计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20180481
56(6):1263-1274,2019

一种基于域名请求伴随关系的恶意域名检测方法

彭成维[1,2] 云晓春[1,2,3] 张永铮[2,3] 李书豪[2,3]
[1](中国科学院计算技术研究所 北京 100190)
[2](中国科学院大学 北京 100049)
[3](中国科学院信息工程研究所 北京 100093)
(pengchengwei@iie.ac.cn)

Detecting Malicious Domains Using Co-Occurrence Relation Between DNS Query

Peng Chengwei[1,2], Yun Xiaochun[1,2,3], Zhang Yongzheng[2,3], and Li Shuhao[2,3]
[1](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)
[2](*University of Chinese Academy of Sciences, Beijing* 100049)
[3](*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093)

Abstract Malicious domains play a vital role in illicit online activities. Effectively detecting the malicious domains can significantly decrease the damage of evil attacks. In this paper, we propose CoDetector, a novel technique to detect malicious domains based on the co-occurrence relationships of domains in DNS (domain name system) queries. We observe that DNS queries are not isolated, whereas co-occur with each other. We base it design on the intuition that domains that tend to co-occur in DNS traffic are strongly associated and are likely to be in the same property (i.e., malicious or benign). Therefore, we first perform coarse-grained clustering of DNS traffic based on the chronological order of DNS queries. The domains co-occurring with each other will be clustered. Then, we design a mapping function that automatically projects every domain into a low-dimensional feature vector while maintaining their co-occurrence relationships. Domains that co-occur with each others are mapped to similar vectors while domains that not co-occur are mapped to distant vectors. Finally, based on the learned feature representations, we train a classifier over a labeled dataset and further apply it to detect unknown malicious domains. We evaluate CoDetector using real-world DNS traffic collected from an enterprise network over two months. The experimental results show that CoDetector can effectively detect malicious domains (91.64% precision and 96.04% recall).

Key words DNS queries; co-occurrence; malicious domains; DNS cut; tensor representation; domain classification

摘 要 恶意域名在网络非法攻击活动中承担重要的角色.恶意域名检测能够有效地减少攻击活动所带来的经济损失.提出 CoDetector 恶意域名检测模型,通过挖掘域名请求之间潜在的时空伴随关系进行恶意域名检测.研究发现域名请求之间存在彼此伴随关系,而并非相互独立.因此,彼此伴随的域名之间存在紧密关联,偏向于同时是正常域名或恶意域名.1)利用域名请求的先后时间顺序对域名数据进行

收稿日期:2018-06-27;**修回日期**:2018-12-10
基金项目:国家重点研发计划项目(2016YFB0801502);国家自然科学基金项目(U1736218)
This work was supported by the National Key Research and Development Program of China (2016YFB0801502) and the National Natural Science Foundation of China (U1736218).
通信作者:云晓春(yunxiaochun@iie.ac.cn)

粗粒度的聚类操作,将彼此伴随出现的域名划分到同一簇中;2)采用嵌入学习构建映射函数,在保留域名伴随关系的同时将每一个域名投影成低维空间的特性向量;3)结合有标记的数据,训练恶意域名检测分类器,用于检测更多未知恶意域名.实验结果表明,CoDetector 能够有效地检测恶意域名,具有 91.64%检测精度和 96.04%召回率.

关键词 域名请求;请求伴随;恶意域名;时间序列切割;向量化表示;域名分类

中图法分类号 TP391

域名系统(domain name system, DNS)是当今互联网中重要的基础核心服务之一,负责提供统一的域名地址空间映射服务,主要将易于人类记忆的域名翻译为易于机器识别的 IP 地址.然而,伴随着域名系统提供正常服务的同时,近年来越来越多的网络非法活动也开始滥用域名系统以达到其恶意目的.例如,僵尸网络利用域名生成算法(domain generate algorithm, DGA)批量生成大量用于僵尸网络命令与控制(command & control, C&C)信道通信的域名,来逃避权威安全防御机构的封杀和屏蔽[1-2].网络诈骗犯注册外观极其相似于知名合法域名(如 alipay.com)的新域名(如 allpay.com),并搭建钓鱼网站来欺骗网络用户,达到窃取账户信息、信用卡密码等目的[3].2016 年思科年度安全报告[4]中指出,高达 91.3%的恶意软件均会对域名系统进行一定程度的滥用,造成大量的经济损失.

近年来出现了大量利用 DNS 流量检测恶意域名的研究工作,其主要目的是为了在用户访问这些恶意域名之前进行防御和预警,从而缓解攻击活动带来的威胁和损害.常见的方法[5-7]是从 DNS 流量、网页信息等数据中为每一个域名人为手动提取特征(例如 TTL(time to live)大小、域名请求模式、解析的 IP 地址数目、涉及到的国家等),随后利用机器学习算法构建分类器.然而,这类基于特征的检测方法能够有效地检测恶意域名的前提是提取特征的有效性,即能否有效地区分黑白域名的行为,并且攻击者不去修改恶意域名的行为来规避这些特征.实际上,之前提出的很多特征已被证实不具有很好的鲁棒性,攻击者可以通过简单地调整来改变这些特征,从而逃避检测.

本文提出 CoDetector 算法,一种基于域名请求之间内在的时空伴随关系(co-occurrence relation)进行恶意域名检测的算法.本文发现域名请求之间不是彼此独立,相反存在时空相似、伴随共现的特性.针对一次域名查询,触发这次查询的底层应用程序同时会触发其他相关的域名查询,这些域名相互伴随出现,协同完成此次网络活动.例如使用浏览器打开链接,如 http://www.baidu.com 会先触发对该页面域名 www.baidu.com 的 DNS 查询,当网页内容开始呈现时,则会进一步触发对页面嵌入内容(如图片、广告等)的链接进行额外的 DNS 查询.这些域名请求是为了共同完成这次页面展示而触发的请求,在 DNS 请求上表现为伴随共现的特性.于此同时,本文发现恶意域名请求之间同样存在时空伴随关系,不同的恶意域名会在一次恶意网络活动中共同伴随地出现.例如一次路过式下载(drive-by download)行为通常由一个长长的 URL 重定向链条来逐步导向到最终的恶意软件.网络黑产中,恶意的搜索引擎优化技术通过构建重复循环的 URL 链条,让搜索引擎的爬虫持续停留在被敌手设计的页面中.

本文的假设是伴随出现的域名之间存在紧密的关联,性质上具有同态性,即和恶意域名伴随出现的域名偏向于是恶意域名,反之亦然.基于以上假设,本文提出 CoDetector 恶意域名检测算法,其主要思路是从 DNS 流量中挖掘域名之间时空伴随关系,然后借鉴深度学习中张量化表示算法(如 word2vec[8])在保留域名彼此时空伴随关系的基础上将每一个域名映射为低维空间的特征向量,最后结合机器学习分类算法构建恶意域名检测分类器.为实现以上目标,本文主要面临 2 个挑战:

1) 如何从 DNS 流量提取彼此具有时空伴随关系的域名?

2) 如何在保留域名彼此时空伴随关系的同时张量化地提取域名特征?

针对挑战 1,本文提出基于域名请求时间间隔的切割算法,将原始 DNS 流量中的 DNS 请求根据时间顺序切割成不同长度的时空伴随域名序列.本文认为在时间上同时触发的域名请求彼此存在伴随关系,存在明显的时间间隔的请求则不具有伴随关系.实际上,通过对真实 DNS 数据分析,本文发现每个用户所触发的 DNS 请求在时间上存在明显的分块

聚簇现象,即 DNS 请求一批接着一批触发,不同批次之间存在明显时间间隔.例如在打开下一个网页之前,会在上一个网页停留一定的时间,从而导致这 2 次页面行为触发的域名请求之间也存在明显时间间隔.因此,如果 2 个域名请求之间的时间间隔大于给定的阈值(例如 5 s),则很有可能是不同网络活动所触发的 DNS 请求,本文便将这 2 个域名划分到不同序列中,反之则划分到同一个序列中.每一个序列有 1 个或者多个域名组成,近似认为是由一次网络活动所触发的 DNS 请求集合,彼此具有时空伴随关系.

针对挑战 2,本文借鉴 Skip-Gram[9] 的词向量表示算法,将每一个域名投影成 d 维实数空间中的一个点(向量),目标是使得具有伴随关系的域名映射成空间中距离相近的点;反之,把不具有伴随关系的 2 个域名映射到空间中相隔较远的位置.考虑到挑战 1 中提取的时空伴随域名序列可能会存在噪声干扰和由序列长度过长带来的计算复杂度问题,本文采用滑动窗口的方式,进一步从时空伴随域名序列中提取具有时空伴随域名对.每个时空伴随域名对由 2 个域名组成,彼此之间具有时空伴随关系,同时,本文采用负采样技术[8]构建不具有时空伴随关系的域名对作为负样本数据,最后结合优化目标迭代地调整每一个域名在 $\mathbb{R}^d$ 空间的位置.优化结束后,本文便可得到每一个域名特征向量.

与前人工作对比,本文工作主要有 3 点不同之处:

1) CoDetector 自动地从 DNS 流量中挖掘潜在的域名伴随关系,并映射为特征向量,无需人工专家经验,省去人工设计特征的繁杂工作.

2) CoDetector 仅利用域名请求的时间顺序挖掘伴随关系,无需额外的附加信息.因此,本文的方法更加轻量实时,能够处理不具有正常应答的恶意域名,例如 DGA 生成的 NXDomain 域名.

3) CoDetector 将具有伴随关系的域名进行了聚类,因此可以用于恶意域名团伙发现.

最后,本文采集一个企业网下近 2 个月的 DNS 流量数据,结合 3 种主流的机器学习分类算法(随机森林、XGBoost 和深度神经网络)来评估 CoDetector 模型的检测效果.实验结果表明,CoDetector 平均能够达到 91.64%的检测精度和 96.40%的召回率.因此,CoDetector 能够有效地通过 DNS 流量挖掘域名请求之间时空伴随关系,并用于检测恶意域名.

总体来说,本文工作具有 4 点贡献:

1) 提出一种基于时间间隔的域名序列切割算法,能够有效地从 DNS 流量中提取具有伴随关系的域名序列.

2) 提出一种无监督的域名张量化表达算法,能够将每一个域名映射为低维空间的特征向量并且保留域名彼此之间的伴随关系.

3) 提出一种基于域名请求之间时空伴随关系的恶意域名检测算法——CoDetector.该模型自动地从原始 DNS 流量挖掘潜在的域名时空伴随关系,并用于检测恶意域名.

4) 结合真实 DNS 数据,对 CoDetector 的可行性和有效性进行评估,实验验证 CoDetector 能够有效地检测恶意域名.

1 相关工作

近年来出现了大量利用 DNS 流量检测恶意域名的研究工作.Antonakakis 等人[5]提出了 Notos 动态域名打分系统.Notos 主要提取 3 种类型的特征:1)基于网络位置的特征(历史上与域名关联的 IP 数量、地理位置的多样性、它们驻留的不同自治系统的数量等);2)基于域名 Zone 文件的特征(如域名不同 gram 分布的平均长度、不同顶级域名的数量、字符频率等);3)基于历史证据的特征(如域名解析的 IP 地址中有多少曾经在恶意样本中出现、该域名是否在蜜罐系统中捕获等).Bilge 等人[6]提出了 Exposure 恶意域名检测系统.Exposure 克服了 Notos 的部分限制,它能够识别之前从未在恶意活动中看到的恶意域名和地址,并且只需要较短时间的训练数据.Bilge 等人从 DNS 流量中为每一个二级域名提取了 15 个特征,其中包括新颖的基于时间的特征(短生命周期、每日访问相似度、重复模式、访问成功的比率等)、基于 DNS 应答的特征(不同 IP 地址的数量、不同国家的数量、共享 IP 地址的域名的数量、反向 DNS 查询结果)、基于 TTL 值的特征(TTL 的平均值、TTL 的标准偏差、不同的 TTL 值的数量、TTL 变化的数量)和基于域名词法的特征(数字字符的百分比和最长有意义的子字符串的标准化长度).Antonakakis 等人[7]提出 Kopis 系统.与 Notos 和 Exposure 相比,Kopis 使用了 DNS 系统中更高层次域名流量数据(顶级域名服务器和权威域名服务器采集到的 DNS 流量).因此,Kopis 从更加全局的角度来提取每一个域名的行为,其中包括基于请求来源分布的统计特征(例如与递归 DNS 服务器相关联的 IP 地址的多样性)和请求来源在每个时期结束时向给定域名的 DNS 流量,递归 DNS 服务

器的相对查询量以及与域名指向的 IP 空间相关的历史信息.这些方法主要是通过针对每一个域名的提取局部特征,并利用机器学习分类器来构建检测模型.可能存在的问题是:如果这些检测方法(特征)被敌手了解之后,很容易通过合理的调整来规避这些特征,逃过检测.本文提出的 CoDetector 模型是考虑了域名之间的潜在时空伴随关联特性.如果敌手想要绕过 CoDetector 的检测,需要消除其使用的恶意域名之间的时空伴随特性.例如每次只使用一个域名,然而这极大地降低了恶意活动的灵活性.

Khalil 等人[10]提出利用域名和 IP 之间的全局关联来检测恶意域名.其主要思路是利用(域名、IP)映射数据构建域名关联图.如果 2 个域名映射到相同的 IP,则在 2 个域名之间添加关联边.随后,在域名关联图使用基于图上路径搜索的机制来推断未知域名的恶意分数.Peng 等人[11]提出了一种基于 DNS CNAME 记录的恶意域检测方法,该方法专注于没有解析到 IP 地址,但出现在 DNS CNAME 记录中的域名.该方法是基于 CNAME 记录连接的 2 个域名存在内在紧密关系,并且偏向于处于同样的性质(即同时是恶意域名或者同时是正常域名).他们提出一种基于置信传播的图推理方法,通过计算未知域名与其他已知恶意和正常域名的全局关联来计算其恶意概率.他们的实验结果表明,该方法可以有效揭露被以往研究工作所忽略的恶意域名.Manadhata 等人[12]提出了一种基于图推断方法的恶意域名检测模型.该模型首先利用企业网络中的 DNS 数据构建了一个主机-域名二分图,其中如果一台主机访问了某个域名,便在该主机和该域名之间添加关联边,最后利用置信传播(belief propagation)算法在图上进行推断.文献[13]提出了利用 DNS 服务器与用户之间通信的 DNS 数据来构建域名主机二分图,其主要假设是访问恶意域名的机器更有可能是感染的机器,反过来感染的机器访问的域名也更有可能是恶意域名.然而,上述方法利用 DNS 服务器与用户之间的 DNS 数据,会带来隐私问题.文献[14]提出利用 DNS 数据检测长周期下 APT 中隐蔽可疑的 DNS 行为.

本文并不是第 1 个提出利用域名请求伴随关系来检测恶意域名的研究工作.Gao 等人[15]提出了一种基于域名请求伴随关系来检测恶意域名团伙的算法.该工作利用部分已知的恶意域名作为种子,统计域名和种子伴随出现的关系,采用 TF-IDF 算法和 XMeans 聚类算法来提取域名团伙.然而,该工作首先未考虑域名请求之间的时间间隔,因此会将属于不同网络活动触发的域名划分到一个团伙中,存在严重的噪声干扰.本文提出的模型能够自动地挖掘域名的伴随关系,且具有线性的时间复杂度,适合在大规模数据上运行.

2 CoDetector 检测模型

本文发现主机层面的域名请求之间存在伴随共现关系,并非互相独立.因此,通过考虑一个域名经常和哪些域名伴随出现能够有效地协助决策该域名是否是恶意域名.本文的基本假设是具有时空伴随关系的域名之间存在紧密的关联,在性质上(恶意性质或者正常性质)具有同态性.具体来说,如果一个域名经常和恶意域名伴随共现,那么该域名偏向于是恶意域名,反之亦然.CoDetector 模型首先从 DNS 流量中提取彼此具有时空伴随关系的域名,形成时空伴随域名序列集合;随后利用深度学习算法在保留域名彼此伴随关系的同时将每个域名投影成低维实数空间的特征向量;最后结合部分已知的黑白域名,利用机器学习分类算法构建恶意域名检测分类器,用于识别未知的恶意域名.

图 1 展示了 CoDetector 检测模型的工作流程,其主要分为 3 个模块.

1) 时空伴随域名序列提取模块.利用域名请求的先后时间顺序将原始域名数据进行粗粒度的聚类,划分为不同的序列,使得每一个序列中的域名彼此具有时空伴随关系.

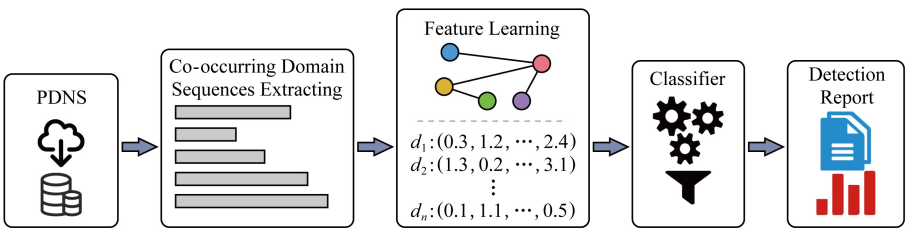

Fig. 1 The workflow of CoDetector

图 1 CoDetector 模型工作流程图

2) 特征学习模块.构建映射函数 $f:D\to\mathbb{R}^d$ 将每一个域名 $x\in D$ 映射为 d 维空间的特征向量 $\boldsymbol{v}_i$,同时保留域名之间的时空伴随关系,即将具有伴随关系的域名映射到 $\mathbb{R}^d$ 空间中距离相近的点,把不具有伴随关系的域名映射到距离遥远的点.

3) 恶意域名检测模块.利用特征学习模块中学习到的每个域名的特征向量,结合部分黑白域名列表,采用有监督的机器学习算法训练恶意域名分类器,并应用于对更多未知属性的域名进行检测.

详细介绍每一个模块的设计,最后给出模型时间复杂度分析.

2.1 时空伴随域名序列提取

本模块首先针对原始 DNS 流量进行粗粒度的聚类操作,将彼此具有伴随关系的域名划分在同一个域名簇中,形成时空伴随域名序列结合.通过对真实的 DNS 流量分析,本文发现每个用户的 DNS 请求具有明显分块聚簇的特点,在时间顺序上存在聚类的现象.用户的域名请求一批接着一批发起,不同批次之间的 DNS 请求具有明显的时间间隔.

图 2 展示了本文采集到的部分用户在 10 min 时间段内的域名请求时间散点分布图,x 轴为时间,图 2 中每个点代表一次 DNS 请求,不同直线代表不同用户的 DNS 请求随着时间的散点分布.本文发现,每一个用户的域名请求在时间线上呈现成块行为.例如用户 u_7,u_8,u_9 的域名请求数据存在明显的聚簇现象,同簇之间的请求几乎同时发起,簇与簇之间存在明显时间间隔.因此,本文近似地按照域名请求时间进行粗粒度的聚类,认为每簇中的域名彼此伴随出现,形成一个时空伴随域名序列.

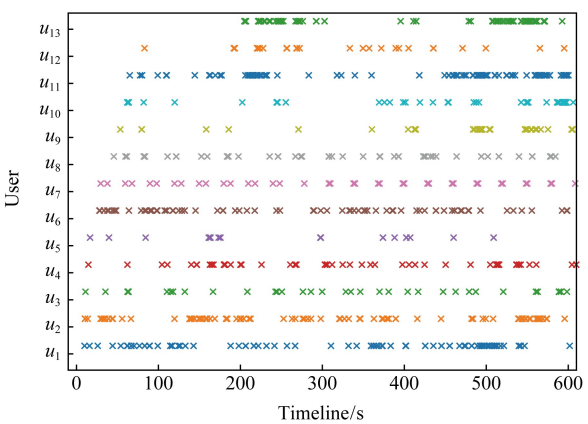

Fig. 2 The scatter of DNS queries in 10 min

图 2 10 min 内部分 DNS 数据请求散点图

假设 $\mathbb{D}$ 是通过 DNS 服务器监听到的 DNS 流量数据,其可以分解为不同终端用户的 DNS 请求集合,即 $\mathbb{D}=\bigcup_i D_i$,其中 $D_i=\{(d_i^1,t_i^1),(d_i^2,t_i^2),\cdots,(d_i^{n_i},t_i^{n_i})\}$ 为用户 u_i 请求域名的集合,(d_i^j,t_i^j) 代表用户 u_i 在时间 t_i^j 请求域名 d_i^j,$t_i^1\leqslant t_i^2\leqslant\cdots\leqslant t_i^{n_i}$.基于 DNS 请求的成块聚簇现象,本文提出一种基于时间间隔的域名时空伴随序列切割算法.具体表现是:如果相邻的 2 个域名请求之间,$\{(d_i^j,t_i^j),(d_i^{j+1},t_i^{j+1})\}$ 的时间间隔 $\Delta t=t_i^{j+1}-t_i^j$ 大于切割阈值 τ(如 5 s),本文便将这 2 个域名请求划分到不同的序列中,反之则划分到同一个序列中.图 3 展示了利用时间间隔切割域名数据的简单示例图.最终,将所有用户的域名数据切割形成的时空伴随域名序列即为从 DNS 流量 $\mathbb{D}$ 中提取的时空伴随域名序列集合.

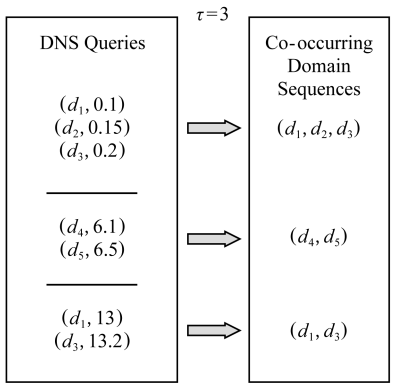

Fig. 3 An illustration for generating co-occurring domain sequences

图 3 时空伴随域名序列生成说明图

2.2 特征学习

假设 $\mathbb{S}=\{S_1,S_2,\cdots,S_n\}$ 为提取到的时空伴随域名序列集合,其中 $S_i=\{d_1,d_2,\cdots,d_{n_i}\}$ 是由 n_i 个域名构成的序列.本模块的目标是在保留域名彼此之间的伴随关系的同时,将每一个域名映射成低维空间的特征向量.本文把映射操作形式化为似然概率最大化问题:

首先,每一个时空伴随域名序列 S_i 的概率 $P(S_i)=Pr(d_1,d_2,\cdots,d_{n_i})$ 为这 n_i 个域名联合出现的概率.因此,映射操作的基本优化目标是最大化序列集合 $\mathbb{S}$ 的概率,即:

$$\max P(\mathbb{S})=\max\prod_{i=1}^{n}P(S_i). \tag{1}$$

然而,最大化该目标函数在现实计算中存在问题,尤其是当序列长度 n_i 过大而带来的联合概率计算呈指数增长的复杂度.近年来,Skip-Gram[9] 语言模型在自然语言处理问题上取得了极大的成功,其主要思想是通过一个单词来预测其上下文出现的

单词.启发于 Skip-Gram 模型,本文提出基于滑动窗口的思想来将长度为 n_i 的序列 S_i 分解为多个短小的子序列,即只保留域名和其窗口内域名的时空伴随关系,而忽略其和更远位置的域名关系,从而极大地降低了计算复杂度.本文假设:

$$P(S_i)\propto\prod_{j=1}^{n_i}Pr(C(d_j))=\prod_{j=1}^{n_i}\prod_{c\in C(d_j)}p(d_j,c),\quad(2)$$

其中,$C(d_j)$为域名 d_j 的上下文.假设窗口大小为 w,则 $C(d_j)=(d_{j-w},d_{j-w+1},\cdots,d_{j-1},d_{j+1},d_{j+2},\cdots,d_{j+w})$,更进一步分解为 $2w$ 组具有时空伴随关系的域名对$\{(d_j,d_{j-w}),(d_j,d_{j-w+1}),\cdots,(d_j,d_{j-1}),(d_j,d_{j+1}),(d_j,d_{j+2}),\cdots,(d_j,d_{j+w})\}$.

图 4 展示了利用滑动窗口从时空伴随序列 S 生成时空伴随域名对的示意图,其中该序列 S 是由 7 个域名组成,滑动窗口 $w=2$.采用滑动窗口的方式将时空伴随域名序列分解成时空伴随域名对的操作能够带来 2 点改进:1)滑动窗口的引入成功地消除了由于序列长度过长而带来的计算复杂度增长的问题,使得在现实中计算成为了可能;2)滑动窗口能够有效地减少噪声伴随关联(将本身不具有时空伴随的域名错误地认为具有时空伴随关系).窗口滑动的方式保证了每一个域名只和其固定窗口大小内的域名具有伴随关系,保证了域名伴随关系的质量.

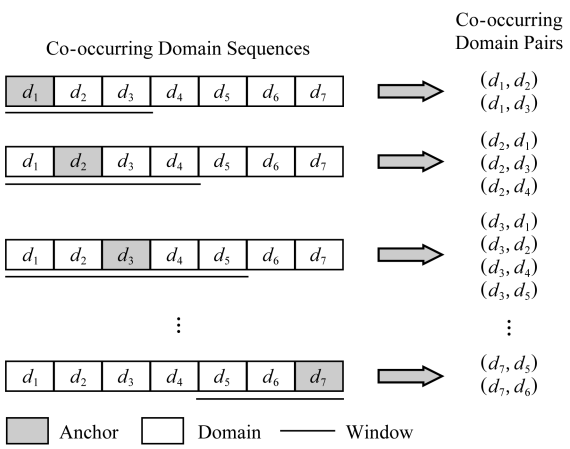

Fig. 4 An illustration for generating co-occurring domain pairs ($w=2$)

图 4 基于滑动窗口($w=2$)的时空相似域名对生成示意图

基于以上改进操作,优化目标函数式(2)为

$$\log P(S)\propto\sum_{i=1}^{n}\sum_{d\in S_i}\log[Pr(C(d))]=\sum_{i=1}^{n}\sum_{d\in S_i}\sum_{c\in C(d)}\log p(d,c)=\sum_{(d_i,d_j)\in\mathcal{P}}\log p(d_i,d_j),\quad(3)$$

其中,$\mathcal{P}$ 为基于滑动窗口操作提取的所有具有时空伴随关系的域名对集合.

假设 $f:D\to\mathbb{R}^d$ 为映射函数,将每一个域名 $d_i\in D$ 投影成实数空间$\mathbb{R}^d$中的一个向量 $\boldsymbol{v}_i$.本文采用 Sigmoid 函数来衡量这 2 个域名 d_i 和 d_j 的联合概率,即

$$p(d_i,d_j)=Sigmoid(\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_j)=\frac{1}{1+\exp(-\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_j)}.\quad(4)$$

更进一步,由于式(3)中只考虑了具有时空伴随关系的域名对之间的映射关系,而忽略了将不具有时空伴随关系的域名投影成$\mathbb{R}^d$空间中距离较远的向量,因此一个通用的映射方式是将全部域名映射到一个点上.为此,本文采用负采样技术[8]构建不具有时空伴随关系的域名对,并最小化这部分域名对的联合概率.所以,最终本文的目标函数是:

$$\min[-\sum_{(d_i,d_j)\in\mathcal{P}}\log p(d_i,d_j)+\sum_{(d_i,d_j)\in\mathcal{N}}\log p(d_i,d_j)],\quad(5)$$

其中,$\mathcal{P}$ 是所有具有时空伴随关系的域名对集合,$\mathcal{N}$ 是通过负采样技术生成的不具有时空伴随关系的域名对集合.本文通过随机梯度下降(stochastic gradient descent, SGD)[16]算法来最小化目标函数.迭代结束之后,本文便得到每个域名 d_i 在$\mathbb{R}^d$空间中的特征向量 $\boldsymbol{v}_i$.

2.3 恶意域名检测模块

通过最小化式(5),本文将$\mathbb{S}$ 中每个域名表达成 d 维空间的特征向量.本文通过提前采集的黑白域名列表匹配 $\mathbb{S}$ 中的域名,从而获取部分有标签的数据,再结合机器学习分类算法构建恶意域名检测分类器,用于检测未知的恶意域名.

2.3.1 训练阶段

结合域名黑白名单,匹配数据集中出现的域名,本文获得部分标记的恶意域名和正常域名.利用成熟的有监督的机器学习分类算法(如随机森林),结合这部分域名在特征学习阶段学习到的特征向量,最终得到恶意域名检测分类器 $\pi:\boldsymbol{v}\to s$,其中 $\boldsymbol{v}$ 为域名对应的特征向量,$s\in[0,1]$为该域名的恶意打分.

2.3.2 检测阶段

针对其余部分未知属性的域名,本文利用训练阶段学习的分类器进行分类,从而检测未知的恶意域名.对于域名待检测域名 d_i,假设其在特征学习阶段对应的特征向量为 $\boldsymbol{v}_i$,则其恶意打分为 $s_i=\pi(\boldsymbol{v}_i)$.如果 $s_i>0.5$,则判定域名 d_i 为恶意域名,其分数越高,则代表域名 d_i 为恶意域名的可能性越大.

2.4 模型时间复杂度分析

CoDetector 共分为 3 个阶段.

阶段 1. 将 DNS 流量切割,按照采集到的 DNS 的先后顺序根据时间便可以切割,因此时间复杂度为 $O(n)$,其中 n 是 DNS 流量中域名请求的数量.

阶段 2. 可以细分为 3 个部分:

1) 利用滑动窗口的方式生成时空伴随域名对,针对每一个域名,最多形成 $2w$ 个伴随域名对,因此,本文最多共有 $2wn$ 个域名对,且时间复杂度为 $2w\times O(n)$.

2) 利用负采样技术生成和正样本同数量级的负样本,假设针对每一个域名通过随机的方式(如 Hash 采样)生成 K 个负样本(本文中 $K=5$),因此,最多有 $2Kwn$ 个负样本域名对,且时间复杂度为 $K\times 2w\times O(n)$.

3) 式(5)可以分解为对每一个时空伴随域名对进行优化调整.结合正负样本,最多有 $2w\times(K+1)\times n$ 个域名对.针对每一个时空伴随域名对优化迭代一次的复杂度为 d,假设迭代 M 次,那么式(3)最终的复杂度为 $M\times 2w\times(K+1)\times d\times O(n)$.

由于 M,K,w,d 均远小于 n,因此第 2 阶段的时间复杂度为 $O(n)$.

阶段 3. 利用已知的黑白域名和其在 $\mathbb{R}^d$ 空间的特征向量来训练模型分类器.其复杂度与分类器算法的选取和黑白域名的样本数量有关系.例如,本文选择随机森林算法作为分类器算法,共 m 棵树,训练样本数量为 N,则复杂度为 $O(m\times d\times N\log N)$,由于 N 为训练样本数量,相对于全部域名数量 n 是很少的一部分,因此这部分的复杂度相对于第 1 阶段和第 2 阶段均为常数.

因此,CoDetector 模型的计算量主要集中在第 1 阶段和第 2 阶段中,综合起来复杂度为 $O(n)$,其中 n 是 DNS 流量中域名请求的数目.

3 实 验

3.1 数据集

1) 域名黑名单.本文通过采集网络公开的黑名单列表来构建本文中的域名黑名单,其中包括 Malware Domains List[17],Phishtank[18],Openphish[19],AbuseList[20].本文从 2017-01-03—2017-10-14,持续不断地收集这些来源的黑名单,最后保留去重的域名列表.除此之外,本文还采集了宙斯(Zeus)病毒中使用的恶意域名和著名的蠕虫病毒飞客(conficker)[21]中通过 DGA 算法生成的恶意域名.这些黑名单列表包含了形式多样的恶意域名,如僵尸网络命令与控制通道的域名、偷渡式下载域名、网络钓鱼、垃圾邮件、网络诈骗勒索等域名,能够很好地覆盖各种不同类型的网络攻击.更进一步,为了保证本文中使用的黑名单列表的可靠性,本文采用 Google Safe Browsing[22] 来进一步针对从 Phishtank 和 Openphish 采集到的域名做 2 次筛选,只有当 Google Safe Browsing 也认为该域名是恶意域名,本文才将该域名保留.

2) 域名白名单.本文根据 Alexa[23] 每天提供的全球访问量最多的 100 万域名(例如 google.com)列表来构建本文的白名单域名列表.本文筛选那些长时间(如 1 年)持续排名在 Alexa Top 20000 的域名作为本文的正常域名.通过持续排名的条件可以有效地删除噪声域名,例如僵尸网络的域名在攻击活动发生时会存在短暂的访问量爆发,从而有可能出现在 Top 20000 列表中.实验中,本文收集了 2015-01-16—2017-03-05 共计 513 天的 Alexa Top 100 万域名列表,本文共发现 9216 个域名持续地出现在这 513 天的 Top 20000 域名中.

3) DNS 数据.本文中使用的 DNS 数据集是在一个企业网内部网关捕获的 2 个月(2017-10-13—2017-12-18)的 DNS 流量,其中共包含 12 291 055 条 DNS 请求数据包.表 1 列举了本文的实验数据,其中包括采用基于时空时间的切割算法生成的时空伴随域名序数目、基于滑动窗口生成的时空伴随域名对数目,以及匹配到的正常域名和恶意域名的数目.

Table 1 Description of Experimental Data

表 1 实验数据描述

τ	w	Co-occurring Domain Sequences	Co-occurring Domain Pairs	Malicious Domains	Benign Domains
2	1	1 490 994	4 862 854	9 445	3 036
2	2	1 490 994	7 470 464	9 447	3 036
2	3	1 490 994	9 822 889	9 447	3 036
3	1	1 414 711	5 029 952	9 445	3 048
3	2	1 414 711	7 724 753	9 457	3 048
3	3	1 414 711	10 161 202	9 457	3 048
5	1	1 311 523	5 236 281	9 458	3 059
5	2	1 311 523	8 042 945	9 460	3 059
5	3	1 311 523	10 589 250	9 460	3 059

3.2 评价指标

为了定量化地衡量 CoDetector 模型对恶意域名检测的效果,本文采用 3 个指标.

1) 召回率(Recall, R).测试集中所有的恶意域名样本,CoDetector 能成功识别出恶意域名的比例.

2) 检测精度(Precision, P).在测试集数据中,CoDetector 判定为恶意域名的样本中确实是恶意域名的比例.

3) F-值(F-Measure, F).F-值是召回率和精度这 2 个指标综合形成的指标,能有效地权衡这 2 个指标.其中 $F=2\times\frac{R\times P}{R+P}$,$F$-值越大,则代表模型整体检测效果越好.

3.3 域名伴随关系分析

CoDetector 模型的基本前提是域名请求之间具有时空伴随关联关系,即在用户层的 DNS 数据中域名请求的时间顺序并非完全随机分布,而是存在伴随关系.给定域名全体集合 X,本文称 φ_d 为域名 d 的伴随分布,即伴随 d 同时出现的域名(在 d 窗口为 w 内的域名)分布.如果 d 在 DNS 请求上具备伴随关系,则 φ_d 在 X 上应为非均匀分布,随机性差;反之应近似为均匀分布,随机性强.本文采用香农熵来衡量一个分布函数的随机性.给定一个密度分布函数 $p(x)$,则其香农熵 $H(p)=-\sum_x p(x)\,\mathrm{lb}\,p(x)$.$H(p)$越大,代表分布函数 $p(x)$分布越均匀;反之代表 $p(x)$分布存在偏向性.针对采集的真实 DNS 数据,为保证统计的有效性和降低计算规模,本文统计出现次数在 1 000 次以上的域名,共计 1 315 个域名,其香农熵统计特征如表 2 所示:

Table 2 Statistic Results of Shannon Entropy Under Different Parameters

表 2 不同参数下的香农熵统计结果

τ	w	Mean	Std	Min	Max
2	1	4.98	1.45	0	8.21
2	2	5.22	1.42	0	8.23
2	3	5.39	1.4	0	8.36
3	1	5.09	1.44	0	8.29
3	2	5.34	1.4	0	8.33
3	3	5.52	1.38	0	8.4
5	1	5.23	1.42	0	8.35
5	2	5.5	1.38	0	8.43
5	3	5.69	1.35	0	8.45

表 2 列举了这 1 315 个域名在不同切割时间间隔 $\tau=\{2\,\mathrm{s},3\,\mathrm{s},5\,\mathrm{s}\}$和不同窗口大小 $w=\{1,2,3\}$下生成的伴随分布及其对应的香农熵统计特征.注意到如果伴随分布是安全随机的,则伴随分布的香农熵应该为 lb 1315=10.36.第 1 列代表时间间隔 τ 和窗口大小 w 的选择,第 2～5 列列举了在该组参数下 1 315 个域名香农熵的统计特征(平均值、标准差、最小值和最大值).

在这 9 组参数下,这 1315 个域名的香农熵均远小于 10.36,其中平均值不到 10.36 的一半,最大值约为 10.36 的 80%.数据结果表明:域名的伴随分布并不是均匀分布,而是存在偏向性.因此验证了用户层域名请求之间具有时空伴随关联关系.

3.4 模型假设验证

CoDetector 模型的基本假设是具有时空伴随关系的域名之间具有强关联,偏向同时属于恶意域名或者正常域名,即同态性.实验中,本文首先利用采集到 DNS 流量对模型假设的合理性进行验证.本文主要从 2 个方面进行验证:1)在包含恶意域名的时空伴随域名序列中,其他域名是否更加偏向于是恶意域名;2)在滑动窗口生成的时空伴随域名对中,如果一个域名是恶意域名,另外一个域名是否偏向于是恶意域名.注意本文只对和恶意域名伴随出现的域名的性质分布进行统计来评估同态性.因为本文的主要目的是为了检测恶意域名.

3.4.1 时空伴随域名序列同态性验证

给定切割时间间隔 τ,本文将原始 DNS 流量切割成具有时空伴随关系的域名序列集合$\mathbb{S}$.利用采集到的域名黑名单,本文标定序列中每一个域名是否是恶意域名.本文定义 $q(S)$为序列 S 中恶意域名的个数,记$\mathbb{S}^n=\{S\mid q(S)\geqslant n,S\in\mathbb{S}\}$,为$\mathbb{S}$ 中包含恶意域名个数大于等于 n 的序列子集合.本文记包含恶意域名的时空伴随域名序列为可疑时空伴随域名序列.统计集合$\mathbb{S}^n(n\geqslant 1)$中,恶意域名所占的比例为 δ,如果 $\delta>0.5$ 则说明在$\mathbb{S}^n$中的域名更多的是恶意域名,从而验证假设.选择 3 种不同的时间间隔 $\tau=\{2\,\mathrm{s},3\,\mathrm{s},5\,\mathrm{s}\}$生成时空伴随的域名序列集合,并针对4 种粒度的包含恶意域名的时空伴随域名序列子集,$n=\{1,2,3,4\}$统计恶意域名所占的比例 δ.

图 5 展示了包含恶意域名的时空伴随域名序列中恶意域名所占的整体比例 δ,其中横坐标代表$\mathbb{S}$中的不同子集$\mathbb{S}^1,\mathbb{S}^2,\mathbb{S}^3,\mathbb{S}^4$.首先,注意到所有子集中恶意域名比例均大于 50%,且使用相对较小的时间间隔 τ 切割生成的时空伴随域名序列中恶意域名所含比例越高.另一方面,结果表明随着序列中包含的恶意域名数目增加,序列在整体上是恶意序列的概率也在变大.

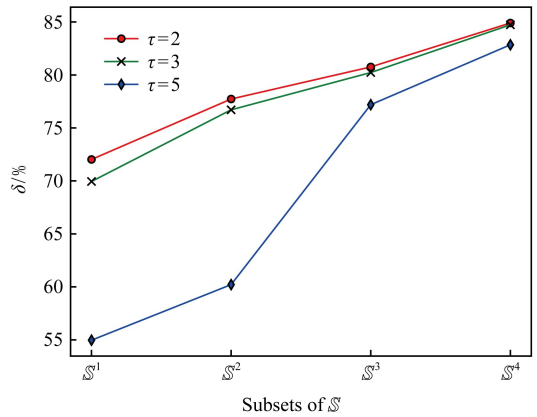

Fig. 5 Homomorphic ratio of malicious property in co-occurrence domain sequences

图 5 伴随域名序列中恶意属性同态比例

3.4.2 时空伴随域名对同态性验证

基于滑动窗口的方式,本文从时空伴随域名序列集合$\mathbb{S}$生成时空伴随域名对集合$\mathcal{P}$.针对集合$\mathcal{P}$中每一个域名对(d_i,d_j),本文统计当域名d_i是恶意域名时d_j同样是恶意域名的比例δ.实验中,本文选择时间间隔参数$\tau=\{2\text{ s},3\text{ s},5\text{ s}\}$,滑动窗口大小$w=\{1,2,3\}$,针对生成的 9 个时空伴随域名对集合$\mathcal{P}$分别计算$\delta$.结果如图 6 所示,其中横坐标$w$为滑动窗口的大小,纵坐标为当域名对$(d_i,d_j)$同时是恶意域名的比例$\delta$.

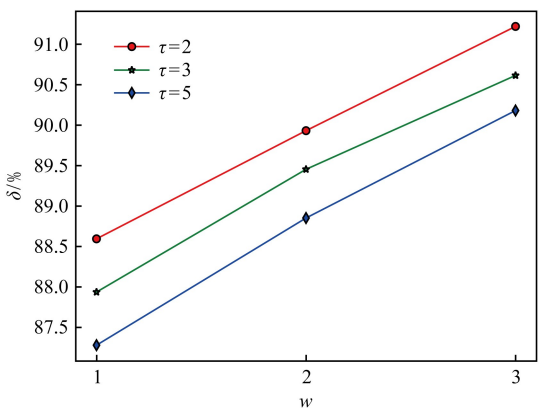

Fig. 6 Homomorphic ratio of malicious property in co-occurrence domain pairs

图 6 伴随域名对中恶意属性同态比例

本文发现:1)在时空伴随域名对中,如果一个域名是恶意域名,另外一个域名也是恶意域名的比例平均为 89.34%,即域名在时空伴随域名对中同样保持同态性;2)通过滑动窗口生成的时空相似域名对能够有效地提高域名之间的同态性.例如当$w=1$时,如果d_i是恶意域名,d_j是恶意域名的概率高达 88%,且随着窗口w的变大,同态性也有所提升.这一结果进一步验证了本文实验的假设,即具有时空相似性的域名之间同时在域名性质上具有同态性.

3.5 模型检测效果

CoDetector 模型涉及到 3 个主要参数:1)序列切割时间间隔τ;2)滑动窗口大小w;3)每个域名的特征的维度d.实验中本文选取$\tau=\{2\text{ s},3\text{ s},5\text{ s}\}$,$w=\{1,2,3\}$和$d=\{100,200,300\}$,共 27 种参数组合.针对每一组参数$\theta=(\tau,w,d)$,本文采用标准的 10-Fold 交叉验证的方式来评估实验效果.首先,本文将训练数据集随机平均地分为 10 份,其中 9 份作为训练集,剩下 1 份作为测试集,获取在测试集上的检测效果.如此重复 10 次,并将 10 次的平均结果作为这组参数的最终实验结果.本实验全部运行在一个具有 8 核 Inter® Core™ I7-6700K CPU@4.00 GHz、内存为 32 GB 的服务器.

本文选用 3 个主流的有监督的机器学习算法作为恶意域名检测的分类器.

1) 随机森林(RandomForest).随机森林是通过集成学习的思想将多棵决策树集成的一种算法.本文利用 scikit-learn[24] 机器学习库来实现随机森林分类器.主要参数设定为:$n\_estimator=200$,$min\_samples\_split=11$.其他参数默认.

2) XGBoost.XGBoost[25] 是一种面向基础的 GradientBoosting 算法的优化版本,具有高效的运行效率、灵活性和可移植性.本文采用其公开的 Python 库来训练本文的分类器模型,参数设置为$max_depth=6$和$num_boost_round=100$,其他参数默认.

3) 深度神经网络(deep neural network, DNN).本文构建了一个 4 层的深层感知机模型.第 1 层是输入层,接受通过时空相似表达模块学习的向量;第 2 层是隐藏层 1,包含nh_1个神经元;第 3 层是隐藏层 2,包含nh_2个神经元;第 4 层是输出层,输出 1 个 2 维的向量,第 1 个元素代表是正常域名的概率,第 2 个元素代表是恶意域名的概率.层与层之间采用全连接网络进行连接,并采用$Relu$非线性激活函数来增加模型的非线性能力.最后 1 层本文采用$softmax$函数来对最终的结果进行归一化.本文结合人为经验设定$nh_1=128$和$nh_2=64$,采用交叉熵损失$cross_entropy$函数来衡量模型的效果.最后,本文通过随机梯度下降 SGD[16] 算法来最小化损失函数.

表 3 列举了 CoDetector 模型在这 27 组参数设置下利用 3 种分类器算法在训练集数据上通过 10-Fold 交叉验证得到的检测效果.RandomForest 平均能够达到 93.97%的 F-Measure、91.80%的精度和 96.30%的召回率.XGBoost 平均能够达到 94.06%的 F-Measure、91.41%的精度和 96.90%的召回率.DNN 平均能够达到 93.85%的 F-Measure、91.76%的精度和 96.05%的召回率.因此,在 3 种不同的分类算法下,CoDetector 模型均能够有效地检测恶意域名.

Table 3 Detail Performance of CoDetector under the 27 Parameter Settings Using 3 Classifiers with 10-fold Cross Validation

表 3 27 组参数下 CoDetector 基于 3 种分类器算法 10-fold 交叉验证的检测效果

Parameters			RandomForest			XGBoost			DNN		
τ	w	d	P/%	R/%	F/%	P/%	R/%	F/%	P/%	R/%	F/%
2	1	100	92.83	96.22	94.49	92.46	96.91	94.63	92.72	96.69	94.66
2	1	200	92.62	96.3	94.43	91.72	97.53	94.54	92.53	96.46	94.46
2	1	300	92.33	96.35	94.3	91.57	97.47	94.42	92.32	96.77	94.49
2	2	100	93.11	96.04	94.55	92.63	97.05	94.78	92.89	96.82	94.81
2	2	200	93.21	96.08	94.62	92.51	97.05	94.72	93.11	96.59	94.82
2	2	300	93.04	95.99	94.49	92.32	97.31	94.75	92.63	96.94	94.73
2	3	100	86.87	97.26	91.77	87.18	96.57	91.63	88.17	94.4	91.17
2	3	200	86.28	97.54	91.56	87.02	95.79	91.19	87.71	92.5	90.03
2	3	300	86.07	97.3	91.34	86.81	95.55	90.97	87.5	92.27	89.81
3	1	100	92.87	96.03	94.42	92.21	97.05	94.57	92.44	96.97	94.64
3	1	200	92.74	96.14	94.41	92	97.38	94.61	92.22	97.07	94.58
3	1	300	92.61	96.16	94.35	91.84	97.43	94.55	91.96	97.08	94.45
3	2	100	93.21	96.23	94.69	92.83	96.9	94.81	92.88	96.49	94.65
3	2	200	86.12	97.07	91.27	86.63	96.05	91.09	87.35	92.29	89.74
3	2	300	85.91	96.84	91.05	86.42	95.81	90.87	87.14	92.06	89.53
3	3	100	93.38	96.21	94.78	92.97	97.01	94.95	93.04	96.91	94.93
3	3	200	93.35	95.76	94.54	92.88	96.91	94.85	92.73	96.88	94.76
3	3	300	93.42	95.94	94.66	92.66	97.11	94.83	93.14	96.46	94.77
5	1	100	92.91	95.89	94.37	92.39	96.93	94.61	92.35	97.03	94.63
5	1	200	92.84	96.17	94.47	92.07	97.11	94.52	92.9	96.29	94.56
5	1	300	92.7	96.27	94.45	91.9	97.15	94.45	92.6	96.76	94.63
5	2	100	93.24	96.08	94.64	92.69	97.03	94.81	92.84	96.98	94.86
5	2	200	93.3	95.87	94.56	92.59	96.87	94.68	92.58	97.06	94.77
5	2	300	93.22	96.08	94.63	92.52	97.21	94.81	92.85	96.7	94.74
5	3	100	93.33	96.18	94.73	93.07	96.82	94.91	92.52	97.1	94.75
5	3	200	93.47	95.86	94.64	92.88	96.89	94.84	92.86	96.74	94.76
5	3	300	93.42	95.76	94.57	92.93	97.01	94.92	93.09	96.65	94.83

4 讨 论

CoDetector 通过挖掘域名请求之间的伴随关系来进行恶意域名判定.因此,对本文的模型可能会存在的问题进行讨论.

1) 噪声伴随.本文通过基于时间间隔的切割算法来获取具有时空伴随的域名序列,然后通过滑动窗口的方式提取时空伴随域名对.其基本的原理是基于域名请求的先后时间顺序,因此后台网络活动触发的 DNS 请求可能会和当前的网络活动触发的 DNS 请求在时间上存在重叠,从而带来噪声干扰.然而,噪声干扰是属于小概率事件,通过长时间的数据累积,即可逐渐消除噪声带来的影响.

2) 攻击绕过.攻击者通过在其网络活动所必需的 DNS 请求中混杂其他非必需的 DNS 请求,从而制造虚假伴随关联关系.例如,在恶意网页中添加虚假外链,链接到已知的正常网站(如 qq.com),从而制造其恶意域名和正常的域名伴随出现的假象.

3) 检测时空.CoDetector 只能检测当前时空下采集到的 DNS 流量中的恶意域名.对于未来出现的未知域名,本文没有提前训练该域名的特征表示,CoDetector 无法对其恶意性质进行判定.本文将在后续的工作中进行补充.

5 结 论

恶意域名是网络攻击活动中主要的基础设施.本文提出一种基于域名请求之间存在时空伴随关联的恶意域名检测算法,想法是如果一个域名经常和恶意域名伴随出现,那么该域名很大可能是恶意域名.本文首先提出基于时间间隔的时空伴随域名序列提取算法,从原始 DNS 流量中提取具有时空伴随关系的域名,接着采用深度学习算法将每一个域名映射为低维空间的特征向量,最后结合分类算法训练模型分类器.实验结果表明:CoDetector 模型能够有效地检测恶意域名,具有 91.64%的检测精度和 96.04%的召回率.

参 考 文 献

[1] Plohmann D, Yakdan K, Klatt M, et al. A comprehensive measurement study of domain generating malware [C] //Proc of USENIX Security Symp. Berkeley, CA: USENIX Association, 2016: 263-278

[2] Antonakakis M, Perdisci R, Nadji Y, et al. From throw-away traffic to bots: Detecting the rise of DGA-based malware [C] //Proc of the 21st USENIX Security Symp. Berkeley, CA: USENIX Association, 2012: 491-506

[3] Szurdi J, Kocso B, Cseh G, et al. The long "taile" of typosquatting domain names [C] //Proc of the 23rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2014: 191-206

[4] Cisco. Cisco 2016 annual security report [R/OL]. 2016 [2018-06-28]. http://www.cisco.com/c/m/en_us/offers/sc04/2016-annual-security-report/index.html

[5] Antonakakis M, Perdisci R, Dagon D, et al. Building a dynamic reputation system for DNS [C] //Proc of the 19th USENIX Security Symp. Berkeley, CA: USENIX Association, 2010: 273-290

[6] Bilge L, Sen S, Balzarotti D, et al. Exposure: A passive DNS analysis service to detect and report malicious domains [J]. ACM Transactions on Information and System Security, 2014, 16(4): 14

[7] Antonakakis M, Perdisci R, Lee W, et al. Detecting malware domains at the upper DNS hierarchy [C] //Proc of the 20th USENIX Security Symp. Berkeley, CA: USENIX Association, 2011(11): 1-16

[8] Mikolov T, Sutskever I, Chen Kai, et al. Distributed representations of words and phrases and their compositionality [C] //Proc of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2013: 3111-3119

[9] Bojanowski P, Grave E, Joulin A, et al. Enriching word vectors with subword information [J]. arXiv preprint arXiv: 1607.04606, 2016

[10] Khalil I, Yu Ting, Guan Bei. Discovering malicious domains through passive DNS data graph analysis [C] //Proc of the 11th ACM on Asia Conf on Computer and Communications Security. New York: ACM, 2016: 663-674

[11] Peng Chengwei, Yun Xiaochun, Zhang Yongzheng, et al. Discovering malicious domains through alias-canonical graph [C] //Proc of the 16th IEEE Int Conf on Trust, Security and Privacy in Computing and Communications. Piscataway, NJ: IEEE, 2017: 225-232

[12] Manadhata P K, Yadav S, Rao P, et al. Detecting malicious domains via graph inference [G] //LNCS 8712: Proc of European Symp on Research in Computer Security. Berlin: Springer, 2014: 1-18

[13] Rahbarinia B, Perdisci R, Antonakakis M. Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks [C] //Proc of the 45th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks (DSN). Piscataway, NJ: IEEE, 2015: 403-414

[14] Wang Xiaoqi, Li Qiang, Yan Guanghua, et al. Detection of covert and suspicious DNS behavior in advanced persistent threats [J]. Journal of Computer Research and Development, 2017, 54(10): 2334-2343 (in Chinese)
(王晓琪, 李强, 闫广华, 等. 高级持续性威胁中的隐蔽可以DNS行为的检测[J]. 计算机研究与发展, 2017, 54(10): 2334-2343)

[15] Gao Hongyu, Yegneswaran V, Jiang Jian, et al. Reexamining DNS from a global recursive resolver perspective [J]. IEEE/ACM Transactions on Networking, 2016, 24(1): 43-57

[16] Kiefer J, Wolfowitz J. Stochastic estimation of the maximum of a regression function [J]. The Annals of Mathematical Statistics, 1952, 23(3): 462-466

[17] DNS-BH. Malware domain blocklist by risk-analytics [EB/OL]. [2017-01-03]. http://www.malwaredomains.com

[18] OpenDNS. Phishtank [EB/OL]. [2017-01-03]. http://www.phishtank.com

[19] OpenPhish. Timely, accurate, relevant threat intelligence [EB/OL]. [2017-01-03]. https://openphish.com

[20] AbuseList. Ransomware tracker [EB/OL]. [2017-01-03]. https://ransomwaretracker.abuse.ch

[21] Porras P A, Saïdi H, Yegneswaran V. A foray into conficker's logic and rendezvous points [EB/OL]. [2017-01-03]. https://www.usenix.org/legacy/event/leet09/tech/full_papers/porras/porras_html/

[22] Google. Google safe browsing [EB/OL]. [2017-10-14]. https://www.google.com/transparencyreport/safebrowsing/

[23] Alexa. Alexa top 1 million [EB/OL]. [2017-03-05]. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

[24] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in python [J]. Journal of Machine Learning Research, 2011(12): 2825-2830

[25] Chen Tianqi, Guestrin C. XGBoost: A scalable tree boosting system [C] //Proc of the 22nd ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2016: 785-794

Peng Chengwei, born in 1993. PhD candidate in the Institute of Computing Technology, Chinese Academy of Sciences, China. His main research interests include network and information security, in particular DNS data mining.

Yun Xiaochun, born in 1971. PhD, professor and PhD supervisor in the Institute of Information Engineering, Chinese Academy of Sciences, China. His main research interests include network and information security.

Zhang Yongzheng, born in 1979. PhD, professor and PhD supervisor in the Institute of Information Engineering, Chinese Academy of Sciences, China. His main research interests include network and information security, particularly cyberspace security situational awareness.

Li Shuhao, born in 1983. PhD, assistant professor in the Institute of Information Engineering, Chinese Academy of Sciences, China. His main research interests include network and information security, particularly mobile malware attack and defense.