基于 RASP 的 Web 安全检测方法

余航，王帅，金华敏
（中国电信股份有限公司研究院，广东 广州 510630）

摘　要：目前，传统的 Web 安全检测方法作用于程序输入输出端，不能防范经变形混淆后绕过检测进入程序内部执行的恶意代码，难以满足当前 Web 应用安全防护新需求。本方法基于对传统数据流监控方法风险的深入分析，结合 RASP 技术特性，提出了基于 RASP 的 Web 安全检测方法，在 Web 应用程序内部的权限判别函数参数、系统命令执行函数参数、数据库操作函数参数处埋下 RASP 探针，在代码解释器层面实时检测数据流的变化。本方法基于 Java 语言进行了实现，在实验室证明该方法在准确率和检测时间上优于传统的 Web 安全检测方法，并在最后分析提出了本方法的部署和应用场景。

关键词：Web 应用程序；网络安全；RASP；安全监测
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2020294

RASP based Web security detection method

YU Hang, WANG Shuai, JIN Huamin
Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

Abstract: At present, the traditional Web security detection methods act on the input and output of the program, which can not prevent malicious code entering the program after being distorted and confused, and it is difficult to meet the new requirements of Web application security protection. Based on the in-depth analysis of the risk of traditional data flow monitoring methods, combined with the technical characteristics of rasp, a Web security detection method based on rasp was proposed. The rasp probe was embedded in the parameters of authority discrimination function, system command execution function and database operation function in Web application, and the change of data flow was detected in real-time at the code interpreter level. This method was implemented based on Java language. It was proved in the laboratory that this method is better than the traditional Web security detection method in accuracy and detection time. Finally, the deployment and application scenarios of this method were analyzed and proposed.

Key words: Web application, network security, RASP, security monitoring

1　引言

在当前数字时代，Web 应用程序在社会活动、经济活动、政府施政中承担着重要角色，各种高价值的数据和信息经由 Web 接收、处理、存储、转发，能带来可观的经济效益和广泛的社会影响，因此 Web 应用程序成为网络攻击的主要目标之一，蠕虫、病毒、后门、注入、仿冒、DDoS 等攻

收稿日期：2020−09−15；修回日期：2020−11−10

击行为不断威胁着 Web 应用程序的安全。

为了保障 Web 应用程序平稳运行，防范数据窃取，人们通常在 Web 应用程序的输入输出位置使用数据流监测系统，实时对攻击行为进行告警和拦截，这种传统的方法在过去能很好地防御 Web 攻击，但在互联网技术发展迅猛的今天，Web 应用程序面临的威胁也在不断变化与升级。基于互联网衍生出来的云计算、大数据、物联网、移动计算等新技术与新模式，让 Web 应用程序的开发框架、运行环境、流量入口、服务器部署都发生了很大的变化，传统的 Web 数据流监测方法已不能应对日益严峻的网络安全态势[1]。

本文提出的基于 RASP 的 Web 安全检测方法，以权限校验函数、系统命令执行函数和数据库操作函数作为攻击监测的切入点，能深入 Web 应用程序源代码的逻辑，对 Web 应用程序实现最细粒度的数据流监测。与传统的数据流监测方法相比，该方法能实现更低的误报率、漏报率和更少的维护成本，是现有 Web 安全监测体系的有力补充。

2 传统数据流监控方法的风险

目前，随着 Web 开发技术的发展以及日益庞大的数据量，Web 应用程序的部署方式和架构已经向集群化、微服务化、云化发展，多层次的架构和部署扩大了 Web 风险暴露面，有明显边界的网络拓扑已经越来越少，这些 Web 应用的新特点让检测层次单一的传统数据流监测方法不能再适应 Web 应用程序的安全需求。

传统的数据流监测方法示意图如图 1 所示。传统的数据流监测方法仅监测用户到 Web 应用程序之间的数据流，在 Web 进行资源操作与调用时没有埋点监测，若 Web 应用程序遭遇反序列化之类的 0day 攻击，或攻击代码经过一系列的“变形”，绕过了监测规则，都会使传统的数据流监测方法识别不出恶意数据流而将其放行，导致安全事件的发生。若攻击者目的仅在于破坏可用性，则只需绕过输入监测即可。在性能上，传统的方法会让所有数据在传入程序之前要经过 SQL 注入过滤、XSS 过滤、命令执行过滤等安全检测，大幅增加了程序的性能开销。

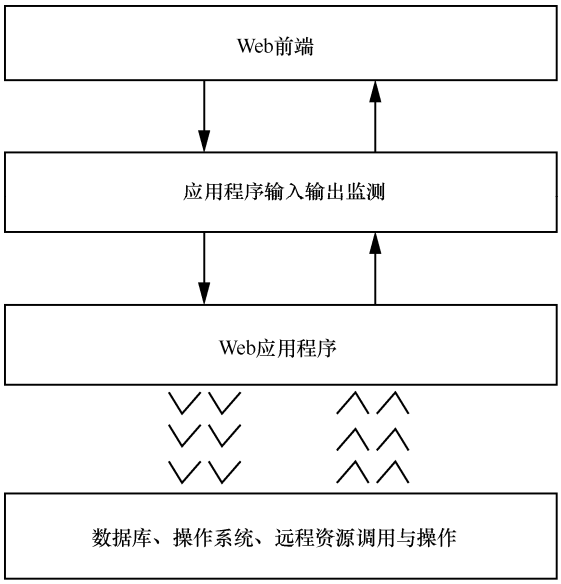


图 1 传统的数据流监测方法示意图

3 RASP 技术与产品发展

Gartner 分析师在 2014 年将 RASP 列为应用安全领域的关键技术。RASP 在设计、实施和执行上与其他应用安全保护技术有本质区别，可以快速地将安全防御功能像疫苗一样注入正在运行的应用程序中，结合应用的逻辑及上下文，对访问应用请求的每一段代码进行检测，弥补了传统边界安全防护产品的先天性防护不足[2]，应对无处不在的应用漏洞与网络威胁，为应用程序提供全生命周期的动态安全保护。

RASP 技术具有普适性，RASP 的开发部署只针对应用程序本身，与程序运行平台和代码逻辑无关，能随着应用程序的迁移而迁移，不会影响应用程序的设计，并且 Web 和非 Web 应用程序都可以通过 RASP 进行保护。

但 RASP 是优点和缺点都十分突出的技术，由于 RASP 是注入 Web 应用程序内部运行，会增

加应用程序的处理开销和工作量，对性能造成一定的影响[3]，再加上 RASP 面世时间短，相关理论和技术尚不成熟，因此业内大多持观望态度。目前国内推出的 RASP 产品包括灵蜥和 Open Rasp，由于这两款产品的设计思路是完全替代旧有监测系统，不可避免地带来系统处理过程复杂、规则库庞大、监测控制点过多的问题，大大增加了应用程序的资源消耗和程序运行时间，这也是该两款产品尚未得到市场认可的原因之一。

4　基于 RASP 的 Web 安全检测方法介绍

本文提出的利用 RASP 提升 Web 安全检测效果的方法发挥了 RASP 技术的优势，以权限校验函数、系统命令执行函数和数据库操作函数作为攻击监测的监测点，从根源上防御大部分 Web 漏洞攻击，该方法还能对以上 3 种函数的参数进行上下文记录，形成参数的上下文链条，方便事后的溯源定位和安全审计，提升应急响应能力。

4.1　技术框架

该方法共有 3 个模块：权限监测模块、系统命令执行函数参数监测模块和数据库操作函数参数监测模块。

（1）权限监测模块

- 记录初始时用户的权限，并一路追踪权限参数的变化并记录。
- 在数据流到达敏感函数，并执行函数功能前，将此时的用户权限与记录的初始权限做比对，若相同则继续执行操作，不同则阻断操作并告警。

（2）系统命令执行函数参数监测模块

- 根据业务需求制定操作系统命令的白名单。
- 标记系统命令执行函数的参数，记录该参数从被赋值到传入系统执行前的变化。
- 数据流到达命令执行函数，执行系统命令前，将此时的命令与系统命令白名单进行比对，若符合白名单则执行系统命令。

由于操作系统只能执行它理解的命令，因此白名单只需要针对 Web 运行的操作系统，无须考虑编码、序列化、拼接等复杂情况，能有效地减小白名单大小，缩短白名单校验消耗的时间和资源。

（3）数据库操作函数参数监测模块

- 根据业务需求指定数据库操作命令的白名单和可操作库表的白名单。
- 标记执行数据库操作函数的参数，记录参数从被赋值到传入函数执行前的变化。
- 在 Web 应用程序输出命令到数据库执行前，与数据库操作白名单对比，若符合则执行数据库操作。

由于参数输入数据库执行前已经经过 Web 应用程序转码、拼接等处理，因此白名单只需针对能被数据库理解的命令制定，能有效减小白名单大小，缩短白名单校验消耗的时间和资源。

通过以上 3 个模块，可以直接在代码执行层面实时分析数据流的变化，记录敏感函数参数的上下文数据，从而在根本上阻断攻击的发生，不需要担心规则被绕过。此外，该方法记录的参数传递路径以及变化情况还有助于安全审计工作，便于准确定位漏洞的触发点。

4.2　部署方案

本文提出的利用 RASP 提升 Web 安全检测效果的方法不是一套独立的能防御所有 Web 攻击手段的方法，它是将监测的重点放到 Web 应用程序内部与 Web 攻击息息相关的函数中，不依赖网络流量分析，因此避免了协议解析、字符解码、复杂的正则表达式匹配以及基于签名的威胁鉴别等麻烦，对于 0day 攻击和规则绕过攻击有很强的防御能力，但对于不涉及程序内部函数的攻击则无能为力，如 URL 重定向攻击和反射型 XSS 攻击、密码爆破等攻击手段[4]，因此它还需要传统的数据流监测方法相配合，让两种方法各司其职，共同构建更完整、更全面、更立体的数据流监测体系。

以 Spring 框架为例，可以将精简后的 WAF

部署在 Web 应用程序外层，专门拦截过滤反射型 XSS、URL 重定向攻击、点击劫持等不涉及程序内部的攻击手法，在减少 WAF 性能消耗的同时减少程序内部 RASP 需要处理的数据流。当数据流运行到 Web 应用程序内部时，RASP 会精确分析数据流的变化情况，根据分析结果区分合法行为还是攻击行为，然后对攻击行为进行拦截。部署方案示意图如图 2 所示。

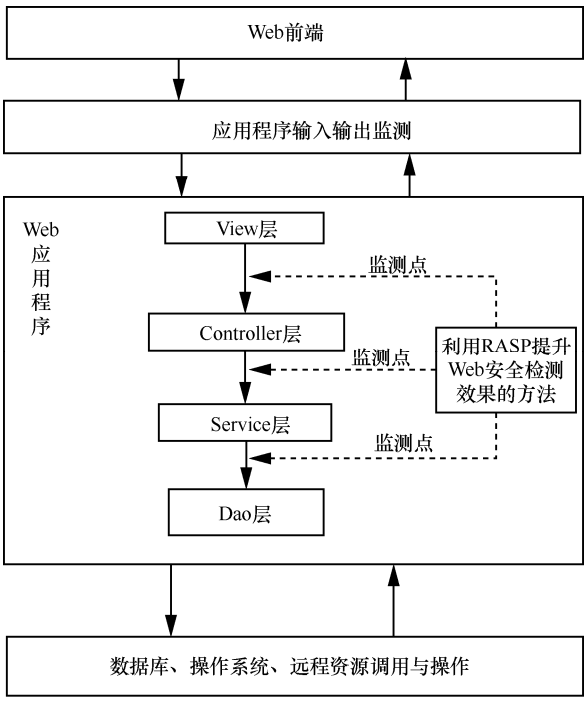


图 2　部署方案示意图

5　基于 RASP 的 Web 安全检测方法实现

5.1　各语言实现的关键技术

RASP 的技术实现与 Web 应用程序的开发语言强相关，不同的语言有不同的实现方式，见表 1。

5.2　基于 Java 的方法实现

本文基于 Java 的 RASP 技术实现，编写了 RASP_SQL_CHECK.jar 作为数据库操作函数参数监测模块，对传入的参数进行监测。

在 RASP_SQL_CHECK.jar 中，ClassTransformer 类继承 ClassFileTransformer 接口，然后在类中编写 transform 方法来获取 Java Web 数据库操作函数的类加载器、类名、字节码等数据，并使用字节码操控框架 ASM 把字节码传递给 TestVisitor 做处理，如图 3 所示。

```
class ClassTransformer implements ClassFileTransformer {


    public byte[] transform(ClassLoader 类加载器, String 类名, byte[] 字节码){
        try {
            if (监测到字节码的类名等于RASP_Process) {
                实现ClassReader类读取读取并解析字节码;
                实现ClassVisitor类,将字节码转到TestVisitor做处理;
                返回处理结果;
            }
            else{System.out.println("null");}

        } catch (Exception e) {
            e.printStackTrace();
        }catch (Throwable t){
            t.printStackTrace();
        }

    }
```

图 3　transform 获取数据库操作函数类

TestVisitor 中使用了 ASM 字节码操作和分析框架的 MethodVisitor 接口，当 TestVisitor 捕获到关键函数 ishack 时转入 MethodVisitor 进行字节码处理，对字节码进行操作，并织入 Rasp 的安全检测函数 TestVisitorAdapter，如图 4 所示。

TestVisitorAdapter 会根据 SqlFilter 的方法对参数进行校验处理（如图 5 所示），可根据需要随

表 1　不同开发语言的不同实现原理

开发语言	实现原理
Java	利用 Java 的 ClassFileTransformer 接口特性，构建独立于 Java 应用的 agent 程序，让 class 文件在 JVM 运行前插入安全监测代码，实现 RASP 功能
PHP	利用 PHP 的拓展模块特性，在执行模块初始化（MINIT）过程中，替换特定 PHP_FUNCTION 对应的函数指针，从而插入安全监测代码，实现 RASP 功能
Python	利用 Python 动态分析中常用的命名空间覆盖的机制、hook 关键函数，将函数的参数和返回值送回到安全监测代码，实现 RASP 功能

```
public class TestVisitor extends ClassVisitor {


    public TestVisitor(ClassVisitor classVisitor,String className) {
        super(Opcodes.ASM5, classVisitor);
    }

    @Override
    public MethodVisitor visitMethod(int access, String name, String desc, String signature, String[] exceptions) {

        if (监测到字节码的函数名等于ishack) {
            将字节码传入安全监测函数TestVisitorAdapter;
            返回结果
        }
    }
}
```

图 4　在关键函数 ishack 前插入探针

```
public class TestVisitorAdapter extends AdviceAdapter {
    protected TestVisitorAdapter(int api, MethodVisitor mv, int access, String name, String desc) {
        super(api, mv, access, name, desc);
    }
    @Override
    protected void onMethodEnter() {
        还原字节码;
        初始化sqlFilter文件,并将还原后的字节码输入到文件进行过滤;
        if(返回的过滤结果为真) {
            应用程序抛出java/sql/SQLException异常类型错误;
            定义异常错误的信息为please stop hacking !!!;
            页面显示异常信息;
        }
    }
}
```

图 5　对参数进行校验

时修改 SqlFilter（如图 6 所示），实现程序的解耦和逻辑优化。

```
public class SqlFilter {
    public boolean filter(Object sql){
        boolean ret=false;
        将数据转换为小写字母;
        定义需要过滤的危险字符;
        if(过滤结果为真) {
            返回过滤结果ret = ture;
        }
        返回结果ret;
    }
}
```

图 6　过滤函数 SqlFilter 样例

最后将整个程序打包成 RASP_SQL_CHECK.jar，在 Java Web 运行前将 jar 添加至 JVM 的启动参数，即可对 Java Web 进行安全检测。

5.3　运行效果

简单创建一个以 Spring 框架为基础的 Java Web 工程，在 Dao 层数据库操作函数前插入 RASP 探针，使程序在 JVM 运行到数据库执行函数时先运行 RASP_SQL_CHECK.jar 的检测方法，若发现攻击特征，会进行告警提示。

由于 Java 操作数据库统一通过 Java.sql.Statement 类执行，因此在实际部署运行时，RASP 可直接检测 Java.sql.Statement 的字节码。（如图 7 所示）在 Statement 对象将 SQL 语句发送到数据库前进行检测，既可以增加检测的准确率，还能减少埋点位置。

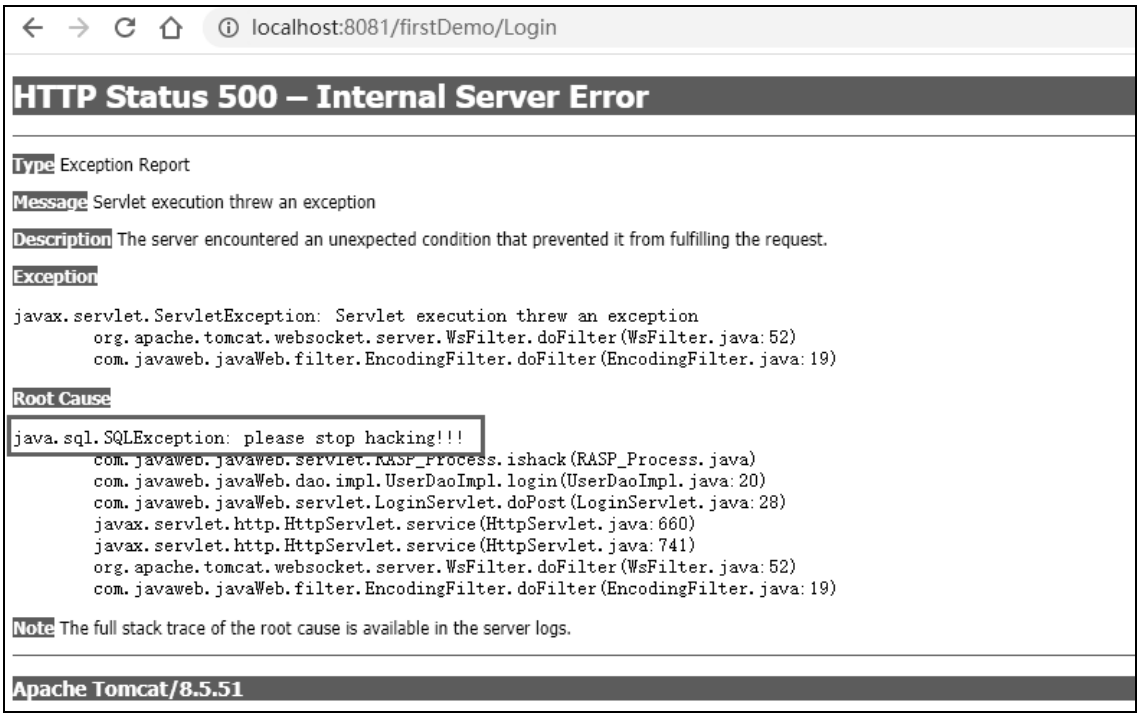


图 6　RASP 触发

5.4　检测结果与性能比较

为了具体验证本方法的优越性，本文在实验环境下将本文方法与不采用安全防护以及采用 OpenRasp 防护、传统数据流监控等其他方法进行了攻击检测效果和性能的比较。实验环境见表 2。

实验步骤如下。

步骤 1　使用 SQLmap 生成 500 个不同攻击特征的请求包并保存。

步骤 2　编写脚本，生成 1 000 个正常的请求包并保存。

步骤 3　使用 Jmeter 将 1 500 个请求包并发地发送到 Java Web 程序。

步骤 4　记录不同方法部署情况下攻击识别的准确率和运行时间。

步骤 5　以上步骤重复 10 次。

各方法检测结果与性能实验结果见表 3。

实验结果表明，本方法能在保证识别准确率的情况下，有效降低资源消耗。

表 2　实验环境

环境组件	Java	Tomcat	WAF	SQLmap	Jmeter	OpenRasp
版本	1.8	8.5.0	https://github.com/chengdedeng/waf	1.4.8	5.2.1	1.3.4

表 3　各方法检测结果与性能实验结果

方法	平均识别准确率	平均运行时间/s
不采用安全防护	—	2.489
本文方法	100%	3.301
采用 OpenRasp 防护	100%	4.831
传统数据流监控	98%	4.653

6　基于 RASP 的 Web 安全检测方法应用场景

6.1　部署场景

本文提出的利用 RASP 提升 Web 安全检测效果的方法本质上是利用程序设计语言的特性[5]，实现对 Web 应用程序内部组件的安全监测，因此该方法能适用于各种 Web 应用场景，与应用程序运行的系统平台无关。

每种 Web 开发语言都有像 Java 执行系统命令的 Runtime.getRuntime().exec()接口、执行数据库命令的 executeQuery()接口等系统接口，在部署本文方法时应理清 Web 应用程序调用了哪些系统接口，并针对性地开发 RASP 监测系统，从而在不修改 Web 应用程序代码的情况下，将保护程序注入运行的 Web 中，实现安全监测和应用程序的解耦。

6.2　漏洞防御场景

攻击者受利益驱使，攻击 Web 应用程序的主要目的是存储在程序内高价值的数据信息，或以 Web 服务器为跳板入侵内网，因此攻击者大多围绕权限控制、命令执行、数据库这 3 个方面进行攻击[6]。根据上述情况，本文提出的利用 RASP 提升 Web 安全检测效果的方法选择了权限校验函数、系统命令执行函数和数据库操作函数作为攻击监测的切入点，对 Web 应用程序进行安全防御。

权限监测模块、系统命令执行函数参数监测模块和数据库操作函数参数监测模块能有效防御注入攻击、失效的身份认证、敏感信息泄露、失效的访问控制、不安全的反序列化等 OWASP Top10 风险类型，这 3 个模块还能防御以下攻击手段。

- 权限监测模块：能有效防御存储型 XSS、任意文件读取下载、文件上传、失效的身份认证、敏感信息泄露、越权等攻击手段。
- 数据库操作函数参数监测模块：能有效防御 SQL 注入、不安全的反序列化、脱库、数据库提权、Webshell 等攻击手段。
- 系统命令执行函数参数监测模块：能有效防御 CSRF、OS 注入、XXE、反序列化、权限控制、木马、Webshell 等攻击手段。

因为数据库操作函数参数监测模块、系统命令执行函数参数监测模块会在执行命令前对参数进行白名单校验，因此模块白名单只需针对能被理解的命令制定，不需要考虑规则会被绕过，能以尽可能简短的安全监测流程实现更完整的防御，从而避免 RASP 带来的资源消耗问题。

7　结束语

当前随着移动互联和云计算的流行，明显边界的网络拓扑已经越来越少，传统的 Web 应用程序数据监测系统已无法满足 Web 应用程序安全防护的新要求，而被寄予厚望的 RASP 技术因为资源消耗高的缺点，难以推广应用。本文提出的利用 RASP 提升 Web 安全检测效果的方法通过在应用程序内部监测常见 Web 攻击的目标函数，实现了高效率的攻击行为监测，与传统的监测方法相结合能进一步减少监测代码的处理时间，避免 RASP 高消耗的问题，相互弥补了不足，具有良好的应用推广前景。

参考文献：

[1] Global runtime application self-protection (RASP) security market to grow at a CAGR of 49.68% during the period 2018-2022[Z]. 2018.

[2] 邱若男, 胡岸琪, 彭国军, 等. 基于 RASP 技术的 Java Web 框架漏洞通用检测与定位方案[J]. 武汉大学学报(理学版), 2020, 66(3): 285-296.
QIU R N, HU A Q, PENG G J, et al. General detection and location scheme of Java Web framework vulnerabilities based on RASP technology[J]. Journal of Wuhan University (Science Edition), 2020, 66(3): 285-296.

[3] BELLESSORT R, RUELLAN H, OUEDRAOGO N. Method and device for safely executing a Web application in a Web runtime environment: GB2554697[P]. 2018-04-11.

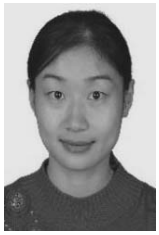
[4] 陈威, 陈乐然, 徐小天, 等. 基于 Web 应用系统脆弱性的攻击及其防御技术[J]. 电信科学, 2017, 33(Z1): 108-116.
CHEN W, CHEN L R, XU X T, et al. Attack and defense technology based on Web application system vulnerability [J]. Telecommunications Science, 2017, 33(Z1): 108-116.

[5] YIN Z X, LI Z F, CAO Y. A Web application runtime application self-protection scheme against script injection attacks[Z]. 2018.

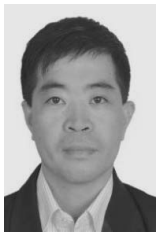
[6] 林锋, 徐柳婧, 陈晓华, 等. 一种基于多视角特征融合的Webshell 检测方法[J]. 电信科学, 2020, 36(6): 125-132.
LIN F, XU L J, CHEN X H, et al. A Webshell detection method based on multi view feature fusion [J]. Telecommunications Science, 2020, 36(6): 125-132.

[作者简介]

余航（1997− ），男，中国电信股份有限公司研究院工程师，主要研究方向为网络安全、安全攻防。



王帅（1979− ），女，中国电信股份有限公司研究院高级工程师，主要研究方向为网络安全、安全攻防。



金华敏（1972− ），男，中国电信股份有限公司研究院高级工程师，主要研究方向为 IP 网、云计算、大数据安全、网络安全。