的还原还需对各阶段行为进行综合关联分析, 因此, 将语义规则用于 APT 攻击生命周期还原将是下一步的研究重点。

## 参考文献

- [1] Gartner. Definition: Threat Intelligence[EB/OL]. <https://www.gartner.com/en/documents/2487216/definition-threat-intelligence>. Jun. 2020.
- [2] Bianco D.. The pyramid of pain[EB/OL]. <http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>, Jun. 2020.
- [3] SANS. 2020 SANS Cyber Threat Intelligence (CTI) Survey [EB/OL]. [https://threatconnect.com/wp-content/uploads/Survey\\_CTI-2020\\_ThreatConnect.pdf](https://threatconnect.com/wp-content/uploads/Survey_CTI-2020_ThreatConnect.pdf). Jun. 2020.

- [4] MANDIANT. IOC Editor User Guide[EB/OL]. <https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf>. Jun. 2020.
- [5] Milajerdi S M, Gjomemo R, Eshete B, et al. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows[C]. *2019 IEEE Symposium on Security and Privacy*, 2019: 1137-1152.
- [6] Kurogome Y, Otsuki Y, Kawakoya Y, et al. EIGER: Automated IOC Generation for Accurate and Interpretable Endpoint Malware Detection[C]. *The 35th Annual Computer Security Applications Conference*, 2019: 687-701.
- [7] Mittal S, Joshi A, Finin T. Cyber-all-Intel: An AI for Security Related Threat Intelligence[EB/OL]. 2019
- [8] Zhu Z Y, Dumitras T. ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports[C]. *2018 IEEE European Symposium on Security and Privacy*, 2018: 458-472.
- [9] Tounsi W, Rais H. A Survey on Technical Threat Intelligence in the Age of Sophisticated Cyber Attacks[J]. *Computers & Security*, 2018, 72: 212-233.
- [10] Liao X J, Yuan K, Wang X F, et al. Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 755-766.
- [11] Milajerdi S M, Eshete B, Gjomemo R, et al. POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting[C]. *CCS '19: The 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019: 1795-1812.
- [12] Strom B E, Applebaum A, Miller D P, et al. Mitre att&ck: Design and philosophy[R]. Technical report, 2018.
- [13] Strom B E, Battaglia J A, Kemmerer M S, et al. Finding cyber threats with ATT&CK-based analytics[R]. Technical report. The MITRE Corporation, 2017.
- [14] Oosthoek K, Doerr C. SoK: ATT&CK Techniques and Trends in Windows Malware[C]. *International Conference on Security and Privacy in Communication Systems*. Springer, Cham, 2019: 406-425.
- [15] The MITRE Corporation. Exploit Public-Facing Application [EB/OL]. <https://attack.mitre.org/techniques/T1190/>. Jun. 2020.
- [16] Moreau L. The foundations for provenance on the web[M]. Now Publishers Inc, 2010.
- [17] Hassan W U, Guo S, Li D, et al. Nodotze: Combatting threat alert fatigue with automated provenance triage[C]. *Network and Distributed Systems Security Symposium*. 2019.
- [18] Liu Q, Li Y, Duan H, et al. Knowledge Graph Construction Techniques[J]. *Journal of Computer Research and Development*, 2016, 53(3): 582-600.  
(刘峤, 李杨, 段宏, 等. 知识图谱构建技术综述[J]. *计算机研究与发展*, 2016, 53(3): 582-600.)
- [19] Bouma G. Normalized (Pointwise) Mutual Information in Collocation Extraction[J]. *German Society for Computational Linguistics (GSCL 2009)*, 2009: 31-40.
- [20] NLTK Project. Natural Language Toolkit[EB/OL]. <http://www.nltk.org/>. Apr. 2020.
- [21] Explosion AI. spaCy: Industrial-strength natural language processing[EB/OL]. <https://spacy.io>. Apr. 2020.
- [22] Universal Dependencies. UD English GUM[EB/OL]. [https://universaldependencies.org/treebanks/en\\_gum/index.html](https://universaldependencies.org/treebanks/en_gum/index.html). Apr. 2020.
- [23] Goldberg Y, Levy O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method[EB/OL]. 2014: ArXiv Preprint ArXiv:1402.3722.
- [24] APT1: Exposing One of China's Cyber Espionage Units[EB/OL]. <https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf>. Apr. 2020.
- [25] Github. Linux Audit Documentation Project[EB/OL]. <https://github.com/linux-audit/audit-documentation/wiki>. Apr. 2020.
- [26] The MITRE Corporation. CVE-2014-6271[EB/OL]. <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271>. Jun. 2020.
- [27] SPADE wiki[EB/OL]. <https://github.com/ashish-gehani/SPADE/wiki>. Apr. 2020.



**潘亚峰** 于 2013 年在信息工程大学网络工程专业获得学士学位。现在信息工程大学计算机技术专业攻读硕士学位。研究领域为网络安全与效果评估。研究兴趣包括: 威胁情报、威胁分析。Email: pyfeng02@163.com



**周天阳** 博士。现任信息工程大学数学工程与先进计算国家重点实验室副教授。研究领域为网络安全与效果评估。Email: aip-teamzhouty@aliyun.com





**朱俊虎** 博士。现任信息工程大学数学工程与先进计算国家重点实验室教授。研究领域为网络模拟与仿真、网络安全测评。  
Email: zhujunhu74@163.com



**曾子懿** 于 2019 年在信息工程大学软件工程专业获得博士学位。现任信息工程大学数学工程与先进计算国家重点实验室讲师。研究领域为网络建模仿真、网络空间安全。  
Email: zyzeng7@163.com