

面向高速网络流量的加密混淆型 WebShell 检测

曹艺怀¹, 陈伟¹, 张帆², 吴礼发¹

(1. 南京邮电大学计算机学院, 江苏 南京 210023; 2. 武汉轻工大学数学与计算机学院, 湖北 武汉 430048)

摘要: WebShell 是一种常见的 Web 脚本入侵工具。随着流量加密和代码混淆等技术的逐渐发展, 使用传统的文本内容特征和网络流特征进行匹配的检测手段越来越难以防范生产环境下复杂的 WebShell 恶意攻击事件, 特别是对于对抗性样本、变种样本或 0Day 漏洞样本的检测效果不够理想。搭建网络采集环境, 在高速网络环境中利用数据平面开发套件 (DPDK, data plane development kit) 技术捕获网络数据包, 标注了一套由 1 万余条不同平台、不同语言、不同工具、不同加密混淆方式的 WebShell 恶意流量与 3 万余条正常流量组成的数据集; 通过异步流量分析系统框架和轻量型日志采集组件快速地解析原始流量, 并融合专家知识深度分析几种流行的 WebShell 管理工具通信过程中的 HTTP 数据包, 从而构建面向加密混淆型 WebShell 流量的有效特征集; 基于该有效特征集使用支持向量机 (SVM, support vector machine) 算法实现对加密混淆型 WebShell 恶意流量的离线训练和在线检测。同时, 利用遗传算法改进参数搜索方式, 克服了由人工经验设置参数方位以及网格搜索陷入局部最优解的缺点, 模型训练效率也得到提升。实验结果显示, 在自建的 WebShell 攻击流量数据集上, 保证了检测高效性的同时, 检测模型的精确率为 97.21%, 召回率为 98.01%, 且在对抗性 WebShell 攻击的对比实验中表现良好。结果表明, 所提方法能够显著降低 WebShell 攻击风险, 可以对现有的安全监控体系进行有效补充, 并在真实网络环境中部署和应用。

关键词: WebShell; DPDK; 流量分析; 异常检测

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022055

Encrypted and obfuscation WebShell detection for high-speed network traffic

CAO Yihuai¹, CHEN Wei¹, ZHANG Fan², WU Lifa¹

1. School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

2. School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430048, China

收稿日期: 2021-11-18; **修回日期:** 2022-01-07

通信作者: 陈伟, chenwei@njupt.edu.cn

基金项目: 国家重点研发计划 (2019YFB2101704); 湖北省自然科学基金 (2020CFB761)

Foundation Items: The National Key R&D Program of China (2019YFB2101704), The National Science Foundation of Hubei Province (2020CFB761)

引用格式: 曹艺怀, 陈伟, 张帆, 等. 面向高速网络流量的加密混淆型 WebShell 检测[J]. 网络与信息安全学报, 2022, 8(4): 119-130.

Citation Format: CAO Y H, CHEN W, ZHANG F, et al. Encrypted and obfuscation WebShell detection for high-speed network traffic[J]. Chinese Journal of Network and Information Security, 2022, 8(4): 119-130.

Abstract: With the gradual development of traffic encryption and text obfuscation technologies, it is increasingly difficult to prevent complicated and malicious WebShell attack events in production environment using traditional detection methods based on text content and network flow features, especially for adversarial samples, variant samples and 0Day vulnerability samples. With the established network traffic collection environment, DPDK technology was used to capture network traffic in the high-speed network environment, and a dataset was marked with label. The dataset consisted of more than 24,000 normal traffic and more than 10,000 malicious WebShell traffic under different platforms, different languages, different tools, different encryption and obfuscation methods. Then Asynchronous traffic analysis system framework and lightweight log collection components were used to efficiently parse raw traffic. Expert knowledge was integrated to analyze HTTP data packets during the communication process of several popular WebShell management tools, and the effective feature set for encrypted and obfuscation WebShell was obtained. Support Vector Machine (SVM) algorithm was used to realize offline training and online detection of complicated WebShell malicious traffic based on the effective feature set. Meanwhile, improving the parameter search method with the genetic algorithm promoted the model training efficiency furthermore. The experimental results showed that the detection efficiency can be guaranteed based on the self-built WebShell attack traffic dataset. Besides, the detection model has a precision rate of 97.21% and a recall rate of 98.01%, and it performed well in the comparative experiments of adversarial WebShell attacks. It can be concluded from the results that the proposed method can significantly reduce the risk of WebShell attack, effectively supplement the existing security monitoring system, and be applied in real network environments.

Keywords: WebShell, DPDK, traffic analysis, abnormal detection

0 引言

随着互联网技术的广泛应用和快速发展, Web 应用程序已经渗透到各行各业中, 政府、教育机构、大型企业、电商和社交网络等建立了网上服务站点, 以便开展业务和用户浏览信息, 由此网站的信息量和规模在不断增大, 随之带来的 Web 安全问题日益严重。国家互联网应急响应中心 (CNCERT) 在 2021 年 5 月发布的《2020 年我国互联网网络安全态势综述》^[1] 中指出, 2020 年度境内外 2.6 万个 IP 地址对我国境内约 5.3 万个网站植入 WebShell。WebShell 俗称网站后门, 攻击者通过上传 WebShell 到网站目录获取目标权限。随着现代红蓝对抗逐渐升级, 无论是在企业日常安全演练, 还是黑灰产非法侵入、破坏基础信息设施等场景中, 攻击者越来越频繁地使用如蚁剑 (AntSword)、冰蝎 (Behinder)、哥斯拉 (Godzilla) 等流行的国产 WebShell 管理工具, 来实现主机权限获取、内网横向移动、数据窃取修改等大量恶意攻击行为。因此, 研究出可以准确、快速地检

测面向不同类型的加密混淆型 WebShell 管理工具的攻击流量对网站安全和用户信息安全有着重要的现实意义。

基于目前的网络环境和技术手段, 针对 WebShell 的检测和分析技术, 通常是主机端特征对比和敏感函数发现等静态检测方式, 这种基于规则匹配的方法仅对已知的 WebShell 样本具有较高检出率, 对于一些未知或混淆加密的 WebShell 样本检测召回率较低、普适性较差。另外, 通过结合机器学习和基于访问情况、行为模式等动态检测机制在网络流量侧发现异常行为的检测方式面临着特征选取不充分、分类器效果差等问题。

针对这些挑战, 本文的主要工作如下。

1) 标注了一套面向 HTTP 的 WebShell 恶意流量检测的标注数据集, 并上传共享。数据集由两部分组成: ① 1 万条以上的 HTTP WebShell 恶意流量, 包括不同系统、不同语言、不同工具、不同混淆加密方式的 WebShell 攻击流量; ② 2 万条以上的 HTTP 正常流量, 涵盖了主流的 HTTP 协议应用类型。

2) 提出一种融合专家知识的加密混淆型 WebShell 恶意流量特征提取方法。利用异步流量分析系统框架快速地解析流量数据得到包含若干信息字段的文本, 再结合关于加密混淆型 WebShell 攻击特征、绕过方式等专家知识, 提取有效的关键信息字段。本文基于对攻击和数据的理解提出的提取特征方法, 不依赖复杂的机器学习算法就可以达到较好的检测效果。

3) 设计并实现了一套基于高速网络环境的加密混淆型 WebShell 恶意流量识别的原型系统, 自动识别 WebShell 恶意流量。将该系统部署在服务器上, 使用 DPDK 技术提供的模块捕获网络流量数据包, 针对加密混淆型 WebShell 通信特征, 实现对不同种类的 WebShell 恶意流量进行检测。实验结果表明, 该原型系统精确率达到 98.51%, 召回率达到 99.15%, 并且可以识别出部分未知加密混淆方式的 WebShell 恶意流量。

1 相关工作

基于目前的网络环境和技术手段, 研究学者针对 WebShell 的检测方法总体上分为 3 类, 分别从基于文本内容、Apache/Nginx 日志内容和网络流量进行了广泛而深入的研究。

第 1 种基于 WebShell 文本内容的检测方法大致可以划分为静态检测和动态检测。静态检测主要是利用统计学方法, 如文件中的特征代码、信息熵、最长单词、重合指数、压缩等特征进行异常检测。动态检测是通过监控代码行为来判断是否为 WebShell, 借助动态污点追踪技术检测或阻断风险操作。其中, 被广泛提到的是 Hagen 于 2011 年设计实现的 NeoPI^[2], 这款工具对某些混淆、变形的 WebShell 文件具有很好的识别效果, 但是只做定量检测, 并不对文件做定性评价, 并且存在误报率和漏报率较高的问题。另外, 基于文件内容的检测需要在检测前获取 WebShell 文本, 这使该技术在针对无文件 WebShell 进行检测时存在瓶颈。

第 2 种基于 Web 日志的检测方法通过分析日志内容, 可以溯源取证整个攻击过程, 比较适合分析较为复杂的、未被发现的 WebShell 的绕过策略。石刘洋等^[3]在检测日志文本特征的同时, 提

出结合统计特征以及文件访问之间的关联特征。通过研究发现, 日志中对 WebShell 的访问频率记录与正常记录存在明显的差异, 利用这种差异性可以有效提升日志检测的准确率。基于日志的检测技术可以利用上下文信息获取攻击者的意图和确切行为, 但是由于完整的日志一般记录在攻击行为发生之后, 这种检测方法存在一定的滞后性, 需要进一步研究如何结合机器学习的优势及时地预测和阻断攻击者的下一步非法行为。

第 3 种基于网络流量的检测方法通过流量数据分析行为, 结合访问日志、数据包内容等信息进行建模分析, 从中挖掘可能出现的异常点。文献[4-5]从隐匿性、接口特性和逃逸检测等方面进行分析, 通过部署蜜罐对恶意样本进行观察, 总结了不同类型的恶意样本在执行敏感操作时的客户端行为和服务端行为。然而, 针对流量技术的检测面临流量噪声、实时性能、特征维度、样本不均衡等多方面因素的制约^[6]。此外, 根据加密通信流量的情况可能会进一步提高检测难度。

2 WebShell 数据集与特征提取

2.1 加密混淆型 WebShell 数据集

关于 WebShell 检测的公开数据集大多聚焦于恶意文件样本, 而缺少 WebShell 通信流量的数据集。基于当前红蓝对抗的现状和 WebShell 管理工具的使用习惯, 本文从两大系统、3 种工具、3 种程序语言、2 种模式作为切入点, 广泛地采集不同环境下的 WebShell 通信流量数据。如图 1 所示, 两大系统分为 Windows 系统和 Linux 系统, 3 种 WebShell 管理工具分为蚁剑、冰蝎和哥斯拉, 3 种程序语言分为 PHP、JSP 和 ASP, 2 种模式分为原始模式和混淆模式^[7]。其中, 混淆模式又分为特征混淆和编码混淆, 依据不同的 WebShell 管理工具的特点, 具体的混淆模式如表 1 所示。为了尽可能地模拟真实攻击环境, 本文构建了存在若干漏洞的蜜罐来捕获和分析 WebShell 攻击行为, 并且邀请了数位有丰富渗透测试经验的安全技术人员来攻击这个蜜罐, 即通过某种漏洞拿到靶机环境权限, 进而使用上述 3 种常用的

WebShell 管理工具进行命令执行、文件浏览、权限维持、特权提升和横向移动等恶意攻击行为^[8]。最后，部署在靶机的流量采集模块共捕获到 11 408 条 WebShell 通信流量作为恶意样本，其中包含未混淆的 WebShell 流量 5 375 条，加密混淆后的 WebShell 流量 6 033 条，并且从真实网络环境中收集到正常 HTTP 通信流量 24 788 条作为正常样本，包含注册、登录、增删改查等正常业务请求。

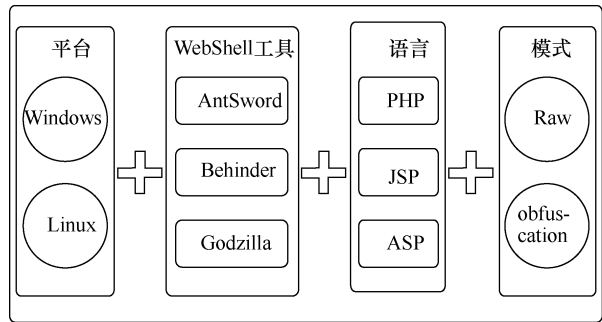


图 1 数据集的不同组合方式
Figure 1 Different combination of dataset

表 1 不同 WebShell 攻击的混淆模式
Table 1 Confusion types of different WebShelltools

WebShell 工具	特征混淆	编码混淆
AntSword	增加若干 HTTP Header 和 Body Data 信息	BASE64
	使用随机英文单词作为参数名	CHR
	在 HTTP Body 中增加垃圾数据	CHR13
	使用 multipart 格式进行数据传输	ROT13
Behinder	增加若干 HTTP Header 信息	AES_BASE64
	在 HTTP Body 增加若干 leftData 和 rightData	XOR_BASE64
Godzilla	增加若干自定义的 HTTP Header 信息	无

2.2 加密混淆型 WebShell 特征工程

加密混淆型 WebShell 特征工程主要分为 3 个模块，分别是解析流量数据、采集和标准化流量日志以及流量特征提取。

2.2.1 解析流量数据

解析流量数据主要功能为基于 Zeek^[9]入侵检测系统的流量解析和原始日志输出，大致流程如图 2 所示。

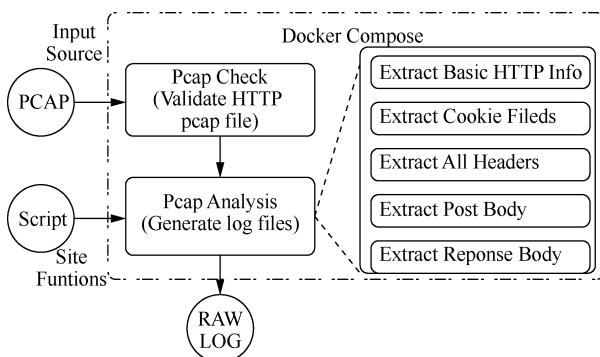


图 2 解析流量数据流程
Figure 2 Flow of analyzing pcap file

首先子模块会监控输入源，获取新产生的网络流量 PCAP 文件，然后为每一个 PCAP 文件创

建一个子进程，在子进程中，模块加载默认的 HTTP 协议数据提取脚本，以及由用户自定义编写的 Zeek 功能拓展脚本，调用 Zeek 入侵检测系统对 PCAP 文件进行解析和处理，得到由 Zeek 日志系统生成的日志文件。根据 WebShell 与正常文件的连接特征和行为特征之间的差别，该模块在 /zeek/scripts/sites/_load_.zeek 配置文件中引入了 4 类 Zeek 脚本，对系统的功能进行了拓展。

(1) 记录 HTTP 请求头的 COOKIE 字段

在针对 WebShell 连接行为的流量分析中，HTTP 请求头的 COOKIE 字段是一个重要考察点。正常网站的访问过程中会存在多个不同的有意义的 COOKIE 字段，如记录会话信息的 SessionID、记录身份认证信息的 Token 以及时间戳等其他信息。WebShell 由于其功能较为单一，在不混淆的前提下，其连接过程中通常只会有一个 COOKIE 字段，即 SessionID 字段。脚本从 HTTP 请求连接的流量中提取得到请求头，并将请求头的 COOKIE 字段添加到 http.log 日志文件中。

(2) 记录 HTTP 请求头的自定义字段

根据 RFC2616 和 RFC7230 的标准定义，

HTTP 请求头中的字段具有注册认证的作用。同样,网站开发者会根据具体的需求自定义字段,这些字段包含关于 HTTP 流量的重要信息。而 WebShell 只是一个单独的脚本文件,会包括一些基本的 HTTP 请求头字段。与上一脚本类似,脚本从请求头中提取出自定义的字段,并将其记录在 http.log 日志文件中。

(3) 记录 HTTP 连接中 POST 请求数据

POST 是 HTTP 请求方法之一,用于向服务端提交用户数据。根据 RFC2616 的标准定义,分析 POST 请求包的传输数据可以判断网络中上载的行为是否存在恶意攻击。因此,通过 http_entity_data 事件脚本提取 POST 方法传输的数据内容,将其记录在 http.log 日志文件中。

(4) 记录 HTTP 连接中 RESPONSE 返回数据

RESPONSE 数据包是服务端返回给客户端的数据,可以分析返回数据是否包含命令执行、SQL 注入等恶意攻击的结果。与上一个脚本类似,通过 HTTP 连接中传输的协议包实体数据,脚本会取出 RESPONSE 数据包的内容,并记录在 http.log 日志文件中。

经过上述脚本处理后,一个 HTTP 协议的流量数据包会解析成一条包含 23 个字段的 JSON 格式记录,每个 PCAP 文件生成一个 http.log 日志文件,其形式如图 3 所示。

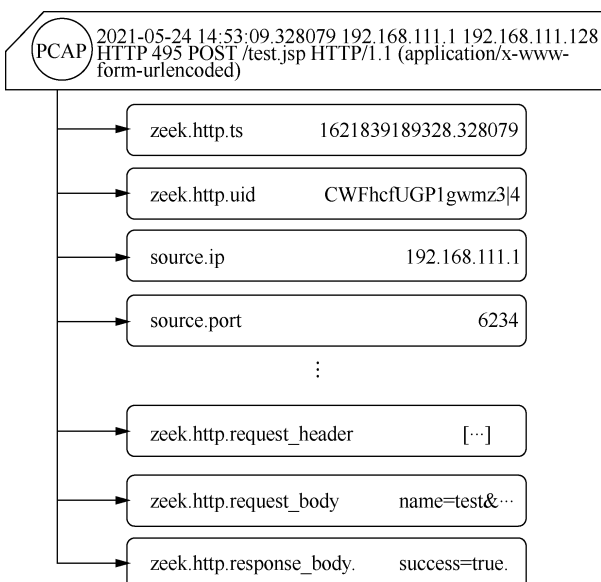


图3 http.log 日志文件示意
Figure 3 Schematic diagram of http.log file

2.2.2 采集和标准化流量日志

采集和标准化流量日志模块选用 Filebeat^[10] 采集日志和 Logstash^[11] 处理日志的技术栈。Filebeat 是使用 Golang 实现的轻量型日志采集器,它会监视配置中指定的日志文件或位置,收集日志事件,Filebeat 会采集由上一个模块中 Zeek 产生的 http.log 日志文件,并将它们转发到 Logstash 组件进行下一步处理。Logstash 是一款开源的服务器端日志处理管道,它能同时对来自不同数据源的数据进行采集,通过编写自定义的规则,对采集到的数据进行处理和转换,并配置到不同输出源,如文件、数据库和 Kibana 等组件。Logstash 主要负责将 Filebeat 传输的文件做如下过滤步骤。

步骤 1 将由 Unix 时间戳 (Unix timestamp) 表示的时间格式转换为更加简洁明了、易读的 UTC+8:00 的时间格式。

步骤 2 将不易识别、较为复杂的字段名重命名为简单易懂的字段,如“source.ip”转换成“src_ip”,“source.port”转换成“src_port”,“http.request.body.bytes”转换成“request_body_len”等。

步骤 3 去除由 Filebeat 组件产生的不必要的字段,如“fileset”“event”“network”等字段,减少日志文件存储空间的压力。

采集和标准化流量日志模块均使用 Docker 实现配置和运行,Filebeat 和 Logstash 两个组件的大致工作流程如图 4 所示。

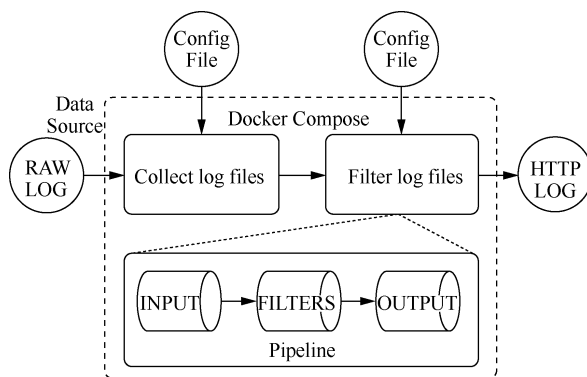


图4 采集和标准化流量日志流程
Figure 4 Flow of collecting and standardizing traffic logs

2.2.3 流量特征提取

流量特征提取模块提出了对 HTTP 流量进行特征工程的转化方案。经过数据清洗和格式化后,

提取流量数据的内容特征,生成用于模型识别的特征向量,如表 2 所示。

表 2 加密混淆型 WebShell 流量特征
Table 2 Features of encrypted and obfuscation WebShell flow

特征类别	特征名称
基于文本的特征	1) URL 特征;
	2) 请求流量中是否有 referer 字段;
	3) 请求流量中 COOKIE 个数;
	4) 请求和返回流量中 HTTP 首部字段个数;
	5) POST BODY 长度;
	6) POST BODY 参数个数;
	7) POST BODY 是否是特殊编码方式
基于统计的特征	1) POST BODY 信息熵;
	2) RESPONSE BODY 非数字字母的比率;
	3) POST BODY 最长参数长度占总长度比率;
	4) POST BODY 重合指数

(1) 基于文本的特征

① URL 特征。出于攻击隐匿性的考虑, WebShell 落地文件通常位于敏感的目录(如“images/”“css/”等),文件通常以敏感名称命名(如“shell.php”“1.php”等),文件后缀一般是 ASP、JSP 和 PHP,另外,为了规避上传检测,会用类似“php%00jpg”的混淆后缀名。

② 请求流量中是否有 referer 字段。referer 字段包含了当前请求页面的来源页面或站点,即表示当前页面是通过此来源页面中的链接进入的。而 WebShell 脚本的逻辑较为简单,通常是单页面的形式,所以一般情况下不存在 referer 首部字段。

③ 请求流量中 COOKIE 个数。由于 WebShell 通常只会携带一个或两个 COOKIE。

④ 请求和返回流量中 HTTP 首部字段个数。WebShell 只是一个单独的脚本文件,一般只会包括一些基本的 HTTP 请求头字段,如“Host”“Accept-Encoding”“User-Agent”“Content-Type”和“Content-Length”等。

⑤ POST BODY 长度。即 HTTP 首部字段 Content-Length 的值。

⑥ POST BODY 参数个数。通常 WebShell 的功能是基于命令执行的方式,所以参数个数较少,甚至在冰蝎、哥斯拉等 WebShell 管理工具的

通信流量中不存在参数,而是以“application/octet-stream”内容编码方式进行数据传输。

⑦ POST BODY 是否是特殊编码格式。攻击者往往会利用不同的信息编码方式(如 base64、rot13、chr16、chr32 等)来逃避基于关键字特征的内容检测。

(2) 基于统计的特征

① POST BODY 信息熵。信息熵是描述一个系统混乱程度的指标。加密混淆型 WebShell 相对于其他普通的 WebShell 而言,最大的不同在于前者的流量内容是基于端到端的 AES 算法或 XOR 运算加密通信。由此,相对于正常流量而言,加密混淆型 WebShell 流量的信息熵相对较大。

② RESPONSE BODY 非字母数字比率。和上一个特征相同,由于 WebShell 流量加密导致其返回包的流量中会出现很多不可见字符,如图 5 所示。

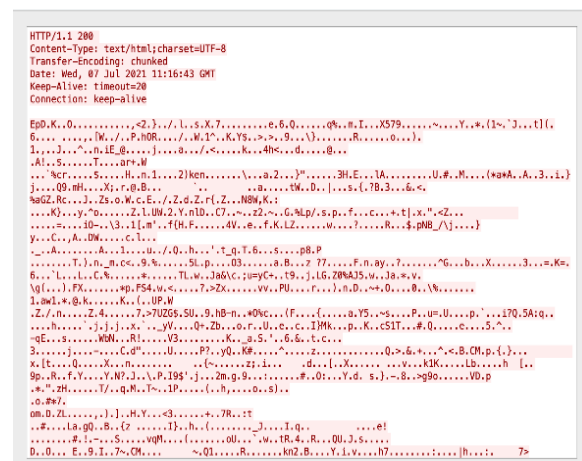


图 5 加密混淆型 WebShell 的返回数据包
Figure 5 Response body of encrypted and obfuscation WebShell

③ POST BODY 最长参数长度占总长度比率。由于 WebShell 的 POST 请求包数据内容较多,参数个数较少,所以往往凸显出最长参数长度占总长度的比率高特点。

④ POST BODY 重合指数(IC, index of coincidence) [12]。

重合指数是衡量频率分布和均匀分布相似程度的度量,可以用来判断文件是否被加密的一种方法,一个完全随机的英文文本的 IC 值接近 0.038 5,而有意义的英文文本的 IC 值接近 0.067。

如果某条流量的 POST BODY 数据内容经过某种加密方式的处理, 其 IC 值会接近 0.038 5。

3 面向流量的加密混淆型 WebShell 检测

本文设计的加密混淆型 WebShell 检测系统整体架构分为 3 个部分, 分别是流量采集模块、流量处理模块和流量检测模块。流量采集模块主要负责在网络出口处把用分光器镜像出来的流量从核心交换机传输到 DPDK^[13]应用服务器, 然后通过 DPDK 应用服务器实现在高速网络流量环境对数据包的捕获。流量处理模块由

Docker 容器搭建, 主要负责通过 Zeek 开源工具对流量数据包被动式解析成 JSON 形式的文本, 然后利用 Filebeat 轻量型日志采集器实时读取 JSON 文本, 并将采集的文本输入 Logstash 日志处理器中对数据清洗和格式化, 最后对数据包的内容进行深度分析, 通过特征挖掘等方式从不同维度提取有效的检测特征。流量检测模块主要是将上一步特征工程的结果应用到改进后 SVM 模型上实现对高速网络流量中 WebShell 行为的分析、检测和溯源。加密混淆型 WebShell 检测流程如图 6 所示。

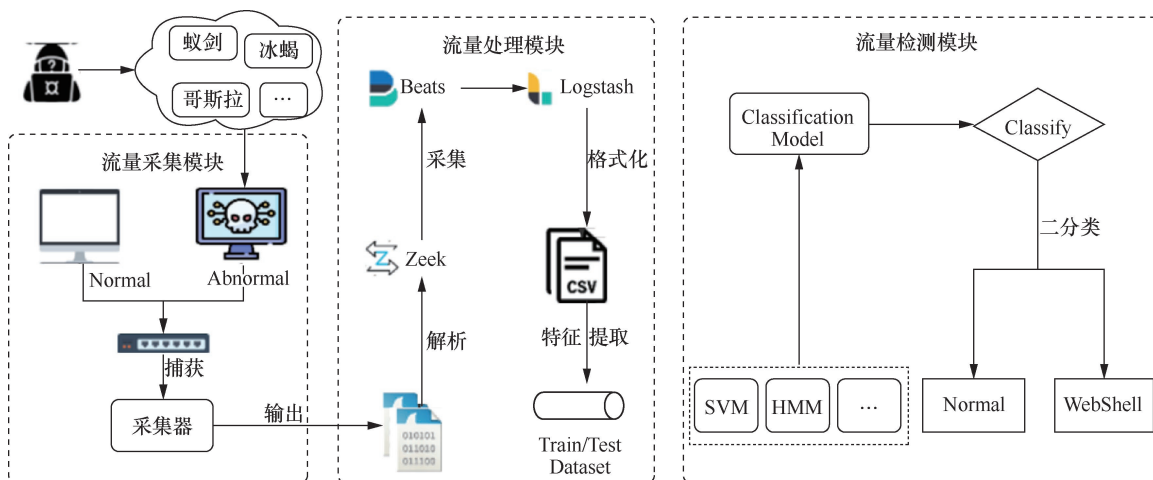


图6 加密混淆型 WebShell 检测流程
Figure 6 Flow of encrypted and obfuscation WebShell detection

3.1 算法原理

支持向量机 (SVM) 由 Vapnic 与其同事首先提出, 是一种基于分类和回归分析的监督型数据分类学习模型算法。SVM 算法是从线性可分情况下广义最优分类面发展而来, 在预测小样本和非线性数据的训练拟合中拥有很多特有的优势。基本思想如图 7 所示, 通过在正例和反例间构建最大化隔离边缘, 由此转换为凸二次规划问题求解。

3.2 参数优化

在机器学习模型的训练过程中, 其训练效果和性能受模型的超参数影响, 根据专家经验, 支持向量机中比较重要的 3 个参数为惩罚系数“C”、核函数“kernel”和核函数参数“r”。主流的核函数有 4 种, 分别是线性核函数“Linear”、多项式核函数“Polynomial”、径向基核函数“RBF”

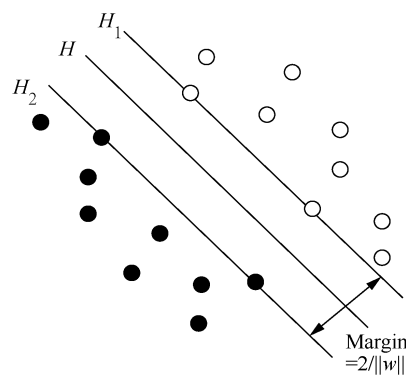


图7 二维空间中的分类
Figure 7 Classification in two-dimensional space

和生长曲线核函数“Sigmoid”。RBF 核函数不易受训练样本数量及分类特征维数的影响, 计算复杂度低, 因此本文选取 RBF 核函数。惩罚系数的主要作用是调节特征子空间中支持向量机的置信范围与经验风险的比值, 使模型的泛化能力达到

最佳。核函数的参数决定着函数的径向尺度范围， r 参数设置过小会导致欠拟合现象，设置过大会降低模型的泛化能力^[14]。

在寻找 SVM 参数最优解时，有时会选择经验法、试探法等偶然性较强的方法，更多时候是用网格搜索法。然而，网格搜索法的主要缺点是：当搜索区间和搜索步长确定之后，就只能在一些已确定的离散点中寻找最优解，而很可能错过更优解。

寻找 SVM 最优参数的实质是二次凸规划问题，即求函数的最优值，而遗传算法最主要的用处之一就是函数最优化，是一种高效的随机搜索算法，不会将搜索范围限定在某一些解中，克服了网格搜索的缺点。所以，本文利用一种改进遗传 SVM 参数优化算法获取参数的最优解，缩短搜索时间。首先，该改进算法通过网格搜索方法快速搜索初始参数范围；然后，根据初始参数范围，利用原始遗传算法进行精细搜索，从而克服了原始遗传算法的初始搜索空间根据人工经验设置以及网格搜索陷入局部最优解的缺点，改进遗传 SVM 算法参数优化流程如图 8 所示。

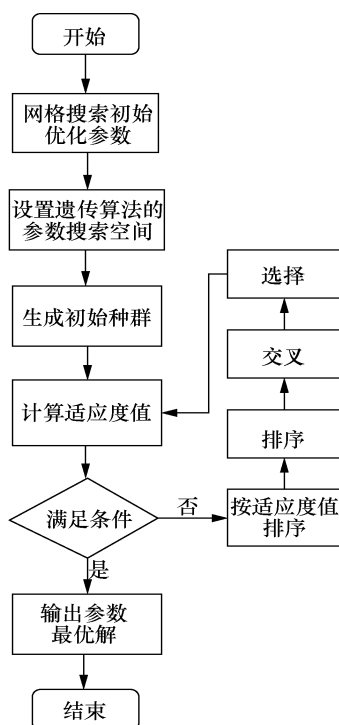


图 8 改进遗传 SVM 算法参数优化流程
Figure 8 Improved genetic SVM algorithm parameter optimization process

3.3 算法适用性

SVM 通过选取少量支持向量的方法确定超平面，支持向量最靠近分类决策面的所有样本的集合，只占有所有样本中很小部分，是全体样本中最具代表性的样本集合，即当训练样本较小时，仍然可以对测试集数据有较高的分类准确率。因此，将支持向量机理论应用于加密混淆型 WebShell 检测，可以在先验数据不足的情况下，SVM 算法的分类能力依然较强，从而使加密混淆型 WebShell 检测系统的检测能力得到保障。

4 实验与分析

4.1 实验环境

实验环境分为训练环境、靶机环境和攻击环境。训练环境的操作系统：MacOSBigSur 11.5.2；处理器：2 GHz 四核 Intel Core i5；内存：16 GB 3733 MHz LPDDR4X；脚本解释器：Python3.6；第三方 Python 模块：scikit-learn v0.23.3、numpy v1.19.2 和 pandas v1.1.5。

靶机环境均使用 VMwareWorkstationPro15 搭建，其中 Windows 系统环境使用 Windows 10 20H2，64 位；Linux 系统环境使用 Ubuntu 20.04 LTS；PHP 环境使用 PHP 7.1.2；Java 环境使用 JDK 8u71；ASP.NET 环境使用 .NET Framework 3.5。

攻击环境的操作系统：Windows 10；处理器：AMD A8-8600P Radeon R6，10 Compute Cores 4C+6G 1.6GHz；WebShell 管理工具：AntSword v2.1.14，Behinder v3.0 Beta11，Godzillav3.03。

4.2 评价指标

本文使用十折交叉验证，即数据集平均分为 10 份子集，轮流将其中 9 份作为训练集，1 份作为测试集，取平均 10 次交叉验证的准确率作为评判结果。训练数据是不平衡的，往往会导致在测试时准确率较高但召回率很低。为了解决这一问题，本文选用 $F1$ 分数 ($F1$ score) 作为评估指标。 $F1$ 分数是一种在统计学中用于衡量二分类模型效果的一种指标，它同时兼顾了模型的准确率和召回率，可以视作上述两种指标的调和平均。利用混淆矩阵表示数据的分类结果如表 3 所示。

表3 混淆矩阵
Table 3 Confusion matrix

分类	实际异常	实际正常
预测异常	TP	FP
预测正常	FN	TN

根据表3,精确率(Precision)、召回率(Recall)和 $F1$ 分数的定义如式(2)~式(4)所示。

$$\text{Precision} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FP_i} \quad (2)$$

$$\text{Recall} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FN_i} \quad (3)$$

$$F1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

4.3 实验结果与分析

为了验证本文提出的基于加密混淆型 WebShell 流量检测算法的有效性,设计如下三组实验。

第一组实验目的是对比使用网格搜索^[15]和改进后参数优化算法之后 SVM 训练达到收敛时所使用的迭代次数。如图9所示,经网格搜索方法寻找最佳参数的 SVM 训练达到收敛时迭代次数大约是80次;通过改进原始遗传算法,加快了 SVM 训练收敛速度,大约迭代40次即可收敛,有效提高了参数优化速度,降低了 SVM 训练时间。其中,遗传算法的参数设定为:最大遗传代数 MAXGEN=300,种群数量 NIND=50,变异率 pm=0.01,交叉率 pc=0.6。

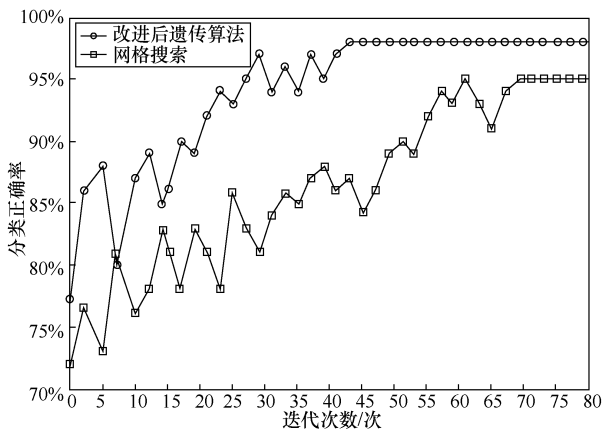


图9 经网格搜索和改进后遗传算法优化的 SVM 训练达到收敛时迭代次数

Figure 9 The number of iterations when the SVM training optimized by the grid search and improved genetic algorithm reaches convergence

第二组实验对比了使用 CICFlowMeter^[19]工具自动提取特征和本文提出的融合专家知识的加密混淆型 WebShell 恶意流量特征提取方法时相同数据集的 $F1$ 分数。CICFlowMeter 是一款网络流量生成器,它可用于生成双向流的80多维流特征,如持续时间、数据包数、字节数、包的长度等常见流相关特征。

如表4所示,在使用相同数据集的情况下,经 CICFlowMeter 提取特征后 $F1$ 分数是81.76%;使用本文提出的特征提取方法后 $F1$ 分数是98.17%。后者的 $F1$ 分数比前者高可能在于 CICFlowMeter 偏向于分析网络流本身的特征,而忽略了网络数据包的内容,并且部分 WebShell 恶意流量与正常流量在流特征上的差异不太明显。

表4 不同特征工程 WebShell 检测能力对比
Table 4 Comparison of WebShell detection capabilities of different feature engineering

特征工程	精确率	召回率	$F1$ 分数
本文方法	97.45%	98.91%	98.17%
CICFlowMeter	84.13%	79.52%	81.76%

第三组实验将数据集分为两组 DatasetA 和 DatasetB, DatasetA 是未经过混淆的原始数据流量, DatasetB 是经过不同混淆方式处理后的数据流量。将实验内容分为3个部分:第一部分实验以 DatasetA 作为训练集和测试集,评估支持向量机和隐式马尔可夫两种不同的算法在此数据集的检测效果;第二部分实验是在使用 SVM 算法的情况下,用 Dataset A 作为训练集预测 DatasetB 的结果和用部分 DatasetB 作为训练集预测剩余部分 DatasetB 的结果;第三部分实验以 DatasetA 和 DatasetB 作为训练集,使用支持向量机算法检测某种未知的加密混淆型 WebShell 流量作为测试集 DatasetC,即该种加密混淆方式没有出现在 DatasetA 和 DatasetB 中。

第一部分实验结果如表5第一行所示,使用相同数据集 DatasetA 的情况下, SVM 算法的性能优于 HMM 算法。使用 SVM 算法的 $F1$ 分数是98.35,而 HMM 的 $F1$ 分数是84.31%,召回率较低可能是因为 HMM 算法是基于马尔可夫性假设,即当一个随机过程在给定现在状态及所有过

去状态情况下,其未来状态的条件概率分布仅依赖于当前状态。

第二部分实验结果如表 5 第二行所示,以未混淆的加密混淆型 WebShell 流量 DatasetA 作为训练集,用 SVM 算法预测混淆后加密混淆型 WebShell 流量 DatasetB 的精确率为 49.20%,召回率为 41.89%, $F1$ 分数是 45.25%。作为对比,

使用十则交叉验证方法划分 DatasetB,平均 10 次交叉验证的 $F1$ 分数为 97.46%。该组实验表明,混淆后的加密混淆型 WebShell 流量与原始数据相比,在内容特征和行为特征等方面均有较大的差异,所以检测模型需要 DatasetB 进一步丰富样本种类以及对加密混淆型 WebShell 的识别能力。

表 5 第三组实验结果
Table 5 Three sets of experimental results

实验组别	模型名称	训练集	测试集	精确率	召回率	$F1$ 分数
第一部分	SVM	Dataset A	Dataset A	97.57%	99.15%	98.35%
	HMM	Dataset A	Dataset A	89.79%	79.46%	84.31%
第二部分	SVM	Dataset A	Dataset B	49.20%	41.89%	45.25%
	SVM	Dataset B	Dataset B	98.74%	96.22%	97.46%
第三部分	SVM	Dataset A+B	Dataset C	97.21%	98.01%	97.61%

第三部分实验目的是验证加入 DatasetB 作为训练集后的模型对未知的加密混淆型 WebShell 流量仍然有优秀的检测效果。首先,蚁剑支持用户自定义插件的功能,其中 x1DecoderPlus 是一种开源的蚁剑流量加密方法,它是基于 XOR 和 Base64 的编码方式,在流量的内容形式上与表 1 所示的蚁剑默认 3 种编码方式完全不同。其次,通过开源的方法对冰蝎进行源码级别的修改,修改后的冰蝎 WebShell 流量去除或变更了原版冰蝎 WebShell 流量的若干固有特征。由于真实环境中,WebShell 流量通常只会占全部流量的百分之几,甚至万分之几,所以本次实验利用 x1DecoderPlus 和修改后冰蝎 WebShell 管理工具共生成 503 条 WebShell 流量和随机选择 2 375 条正常流量作为增量测试集,检测结果如表 5 第三行所示, $F1$ 分数为 97.61%,证实了本文提出的模型有较好的泛化能力,可以检测出相对未知的加密混淆型 WebShell 流量。

通过搭建一个存在历史漏洞的 WordPress 站点来测试本文提出的 SVM 模型的时效性,并将其与其他研究结果进行横向比较。首先,探讨了不同大小的流量数据包对模型运行时间的影响,实验中使用两个流量数据文件,其文件大小分别是 1.4MB 和 88.3MB,并且将两个文件均运行 3 次取平均值

作为实验结果。结果如表 6 所示。当流量数据规模较少时,SVM 模型可以在较短的时间内得到检测结果,这样有助于安全运营人员快速响应潜在的威胁事件。其次,将本次实验结果与 Wu 等提出的基于随机森林算法的 FRF-WD^[17]检测方法作比较。为了保持样本的一致性,验证 FRF-WD 方法的数据集中,WordPress 站点的所有源文件作为正常样本,4 种类型的恶意 WebShell 文件作为异常样本,其中 4 种类型包括一句话 WebShell、利用蚁剑插件生成的加密混淆 WebShell、哥斯拉 WebShell 和冰蝎 WebShell。两种方法的检测效率比较如表 7 所示。

表 6 不同文件大小对实验结果的影响
Table 6 The influence of the different sizes on experimental results

模型名称	文件大小	流量条数	运行时间
SVM	1.4 MB	53 条	4.23 s
	88.3 MB	596 条	43.57 s

表 7 FRF-MD 和本文方法的检测效率比较
Table 7 Comparison of efficiency between FRF-MD and this paper's method

模型名称	文件大小	运行时间
本文方法	88.3 MB	43.57 s
FRF-WD	63.9 MB	41.84 s

由于需要快速处理流量,在算法上选择了

SVM 这种“轻量级”算法；在时间复杂度上，本文利用 Elastic 两大组件 Filebeat 和 Logstash，可以较好地优化流量解析、标准化过程中耗费的时间；在空间复杂度上，本文利用 docker-compose 对各个模块进行编排，耗费内存较小，响应速度较快。即使本文方法的检测对象是流量数据包，需要经过流量解析和日志转换等流程，但总检测时间与 FRF-WD 检测方法的所用时间相近，时效性得到保障。两种方法的检测能力比较如表 8 所示，FRF-WD 检测方法和本文提出的方法都可以检测出较为原始的一句话、小马、大马和未混淆的蚁剑等常规类型的 WebShell 文件，但是前者未能检测出经过混淆后的蚁剑、冰蝎和哥斯拉 WebShell 恶意文件，而本文方法仍然可以有效地检测出受害主机与攻击主机之间的通信流量。

表 8 FRF-MD 和本文方法的检测能力比较
Table 8 Comparison of capabilities between FRF-MD and this paper's methods

WebShell 文件类型	是否混淆加密	FRF-WD	SVM
一句话	否	•	•
小马	否	•	•
大马	否	•	•
蚁剑	否	•	•
	是	○	•
冰蝎	是	○	•
哥斯拉	是	○	•

注：•代表能够检测出 WebShell 攻击，○代表不能检测出 WebShell 攻击。

5 结束语

以往的 WebShell 检测研究多数通过从 WebShell 文件样本中提取大量的语义、语法等特征，并将这些特征作为关键词来判断是否存在 WebShell。随着攻防对抗的日益升级，以及加密和混淆技术的逐渐发展，这些方法很难适用。本文以 HTTP 请求和响应流量作为研究对象，在对蚁剑、冰蝎、哥斯拉 3 种主流的 WebShell 管理工具攻击原理与方法进行深入研究之后，结合专家知识分析 WebShell 管理工具的攻击行为在流量报文上体现出的文本特征和统计特征，利用可以在高速网络环境中捕获 HTTP 流量数据包的

DPDK 技术，采集了上述 3 种工具的加密混淆型 WebShell 流量数据集，构建了加密混淆型 WebShell 流量检测方案，分别进行 WebShell 攻击行为检测和 WebShell 攻击成功识别。实验结果表明，本文提出的融合专家知识的特征提取方案相比 CICFlowMeter 工具自动提取特征的检测能力更为优秀，并且检测模型在面对未知的加密混淆型 WebShell 流量时表现出良好的检出能力。此外，本文的模型在保持较好检测性能的同时，与文中引用的基于文件内容检测的 FRF-WD 方法相比，检测效率并未有明显下降，运行时间只增加了 4%，但能识别出 FRF-WD 方法不能识别出来的潜在 WebShell 攻击威胁。

虽然本文方法能有效地对加密混淆型 WebShell 进行检测，但还存在提升空间。例如，由于请求报文较长，数据包会出现分片传输，这会导致 HTTP 数据包分割成若干个数据包。因此，可以尝试在检测之前将分片的请求包重组到一起再进行检测，以达到更好的检测效果。另外，随着攻防技术的发展，加密混淆型 WebShell 特征可能会出现一些变化，需要日后进一步丰富恶意 HTTP 通信样本的种类。

参考文献：

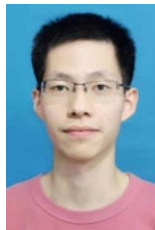
- [1] 国家互联网应急响应中心. 2020 年我国互联网网络安全态势综述[EB].
CNCERT/CC. Overview of China's Internet security posture in 2020[EB].
- [2] BEHRENS S, HAGEN B. Web shell detection using NeoPI [EB].
- [3] 石刘洋, 方勇. 基于 Web 日志的 Webshell 检测方法研究[J]. 信息安全研究, 2016, 2(1): 66-73.
SHI L Y, FANG Y. Research on Webshell detection method based on Web log[J]. Journal of Information Security Research, 2016, (1): 66-73.
- [4] YANG C H, SHEN C H. Implement web attack detection engine with snort by using modsecurity core rules[C]//Fourth the E-Learning and Information Technology Symposium (EITS 09). 2009.
- [5] 宋明璐. 基于主动学习的 CNN Webshell 检测[D]. 北京: 北京交通大学, 2021.
SONG M L. CNN Webshell detection based on active learning[D]. Beijing: Beijing Jiaotong University, 2021.
- [6] 龙啸, 方勇, 黄诚, 等. Webshell 研究综述: 检测与逃逸之间的博弈[J]. 网络空间安全, 2018, 9(1): 62-68.

- LONG X, FANG Y, HUANG C, et al. Webshell research review: the confrontation between detection and escape[J]. Cyber Security, 2018, 9(1):62-68.
- [7] STAROV O, DAHSE J, AHMAD S S, et al. No honor among thieves: A large-scale analysis of malicious web shells[C]//Proceedings of the 25th International Conference on World Wide Web. 2016: 1021-1032.
- [8] 赵运弢, 徐春雨, 薄波, 等. 基于流量的 WebShell 行为分析与检测方法[J]. 网络安全技术与应用, 2018, 18(4): 8-9.
- ZHAO Y T, XU C Y, BAO B, et al. WebShell behavior analysis and detection method based on traffic[J]. Network Security Technology & Application, 2018, 18(4): 8-9.
- [9] HAAS S, SOMMER R, FISCHER M. Zeek-osquery: host-network correlation for advanced monitoring and intrusion detection[C]//IFIP International Conference on ICT Systems Security and Privacy Protection. Springer, Cham, 2020: 248-262.
- [10] SANJAPPA S, AHMED M. Analysis of logs by using logs-tash[C]//Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Springer, Singapore, 2017: 579-585.
- [11] CHEN L, LIU J, XIAN M, et al. Docker container log collection and analysis system based on ELK[C]//2020 International Conference on Computer Information and Big Data Applications (CIB-DA). IEEE, 2020: 317-320.
- [12] HARREMOËS P, TOPSOE F. Inequalities between entropy and index of coincidence derived from information diagrams[J]. IEEE Transactions on Information Theory, 2001, 47(7): 2944-2960.
- [13] 曾理, 叶晓舟, 王玲芳. DPDK 技术应用研究综述[J]. 网络新媒体技术, 2020, 9(2): 1-8.
- ZENG L, YE X Z, WANG L F. Summary of DPDK Technology Application Research[J]. Journal of Network New Media, 2020, 9(2): 1-8.
- [14] 刘奕, 李建华, 张一韬, 等. 基于特征属性信息熵的网络异常流量检测方法[J]. 信息安全, 2021, 21(2): 78-86.
- LIU Y, LI J H, ZHANG Y T, et al. Network abnormal traffic detection method based on characteristic attribute information entropy[J]. Netinfo Security, 2021, 21(2): 78-86.
- [15] HESTERMAN J Y, CAUCCI L, KUPINSKI M A, et al. Maximum-likelihood Estimation With a Contracting-grid Search Algorithm[J]. IEEE Transactions on Nuclear Science, 2010, 57(3): 1077-1084.
- [16] SHARAFALDIN I, LASHKARI A H, GHORBANI A A. Toward generating a new intrusion detection dataset and intrusion traffic

characterization[C]//International Conference on Information Systems Security & Privacy (ICISSP). 2018: 108-116.

- [17] WU Y, SUN Y, HUANG C, et al. Session-based webshell detection using machine learning in web logs [J]. Security and Communication Networks, 2019, 2019

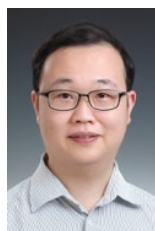
[作者简介]



曹艺怀 (1996-), 男, 湖北黄冈人, 南京邮电大学硕士生, 主要研究方向为 Web 安全、攻防技术和流量分析。



陈伟 (1979-), 男, 江苏淮安人, 南京邮电大学教授, 主要研究方向为无线网络安全、移动互联网安全。



张帆 (1977-), 男, 湖北当阳人, 武汉轻工大学副教授, 主要研究方向为信息系统安全、可信计算和机器学习及安全。



吴礼发 (1968-), 男, 湖北蕲春人, 南京邮电大学教授、博士生导师, 主要研究方向为网络协议逆向、软件安全漏洞挖掘及逆向和入侵检测。