计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190200332

改进的 TLS 指纹增强用户行为安全分析能力

胡建伟　徐明洋　崔艳鹏
西安电子科技大学网络与信息安全学院　西安 710071

摘　要　随着攻防对抗的升级，用户行为分析与网络安全的结合逐渐进入了研究者的视野。用户行为分析技术可以做到在被成功攻击前识别不可信用户，遏制入侵，达到主动防御的效果。当前在 Web 安全中用户行为分析所使用的数据源主要是应用层 HTTP 维度的数据，这不足以确定用户身份，容易造成漏报。在安全性和隐私性更好的 HTTPS 技术被大规模应用的情况下，文中提出了基于 n-gram 和 Simhash 的改进的 TLS 指纹数据，该方法提高了现有 TLS(Transport Layer Security)指纹的容错性。将该指纹应用到用户行为分析中可提高用户身份判定的准确率。对比实验使用卷积神经网络对从真实环境中得到的指纹数据和日志型用户行为数据进行建模分析。结果表明，改进的 TLS 指纹数据可以更有效地识别用户和黑客，将准确率提高了 4.2%。进一步的分析表明，通过改进的 TLS 指纹关联用户行为和时间轴回溯，还能在一定程度上对黑客进行追踪溯源，从而为安全事件调查提供情报上下文。

关键词：TLS 指纹；卷积神经网络；用户行为分析；Web 安全
中图法分类号　TP393

Improved TLS Fingerprint Enhance User Behavior Security Analysis Ability

HU Jian-wei, XU Ming-yang and CUI Yan-peng
School of Cyber Engineering, Xidian University, Xi'an 710071, China

Abstract　With the upgrade of offensive and defensive confrontation, the combination of user behavior analysis and network security has gradually entered the researchers' field of vision. User behavior analysis technology can achieve active defense by identifying untrusted users and preventing the intrusions before being attacked successfully. Currently, the datasets used in user behavior analysis in Web security are mainly the application layer HTTP data, which is insufficient to identity user and is likely to cause false negatives. This paper proposed an improved TLS fingerprint data based on n-gram and Simhash, which enhances the fault tolerance of the existing TLS fingerprint. The application by using the improved fingerprint to user behavior analysis can improve the accuracy of user indentification. The comparative experiment used convolutional neural network to model and analyze the fingerprint data and log-type user behavior data obtained from the real environment. The results show that the improved TLS fingerprint data can identify normal users and hackers more effectively, and the accuracy is improved by 4.2%. Further analysis shows that the improved TLS fingerprint can trace hackers to a certain extent by correlating user behaviors and timeline backtracking, thus providing an intelligence context for security incident investigation.

Keywords　TLS fingerprint, Convolutional neural network, User behavior analysis, Web security

1　引言

随着 Web 技术的升级，企业和个人的业务逐渐互联网化和大规模化，这些变化引入了更多的攻击面，由此产生了大量新的攻击方法。攻防对抗的升级导致很多传统的防护手段呈现颓势，如基于规则的 Web 应用防火墙等。这些防护手段由于需要大量专家分析攻击方法和提取规则，使得人工成本不断提高。并且规则也需要人工维护，一旦规则规模过大，维护工作就变得难以为继。因此，在人工智能技术飞速发展的情况下，数据驱动安全的概念应运而生，并且 Web 安全领域中的人工智能技术已经逐渐从传统的机器学习方法过渡到深度学习方法。作为一种以机器学习为核心的高级网络威胁检测手段，用户行为安全分析正逐渐受到高度关注。用户行为分析可以帮助企业解决其最重视的安全风险。

Web 安全领域中的用户行为分析技术使用的数据源多为 HTTP 维度的数据[1-2]，主要是 Web 服务器的日志记录，其包含请求时间、访问者 IP、HTTP 请求首行数据、HTTP 返回状态码、返回文档长度、User-Agent 值等。这些数据对于用户身份的判定仍显不足，在用户行为分析中无法根据这些数据准确地确定多次行为是同一用户所为。例如在一个黑客使用和不使用匿名代理攻击站点时，应该认为这是同一个黑客，但是由于一些原因确定是否是同一个黑客这个问题难以解决。首先由于 IPv4 地址有限，互联网接入时大量使用了网络地址转换(Network Address Translation, NAT)技术，因此

到稿日期：2019-02-20　返修日期：2019-04-03　本文已加入开放科学计划(OSID)，请扫描上方二维码获取补充信息。
通信作者：胡建伟(1270547909@qq.com)

大量用户共用一个出口公网 IP 地址，这使得通过来源 IP 判断用户变得不可靠。其次，由于 HTTP 是无状态的协议，引入了 Cookie 来判断用户身份，但 Cookie 易被篡改，因此仅依赖 Cookie 是不可靠的，同理 HTTP 头中的 User-Agent 字段也因易被篡改而常常被伪装。

为了更精确地识别用户或客户端，出现了一系列基于前端技术栈的指纹识别方法，如使用 JavaScript 计算指纹[3]识别浏览器，该技术通过查询得到浏览器的 User-Agent 值、屏幕分辨率、支持语言、安装的插件与支持的 MIME 类型、所在时区等参数，然后通过散列函数计算得到指纹。但该技术容易产生指纹冲突，仅能作为辅助技术。为了解决指纹易冲突问题，出现了基于 HTML5 技术的 Canvas 指纹和 AudioContext 指纹[4]，也出现了跨浏览器的指纹识别方法[5]。然而，这些前端技术都需要浏览器的参与才能得到相应的指纹，同时也需要专业的前端专家的配合，因此该技术与业务之间耦合度较高，限制了其发展和规模化应用。最重要的是，在应对非浏览器的自动化工具时，如自行开发的爬虫程序等，由于没有 JavaScript 引擎和 CSS 引擎的解析，通过 JavaScript 和 CSS 进行指纹提取的方法就会失效。因此，基于前端技术栈的指纹在应对黑客行为时不适用。

随着人们对安全和隐私的要求越来越高，HTTPS 已经得到大规模的应用，全网普及率已经越来越高。据 Google 的统计[6]，截至 2018 年 12 月，各个平台上通过 Chrome 浏览器使用 HTTPS 加载的网页所占的比例都已超过 70%，Chrome 平台上甚至达到了 91%。而且根据 W3Techs 的统计，全网有 45.1%的网站默认使用 HTTPS[7]。HTTPS 是 TLS 和 HTTP 的结合，即 TLS 为 HTTP 传输提供加解密和相关认证服务。此时 HTTP 数据是以密文形式传输，但 TLS 握手信息是以明文形式传输，因此可以被识别。同时，不同的软件在实现 TLS 协议时因自身需要不同，TLS 握手数据会有所区别，如软件支持的密钥套件(Cipher List)、TLS 扩展(TLS Extension)等存在差异，这些差异就可以被用于 TLS 的指纹识别。因此，引入 TLS 指纹可以更好地识别客户端的身份，区分正常浏览器用户和各种黑客工具。TLS 指纹技术是基于传输层中的 TLS ClientHello 握手过程，可以解决 IPv4 的 NAT 问题和前端指纹不解析问题。

本文使用 2-gram 对 TLS 指纹提取特征并 simhash 化，改善了原有 TLS 指纹容错性差的问题，并将改进的 TLS 指纹数据引入用户行为分析模型中，提高了确认用户身份的准确度。本文以基于卷积神经网络的模型为实验环境设计了对比实验，得出引入 TLS 指纹数据可将用户行为安全分析对不可信用户的识别准确率提高 1.8%；而本文改进的 TLS 指纹数据可将该准确率提高 4.2%。经过复盘分析可知，改进的 TLS 指纹数据增强了发现分布式异常行为的能力，并在发现威胁后根据 TLS 指纹关联和时间轴回溯对黑客进行追踪溯源。

2 相关工作

2.1 TLS 指纹

早在 2009 年，Ivan[8]提出了从 SSL Client Hello 握手包中提取 Cipher List 作为客户端指纹。2012 年，Marek[9]提取了 SSL 版本、Cipher List、TLS 扩展和其他的一些 TLS 的特性，如 Chrome 支持将 SSL 压缩特性作为指纹，并做成了 p0f 的模块，可以解析 SSLv2，SSLv3 和 TLS 的握手包。2015 年，Lee[10]将其应用为 IDS 指纹，用于检测恶意软件(如 SuperFish 等)，并提供了可供他人使用和扩展的工具。同年，Husák 等将 TLS 指纹识别和 HTTP User-Agent 相结合来判断客户端标识[11]。2017 年，Althouse 等基于已有的工作将 TLS 指纹识别标准化，提出了使用 MD5 进行指纹精简和标准化，并开发了 JA3[12]项目。本文基于修改后的 Nginx 模块 JA3，在修复了其中一些错误后构建了用户行为安全分析系统。

SSL/TLS 总是由客户端发起握手连接(ClientHello)，后续过程因 ClientHello 中提供的信息不同而不同，所以 ClientHello 是对客户端进行指纹识别的重要数据结构。ClientHello 包含客户端版本、随机数、会话 ID、密码套件、压缩方法、扩展列表[13]。各项信息如表 1 所列。

表 1 ClientHello 各字段

Table 1 ClientHello struct

字段	说明
版本	SSLv2，SSLv3，TLSv1.0，TLSv1.1，TLSv1.2，TLSv1.3，客户端根据自身支持选择不同的版本
随机数	客户端随机选择的一个结构
密码套件	列出了客户端支持的一组密码学选项，一般一些对安全性比较敏感的开发团队会选出一组自己认为较安全的密码套件列表顺序，因此不同软件支持的密码套件是不同的，顺序也有差异
压缩方法	列出了客户端支持的压缩方法
TLS 扩展	允许客户端向服务端请求一些扩展功能，如椭圆曲线扩展、服务器名称扩展等

由表 1 可知，在 ClientHello 中可以用来做指纹识别的字段有客户端版本、密码套件、扩展列表。为了避免服务端在处理未知类型的 TLS 扩展时出错，提升兼容性，Google 提出了 Generate Random Extensions And Sustain Extensibility[14]机制。该机制会在 ClientHello 包的 TLS 扩展和其他位置中随机添加一些保留关键字的条目，这些保留关键字的 16 进制形式是 xAxA，其中 $x\in[1,2,\cdots,F]$。为了得到更准确的指纹，需要去除这些保留关键字。

2.2 CNN

CNN[15-17]是一种深度前馈网络，在图像分类和计算机视觉领域取得了巨大的成功，在自然语言处理等其他领域也有较多的应用。其设计参考了生物神经元的局部感知特性[18]，通过局部感知和参数共享大大简化了训练参数，并且可以并行计算，使得训练更高效。

CNN 由输入层、卷积层、池化层、激活函数和全连接层组成。卷积层是其核心，通过卷积操作进行特征提取，得到特征矩阵(Feature Map)。在进行卷积操作时，CNN 会选取多个不同大小的卷积核，按照设置的步长对原数据进行卷积。卷积的表达式如下：

$$z(u,v)=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}x_{i,j}\cdot k_{u-i,v-j} \tag{1}$$

为了减少计算量和下一层的输入维度，并有效防止过拟合，常常会引入池化操作。常见的池化操作有最大池化、平均池化和随机池化等。最大池化的操作过程是先选取池化窗口的大小和步长，然后将窗口在原数据上按步长平移，在平移时取窗口中的最大值作为窗口内值的代表。最大池化的结果是：

$$f=\max(w) \tag{2}$$

为了提升表征能力，还需要引入非线性因子，即激活函数。通过激活函数的非线性映射使得整个网络能够取得线性网络无法逼近的效果。常用的激活函数有 ReLU 函数、Tanh 函数和 Sigmoid 函数等。其中，ReLU 函数[18]自提出后，在卷积神经网络中得到了广泛应用，其表达式如下：

$$f(x)=\begin{cases}0, & x<0\\ x, & x\geqslant 0\end{cases} \tag{3}$$

该函数的优点是：当 $x>0$ 时，梯度恒为 1，没有梯度耗散问题，收敛速度快；当 $x<0$ 时，该层输出为 0，增大了网络的稀疏性，使得提取出的特征更具有代表性，使得网络泛化性能更好。该函数的提出很大程度上解决了 BP 算法在优化深层神经网络时的梯度耗散问题。

CNN 在特征提取后，通过全连接层将所有特征连接在一起，然后输出给分类器进行输出判别，最后得到输出结果。

3 基于 TLS 指纹的用户行为安全分析

3.1 数据采集架构

为了收集客户端的 TLS 指纹并进行实时的用户行为分析，最好的方法是在 HTTP 服务器上采集数据。本文通过在应用服务器和客户端之间架设一个反向代理来完成该功能。该反向代理由 OpenResty 和一些自定义模块共同组成。其架构如图 1 所示。

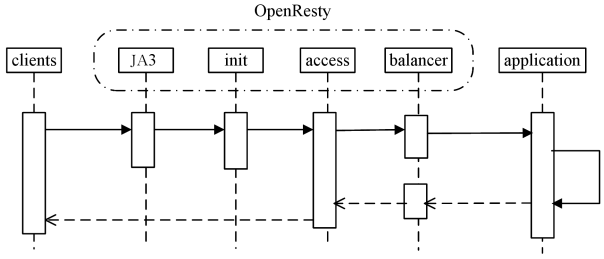


图 1 数据采集架构

Fig. 1 Architecture of data collection

此类架构的主要优点是：对原有业务的侵入性小，仅需要在应用服务器前端加一个反向代理服务器即可；同时，其可扩展性较强，通过负载均衡可以轻松进行扩展。其中，OpenResty 是一个基于 Nginx 和 LuaJIT 的高性能 Web 平台，Nginx 是一个高性能的异步 HTTP 服务器和反向代理，支持自定义模块；LuaJIT 使开发者可以用 Lua 脚本语言调动 Nginx 支持的各种 C 以及 Lua 模块。通过自定义模块 JA3 可以得到 TLS 指纹，然后编写 Lua 脚本得到 JA3 模块的指纹和其他 HTTP 层的行为日志信息，进而进行数据收集和安全分析。

上述架构进行数据收集和用户行为安全分析的工作流程如下：

1）客户端发起一个请求，该请求首先进入 OpenResty，由自定义模块 JA3 解析 SSL/TLS 的 ClientHello 包，得到 TLS 指纹 F。

2）请求信息进入 init_by_lua 阶段，在该阶段 OpenResty 会初始化一些记录数据的代码。

3）该请求进入 access_by_lua 阶段，系统可以在此处进行用户行为分析。如果系统判定用户存在异常请求行为则阻断访问，反之进入 balancer_by_lua 阶段，将该请求交给后端真正的应用服务器进行处理。

JA3 模块中得到的 TLS 指纹由 3 部分组成：客户端支持的最高 SSL/TLS 版本、密码套件、TLS 支持扩展列表，以顺序的形式记录下来，具体格式为：3 个部分间使用逗号分隔，各个密码套件之间使用横杠分隔，各个扩展列表之间也用横杠分隔。F 和示例数据如下所示。

$$F=\{\text{Version},\text{CipherList},\text{ExtensionList}\} \tag{4}$$

$F=\{769,47\text{-}53\text{-}5\text{-}10\text{-}49171\text{-}50\text{-}4,65281\text{-}10\text{-}11\}$

3.2 改进的 TLS 指纹

我们将此套系统部署在一个内部 BBS 论坛上，收集和分析小范围数据后发现如下现象：

1）Chrome，Firefox，Safari 等浏览器在访问一个网站时，多个请求的 ClientHello 包中的 Extension 部分会存在差异，如第一个请求的最后一个 Extension 类型是填充，而后续的请求不再包含这个类型的 Extension。

2）Safari 等为了完成应用层协议的协商，会在第一个请求的 ClientHello 中加入 Next Protocol Negotiation 扩展和 ApplicationLayer Protocol Negotiation 扩展，当服务端不支持时，随后的请求中不会再出现这些扩展。

3）同一种工具分别使用域名和 IP 的方式访问时，前者会在 TLS 扩展中添加 SNI 扩展，而后者不会。

4）Safari，Firefox，Chrome，curl 和 wget 等软件的指纹相差很大，具备较好的区分能力。

5）使用不同的编程语言编写 TLS 客户端进行测试时，得到的 TLS 指纹具有较大的差异，当使用同一种语言和不同的库时，得到的指纹也有差异。

6）很多客户端为实现兼容性而支持比较多的密码套件和 TLS 扩展，导致得到的 TLS 指纹比较长。

根据现象 1）－3）可知，当把 TLS 扩展类型加入指纹参考时，同一个软件的多个请求会存在细微的差异。当使用文献[11]中的方法将 TLS 指纹进行 MD5 哈希时，因为哈希函数局部不敏感，会造成同一客户端得到不同的指纹，从而使容错性较低。根据现象 6），为了得到精简有效的 TLS 指纹，需要对其进行降维。为了解决这个问题，本文使用局部敏感哈希函数 Simhash 对 F 进行处理，得到的 Simhash 值可以通过计算距离对数据相似性进行比较。

TLS 指纹由 3 部分组成，它们都是 2 字节的数字，因此每个部分可能完全相同但具有不同的含义。为了在提取特征时能够得以区分，需要对其进行处理。本文使用了加一位前缀的方式，在 TLS 版本值处加前缀 0，对密码套件的每一项加前缀 1，对扩展列表的每一项加前缀 2。考虑到提取的特征的前后顺序，采用 2-gram 进行特征选取，最后进行 Simhash 计算。图 2 给出对 3.1 节中的示例数据 F 加前缀并使用 2-gram 进行特征提取得到特征集的过程。基于 2-gram 和 Simhash 对 F 进行局部哈希的算法如下。

算法 1 基于 2-gram 提取特征的 Simhash 化 TLS 指纹提取算法

1. Set features←[]
2. Set v←[0] * L//L 为哈希函数得到的值的长度
3. For i←0 to 2 do//示例如图 2

```
4.      For j←0 to len(F[i]) do
5.          feature←加前缀并提取 2-gram 特征
6.          features.append(feature)
7.      Endfor
8. Endfor
9. For f in features// 对特征集进行 simhash 化
10.  h←hashFunc(f)
11.  对 h 的每一位进行加权,权值都取 1,得到 h2
12.  对每一个特征的 h2 进行逐位累加合并到 v
13. Endfor
14. 对 v 进行逐位降维,得到最终 simhash 值
```

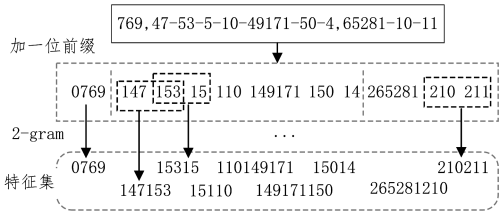


图 2 对示例 F 加前缀的 2-gram 特征提取过程

Fig. 2 2-gram feature extraction process of prefixed F

将多个软件程序的 TLS 指纹通过算法 1 进行计算后,得到每项之间的相似距离,如表 2 所列。

表 2 各软件之间的 TLS 指纹距离

Table 2 TLS fingerprint distance between softwares

	c1	c2	f1	f2	s1	s2	p2r	p2h	p2s	ja	bp
c1	0	3	14	15	25	27	30	32	32	27	29
c2	3	0	13	14	26	28	29	31	31	30	32
f1	14	13	0	1	29	27	24	32	32	29	29
f2	15	14	1	0	28	26	23	33	33	28	28
s1	25	26	29	28	0	4	29	31	31	32	32
s2	27	28	27	26	4	0	25	33	33	30	30
p2r	30	29	24	23	29	25	0	30	28	31	33
p2h	32	31	32	33	31	33	30	0	2	33	33
p2s	32	31	32	33	31	33	28	2	0	33	33
ja	27	30	29	28	32	30	31	33	33	0	6
bp	29	32	29	28	32	30	33	33	33	6	0

其中,c 表示 Chrome 浏览器,f 表示 Firefox 浏览器,s 代表 Safari 浏览器,p2r 表示 Python2 的 request 库,p2h 表示 Python2 的 httplib 库,p2s 表示使用 Python2 运行的 SQL 注入工具 sqlmap,bp 表示渗透测试工具 BurpSuite,ja 表示使用 Java 语言的内置 SSL 库编写的演示软件。

由表 2 可知,同一种工具得到的 Simhash 指纹的海明距离在 5 以内,Chrome,Firefox 和 Safari 分别是 3,1 和 4。同一种语言不同库的指纹可能相差很大,也可能相差很小,如 Java 内置 ssl 库和 BurpSuite 的距离仅有 6,这也表明 BurpSuite 使用的 SSL 库很可能就是内置库。使用 Python2 运行的 sqlmap,由于使用了标准库 httplib,因此与 Python2 用 httplib 编写的 demo 软件的指纹距离只有 2。从表中可以得出,浏览器和其他工具的指纹相差较大,普通用户一般使用浏览器进行网页浏览,而其他用户如黑客、极客等可能会用各种其他工具或自行开发的程序,因而可以用改进的 TLS 指纹来区分。

4 实验设计及结果

4.1 CNN 网络设计

深度卷积神经网络具有强大的自动特征提取能力,避免了复杂的特征工程且易于并行计算,在多个领域都取得了较好的性能。因此,本文实验中,使用 CNN 验证 TLS 指纹对于用户行为分析识别不可信用户的准确率的贡献,然后探讨在检测出威胁后,改进的 TLS 指纹在安全事件调查回溯中的应用。

使用深度卷积神经网络进行用户行为分析的模型如图 3 所示。输入数据分为两个分支,分支一是图 1 中 OpenResty 的后 3 个部分得到的行为数据(LogData),其中包括时间戳、请求方法、请求参数、HTTP Referrer 头、HTTP User-Agent 头、Cookie 头、访问的 URI、使用的协议、Host 头、来源 IP、返回状态码;分支二是图 1 中 JA3 模块得到的 TLS 指纹(TLS FP)。对于 LogData,首先进行字符级 ASCII 码转码得到数值型数据;然后,将其输入字符级嵌入层进行嵌入映射操作,得到字符嵌入矩阵;随后将字符嵌入矩阵送入卷积层,采用 64 个大小为 3×3、步长为 2 的卷积核进行卷积特征提取,同时进行最大池化,并将卷积层的输出送入全连接层。分支二中,TLS 指纹经过算法 1 进行提取特征和 Simhash 计算,从而得到 Simhash 值,然后进入一个全连接层进行神经网络化,并在全连接层后与分支一的全连接层输出进行连接操作来融合两部分数据,然后再进行一次全连接层处理,最后将全连接层输出给 Softmax 分类器进行分类判别,从而得到检测结果。

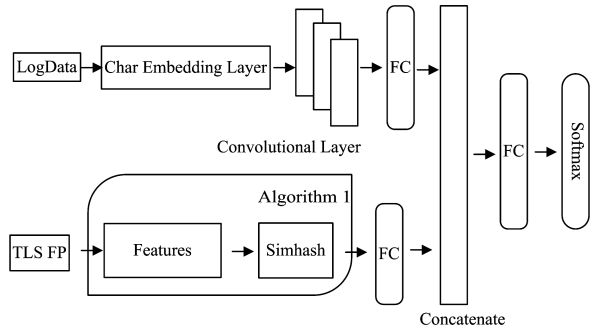


图 3 CNN 网络模型

Fig. 3 CNN model

4.2 实验结果

将上述数据采集系统部署在一个真实环境中一段时间后,收集到数据 72536 条,经过筛选去重和加标签后得到有效数据 23877 条,包括 11229 条正常数据和 12648 条异常数据。将这些数据按 3 :1 进行分割,得到训练数据集和测试数据集。设置对照实验,其中实验组采用的判别模型是图 3 的两个分支,即引入了改进的 TLS 指纹数据;对照组 1 仅使用图 3 中的分支一,即不引入 TLS 指纹数据;对照组 2 使用图 3 中的两个分支,但是分支二中算法 1 部分变为原始 JA3 的处理方法,即使用 MD5 算法处理。对照实验采取的评价指标是准确率。实验结果如表 3 所列。

表 3 实验结果

Table 3 Experimental results

	实验数据	准确率/%
实验组	行为数据+改进的 TLS 指纹	98.3
对照组 1	行为数据	94.1
对照组 2	行为数据+TLS 指纹	95.2

由表 3 可知,引入 TLS 指纹可以提升用户行为安全分析识别的准确率,而且本文提出的改进的 TLS 指纹数据又可以

将准确率进一步提升 3.1 个百分点。进一步的分析发现,引入改进的 TLS 指纹数据的实验组相比对照组多发现的异常行为主要是分布式恶意黑客行为,其变换了 IP 地址、Cookie 和 User-Agent 值,伪装成正常流量,但是其 TLS 指纹距离小于 3,通过对比 TLS 指纹 Simhash 值可以确认这是同一个客户端经过高匿名代理的行为数据。从实验结果还可以看到,改进的 TLS 指纹具备一定的追踪溯源的能力,在检测到威胁后,能根据指纹距离查找可以发现具有威胁的客户端的其他行为和其使用过的代理 IP;如果攻击者在开始探测时并没有使用代理服务器,则其通过时间轴追溯可以在一定几率下找到攻击者的真实 IP 来源。通过改进的 TLS 指纹关联和时间轴追溯,还能发现黑客的行为轨迹,为安全事件调查提供情报上下文。如图 4 所示,黑客先使用了探测目录的信息搜集手段,然后变换 IP 地址进行 SQL 注入漏洞探测。

```
timestamp source_ip uri
1548430429 125.x.180.62 /favicon.ico
…
1548430431 125.x.180.62 /3/favicon.ico
1548430432 125.x.180.62 /
…
1548431782 5.x.40.103 /all?page= 2'vbFEkk(")YjehAV
1548431784 5.x.40.103 /all?page=2) AND 8728=4333 AND…
…
1548431806 5.x.40.103 /all?page=2%' AND 6413=6338…
1548431807 5.x.40.103 /all?page=(SELECT (CASE WHEN…
```

图 4 黑客试探行为轨迹

Fig. 4 Hacker probe behavior track

结束语 本文详细分析了 TLS 指纹的原理,对比分析了不同软件、不同库产生的 TLS 指纹,并提出了使用 2-gram 和 Simhash 对指纹进行哈希化,得到精简有效且可计算距离的哈希指纹,解决了原有指纹容错性低的问题。最后将改进的 TLS 哈希指纹和其他行为数据整合,通过卷积神经网络进行用户行为安全分析识别,验证了在本文所处的实验环境中,TLS 指纹数据可将用户行为分析检测不可信用户的准确率提高1.8%;而改进的 TLS 指纹数据可将准确率提高 4.2%,并且通过计算指纹距离可以为安全事件调查提供情报上下文。虽然改变 TLS 指纹需要改动底层库,要求攻击者具备更高的能力,但所提方法仍然存在一个缺陷,即存在修改底层库达到 TLS 指纹欺骗的可能。因此 TLS 指纹更应该作为一种辅助手段。探讨 TLS 指纹在其他领域的作用,如基于 TLS 的追踪溯源、行为模式追踪研究等,将是未来研究的重点。

参 考 文 献

[1] YONG B, LIU X, LIU Y, et al. Web Behavior Detection Based on Deep Neural Network[C]// 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing. IEEE, 2018: 1911-1916.

[2] PENG T, QIU W D, ZHENG H, et al. SQL Injection Behavior Mining Based Deep Learning[C]// Proceedings of 14th International Conference. Nanjing, China, 2018.

[3] ECKERSLEY P. How unique is your web browser? [C]// Proceedings of the 10th International Conferenceon Privacy Enhancing Technologies. Berlin: Springer, Heidelberg, 2010: 1-18.

[4] NAKIBLY G, SHELEF G, YUDILEVICH S. Hardware fingerprinting using HTML5[J]. arXiv: 1503.01408, 2015.

[5] CAO Y Z, LI S, WIJMANS E. Browser fingerprinting via OS and hardware level features[C]// Proceedings of Network & Distributed System Security Symposium (NDSS). 2017.

[6] GOOGLE. HTTPS encryption on the web [EB/OL]. https://trans parencyreport. google. com/https/overview.

[7] W3TECHS. Usage of Default protocol https for websites[EB/OL]. https://w3techs. com/technologies/details/ce-httpsdefault/all/all.

[8] IVAN. Examples of the information col- lected from SSL handshakes [EB/OL]. http://blog. ivanristic. com/2009/07/examples-of-the-information-collected-from-ssl-handshakes. html.

[9] MAREK. SSL fingerprinting for p0f [EB/OL]. https://idea. popc ount. org/2012-06-17-ssl-fingerprinting-for-p0f

[10] LEE B. Stealthier Attacks & Smarter Defending with TLS Fingerprint [EB/OL]. http://blog. squarelemon. com/tls-fingerprinting.

[11] HUSÁK M, ČERMÁK M, JIRSÍK T, et al. HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting[J]. EURASIP Journal on Information Security, 2016, 2016(1): 6.

[12] ALTHOUSE J. Open Sourcing JA3 [EB/OL]. https://engineering. salesforce. com/open-sourcing-ja3-92c9e53c3c41.

[13] DIERKS T, RESCORLA E. The transport layer security (TLS) protocol version 1. 2[OL]. https://datatracker. ietf. org/doc/rfc5246/.

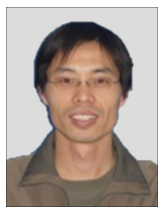
[14] GOOGLE. Applying GREASE to TLS Extensibility, IETF Draft [OL]. https://mailarchive. ietf. org/arch/msg/ietf-announce/15r5EP6SEBb8zA-T5UoeMo5OFyg/.

[15] ZHANG M, XU B Y, BAI S, et al. A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN[C]// International Conference on Neural Information Processing. Springer, Cham, 2017: 828-836.

[16] SAXE J, BERLIN K. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys[J]. arXiv: 1702.08568, 2017.

[17] LE H, PHAM Q, SAHOO D, et al. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection[J]. arXiv: 1802.03162, 2018.

[18] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems. 2012: 1097-1105.



HU Jian-wei, born in 1973, Ph.D, associate professor. His main research interests include cyberspace security and so on.