计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(12):226−231 [doi: 10.15888/j.cnki.csa.007198]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于文档分层表示的恶意网页快速检测方法①

袁　梁[1], 林金芳[2]

[1](无锡城市职业技术学院 师范学院, 无锡 214153)
[2](国防科技大学 系统工程学院, 长沙 410073)

摘　要: 近年来, 恶意网页检测主要依赖于语义分析或代码模拟执行来提取特征, 但是这类方法实现复杂, 需要高额的计算开销, 并且增加了攻击面. 为此, 提出了一种基于深度学习的恶意网页检测方法, 首先使用简单的正则表达式直接从静态 HTML 文档中提取与语义无关的标记, 然后采用神经网络模型捕获文档在多个分层空间尺度上的局部性表示, 实现了能够从任意长度的网页中快速找到微小恶意代码片段的能力. 将该方法与多种基线模型和简化模型进行对比实验, 结果表明该方法在 0.1% 的误报率下实现了 96.4% 的检测率, 获得了更好的分类准确率. 本方法的速度和准确性使其适合部署到端点、防火墙和 Web 代理中.

关键词: 深度学习; 恶意 Web 内容; 网页分类; 恶意网页识别

引用格式: 袁梁,林金芳.基于文档分层表示的恶意网页快速检测方法.计算机系统应用,2019,28(12):226–231. http://www.c-s-a.org.cn/1003-3254/7198.html

Hierarchical Representation Approach to Fast Detection of Malicious Webpages

YUAN Liang[1], LIN Jin-Fang[2]

[1](Normal College, Wuxi City College of Vocational Technology, Wuxi 214153, China)
[2](Collcge of Systems Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: In recent years, the web content detection mainly focuses on how to extract features from HTML document through semantic analysis or emulation execution, while it is undesirable, because it significantly complicates implementation which requires high computational overhead, and opens up an attack surface within the detector. A deep learning approach to detect malicious web pages is proposed. Firstly, we take advantage of the non-complex regular expression to extract tokens from static HTML document, then capture locality representation at multiple hierarchical spatial scales over the document with neural network model, by which the mode can quickly find tiny fragments of malicious code in any length of web pages. The experimental results show that this approach achieves a detection rate of 96.4% at a false positive rate of 0.1%, much better than the baseline and simplified model at the classification accuracy. The speed and accuracy of proposed approach makes it appropriate for deployment to endpoints, firewalls and web proxies.

Key words: deep learning; malicious Web content; Web page classification; malicious Web page detection

当前, 利用恶意网页的攻击非常普遍. 360 安全报告指出, 2018 年上半年, 360 互联网安全中心共截获各类新增恶意网页 1622.6 万个, 同比 2017 年上半年 (201.5 万个) 上升了 7 倍[1]. 如何有效识别这些恶意网页面临多种挑战: 首先检测方法必须在用户无感的情况下运行; 其次检测方法必须能够识别恶意网页以躲

① 基金项目: 国家自然科学基金 (91430214)
Foundation item: National Natural Science Foundation of China (91430214)
收稿时间: 2019-04-25; 修改时间: 2019-05-21; 采用时间: 2019-06-24; csa 在线出版时间: 2019-12-10

避检测为目的的各种代码混淆技术; 最后检测方法必须能够在海量的 Web 网页访问中, 快速找出嵌入在正常网页中的恶意代码片段[2].

为了应对这些挑战, 本文提出了一种基于深度学习的恶意网页快速检测方法, 使用简单的 12 个字符正则表达式标记化 Web 内容, 然后在多个分层空间尺度上检测这些内容. 分层空间尺度是指不是简单地将整个 HTML 文件上的令牌 (token) 集合作为输入, 而是计算令牌集合在多个本地特定子区域上的表示: 将 HTML 文件分成 1/2、1/4、1/8 和 1/16, 然后在这些层级上运用两个全连接层 (fully connected layer) 来提取这个 HTML 文件的表示. 该方法基于简单的令牌流输入来学习 Web 内容的高质量表示, 可以从任意长度的网页中快速找到微小恶意代码片段.

1 相关工作

恶意网页检测研究工作的一个重点是仅使用 URL 字符串来检测恶意 Web 内容. 文献[3]基于黑名单的方法, 首先对恶意 URL 进行标注, 然后利用字符串匹配等技术实现恶意 URL 的识别; 文献[4]以 URL 词汇特征和主机特征为基础建立统一的分类模型, 进而根据已有标注集合识别恶意 URL, 他们专注于人工标注来最大化检测精度; 文献[5]使用 URL 作为检测依据, 但也包含其他信息, 例如 Web 链接中的 URL 引用, 将手工提取特征作为 SVM 和 *K*-最邻近分类器的输入. 但是这些方法只关注 URL 相关信息, 因此无法利用 Web 内容中的恶意语义. 虽然基于 URL 的系统具有轻量级的优势, 并且可以在没有完整 Web 内容的环境中进行部署, 但本文的工作重点是 HTML 文件, 因为它们能检测出更深层次的威胁.

文献[6–8]尝试从 HTML 和 JavaScript 中提取特征并将其提供给机器学习或启发式检测系统来检测恶意 Web 内容. 文献[6]提出一种基于机器学习分类器的网页恶意代码检测方法, 通过分析 JavaScript 脚本本身特征, 来检测网页恶意 JavaScript 脚本; 文献[7]提出一个 Web 网页抓取器, 用于 JavaScript 反混淆和分析, 再将全部抽取的 JavaScript 代码用自定义的基本词表示, 然后利用 3 种机器学习算法通过异常检测模型检测恶意网页; 文献[8]提出一种手动定义的启发式方法, 使用静态特征提取来检测恶意 HTML 文档. 这些方法与本文的研究类似, 不同之处在于, 本文使用无解析器标记化方法来计算 HTML 文件的表示, 而不是显式地解析 HTML, JavaScript 或 CSS, 甚至模拟 JavaScript 的执行. Web 内容的无解析器表示允许对恶意文档和良性文档的语法和语义做出最少量的假设, 从而使深度学习模型在学习 Web 内容的内部表示方面具有最大的灵活性.

在 Web 内容检测之外, 国内外学者在基于深度学习的文本分类领域做出了广泛的研究. 例如, 文献[9]提出的单层卷积神经网络 (CNN), 使用无监督 (Word2Vec) 序列和词嵌入 (word embedding) 序列, 在句子分类任务中, 可以提供良好的性能; 文献[10]对使用单层 CNN 进行文本分类所需要的超参数进行调整, 进一步优化模型性能; 文献[11]的模型则表明在文本分类问题上, 直接从字符输入中学习表示的 CNN 方法更具竞争力. 本文的工作涉及了这些方法, 但不同之处, 本文模型是运行在 HTML、JavaScript 和 CSS 多种格式混搭的 HTML 网页上, 这些格式中可能包含任意源代码、攻击负载和自然语言. 由于 HTML 文件内容的不确定性使得定义离散词汇变得相对困难, 因此本文不使用词嵌入作为模型输入, 相反使用简单的、格式无关的标记化方法对 Web 文档分层表示.

2 模型设计

2.1 设计原则

本文模型是基于恶意网页检测的下述特殊性提出的:

(1) 恶意内容片段通常很小, 但 HTML 文档的长度差异非常大, 这意味着恶意内容在 HTML 文档中的长度比例是可变的. 因此需要模型在多个空间尺度上检查文档, 以识别给定文档是否是恶意的.

(2) HTML 文档的显式解析实际上是 HTML、JavaScript、CSS 和数据的执行, 这是不可取的, 因为这可能需要高计算开销, 并且在检测器内打开攻击面, 存在被攻击者利用的可能.

(3) JavaScript 的仿真执行、静态分析或符号执行也不可取, 因为同样存在增加计算开销和在检测器内打开攻击面的可能.

基于上述问题, 模型设计遵循了下述原则:

(1) 模型不对 HTML 文档执行详细解析、静态分析、符号执行或内容模拟, 而是对文档组成格式做最小假设, 进行简单的词袋样式标记, 本文称为令牌袋 (Bag of Token, BoT).

(2) 模型在代表不同区域和聚合层级的多个尺度空间上捕获局部性表示, 而不是简单地将整个文档以单层的令牌袋表示.

2.2 设计方法

模型涉及一个特征提取器, 负责从 HTML 文档中解析出一系列令牌 (token); 一个神经网络模型, 包括两个逻辑组件.

(1) 检查员: 在分层空间尺度上应用共享权重 (shared-weight), 并将文档的信息聚合成 1024 长度的向量.

(2) 主网络: 对检查员的输出做出最终的分类决策.

(3) 特征提取器: 首先, 使用正则表达式 ([^\x00-\x7F]+|\w+)+) 对目标文档进行标记, 它将文档沿非字母数字字边界进行分割. 然后将得到的令牌流划分为 16 个相等长度的连续块, 其中长度定义为令牌数, 如果文档的令牌数量不能被 16 整除, 则最后一个块中的令牌要少. 标记化和分块功能系统实现的 Python 代码函数 TokenizeChunk 如下.

```
import numpy as np
def TokenizeChunk(data, steps=16, dims=16 384):
data=TokenizeLengthHash (data, steps=steps, dims=dims)
ret = []
stepsize = int(len(data) / float(steps))
for percent in np.arange(0, 1, 1 / float(steps)):
idx = int(len(data) * percent)
unq, cnt = np.unique(data[idx:idx + stepsize], return_counts=True)
newarray = np.zeros(dims / steps)
for v, c in zip(unq, cnt):
newarray[v] = c
ret.append(newarray)
return ret
```

接下来, 使用函数 TokenizeLengthHash, 它利用一个修改过的 hashing trick, 使用 1024 个 bin, 为每个块创建令牌袋.

```
import re
import murmur # the murmur hashing library
def TokenizeLengthHash (data, steps=16, dims=16 384):
feats = re.findall(r"([^\x00-\x7F]+|\w+)", data)
final_feats = []
for feat in feats:
loglength=int(min(8, max(1, math.log(len(feat), 1.4)))) - 1 # 0-7
shash=murmur.string_hash (feat) % (dims/steps/8)
final_feats.append (loglength*(dims/steps/8)+shash)
return final_feats
```

整个流程的结果是, 先将 HTML 文档标记, 再将得到的令牌集合分成 16 个相等长度的块, 然后将每个块散列到 1024 个 bin 中, 得到一个 16×1024 tensor, 表示从 HTML 文档中提取的令牌袋序列, 其中序列中的每个元素表示输入文档的连续 1/16 上的聚合.

(4) 检查员组件: 检查员负责将提取的特征表示输入到神经网络中. 如图 1 所示, 第一步, 创建一个层级表示令牌序列, 在这个层级中, 先将最初的 16 个令牌袋折叠成 8 个令牌袋, 再将 8 个令牌袋折叠成 4 个, 4 折叠成 2, 2 折叠成 1. 这样就在多个空间尺度上获得了多个令牌袋, 用于捕获令牌的出现. 整个折叠过程的平均窗口长度为 2, 步长为 2, 直到递归得到一个令牌袋. 采用平均而不是求和的方法, 对于给定文档保证了每个表示级别的范数相同.

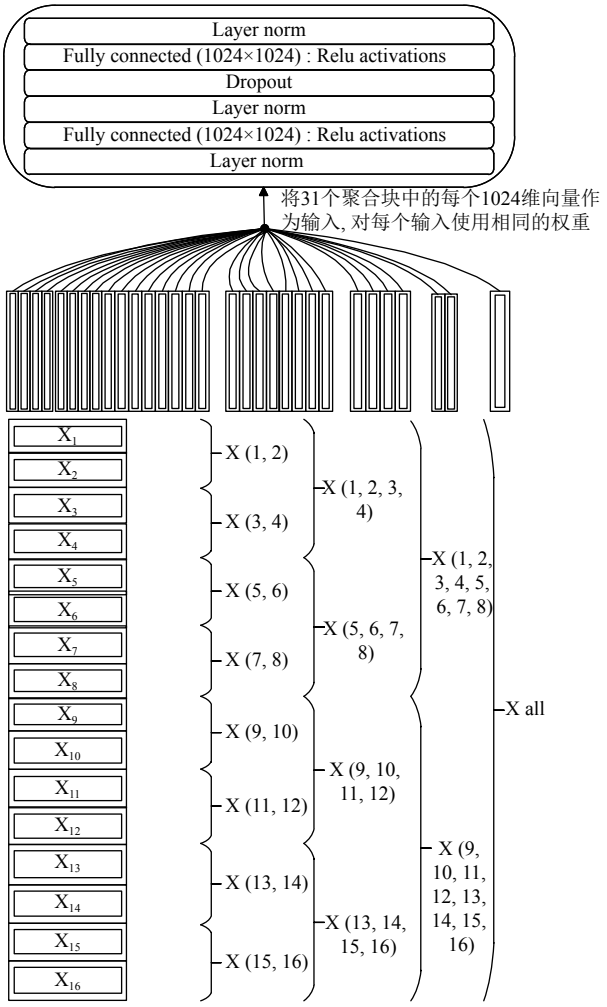


图 1 检查员组件逻辑

一旦检查员创建了这种分层表示, 它就会继续访问聚合树中的每个节点并计算输出向量. 检查员是一个前馈神经网络, 有两个完全连接的层, 每个层有 1024 个 ReLU 单元. 本文使用 layer normalization[12]来防止梯度弥散, 使用 dropout[13]规范化检查员, dropout=0.2.

在检查器访问每个节点后, 为了计算 1024 维向量输出, 模型从 1024 个神经元中获取最大激活, 其结果是用最后一个层中的每个神经元的最大输出表示文档的最终向量. 简单地说, 这将增强输出向量捕获模式, 因为这些模式最好地匹配了用于预测恶意内容的已知模板特征, 无论它们出现在文档中的什么位置, 或者整个文档有多长.

(5) 主网络组件: 一旦检查员计算了目标文档的 1024 维输出向量, 向量就被输入到模型的主网络中. 如图 2 所示, 主网络也是一个前馈神经网络, 有两个完全连接的层, 在每个完全连接的层之前是 layer normalization 和 dropout, 与检查员的一样, dropout=0.2.

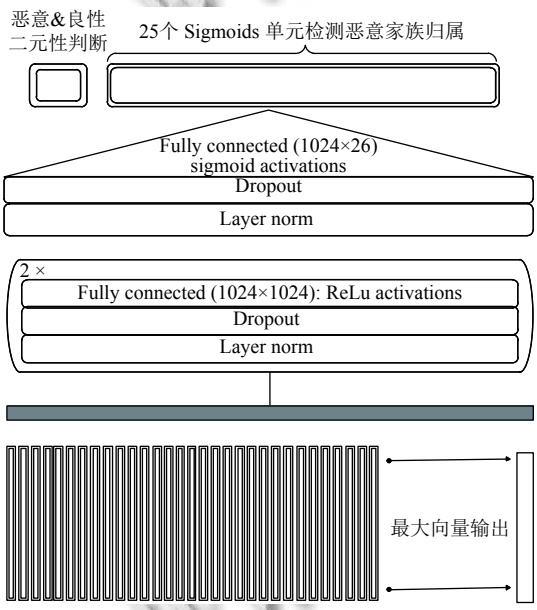


图 2 主网络组件逻辑

模型的最后一层由 26 个 Sigmoid 单元组成, 对应于对文档做出的 26 个检测决策. 其中一个 Sigmoid 用于确定目标文档是恶意的还是良性的, 其余 25 个 Sigmoid 检测各种信息标签, 用于确定文档的恶意软件家族归属, 例如确定文档是网络钓鱼网页还是漏洞利用工具包. 为了训练模型, 本文在每个 Sigmoid 输出上使用 binary-cross-entropy 损失函数, 然后平均得到的梯度以计算参数更新. 模型强调评估良好与不良 Sigmoid 输出的准确性, 但也考虑了模型输出的性能.

3 实验

本文的目标是创建一个可以运行在端点、防火墙和 Web 代理上的快速 Web 内容检测模型. 模型的准确性和速度是主要考核指标.

本文以两种方式测试了上述模型的准确性. 首先它与其他词袋模型进行了比较, 这些基线模型代表了标准的文档分类方法, 其次以各种方式修改了模型, 以测试模型设计的合理性.

本文没有直接给出实验用例测试模型的速度. 分层尺度空间模型涉及网页内容的解析, 但同涉及 Web 内容解析或仿真执行的其他方法比较, 40 万个网页的检测耗时仅为这类 WAF 产品的 30%, 速度优势明显.

3.1 实验数据

实验数据集来源于 360 公司的 NGSOC 和 TIP 平台. 360 公司全球化布控的探针每天接收数万个新的 HTML 文件, 使用 60 个 Web 威胁扫描程序扫描它们. 实验数据集是平台 2018 年前 9 个月推送的数据, 如图 3 所示.

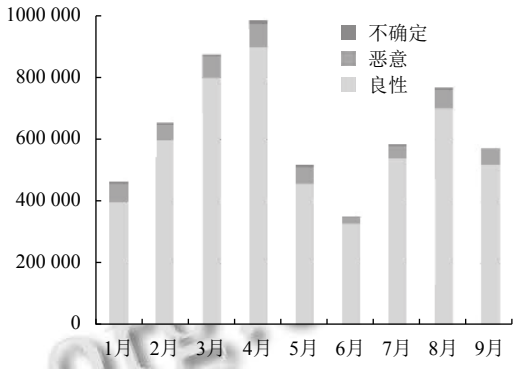


图 3 实验数据集

HTML 文件基于 SHA256 进行唯一标识. 训练/测试集的拆分依据文件的首次推送时间来计算. 这可以确保: 一是训练和测试集是不同的 (因为稍后提交的相同 HTML 文件将解释为重新推送该文件, 会选择忽略); 二是训练和测试实验更加接近实际部署场景.

标记策略是将良性 HTML 文件定义为全部扫描器判断为正常的文件, 即 0 威胁, 将恶意文件定义为 3 个或更多扫描器判定为威胁的文件. 因此实验丢弃了 1 到 2 个扫描器判定为威胁的文件, 因为这一小部分检测意味着安全行业仍然无法确定它们是恶意的还是良性的. 从图 2 中可以看出, 这些不确定的文件占整个文件集很小的一部分.

这种标记方法存在风险, 即模型简单地记住了 360 产品的知识, 而不是学习真正新颖的检测功能, 以

检测 360 可能错过的未知威胁. 因此在实验中, 本文使用了历史模拟法 (historical simulation), 定义如下:

(1) 在 360 首次推送的某个时间点 t 之前 Web 内容文件上训练模型.

(2) 在时间点 t 后两个月内首次出现的 Web 内容文件上测试模型.

此方法弥补了模型检测未知威胁能力不足的问题, 这是因为测试的文件在训练中从没有出现过, 但是 360 有时间通过检测规则和黑名单更新来检测新的威胁. 换句话说, 只要模型能够正确预测未来 Web 内容的标签, 360 就有时间将黑名单或检测规则更新, 从而间接证明本文模型具备检测 0-day 威胁的能力.

另外, 本文还手工检查了标记为良性的, 但模型却显示具有很高恶意概率的样本, 最终发现大多数误报是垃圾内容, 后面会详细讨论.

3.2 实验设置

为了评估分层尺度空间模型的性能, 本文进行了 4 个实验. 其中对比实验将本文模型与其他基线模型进行了比较, 对比实验中涉及的神经网络模型, 模型训练采取 Adam 优化算法、*batch_size*=64, 以及基于验证集的早停法. 2 个对比实验设置如下:

FF-BoT: 一种前馈架构, 使用本文模型提取的令牌作为输入, 但是将其散列为 16 284 长度向量. 因为它与本文模型中使用的 16×1024 表示具有相同的维度, 因此近似于逐个比较. FF-BoT 提供了一个简单的深度学习词袋基线与本文模型进行比较.

XGBoost-BoT: 使用与 FF-BoT 相同的特征输入的梯度提升决策树 (XG-Boost) 模型.

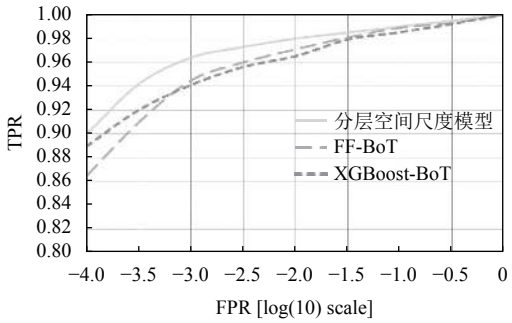
另外本文还对分层尺度空间模型进行了修改, 以确定模型内部流程的合理性. 2 个修改实验如下:

HIM-Flat1: 模型的变体 1, 删除了平均池化步骤, 这样检查员只能看到树的叶节点. 换句话说, 在输入冠军模型的 31 个聚合表示中, 该模型只能看到 16 个连续的块, 而没有更大的聚合窗口. 此实验用于对比在分层空间尺度上检测 HTML 文档的性能增益.

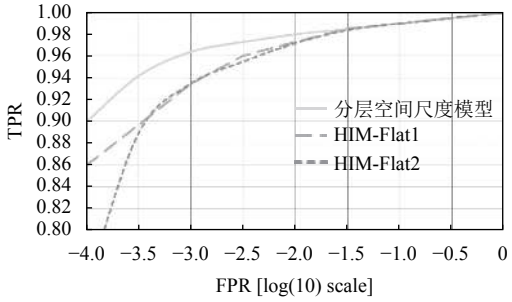
HIM-Flat2: 模型的变体 2, 一个简单的前馈神经网络, 它使用与分层尺度空间模型相同的特征表示. 但是, 它不是在每一步都应用共享权重检查, 而是简单地将 16×1024 连续的令牌向量光栅化为单个 16 284 长度的向量, 并将其输入到前馈神经网络中. 该实验评估了模型相对于只在第一层密集使用共享权重检查的性能增益.

4 实验结果

图 4(a) 和图 4(b) 给出了实验结果的 ROC 曲线, y 轴表示 TPR 真阳率, x 轴表示 FPR 假阳率. 图 4(a) 将 FF-BoT 和 XGBoost-BoT 与本文模型进行了比较, 图 4(b) 将修改模型与完整模型进行了比较.



(a) 分层空间尺度模型与词袋基线模型对比



(b) 分层空间尺度模型与修改模型对比

图 4 实验 ROC 曲线对比图

图 4(a) 的 ROC 曲线显示本文模型要优于其他基线模型. 以 0.1% 的 FPR 比较这些模型的相对性能, 本文模型、FF-BoT 和 XGBoost-BoT 分别达到 96.4%、94.5% 和 94.1% 的检测率. 根据 FPR, 本文模型实现了 3.6% 的假阳率, 而 FF-BoT 和 XGBoost-BoT 大约是 5.5% 和 5.9%. 总体来说, 分层尺度空间模型的整体 ROC 曲线明显优于 FF-BoT 和 XGBoost-BoT.

另外 FF-BoT 参数 (约 2000 万) 远远超过本文模型 (约 400 万), 相对于本文模型表现不佳. 这表明本文模型捕获了更有效的恶意 HTML 文档特征表示, 这要归功于检查员在分层表示中检查的每个空间上下文中使用相同的参数.

图 4(b) 的 ROC 曲线显示在 0.1% 的 FPR 前提下, HIM-Flat1 和 HIM-Flat2 变体的检测率均达到 93.5%. 变体模型的 ROC 曲线明显差于本文模型. 与 HIM-Flat1 的对比实验表明在多个空间尺度上检查内容对于获得良好的准确性至关重要. 类似地, 与 HIM-Flat2 的对比表明, 检查员在检查每个空间上下文和尺度时使

用相同的参数对于产生高检测精度的重要性, 因为 HIM-Flat2 对每个空间上下文使用单独的权重, 从而得到更差的结果.

为了更好地理解分层尺度空间模型学到了什么, 本文采用了恶意软件家族标签来细分恶意样本. 基于 0.1% 的全局误报率阈值, 分层尺度空间模型的总体检测率为 96.4%, 并根据预设的恶意软件家族对检出文件进行了归类. 表 1 中的显示表明, 本文模型在代码注入 XSS, 浏览器漏洞利用和 iFrame 劫持方面取得了理想的检测效果, 但在钓鱼网站上的检测率不高. 由于一个恶意文件可能同时属于多个家族, 故表中的百分比总和要大于 100%.

表 1 不同恶意软件家族检测率及占比

恶意内容家族分类	TPR (FPR=0.1%)	百分比 (%)
代码注入 XSS	0.999	16.1
浏览器漏洞利用	0.998	14.4
IFrame 劫持	0.998	14.9
浏览器重定向	0.99	3.3
搜索引擎毒化	0.987	49.6
Ramnit 恶意家族	0.985	39.7
JQuery 伪装	0.973	0.5
Google Hacking	0.971	13.8
浏览器主页更改	0.935	5.2
勒索软件	0.931	1.2
页面自动点击	0.907	0.6
钓鱼网页	0.895	0.5

另外, 本文还检查了模型与 360 公司标签不一致的情况, 证明模型实际上已经检测到 360 错过的未知恶意 Web 内容. 为进行这项分析, 本文检查 360 公司标记为良性验证集中排名前 20 的评分测试示例, 发现 20 个中的 11 个实际上是恶意的或有潜在威胁的, 9 个是误报. 恶意文件中的 3 个是来自网页内容拦截器的警报页面, 本身不是恶意但表示恶意警告; 3 个是恶意伪装 JQuery 库, 1 个是释放 svchost.exe 木马文件到磁盘的 Javascript, 1 个是页面点击劫持, 1 个是包含下载者执行代码的页面, 2 个是垃圾广告邮件. 该分析验证了模型能够超越标签噪声并识别未知的恶意内容.

5 结论

本文的工作与大多数基于深度学习的文档分类 (如情感分类) 不同, 本文设法捕获试图逃避检测的恶意行为, 而情感分类句子作者并不隐藏他们表达的情绪. 此外, 本文避免使用原始字符序列作为模型的输入, 因为 HTML 文档的长度使得涉及原始字符串的卷积神经网络在端点和防火墙上难以处理. 本文模型采用纯粹基于令牌的静态检测方法, 避免了复杂解析或仿真系统的需求, 能有效处理 Web 页面的检测问题, 而不管 Web 页面的大小如何变化, 以 0.1% 的误报率实现 96.4% 检测性能, 甚至可以识别以前未被安全产商捕获的恶意 Web 内容. 模型良好的速度和准确性使得其适合部署到端点、防火墙和 Web 代理中.

参考文献

1 2018 年上半年中国互联网安全报告. http://zt.360.cn/1101061855.php?dtid=1101062360&did=491357630. [2018-07-30].

2 沙泓洲, 刘庆云, 柳厅文, 等. 恶意网页识别研究综述. 计算机学报, 2016, 39(3): 529–542. [doi: 10.11897/SP.J.1016.2016.00529]

3 Akiyama M, Yagi T, Itoh M. Searching structural neighborhood of malicious urls to improve blacklisting. Proceeding of the 2011 IEEE/IPSJ International Symposium on Applications and the Internet. Munich, Bavaria, Germany, 2011: 1–10.

4 Ma J, Saul LK, Savage S, *et al*. Learning to detect malicious URLs. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 30.

5 Choi H, Zhu BB, Lee H. Detecting malicious web links and identifying their attack types. WebApps' 11 Proceedings of the 2nd USENIX Conference on Web Application Development. Berkeley, CA, USA. 2011. 11.

6 李洋, 刘飚, 封化民. 基于机器学习的网页恶意代码检测方法. 北京电子科技学院学报, 2012, 20(4): 36–40, 12. [doi: 10.3969/j.issn.1672-464X.2012.04.007]

7 马洪亮, 王伟, 韩臻. 基于 JavaScript 的轻量级恶意网页异常检测方法. 华中科技大学学报 (自然科学版), 2014, 42(11): 34–38.

8 Seifert C, Welch I, Komisarczuk P. Identification of malicious web pages with static heuristics. 2008 Australasian Telecommunication Networks and Applications Conference. Adelaide, SA, Australia. 2008. 91–96.

9 Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv: 1408: 5882, 2014.

10 Zhang Y, Wallace B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv: 1510: 03820, 2015.

11 Zhang X, Zhao JB, LeCun Y. Character-level convolutional networks for text classification. arXiv: 1509.01626, 2015.

12 Ba JL, Kiros JR, Hinton GE. Layer normalization. arXiv preprint arXiv: 1607.06450, 2016.

13 黄文坚, 唐源. TensorFlow 实战. 北京: 电子工业出版社, 2017. 2.