软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(7):2683−2698 [doi: 10.13328/j.cnki.jos.006282]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于隐马尔可夫模型的加密恶意流量检测*

邹福泰, 俞汤达, 许文亮

(上海交通大学 网络空间安全学院, 上海 200240)

通信作者: 邹福泰, E-mail: zoufutai@sjtu.edu.cn

摘 要: 近年来, 随着网络加密技术的普及, 使用网络加密技术的恶意攻击事件也在逐年增长, 依赖于数据包内容的传统检测方法如今已经无法有效地应对隐藏在加密流量中的恶意软件攻击. 为了能够应对不同协议下的加密恶意流量检测, 提出了基于 Profile HMM 的加密恶意流量检测算法. 该方法利用生物信息学上的基因序列比对分析, 通过匹配关键基因子序列, 实现识别加密攻击流量的能力. 通过使用开源数据集在不同条件下进行实验, 结果表明了算法的有效性. 此外, 设计了两种规避检测的方法, 通过实验验证了算法具有较好的抗规避检测的能力. 与已有研究相比, 该工作具有应用场景广泛以及检测准确率较高的特点, 为基于加密流量的恶意软件检测研究领域提供了一种较为有效的解决方案.

关键词: 恶意软件; 加密恶意流量检测; 隐马尔可夫模型; 基因序列

中图法分类号: TP309

中文引用格式: 邹福泰, 俞汤达, 许文亮. 基于隐马尔可夫模型的加密恶意流量检测. 软件学报, 2022, 33(7): 2683–2698. http://www.jos.org.cn/1000-9825/6282.htm

英文引用格式: Zou FT, Yu TD, Xu WL. Encrypted Malicious Traffic Detection Based on Hidden Markov Model. Ruan Jian Xue Bao/Journal of Software, 2022, 33(7): 2683−2698 (in Chinese). http://www.jos.org.cn/1000-9825/6282.htm

Encrypted Malicious Traffic Detection Based on Hidden Markov Model

ZOU Fu-Tai, YU Tang-Da, XU Wen-Liang

(School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: In recent years, with the popularization of network encryption technology, malicious attacks using network encryption technology have increased year by year. Traditional detection methods that rely on the content of data packets are now unable to effectively deal with malware attacks hidden in encrypted traffic. In order to deal with the detection of encrypted malicious traffic under different protocols, this study proposes an encrypted malicious traffic detection algorithm based on profile HMM. This method uses the genetic sequence comparison analysis in bioinformatics to realize the identification of encrypted attack traffic by matching key gene sub-sequences. Open source datasets are used to conduct experiments under different conditions, the results demonstrate the effectiveness of the algorithm. In addition, two methods of evasion detection are designed, and experiments have also verified that the algorithm has a better performance to resist evasion detection. Compared with the existing research, the work of this study has a wide range of application scenarios and higher detection accuracy. It provides a more effective solution to the research field of malware detection based on encrypted traffic.

Key words: malware; encrypted malicious traffic detection; hidden Markov model; gene sequence

随着 4G 网络的普及, 移动上网已经成为人们上网的一个重要方式. 根据 ZenithMedia 研究显示, 2018 年, 移动互联网流量占互联网流量的比例高达 80%. 移动互联网应用已经在吃、穿、住、行等众多方面成为人们生活中不可或缺的一部分. 我国目前正积极推进第五代移动通信(5G)和超宽带关键技术的研究, 并启动了 5G 的商用试点. 在未来, 移动数据流量将成为互联网上网的主流方式. 然而在安全方面, 移动互联网仍处于初级

* 基金项目: 国家重点研发计划(2020YFB1807500)

收稿时间: 2020-09-08; 修改时间: 2020-10-20; 采用时间: 2020-12-12

阶段, 缺乏统一的技术标准, 面临诸多严峻的安全问题[1]. 目前, 移动设备操作系统及移动互联网应用的漏洞, 移动互联网通信安全的问题日益凸显, 各种恶意软件造成的隐私窃取、网络金融诈骗等安全事件时有发生. 为此, 网络传输广泛采用了 SSL 这样的加密传输协议以增强安全防御.

不仅仅是移动网络, 报告[2]显示, 2019 年, 已有超过 80%的在线流量被加密, 而且加密流量使用比例趋势逐年增高. 流量加密固然保护了数据的隐私和安全, 然而也给恶意软件提供了藏身之所, 让不法分子利用这些相同的好处来逃避检测和保护他们的恶意攻击行为. 使用网络加密技术能够使得他们传输数据时像隐藏用户隐私信息一样隐藏恶意软件(如僵尸网络、木马和病毒等), 从而被攻击者利用以逃避已有的网络安全审计.

近年来, APT 攻击(advanced persistent threat, 高级持续攻击)造成的影响和危害逐渐引起广泛关注. APT 攻击往往通过持续地高级别渗透技术及社会工程手段以窃取机密情报. 同时, 在攻击过程中, 利用 SSL 等网络加密技术, 混淆传输数据内容以规避侦测软件和防御系统, 传播、扩散恶意可执行文件, 给灾后检测和溯源带来巨大的挑战. 据统计, 目前, 超过 60%的网络流量是通过 TLS/SSL 加密的, 由恶意软件产生的恶意流量占 10%以上[3]. Gartner 预测: 到 2020 年前, 有半数的恶意攻击将通过加密协议传送恶意软件.

在网络加密协议中, 使用频率最高的是 HTTPS 协议. 报告显示, 2019 年第一季度, 已经有 58%的网络钓鱼攻击使用了 HTTPS 协议, 相较 2018 年, 全年增长了近 50%. 而 HTTPS 被恶意软件如此青睐主要有两方面原因: 一是由于使用 SSL 的门槛降低, 攻击者如今可以轻易地创建免费的证书; 二是因为大多数网站已经默认使用 SSL 协议, 而不使用这种协议会被浏览器警告. 另一方面, 大多数僵尸网络的 C&C 服务器都已经使用加密通信协议[4]. 文献[5]统计了僵尸网络 Mirai 及其变种的协议使用情况, 数据显示, Mirai 僵尸网络使用了 HTTPS, FTP, Telnet, CWMP (基于 HTTP 的一种协议)以及 SSH 协议等多种协议, 其中, 使用最多的协议是 HTTPS. 除了 HTTPS 以外, 其他网络加密协议, 如 SSH, 也在统计使用最多几种的协议中出现. 而非加密协议中, CWMP (一种基于 HTTP 的协议, 可以自动配置和远程控制)被大量使用. 文献[6]表明, 僵尸网络使用的协议正以一个非常高的速度演进, 针对某种特定协议的僵尸网络检测方法会很快被淘汰.

然而, 目前各类安全防护软件、探针主要针对非加密流量进行识别和检测, 无法有效应对加密流量中的恶意软件检测. 随着网络加密的迅速普及, 这些设施的防护能力被极大地削弱. 政府企业将重要资产的安全风险暴露在外部网络却无法进行有效的检测和防御, 形势极为严峻. 因此, 研究能够对加密流量的恶意软件检测技术, 具有迫切的现实意义以及更为广阔的应用前景.

随着用户隐私保护和网络意识的增强, 并且加密协议其本身又具备良好的兼容性和延展性[7], 网络加密技术应用愈发广泛, 网络加密流量检测和识别目前已经是一个热门的研究领域. 其中, 大多数研究针对软件的网络通信特征进行分析, 而基于网络流量的恶意软件检测, 主要研究对象为攻击流量和控制端指令.

文献[8−10]总结了目前已有的基于网络流量的检测研究, 虽然已经在网络流量检测领域取得了许多建设性的成果, 并且在工业界不乏成功的应用案例, 例如 Snort 和 nDPI, 但这些研究大多解决的是检测内容可见的流量, 即未经过加密的网络流量. 事实上, 研究检测加密流量与检测非加密流量有着较大差异, 主要有以下 3 点: (1) 由于网络流量经过加密后, 外部可见的内容发生了较大的变化, 大部分非加密流量识别方法很难适用于加密流量, 如常见的基于 HTTP 载荷的恶意攻击检测 Domount[11]和 Decanter[12]; (2) 通常, 攻击者在使用加密协议对流量进行加密的同时, 还会使用流量伪装技术对流量进行混淆或模仿正常流量通信模式, 从而规避检测; (3) 不同的网络加密协议可以选择使用不同的加密算法以及封装方式, 这种没有统一标准的情况往往会给检测任务带来巨大的困难.

对加密网络流量进行检测和分类研究者通常采用机器学习方法. Arndt 等人[13]使用 C4.5, *k*-means 和 Multi-Objective Genetic Algorithm (MOGA)等方法对加密流量进行分类, 测试结果表明, C4.5 具有更强的预测能力. Kumano 等人[14]则使用了 C4.5 和 SVM 算法对使用加密流量的应用实现鉴别. 上述方法基于机器学习中的有监督学习, 利用训练数据集分类器模型, 并通过验证数据集调整模型参数, 检测结果很大程度上依赖于数据标注的准确性. Anderson 等人[15]总结了恶意软件产生的加密流量中的敏感字段, 并使用线性回归、决策树、随机森林、SVM 和多层感知器等多种机器学习算法对这些字段进行训练, 从而使用训练生成的分类器检测恶意

软件产生的加密流量. 实验结果表明, 随机森林算法具有更高的准确率. 此外, Anderson 等人的研究发现: 错误的数据标签会对大部分机器学习算法结果产生影响, 而相对的, 随机森林和多层感知器受数据噪声的影响较小. Anderson 等人得到的实验结果准确率极高, 但其算法仅对是否为恶意软件产生的加密流量进行分类, 并没有对恶意软件的种类进行区分. 考虑到标定数据的工作量问题, 一些研究采用半监督学习, 使用带标签的流量数据训练模型, 从而对未标定的数据进行预测. Bernaille 等人[16]对流量进行聚类, 从而检测使用 SSL 协议加密的应用. Backquet 等人[17]使用 MOGA, 用流特征子空间和参数实现聚类算法. Bar 等人[18]通过 *K*-means 和 KNN 聚类算法实现了一个即时加密流量分类器. Zhang 等人[19]提出了对 *K*-means 聚类算法的改进, 从而检测加密流量. 另外一些研究集中到具体的应用协议上. Conti 等人[20]采用动态时间规整、层次聚类和随机森林分类器等机器学习技术来识别用户在其移动应用程序上执行的特定操作. Shen 等人[21]和 korczynski 等人[22]都使用了马尔可夫链构建 TLS/SSL 协议下的应用流量指纹, 从而对不同应用产生的加密流量进行识别. Fahad 等人[23]利用样本建立隐马尔可夫模型(hidden Markov model, HMM)来识别网络流量, 实验结果还表明, 该方法可以有效地处理背景噪声. Gu 等人[6]介绍了一种不依赖于协议的僵尸网络检测方法, 从主机之间的通信行为特征和单个主机表现出的恶意行为特征两个维度评判僵尸主机. 实验结果表明, 该方法能够区分使用不同网络协议的僵尸网络. 俞汤达[24]对基于 HTTPS 的恶意软件流量进行研究, 设计实现多层自动编码器的加密恶意流量检测算法, 可以区分不同类型的恶意软件的加密流量. 潘雨婷等人[25]提出一种基于信任的加密流量 DDoS 发现方法, 融入基于签名和环境因素的信任评估机制, 基于特征构建 Ball-tree 并使用 *k* 近邻分类算法发现恶意流量. 实验结果表明, 该方法可以更好地发现加密流量 DDoS 攻击, 同时实现对合法租户敏感流量信息的保护.

以上研究大多集中在检测特定类型的加密协议. 根据文献[5], 僵尸网络 Mirai 和其变种有多达 11 种协议与其 C&C 服务器通信. 显然, 对恶意软件使用的网络加密协议逐一分析并设计一种特定的检测算法是十分繁琐且低效的. 为了应对网络环境中包含多种不同协议的应用场景, 本文研究使用基于 Profile HMM 的方法检测包含恶意软件加密流量, 将基因序列与恶意流量产生的上下文进行对应, 通过分析恶意流量的隐藏状态关系搜索同源基因序列, 从而达到分类恶意流量的效果.

本文的主要贡献是: (1) 将 Profile HMM 应用于加密恶意流量检测, 为对抗恶意软件通过加密技术躲避检测提供了新颖的可行方案; (2) 能够检测识别多种恶意软件家族产生的加密恶意流量, 而不仅仅识别流量是否具有恶意性; (3) 在真实数据集上使用 Profile HMM 进行检测, 达到 91.51%的多分类准确率, 具有较强的识别能力.

本文第 1 节介绍 Profile HMM 的概念及本文在此基础上的应用. 第 2 节具体展示实验过程. 在第 3 节对实验结果进行详细的解释. 最后, 第 4 节进行全文总结.

1 基于 Profile HMM 的加密流量恶意软件检测

1.1 Profile HMM概述

首先, 我们将对 Profile HMM 进行简要概述. Profile HMM 最早由 Krogh[26]提出. 文献[27]对 Profile HMM 进行了详细的分析. 最初, Profile HMM 主要用于生物序列分析, 如同源基因序列搜索、判断一个新出现的序列是否属于同一个序列家族等. 后来利用这一特性, 在语音识别、文本分类等多个领域都有应用. 目前已有各种基于 HMM 的 Profile HMM 及其相关模型, 因此首先介绍 HMM 的原理, 以便于理解 Profile HMM.

1.1.1 隐马尔可夫模型

隐马尔可夫模型描述了潜在无限个序列上的概率分布. 因为概率分布必须和为 1, 所以 HMM 分配给序列的概率是有限制的. 如果不减少一个或多个其他序列的概率, 就不能增加一个序列的概率.

图 1 为一个简单的 HMM 示例, 它对由两个字母(a,b)组成的序列进行建模. 从一个初始状态开始, 我们选择一个具有某种转移概率的新状态(要么以 $t_{1,1}$ 的转移概率停留在状态 1, 要么以 $t_{1,2}$ 的转移概率移动到状态 2); 然后生成一个发射概率为的状态(如以概率 $P_1(a)$选择字母 a). 我们重复这个转移/发射过程, 直到到达一个终

点状态. 在这个过程的最后, 得到一个没有观察到的隐藏状态序列π和一个观察到的符号序列 X. “隐马尔可夫模型”的名称来源于其隐藏状态序列π是一阶马尔可夫链, 只有符号序列是可以直接观测到的.

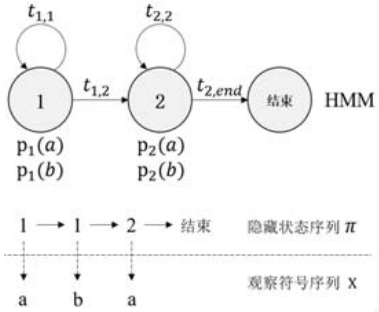

图 1 HMM 示例

1.1.2 Profile HMM

如图 2 所示, Profile HMM 在 HMM 的基础上新增了两个状态 Insert 状态和 Delete 状态, 允许在该状态和下一个状态之间插入一个或多个状态, 或用于删除状态.

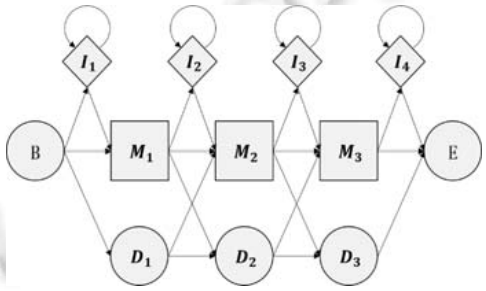

图 2 Profile HMM 示例

Profile HMM 基于传统隐马尔可夫模型, 但与传统的隐马尔可夫模型相比, Profile HMM 有如下优势.

1) 比较明显的一点是, Profile HMM 新加入了 Insert 状态和 Delete 状态, 这使得该模型具有更加健壮的匹配能力, 能够通过插入或删除值来应对异常情况.
2) Profile HMM 通过多序列对齐以及控制 Insert 状态和 Delete 状态的发生概率, 记录某个状态位置信息, 然而传统的隐马尔可夫模型无法保留这些信息.

简而言之, 基于此结构的 Profile HMM 与传统的隐马尔可夫模型相比有两个优点: 1) Profile HMM 利用了观测序列中包含的位置信息, 而标准的 HMM 忽略了这些信息; 2) Profile HMM 允许空转换, 以便模型能够匹配包含插入或删除的序列. 举例来说, 如图 2 所示, D 表示删除状态, I 表示插入状态, M 表示匹配状态.

此外, 模型还包含开始状态 B 和结束状态 E. 插入状态 I 使得序列能够在匹配状态 M 或删除状态 D 之前在特定位置插值. 给定符号序列 S, 如果匹配状态 M_i 和 M_j 之间需要匹配的符号比该序列中所包含的符号要少, 那么一个符号序列在匹配模型时, 将会从 M_i 进入后, 经过若干次匹配插入状态 I 后, 从插入状态 I 再转移到下一个匹配状态 M_j. 类似地, 删除状态 D 使得模型能够在特定位置删除符号, 当序列符号不在模型匹配状态 M_i 与 M_j 之间的符号中时, 该路径将从匹配状态 M_{i-1} 转移到匹配若干个删除状态 D, 直到最终能匹配到 M_{j+1} 状态.

作为 Profile HMM 在网络安全另一个领域的尝试, 本文将把加密流量构建成基因序列, 然后运用同源基因搜索算法, 实现检测包含恶意攻击的加密流量.

1.2 思路介绍

1.2.1 Profile HMM 与加密流量

根据这种 Profile HMM 的结构(如图 2 所示), 我们将解释如何将 Profile HMM 用于将数据包所表现的特征

与模型整合. 我们将流量序列特征考虑为基因序列, 与基因序列相同, 流量特征也会随着时间和空间上的变化而产生改变. 与之类似, 我们知道生物基因会在特定情况下产生变异, 而加密流量的特征也会随着混淆数据的加入而发生变化. 也就是说, 基因序列与加密流量类似, 具有上下文相关性, 关键子序列能够反映整个序列的分类. 同时, 基因子序列存在一定的不确定性, 可以类比于加密恶意流量中的混淆数据包. 然而从生物信息学的角度, 即使存在这些变化, 但其主体功能并未发生实质改变, 从网络安全的角度来说, 一次恶意攻击产生的流量序列必然会有一段或者多段关键序列仍然保留其恶意攻击的特征, 类比生物基因同源分析中, 同源的基因序列通常会带有核心基因片段来溯源其所属的基因家族, 因此本文通过分析加密流量序列中的"关键基因"来帮助检测恶意攻击.

具体地说, 为了允许在同一协议的连接中观察到的数据包序列之间的变化, 该模型对链中的每个位置都有两个额外的状态. 其中一个称为插入状态 Insert, 插入状态可以在一次会话的两个数据包之间按一定顺序插入一个或多个额外的数据包; 另一种称为删除状态 Delete, 删除状态允许从会话序列中删除给定位置的数据包. 因此, 可以理解为插入状态表示重复的包或重发数据包, 而删除状态表示在网络中丢失数据包. Profile HMM 上每个链在加密流量中的每个数据包都对应一种状态, 每个状态发出的符号具有特定于其在链中位置的概率分布. 这些中心链中的状态称为匹配状态, 因为它们的符号发射概率分布与通信过程产生的正常数据包序列相匹配.

1.2.2 同源基因搜索与加密流量检测

由第 1.1.2 节的介绍可以看出, 若存在序列和 HMM 匹配, 则通过该模型的最短路径为 $B\rightarrow M_1\rightarrow\ldots\rightarrow M_n\rightarrow E$. 如果序列包含 Insert 和 Delete 两种状态, 则这条路径可能通过多个插入或删除状态直至匹配模型后到达结束状态 E. 但这种情况在实际情况中, 流量序列未必能从开始状态 B 开始匹配, 然后直至序列尾能够匹配上模型的结束状态 E. 通常, 一条基因可能存在多处变异基因, 类似的, 真实的网络环境中也存在噪声. 对于采取全序列匹配未必能找到所需的基因序列, 因此在同源基因序列搜寻算法中, Profile HMM 一般由多个关键的基因序列域组成, 而不是整个待检测序列的模型. 考虑网络包序列, 包含关键攻击指令的加密流量序列往往被埋在更长的序列中, 因此一段流量序列中可能有多个关键序列. 在这种情况下, 需要使用局部匹配算法来查找待检测序列的子序列是否能匹配 Profile HMM 检测模型.

图 2 展示的是原始的 Profile HMM 结构图, 这种结构只能将一个序列作为一个整体进行匹配, 并不能满足实际流量序列检测的条件. 为了匹配多个子序列, 需要在原始 Profile HMM 中添加额外的状态. 如图 3 所示, 新的 Profile HMM 引入了 5 个新的状态: 新的开始状态 S 和新的结束状态 T 以及 3 个中间状态 N, C 和 J 用于生成不在 Profile HMM 模型内的随机序列. 实现了这一功能, 新的模型就可以实现忽略加密流量中的混淆数据而匹配决定其攻击特征的加密流量序列, 同时避免一次性完成的全局匹配.

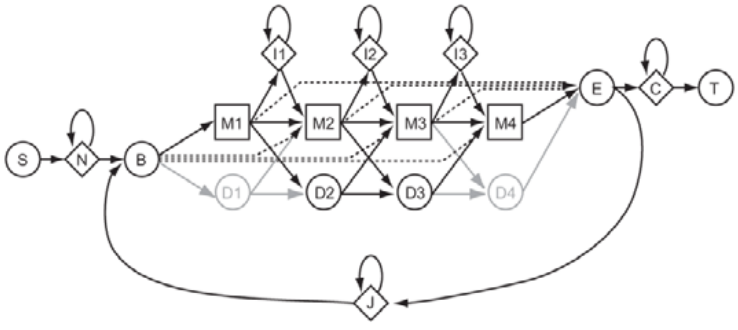

图 3 HMMER 中使用的 Profile HMM 结构图, 该结构可以应用于多段子序列匹配

下面我们介绍模型的参数. 首先, 匹配状态 M、插入状态 I 和删除状态 D 为我们需要关注的关键部分, 这几个状态对应的概率参数需要通过模型训练, 从序列数据中学习得到; 而剩下其他几个状态控制模型的特性

不是从序列中获得的, 需要根据模型的功能配置其参数值. 例如, 状态 E 对应的参数可以用来控制模型是否进行全局匹配, 我们将 $E\to J$ 的条件转移概率设置为 0, 那么模型在匹配完成一次之后会直接结束匹配. 而默认情况下, 这个概率设定为大于0的情况, 则模型可以进行多次匹配得到多条满足条件的序列结果. 在本文实验中, 我们将状态(N,C,J)对应的自循环概率设置为 $\frac{L}{L+3}$, 而控制模型多序列匹配的 $E\to J$ 的转移概率设置为 0.5. 其中, L 为基因序列的长度.

1.3 基于Profile HMM的检测模型

1.3.1 基于 Profile HMM 构建检测模型

图 4 展示了基于 Profile HMM 的加密流量检测模型建立流程. 根据文献[15], 攻击流量相较于正常流量在时间轴上往往表现出脉冲的特征, 而攻击者为了藏匿这种特性, 会采取延时发送、加入混淆数据等方式. 但与此同时, 攻击者对于恶意攻击的时间成本以及数据长度开销增加, 这种隐含的混淆数据模式同样可被用来提高检测效果. 与传统的方法一样, 我们对流量进行抓取和过滤, 以获得相应的数据包序列. 然后对于每个包序列, 我们需要对其进行符号化. 符号化可以减少隐马尔可夫模型中可观察状态的数目, 使模型具有更强的鲁棒性. 经过上述步骤, 我们可以将所有来自同一恶意样本的序列生成一个仅有单一的基因序列的模型. 为了更好地实现训练效果, 可以采用多个序列作为样本进行训练. 最后, 我们通过将检测样本转换成符号序列并与已知的序列进行比较, 以达到检测的目的.

以上是基于 Profile HMM 构建检测模型的具体流程, 我们将进一步解释为什么 Profile HMM 模型适用于检测加密流量. 如图 5 所示, 假设有一段恶意样本的模型, 有一段待检测序列, 其中包含了包含攻击指令的两个序列 1 和 3, 如果检测模型没有利用时间间隔分割流量数据, 而是整个序列进行匹配, 那么模型就会把时间间隔内的所有其他数据都作为关键子序列进行识别. 这样会导致关键子序列被检测模型忽略, 对于攻击者蓄意有间隔的发送混淆数据包的情况几乎无法进行检测. 但我们使用的检测模型并非全序列匹配, 而是通过将序列分段考虑, 将分段发送的数据包作为子序列, 无论攻击者以何种顺序发送数据包, 该模型都可以匹配出对应的关键子序列. 也就是说, 无论攻击者是否加入混淆数据, 该模型都能将恶意样本检测出来.

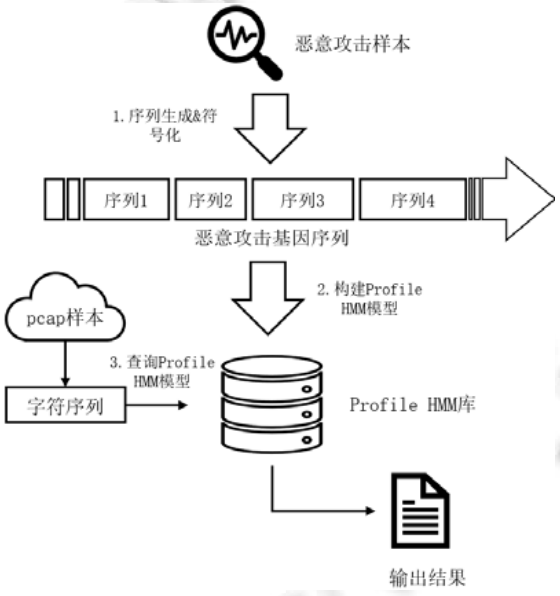

图 4 基于 Profile HMM 的恶意流量检测

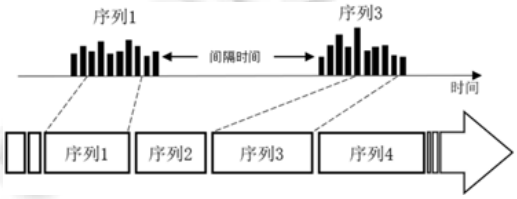

图 5 分析检测模型

1.3.2 模型参数估计

对于一个 HMM, 我们可以令模型为$\lambda=(A,B,\pi)$, 模型由隐藏状态初始概率分布π、状态转移概率矩阵 A 和观测状态概率矩阵 b 共同决定. 而根据第 1.2.2 节的分析, 模型λ需要通过训练已知序列来评估参数, 其中包括匹配状态 M、插入状态 I 和删除状态 D 对应的概率参数. 在模型训练之前, 我们使用多序列对齐(multiple sequence alignment)对训练集进行预处理.

多序列对齐是对两个以上的序列进行比对, 通过在字符之间插入"-"符号, 使每个序列的同源部分在同一列中相互对齐. 使用多序列对齐有以下优势: (1) 多序列对齐能够发现公共子序列, 便于模型的训练和验证; (2) 通过对齐可以提高相似子序列的命中, 以免模型陷入局部最优解; (3) 实验发现, 对齐提高了我们的检测准确率. 如图 6 所示, 我们选取了 3 个来自同一恶意样本流量数据进行序列化后, 得到了 3 条基因序列片段, 图中的字符序列对应了样本流量数据的数据包通信行为. 经过对齐之后, 可以看到, 阴影部分匹配到的公共部分更长. 显然, 这些较长的子序列更加具有关键基因的特性. 也就是说, 数据包经过对齐之后, 可以使攻击行为对应的序列确定在长序列中的某些特定位置, 之后如果出现攻击, 很大概率也会在这些位置出现.

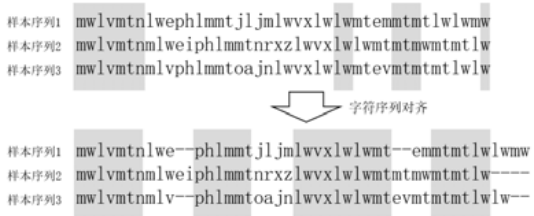

图 6 多序列对齐示例

完成了序列对齐后, 我们就要使用这些序列作为训练数据求解模型参数. 首先, 我们已经定义了模型$\lambda=(A,B,\pi)$, 同时, 我们已知观测序列 $O=o_1,o_2,\ldots,o_3$, 求解模型参数也就是要求观测序列 O 在模型π条件下出现的条件概率 $P(O|\lambda)$. 对于参数的估计, 我们使用 Baum-Welch 算法[27]进行求解.

Baum-Welch 算法的原理基于最大期望(EM)算法, 通过 E-Step 求出联合分布 $P(O,I|\lambda)$基于条件概率 $P(O,I|\bar{\lambda})$ 的期望, 其中, $\bar{\lambda}$ 为当前计算得到的模型参数. 然后, 通过 M-Step 最大化该期望, 得到更新的模型参数λ. 接着, 通过迭代执行 E-Step 和 M-Step, 直到模型参数的值收敛为止. 首先, 对于 E-Step, 当前模型参数为 $\bar{\lambda}$, 联合分布 $P(O,I|\lambda)$基于条件概率 $P(O,I\mid\bar{\lambda})$ 的期望表达式为

$$L(\lambda,\bar{\lambda})=\sum_I P(I\mid O,\bar{\lambda})\log P(O,I\mid\lambda).$$

对于 M-Step, 通过求解上式的最大值, 得到更新后的模型参数如下:

$$\bar{\lambda}=\arg\max\sum_I P(I\mid O,\bar{\lambda})\log P(O,I\mid\lambda).$$

通过迭代 E-Step 和 M-Step, 直到 $\bar{\lambda}$ 收敛. 基于 Baum-Welch 算法, 我们可以求得模型参数$\lambda=(A,B,\pi)$. 至此, 我们的检测模型就可以进行预测, 可以使用 Vertibi 算法来计算和观测序列 O 最匹配的序列.

2 实验过程

2.1 数据获取

本文实验使用开源数据集 CTU-13, CICIDS2017 和 SSL Blacklist 收集得到的流量来验证方法的有效性.

(1) CTU-13: CTU-13 是捷克共和国 CTU 大学在 2011 年捕获的一个僵尸网络流量数据集, 它包含 13 个不同的恶意软件, 捕获环境均基于真实网络. 虽然该数据集相对有些过时, 但包含了许多典型的恶意行为, 如垃圾邮件、点击欺诈等仍具有研究价值. 全部 13 个数据集中共包含 20 643 076 个网络流量, 其中, 使用加密协议且带有标记的数据为 10 615 条.

(2) CICIDS2017: CICIDS 的数据集包含截至 2017 年的良性和最新的常见攻击, 与真实世界的数据

(PCAPs)类似. 数据集基于 HTTP, HTTPS, FTP, SSH 和电子邮件协议构建了 25 个用户的抽象行为. 收集了共计 5 天的数据, 实现的攻击包括暴力破解 FTP, 暴力破解 SSH, DDoS 等等. 我们从数据集中提取了 28 605 条与加密协议相关的流量数据.

(3) SSL Blacklist: SSL Blacklist (SSL BL)数据集收集恶意 SSL 相关的带有恶意性的证书、域名等. 此外, 还提供了 Suricata 的规则集, 以使用它们提供的恶意 IP 地址来识别恶意软件. 表 1 展示了我们从中收集的 16 种家族的恶意软件, 其中包含了有 1 605 个恶意样本.

表 1 加密恶意流量各个家族数据量

恶意软件家族	样本数	样本流量数据量
IcedID	201	8 251
CobInt	33	4 687
Gozi	185	7 671
OrcusRAT	153	6 972
Adwind	52	4 937
AsyncRAT	47	5 640
Ostap	137	6 839
TA505	38	4 468
FindPOS	177	7 490
TinyNuke	115	6 003
ServHelper	25	3 528
PsiXBot	94	5 586
AZORult	61	5 309
PredatorStealer	103	7 195
NetWire	47	4 115
PandaZeus	137	6 898

对于 CTU-13 和 CICIDS2017 直接提供 pcap 包, 以及较为详细的标注, 我们仅处理空数据包以及过滤加密协议相关的流量. 而对于提供恶意样本的数据集, 我们使用 Hybrid-analysis 根据收集的恶意软件 md5 抓取对应的 pcap 包. 如表 1 所示, 我们共收集了 16 种家族的恶意软件, 他们均使用了加密协议进行攻击. 正常数据流量我们使用随机采样的方式在实验过程中插入到训练集和验证集中, 以模拟现实环境中的随机性. 经过以上处理, 我们共得到了 29 539 条网络流(netflow)数据, 扩展出了 485 518 条数据流量, 其数据包大小和分布情况如图 7 所示.

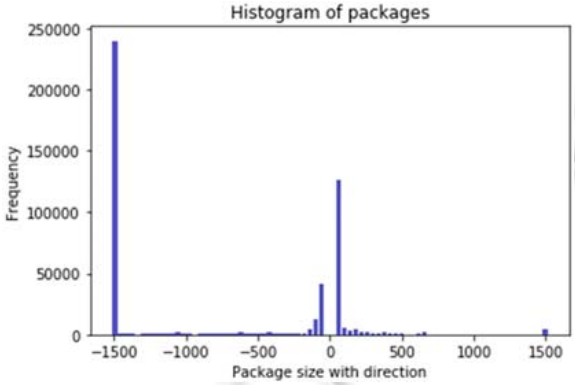

图 7 数据包大小和方向分布情况

2.2 数据预处理和建立模型

在获取记录恶意软件行为的数据包后, 我们考虑对数据进行符号化, 需要设计一种映射算法, 将样本流量数据的数据包通信行为转换为可以被匹配的字符序列.

首先, 我们需要将一个恶意样本的数据进行序列化. 具体实验中, 采用 MapReduce[28]的方式将一定时间内产生的加密流量进行聚合, 得到一组数据包序列. 这组数据包记录了恶意样本利用网络加密协议进行恶意

攻击的通信行为.

接着, 对于一个加密流量的数据包, 我们可以获取数据包的大小和方向. 考虑到数据包大小最大为 1 500 字节, 且存在发送和接受两种方向的数据包, 则存在 1500×2=3000 种可能性. 若对这 3 000 种情况分别进行符号化, 过多符号会导致模型训练过程较长, 且泛化能力较差. 因此, 我们采取分组符号化的方式, 通过设定 *capacity*, 将这 3 000 种情况分在 3000÷*capacity* 个组中. 显然, *capacity* 的值过小会导致模型训练过程较长, 且泛化能力较差; 反之, 会导致检测准确率大幅降低.

在确定了单个数据包如何符号化之后, 我们需要循环处理所有数据包, 以完成这个数据预处理的过程. 此外, 我们对一些边界条件进行约束. 对于过长的数据流, 我们根据第 1.2.2 节定义的基因长度 L 进行截断; 对较短的数据包, 我们通过"-"占位符插入到序列末尾. 具体算法如图 8 所示.

Input: 加密流量数据包序列 P_{encry}, 基因最大长度 L
Output: 符号序列 $Sequence$

```
1  Seq = {}, Out = −1, Size = 0;
2  for packet p ∈ P_encry do
3      if Size < L then
4          if p is an outgoing packet then
5              p → length = p → length × Out;
6          else
7              p → length = p → length × In;
8          Seq = Seq + Symbolize(p → length);
9          Size = Size + 1;
10 while Seq → length < L do
11     Seq = Seq+"-";
12     Seq → length = Seq → length + 1;
13 return Seq
```

图 8 加密流量序列符号化算法

算法中, *Symbolize*(*p*→*length*)表示根据我们设定的 *capacity* 将长度为 *length* 的数据包转换成对应分组的符号, 为了便于后续分析, 我们采用双字母作为每个分组的符号. 例如, 大小为 1 字节的接受数据包落在+0, +1,…,+*capacity* 分组中, 其分组标识我们根据字母按照 aa,ab,...的顺序分配, 若设定 *capacity*=10, 则对应分组为 *fl*, 即该数据包经过符号化后映射为字符 *fl*.

本实验中, 我们使用 MAFFT[29]进行多序列对齐. 文献[30]比较了常用的对序列对齐算法, 包括 CLUSTAL OMEGA, Probcons, T-Coffee, MAFFT, Probalign 等, 结果表明, MAFFT 在准确性和效率上有优势, 且对于长序列处理能力较好, 因此我们选择该算法进行序列对齐. 而对于 Profile HMM 的模型建立, 我们借助于 HMMER 的工具包. 目前虽有较多开源的 Profile HMM 模型训练程序, 实际操作中发现可扩展性较差, 而选择 HMMER 对模型接口的支持程度更高, 便于通过调用工具来实现模型训练和验证.

图 9 显示了具体实现序列对齐、模型训练和检测的过程. 训练集和验证集都以符号化基因序列的形式储存为 fasta 格式文件[31]. 通过工具包调用接口, 可以实现模型训练和模型查询, 返回的结果为一个概率向量, 使用 *softmax* 确定被检测的加密流量对应标签.

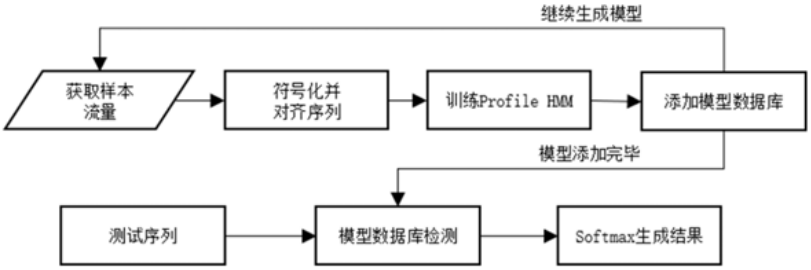

图 9 检测方法具体流程

2.3 评估方法

我们采取十折交叉验证的方式, 将每个恶意样本产生的流进行随机化处理, 然后将其划分为 10 份分别加入分割后的数据集中. 在每一轮训练时, 我们选取其中一个数据集作为验证集, 其余作为训练集以训练检测模型. 此外, 我们在训练过程中对训练数据同样采取随机选取的操作, 从每个样本中选择100个符号序列进行模型的生成. 这样做的目的是提高训练阶段的不确定性, 挑战模型的泛化能力. 经过训练集和验证集的轮换, 我们重复地进行 10 次实验, 最终得到的检测率为这 10 次实验的平均值.

本文设计了多组对比实验以观察不同参数和实验条件对结果的影响, 包括设置不同的序列长度 L、是否使用多序列对齐预处理数据包经过映射后的符号序列. 此外, 本文还将对方法在传统非加密流量中的检测效果进行实验, 并与一些传统方法的效果进行对比, 以讨论本方法的适用范围.

为了能更好地评估实验效果, 本文将会使用如下指标: 准确率(*accuracy*)、精度(*precision*)、召回率(*recall*)、综合评价指标(*F-measure*)以及 *AUC*(area under the curve of ROC). 我们给定一系列标记如下.

1) 真实情况为正例且预测结果为正例的样本为 *TP*(true positive).
2) 真实情况为正例且预测结果为反例的样本为 *FN*(false negative).
3) 真实情况为反例且预测结果为正例的样本为 *FP*(false positive).
4) 真实情况为反例且预测结果为反例的样本为 *TN*(true negative).

可计算得到上述指标:

$$Precision=\frac{TP}{TP+FP},$$

$$Recall=\frac{TP}{TP+FN},$$

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN},$$

$$F=\frac{\alpha^2+1}{\alpha^2}\cdot\frac{Precision\cdot Recall}{Precision+Recal}.$$

需要说明的是, 在 *F-measure* 的实验测量中, 我们令参数α=1, 即实际使用的综合评价指标为 $F1$. 另外, *AUC* 的值需要通过计算 ROC 曲线下方面积得到.

3 实验结果

3.1 效果评估

我们首先对 16 种恶意软件进行分类, 如图 10 显示了每种恶意软件的检测准确率, 经过十折交叉验证法得到平均后的恶意程序检测准确率为 91.51%. 实验结果表明, 多序列对齐对准确率有一定的提升. 但多序列对齐本身依赖额外消耗计算资源, 当基因序列长度 L 被设定为一个很大的值时, 需要花费较多时间成本. 考虑到模型训练的过程相互独立, 因此我们借助了分布式计算框架, 通过并行处理不同组的序列加速这一步骤.

如图 11 所示, 我们以同样的方式对基因序列长度 L 设定不同的值进行实验, 结果表明, 序列长度对准确

率有一定影响. 根据我们前文对模型的分析, 较长的字符序列能够匹配到更多的子序列, 因此模型会提高其对序列的匹配程度. 而我们通过设定参数值消除了模型陷入这种局部最优的情况. 因此, 可以理解选择较长的序列长度L并不一定会导致最后准确率的提升. 此外, 我们在上一个实验中分析了序列长度 L 的长度越长, 时间训练和验证的成本越高, 因此, 我们后续以表现最好的 L=200 进行其余对比实验.

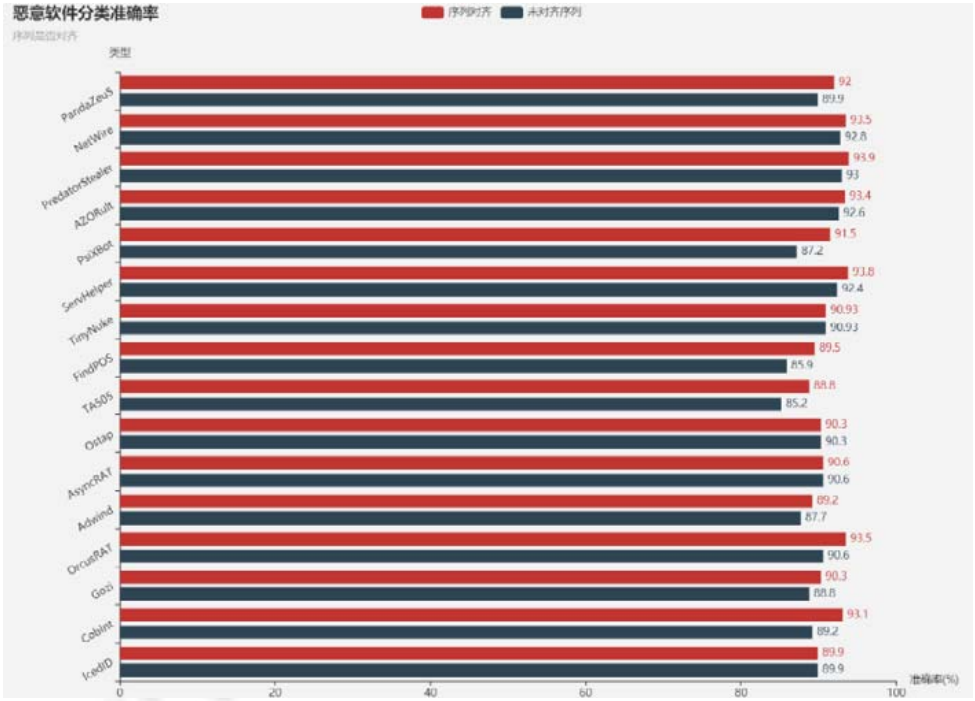

图 10　序列对齐对准确率的影响

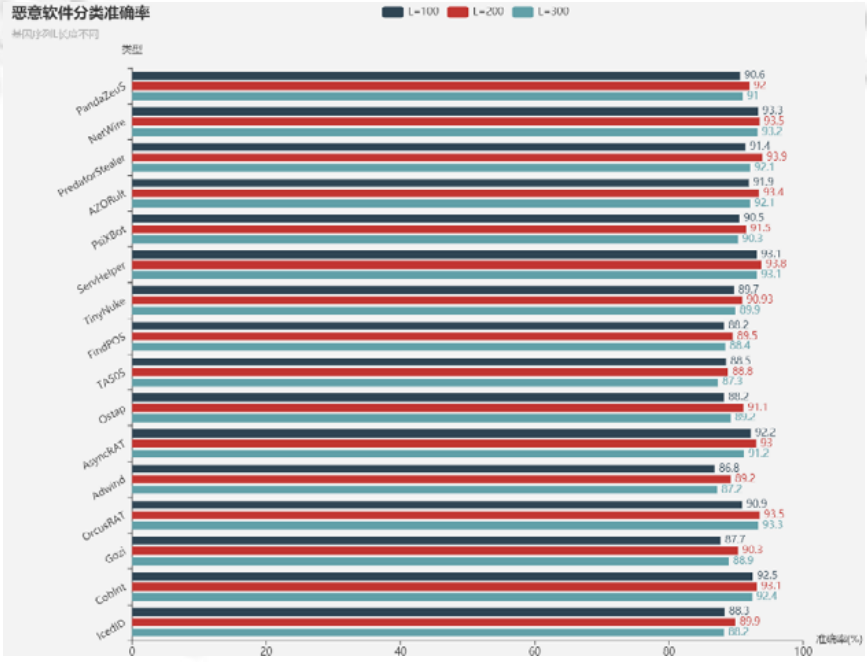

图 11　基因序列长度对准确率的影响

另外, 我们注意到图 7 中有相当一部分 1 500 字节的数据包, 即最大传输单元(maximum transmission unit, MTU). 观察到这些数据包通常在整个基因序列中占据较多部分, 使得较多包含这种数据包的序列有较多相似处, 增加了整体匹配的难度以及无意义的匹配长度. 在实验中, 我们对包括 MTU 在内的较为特殊的数据包进行处理, 限制了出现频率过高的数据包出现次数, 将其产生序列的长度随机缩减为原来的 10%. 图 12 显示了检测恶意软件生成的混淆矩阵, 可以看到, 我们的方法可以较为准确地区分恶意软件.

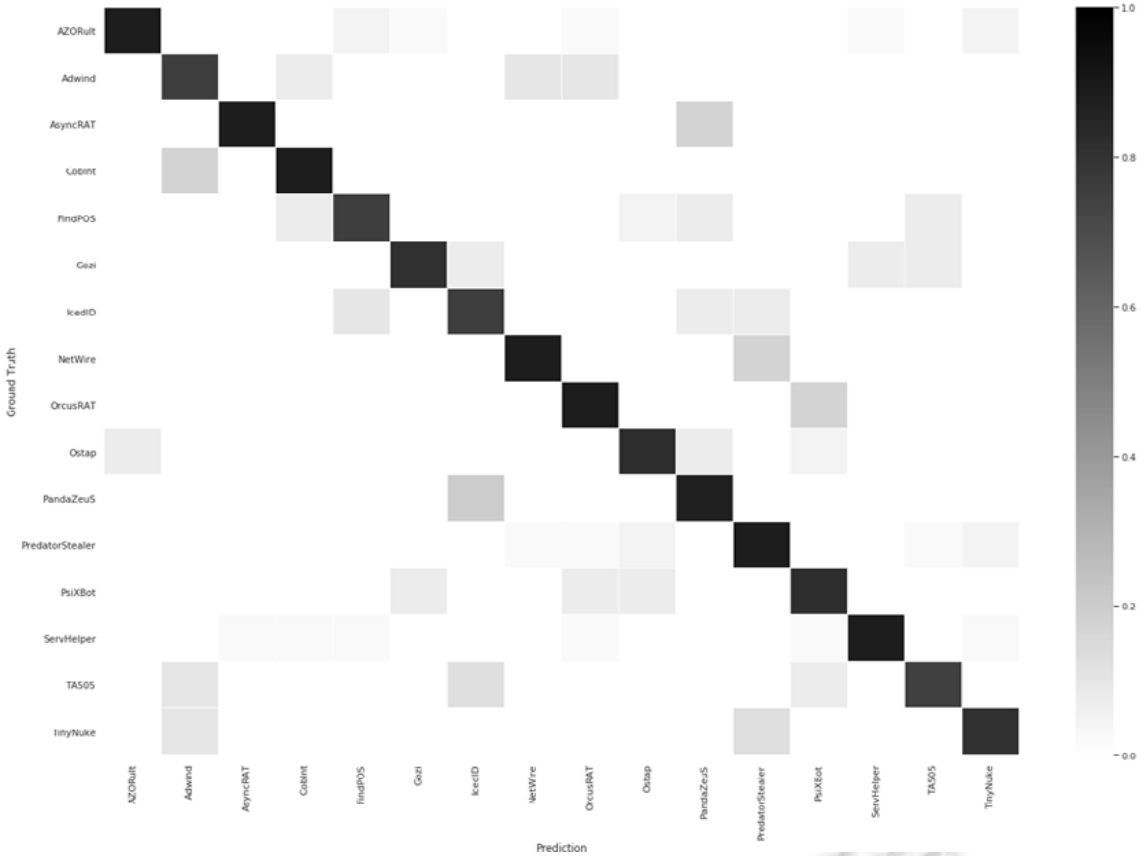

图 12 恶意软件混淆矩阵

3.2 算法效果对比

为了评估我们所提出的方法与其他常见的机器学习方法在同样序列条件下的结果, 我们使用完整的数据集来评估识别准确率. 由于有些机器学习算法仅支持解决二分类问题, 为了便于应用其他算法, 我们采取将分类标签改为两类标签: 恶意和良好, 用作区分恶意软件产生的流量和正常流量. 实验中, 共选取了 3 种较为常用的机器学习分类算法, 分别是随机森林(random forest, RF)、支持向量机(support vector machine, SVM)以及多层感知器(multilayer perceptron, MLP).

下面我们简要介绍这 3 种算法.

(1) RF 算法: 随机森林算法是基于决策树的一种集成算法. 随机森林将每个决策树作为一个分类器训练(假定为解决分类问题), 其最终结果为每棵树得到的分类结果进行投票后得到的得分最高的结果, 从而解决了单棵决策树容易产生过拟合的问题.

(2) SVM 算法: SVM 是通过寻求最小结构化风险来提高泛化能力. 该算法通常为二分类模型, 可以被定义为特征空间上间隔最大的线性分类器. 具体来说, 就是支持向量机的学习策略是让两个类之间的距离最大, 最终转化为一个凸二次规划问题求解. 分类器模型利用求解出的 n 维(n 的大小等于特征

的维数)空间中满足条件的一个超平面, 然后根据点在超平面两侧空间上的位置进行分类.

(3) MLP 算法: MLP 算法是一种前向结构的人工神经网络, 映射一组输入向量到一组输出向量. MLP 源于对大脑神经网络的理解, 将类比于大脑的突触信号通过计算单元模拟. 在 MLP 结构中至少包含 3 层节点, 除了输入节点外, 每个节点都是一个带有非线性激活函数的神经元. 此外, 由于网络是全连接的形式, 在第 N 层的每一个节点 n_i 以权重 w_{ij} 连接下一层第 N+1 节点 n_j. MLP 算法利用反向传播的方式逐步调整每层权重 w_{ij} 以求解更小的损失函数, 若结果收敛, 则可得到一个分类器模型.

图 13 显示了我们采取十折交叉验证的方式对比上述 3 种算法对于恶意软件产生的加密流量的分类结果, 最终产生的结果为这 10 次结果的平均值. 可以看到, 在提供相同完整的流量数据包序列进行训练时, 我们的方法准确率高于其他几种机器学习算法. 这是因为我们使用的检测模型考虑了前后子序列的内在联系, 通过将序列分成若干子序列, 无论攻击者以何种顺序发送数据包, 模型都可以匹配出对应的关键子序列, 从而捕捉到了加入混淆数据包的恶意流量.

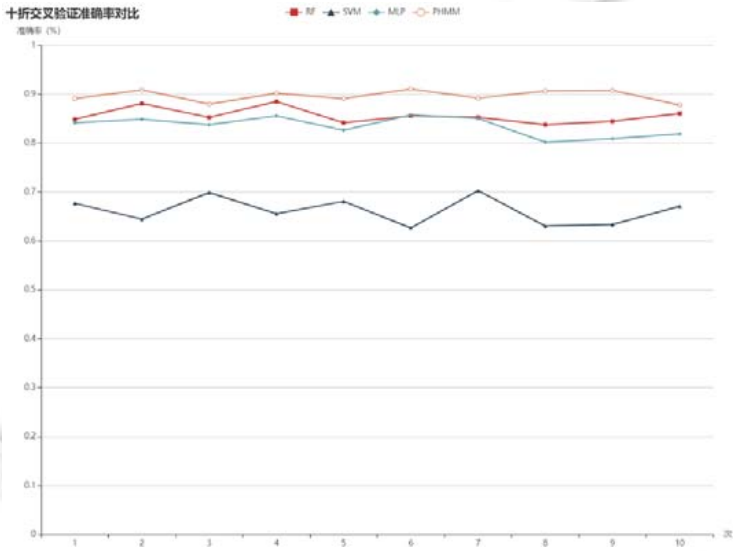

图 13 十折交叉验证实验对比

3.3 规避检测讨论

根据我们提出的利用生物信息学中基因序列的检测方法, 这种方法与使用协议无关, 不依赖于数字证书的特征, 因此可以应用于使用其他网络加密技术的流量检测中. 我们通过不同情况下的实验证明了方法的可行性. 由于这种方法对数据包大小的依赖程度很高, 规避检测的方法可以考虑采取数据包填充的方式. 在实验中, 我们发现数据流中有大量最大传输单元(MTU), 这导致了许多符号化后的序列有中间很大一部分重复的符号, 导致算法在识别中认为长序列的最大传输单元产生的符号为同一种基因家族, 使得检测结果变差. 因此, 我们考虑模拟这种现象, 将数据包填充到最大传输字节数(通常为 1 500 字节), 这样就可以欺骗我们的检测算法, 导致检测准确率降低. 另一方面, 我们的模型具备匹配多段子序列的能力, 而对于一次恶意攻击产生的流量序列, 其中必然会有一段或者多段关键序列仍然保留其恶意攻击的特征, 我们的算法能够识别这种情况. 也就是说, 对于混淆数据包同样具有防御能力. 但考虑一种极端情况, 如果混淆数据包远远多于原始数据包, 我们需要测试算法对于这种情况的宽容度, 以判断在何种极端情况下, 算法会丢失找到关键性基因的能力从而导致检测能力下降. 基于以上两方面, 我们有针对性地提出了两种方法对算法进行逃避检测.

(1) 数据包填充: 将数据包填充至最大传输字节数.

(2) 混淆数据包注入: 伪造数据包插入数据包序列中, 其方向和大小均随机出现.

为了能够直观地显示这两种攻击方式对算法检测能力的影响程度, 我们根据流量序列本身的长度及数据

包大小, 按一定比例修改其数据包大小或是插入一定数量的伪造数据包. 例如, 我们从序列中随机选择 10%的数据包, 将其大小修改为 1 500 字节, 或者我们选择在序列中随机位置插入原序列长度 10%数量的随机大小方向的数据包. 图 14 显示了数据包填充和混淆数据包注入对于算法检测能力影响的效果. 从图 14 可以看出, 30%内填充或注入对算法性能影响很小; 在超过 50%填充或注入后, 也能有 60%以上的准确率. 说明模型具有较好的反逃避检测能力.

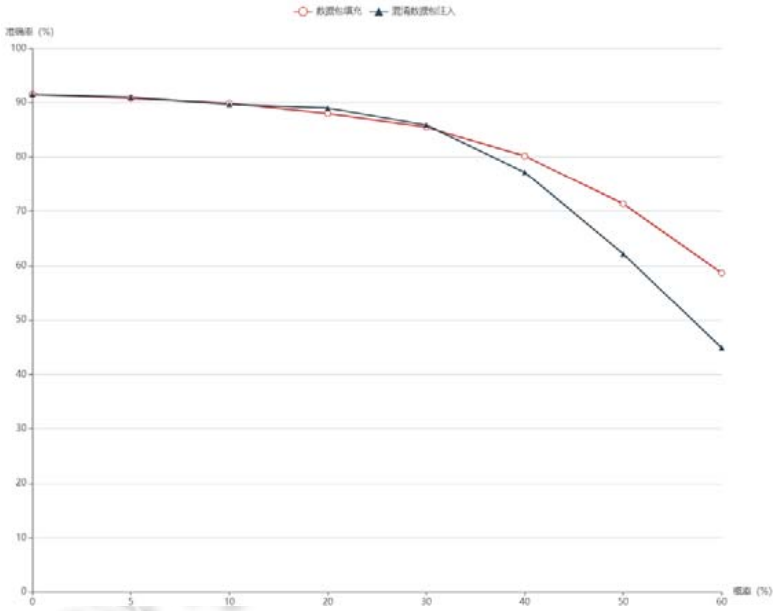

图 14 算法抗逃避检测效果

3.4 算法性能对比

当前, 在恶意加密流量检测领域的比较前沿的工作是思科公司. 他们在这个领域做了一系列的研究工作, 比较有代表性的是发表在 KDD2017 的采用机器学习检测加密恶意流量工作[15]. Anderson 等人提供了他们的方法中用到的流量分析工具 joy[32]. Joy 可以用来处理 pcap 文件, 从中提取 ssl/tls 的证书相关信息, 比如证书的签发者、证书扩展、证书加密套件等; 也会提取流量相关信息, 比如前 n 个数据包的大小以及方向等. 采用 joy 工具流量数据处理后, 再采用文献[15]中所提到的最优算法即随机森林算法进行分类检测. 为了对比, 实验采用第 3.1 节所用的 SSL Blacklist 恶意软件流量数据集以及良性流量数据集, 使用 70%作为训练集, 30%作为测试集, 经过十折交叉验证法得到平均后的恶意程序检测准确率结果为 97.81%. 由于文献[15]的算法只是区分良性/恶意的二分类检测, 我们也修改我们的检测方式由多分类为二分类, 得到的平均后的恶意程序检测准确率为 98.27%. 我们的结果略优于文献[15]的算法.

3.5 检测开销讨论

为了评估检测开销, 我们针对 576 个 pcap 流量文件进行处理, 共计 403 MB, 包括了 48 万个各种类型包、8 698 个 TCP 流. 采用的实验机器是 DELL R730, 具有 2 个 10 核 20 线程 Intel Xeon E5-2630 v4 的 CPU, 主频为 2.20 GHz, 内存为 32 GB. 我们的检测开销时间包括流量预处理时间、特征提取时间以及模型分类时间. 实验测试得到预处理耗时 25.32 s, 特征提取耗时 6.21 s, 流量分类时间耗时 0.042 s. 因此, 检测总开销时间为 31.572 s, 由此得到我们的检测算法每秒能够处理 275 个 TCP 流、15 203 个包. 从流量处理速率来看, 检测总开销时间为 31.572 s, 处理 403 MB 流量, 达到 102 Mb/s. 也就是说可以支持 100 Mb/s 左右的流量, 这个处理速率对于一般中小企业大致可用. 上述实验中, 我们的处理操作是串行进行的, 还可以对数据包进行并行化执行, 检测性能可以得到进一步的提升. 此外, 我们的检测也支持分布式处理, 如在对较大流量的高速网络环

境中, 就可以通过对流量镜像到多个处理机器, 进行分布式协同处理.

4 结 论

本文主要针对使用加密网络通信协议进行攻击的恶意软件检测技术进行研究, 提出了一种基于 Profile HMM 的检测算法. 通过调查现有僵尸网络的协议情况, 发现恶意软件使用的加密协议除了 HTTPS 以外, 还会有大量的 SSH, Tor 等加密协议, 而非加密协议中 HTTP 使用占比最高. 为了能够应对这种不同协议下的加密恶意流量检测场景, 本文提出了一种基于 Profile 隐马尔可夫模型(Profile HMM)的恶意加密流量检测算法. 该方法仅使用了网络数据包信息进行分析, 利用生物信息学上的基因序列分析来实现检测. 本文在开源数据集 SSL Blacklist, CTU-13, CICIDS 上进行了不同条件的实验, 对 16 种恶意软件进行分类, 经过十折交叉验证, 得到了平均后的恶意程序检测准确率为 91.51%. 结果表明了算法的有效性. 最后, 本文还给出了针对检测算法的规避方法, 并通过实验验证了算法具有较好的抗规避的能力. 综上所述, 本文提出的方法具有检测时间相对较短、准确率相对较高且不依赖于协议的广泛应用场景.

References:

[1] Barmpatsalou K, Cruz T, Monteiro E, Simoes P. Current and future trends in mobile device forensics: A survey. ACM Computing Surveys (CSUR), 2018, 51(3): 1−31.

[2] Yaacoubi O. The rise of encrypted malware. Network Security, 2019, 2019(5): 6−9.

[3] Anderson B, Paul S, Mcgrew D. Deciphering Malware's use of TLS (without decryption). Journal of Computer Virology and Hacking Techniques, 2018, 14(3): 195−211.

[4] Rossow C, Dietrich CJ. ProVeX: Detecting botnets with encrypted command and control channels. In: Proc. of the 10th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Heidelberg: Springer, 2013. 21−40.

[5] Antonakakis M, April T, Bailey M, *et al*. Understanding the Mirai botnet. In: Proc. of the 26th USENIX Security Symp. (USENIX Security 2017). 2017. 1093−1110.

[6] Gu G, Perdisci R, Zhang J, *et al*. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proc. of the Conf. on Security Symp. USENIX Association, 2008. 139−154.

[7] Zhang XB, Lam SS, Lee DY, *et al*. Protocol design for scalable and reliable group rekeying. IEEE/ACM Trans. on Networking, 2003, 11(6): 908−922.

[8] Nguyen TTT, Armitage G. A survey of techniques for Internet traffic classification using machine learning. Communications Surveys & Tutorials, 2008, 10(4): 56−76.

[9] Dainotti A, Pescape A, Claffy KC. Issues and future directions in traffic classification. Network IEEE, 2012, 26(1): 35−40.

[10] Namdev N, Agrawal S, Silkari S. Recent advancement in machine learning based Internet traffic classification. Procedia Computer Ence, 2015, 60: 784−791.

[11] Schwenk G, Rieck K. Adaptive detection of covert communication in HTTP requests. In: Proc. of the 7th European Conf. on Computer Network Defense. IEEE, 2012. 25−32.

[12] Bortolameotti R, Caselli M, *et al*. Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In: Proc. of the 33rd Annual Computer Security Applications Conf. 2017. 373−386.

[13] Perera P, Tian YC, Fidge C, *et al*. A Comparison of Supervised Machine Learning Algorithms for Classification of Communications Network Traffic. In: Proc. of the 24th Int'l Conf. on Neural Information Processing. Cham: Springer, 2017. 445−454.

[14] Kumano Y, Ata S, Nakamura N, *et al*. Towards real-time processing for application identification of encrypted traffic. In: Proc. of the 2014 Int'l Conf. on Computing, Networking and Communications (ICNC). IEEE, 2014. 136−140.

[15] Anderson B, McGrew D. Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity. In: Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2017. 1723−1732.

[16] Bernaille L, Teixeira R. Early recognition of encrypted applications. In: Proc. of the Int'l Conf. on Passive and Active Network Measurement. Berlin, Heidelberg: Springer, 2007. 165−175.

[17] Bacquet C. Genetic optimization and hierarchical clustering applied to encrypted traffic identification. In: Proc. of the Computational Intelligence in Cyber Security. IEEE, 2011. 194−201.

[18] Bar-Yanai R, Langberg M, Peleg D, *et al*. Realtime classification for encrypted traffic. In: Proc. of the Int'l Symp. on Experimental Algorithms. Berlin, Heidelberg: Springer, 2010. 373−385.

[19] Zhang M, Zhang H, Zhang B, *et al*. Encrypted traffic classification based on an improved clustering algorithm. In: Proc. of the Int'l Conf. on Trustworthy Computing and Services. Berlin, Heidelberg: Springer, 2012. 124−131.

[20] Conti M, Mancini LV, Spolaor R, *et al*. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In: Proc. of the 5th ACM Conf. on Data and Application Security and Privacy. 2015. 297−304.

[21] Shen M, Wei M, Zhu L, *et al*. Classification of encrypted traffic with second-order Markov chains and application attribute bigrams. IEEE Trans. on Information Forensics and Security, 2017, 12(8): 1830−1843.

[22] Korczyński M, Duda A. Classifying service flows in the encrypted skype traffic. In: Proc. of the 2012 IEEE Int'l Conf. on Communications (ICC). IEEE, 2012. 1064−1068.

[23] Fahad A, Alharthi K, Tari Z, *et al*. CluClas: Hybrid clustering-classification approach for accurate and efficient network classification. In: Proc. of the 39th Annual IEEE Conf. on Local Computer Networks. IEEE, 2014. 168−176.

[24] Yu TD. Research on detecting malware with encrypted traffic [MS. Thesis]. Shanghai: Shanghai Jiao Tong University, 2020 (in Chinese with English abstract).

[25] Pan YT, Lin L. A trust-based DDoS discovery approach for encrypted traffic in cloud environment. Journal of Computer Research and Development, 2021, 58(4): 822−833 (in Chinese with English abstract).

[26] Krogh A, Brown M, Mian IS, *et al*. Hidden Markov models in computational biology: Applications to protein modeling. Journal of Molecular Biology, 1994, 235(5): 1501−1531.

[27] Ghahramani Z, Jordan MI. Factorial hidden Markov models. Machine Learning, 1997, 29(8): 245−273.

[28] Zaharia M, Xin RS, Wendell P, *et al*. Apache spark: A unified engine for big data processing. Communications of the ACM, 2016, 59(11): 56−65.

[29] Katoh K, Asimenos G, Toh H. Multiple alignment of DNA sequences with MAFFT. In: Proc. of the Bioinformatics for DNA Sequence Analysis. Humana Press, 2009. 39−64.

[30] Pais FSM, de Cássia Ruy P, Oliveira G, *et al*. Assessing the efficiency of multiple sequence alignment programs. Algorithms for Molecular Biology, 2014, 9(1): 1−8.

[31] Pearson WR. Using the FASTA program to search protein and DNA sequence databases. In: Proc. of the Computer Analysis of Sequence Data. Humana Press, 1994. 307−331.

[32] McGrew D, Anderson B, Hudson B, Perricone P. joy. 2017. https://github.com/cisco/joy

附中文参考文献:

[24] 俞汤达. 面向加密流量的恶意软件检测研究 [硕士学位论文]. 上海: 上海交通大学, 2020.

[25] 潘雨婷, 林莉. 云环境下一种基于信任的加密流量 DDoS 发现方法. 计算机研究与发展, 2021, 58(4): 822−833.

邹福泰(1973－), 男, 博士, 高级工程师, CCF 高级会员, 主要研究领域为网络威胁检测与防御, 恶意代码分析.

俞汤达(1995－), 男, 硕士, 主要研究领域为加密恶意流量检测.

许文亮(1997－), 男, 硕士, 主要研究领域为暗网, 加密恶意流量检测.