doi :10.3969/j.issn.1671-1122.2025.01.012

一个用于Java应用程序运行时保护的混合系统

江昊，刘成杰，文伟平
（北京大学软件与微电子学院，北京 100091）

摘　要：近年来，应用程序运行时自我保护RASP技术作为一种嵌入式防护机制，广泛应用于检测和防御Web应用程序中的常见攻击，如SQL注入、跨站脚本XSS攻击以及Java反序列化攻击。然而，现有RASP系统多采用基于黑名单的检测方法，容易被绕过且难以应对新型攻击。为此，文章提出一种混合系统HP-RASP，该系统结合启发式规则和深度学习模型，在应用程序运行时提供自适应的安全保护。文章将BERT模型引入RASP系统，用于分析和检测SQL注入攻击，同时通过对常见方法调用栈进行监控和黑名单匹配，防御XSS和反序列化攻击。HP-RASP利用Java插桩技术，动态插入关键类和方法的监控逻辑，实现对Web请求的实时分析。文章在多个开源数据集上评估了该系统的检测性能，并将其与当前主流RASP系统OpenRASP进行了对比。实验结果表明，在检测准确率、性能开销和系统鲁棒性方面，HP-RASP相较现有方案均有显著提升；在SQL注入方面，准确率达到81.9%，比OpenRASP提升了1.84倍，召回率和*F*1分数也显著高于OpenRASP；在XSS防护方面，HP-RASP对反射型XSS和存储型XSS的召回率均达到99.9%，对反序列化攻击的召回率达到84.6%；在响应时间和资源消耗方面，HP-RASP表现良好，并未显著增加响应时间和资源消耗。

关键词：RASP；BERT模型；软件安全；Java网络应用程序

中图分类号：TP309　**文献标志码：**A　**文章编号：**1671-1122（2025）01-0134-14

中文引用格式：江昊，刘成杰，文伟平. 一个用于Java应用程序运行时保护的混合系统[J]. 信息网络安全，2025，25(1)：134-147.

英文引用格式：JIANG Hao, LIU Chengjie, WEN Weiping. A Hybrid System for Runtime Protection inside Java Application[J]. Netinfo Security, 2025, 25(1): 134-147.

A Hybrid System for Runtime Protection inside Java Application

JIANG Hao, LIU Chengjie, WEN Weiping
(*School of Software & Microelectronics*, *Peking University*, *Beijing* 100091, *China*)

Abstract: In recent years, Runtime Application Self-Protection (RASP) has emerged as an embedded defense mechanism widely used to detect and prevent common web application attacks, such as SQL injection, cross-site scripting (XSS), and Java deserialization attacks.

收稿日期：2024-11-04
基金项目：国家自然科学基金[61872011]
作者简介：江昊（2000—），男，湖北，硕士研究生，主要研究方向为软件与系统安全、漏洞挖掘；刘成杰（1998—），男，湖南，博士研究生，主要研究方向为软件安全、漏洞挖掘和入侵检测；文伟平（1976—），男，湖南，教授，博士，主要研究方向为系统与网络安全、大数据与云安全、智能计算安全。
通信作者：文伟平 weipingwen@pku.edu.cn

However, existing RASP systems often rely on blacklist-based detection, which is prone to evasion and struggles against novel threats. This paper introduced a hybrid system, HP-RASP, which combined heuristic rules and deep learning models to provide adaptive security at runtime. Notably, it incorporated a BERT model into the RASP framework to analyze and detect SQL injection attacks, while employing stack monitoring and blacklist matching to defend against XSS and deserialization attacks. HP-RASP used Java instrumentation to dynamically insert monitoring logic into critical classes and methods, enabling real-time analysis of web requests. The system was evaluated on multiple open-source datasets and compared to the current mainstream RASP system, OpenRASP. Experimental results demonstrate significant improvements in detection accuracy, performance overhead, and robustness over existing approaches. For SQL injection, HP-RASP achieved an accuracy of 81.9%, 1.84 times higher than OpenRASP, with recall and *F*1 scores also notably surpassing OpenRASP. For XSS protection, HP-RASP achieved a 99.9% recall rate for both reflective and stored XSS attacks, and an 84.6% recall rate for deserialization attacks. HP-RASP also performed well in terms of response time and resource consumption, without significant increases in either metric.

Key words: RASP; BERT model; software security; Java web application

0 引言

一直以来，网络攻击的风险一直是网络安全的重大威胁。在SANS于2022年发布的报告中，针对网络应用程序的攻击在“SANS 2022 Top New Attacks and Threat Report”[1]排名前五。报告显示，尽管如今部署了大量的安全技术，网络安全仍不容忽视。在众多网络攻击中，比较知名和危害性较大的攻击方法是SQL注入攻击和跨站脚本XSS攻击[2,3]。

最常用的网络应用程序保护技术是网络应用防火墙（Web Application Firewall，WAF），其主要思想是基于应用程序和互联网之间的流量特征来过滤和监控流量。WAF可以分为两类：基于签名的WAF和基于机器学习的WAF[4,5]。在机器学习广泛使用之前，基于签名的WAF已被普遍接受。RAZZAQ[6]等人总结了这一点，并提出仅基于签名的网络安全保护是不足的。随着机器学习技术的发展，许多基于异常的WAF[7,8]和混合型WAF[9,10]被投入使用，推动了软件保护技术的发展。虽然WAF技术在识别某些类型的恶意行为方面做出了杰出贡献，但也存在一定的局限性[11,12]，即使配备了一些机器学习和深度学习技术，但在应对从未见过的恶意流量和某些混淆流量时表现不佳。原因在于WAF技术部署在应用程序之外，仅依赖分析流量的外部特征，拦截其认为恶意的请求，而不分析这些请求在应用程序中如何处理[13]。因此其分析的特征是经过攻击者加密封装后的内容，想要进一步提高检测效果，必须深入到真实的执行流信息中。

当关注恶意请求如何破坏网络应用时，可以发现无论伪装多么巧妙，恶意请求最终总会在攻击执行过程中暴露真面目。如果能在应用程序内部附加监控器，攻击者将无处藏身。运行时应用程序自我保护（Runtime Application Self-Protection，RASP）技术就是这种类型的技术。RASP技术能够实时监控并阻止攻击，使应用程序本身具备自我保护能力。GARTNER于2012年提出了内置防火墙的概念，随后有大量相关工作展开[14,15]。ČISAR[16]等人在2016年总结了RASP的框架，展示了这种技术的优缺点。与WAF技术相比，RASP不需要广泛的测试和配置，并且表现更为稳定。然而，现有的RASP系统在检测和防御复杂攻击时仍然存在很多不足。多数系统依赖基于黑名单的静态规则进行攻击检测，虽然对已知的攻击方式有较高的识别率，但面对新型或变异攻击时，往往会失效。本文发现大量的有效载荷可以从RASP系统中解析出来，如果正确使用机器学习和自然语言处理技术来识别这些有效

载荷，对恶意请求的拦截准确率也会得到提升。

2018年，BERT模型[17]的发布为恶意有效载荷的识别带来了新的视角，因为BERT模型只需要微调即可适应某些任务类型，甚至可能用于SQL语句的分类。BERT模型使用了掩蔽语言模型（Masked Language Model，MLM）[18,19]和下句预测（Next Sentence Prediction，NSP）[20,21]来通过掩蔽和预测学习词语与句子之间的关系。该模型在分类任务中表现良好，可以通过将请求的有效载荷评估为高维矩阵在较短时间内确定其为恶意或良性的可能性，因此BERT模型可以满足本文对恶意载荷识别的要求。

综上所述，本文提出了一种混合型的RASP系统——HP-RASP（High Performance RASP）。该系统结合了基于启发式规则的传统检测方法与深度学习，以实现对多种攻击类型的实时、动态防护。特别地，本文在RASP系统中引入了BERT模型，用于对攻击流量进行深度分析与检测。BERT模型的双向上下文理解能力，使其在检测恶意SQL请求时具有更强的适应性和鲁棒性。此外，HP-RASP还通过监控Java方法调用堆栈并结合黑名单匹配，实现了对XSS攻击和Java反序列化攻击的高效防护。

在系统设计中，本文利用Java字节码插桩技术，将安全监控逻辑动态注入到关键的类和方法中，无需对原有的应用程序进行修改。这不仅增强了系统的可维护性，也使得安全机制能够对Web请求进行实时分析与拦截。本文在多个开源数据集上对HP-RASP进行了广泛的实验评估，并与当前主流RASP系统——OpenRASP[22]进行了对比。实验结果表明，HP-RASP在检测准确性、性能开销和系统稳定性方面均优于现有的解决方案，尤其在SQL注入检测方面具有显著优势。

本文的主要贡献包括：1）首次在RASP系统中引入BERT模型，显著提升了对SQL注入攻击的检测精度；2）设计并实现了一种结合启发式规则与深度学习模型的高效运行时监控框架，能够对SQL注入、XSS攻击和Java反序列化攻击进行多层次防护；3）通过广泛的实验验证，证明了HP-RASP系统在真实场景中的检测效果和性能优势。

1 研究动机

近年来，随着Web应用程序在各行各业的广泛应用，网络安全威胁的复杂性和频率不断增加。SQL注入、XSS攻击和Java反序列化攻击等漏洞，成为攻击者常用的手段，给应用程序带来了严重的安全风险。为了应对这些问题，可以将恶意流量拦截于应用程序之外的WAF技术得到广泛使用。然而由于WAF的静态特性，其仅能通过流量的外部特征进行分析，因此恶意流量可以通过加密等自我伪装方式绕过WAF的安全检测。为了应对这些问题，RASP技术成为一种有效的防护机制。RASP通过在应用程序运行时的环境中直接嵌入安全监控模块，能够实时分析并检测恶意攻击行为，且无需修改应用程序的源代码。这使得RASP在动态防护中具备独特的优势。

尽管当前许多平台已内置基本的SQL注入和XSS检测功能，如通过静态规则检测和模式匹配实现。然而，这些方法的主要局限性在于缺乏上下文语义理解，难以应对复杂和变种攻击，导致现有的RASP系统在检测和防御复杂攻击时仍然存在不足。本文引入BERT模型，以提升检测的灵活性和精度，特别是在应对新型攻击时具备显著优势。本文将在1.1节以一个具体的例子详细阐述现有的RASP系统的局限性，并在1.2节提出本文的威胁模型。

1.1 现有 RASP 系统的局限性

当前的RASP系统主要依赖静态黑名单规则进行攻击检测。这种方法虽然在识别已知攻击时有效，但在应对新型和变种攻击时存在局限性。黑名单策略依赖预定义的攻击模式，攻击者只需稍作修改，如混淆SQL语句或采用新颖的攻击载荷，即可绕过检测。以百度的开源框架OpenRASP[22]为例，其SQL注入检测逻辑基于关键词匹配和函数黑名单。例如，检测机制要求SQL语句中包含指定的敏感关键词（如select、file

等）或特征（如16进制字符或联合查询）才能触发安全警报。然而，这种规则检测方式在面对复杂的变种攻击时显得无力。新型攻击方式绕过静态检测规则的流程如图1所示，虚线路径表示攻击者绕过检测逻辑时的执行路径。攻击者可以通过规避关键词或修改语法结构绕过检测，诸如盲注、条件响应或时间延迟注入等方式可能避开静态规则的防护。此外，面对新的攻击模式时，预设规则未涵盖其特征，检测效果较差，因此，基于静态特性的检测方法使RASP系统过度依赖开发者对已知攻击的理解，缺乏自适应能力，难以有效应对未知威胁。

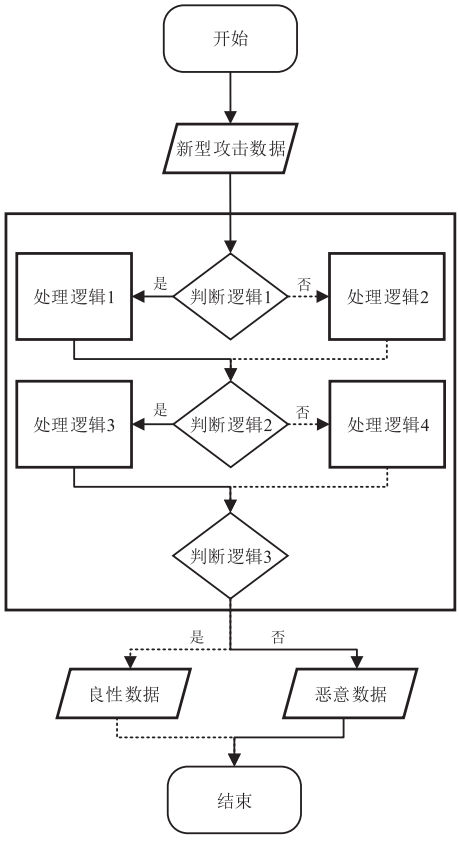


图 1 新型攻击方式绕过静态检测规则的流程

为了解决这些问题，本文提出基于深度学习的防护机制，特别是将BERT模型引入到RASP系统中。通过分析SQL注入、XSS攻击和反序列化攻击的上下文特征，BERT模型能够有效捕捉攻击载荷中的深层语义关系，从而提高检测精度，避免静态规则的局限性。

1.2 威胁模型

本文的威胁模型主要关注Java应用程序在运行时的SQL注入、XSS攻击和反序列化攻击。攻击者的目标是绕过现有的检测机制，成功发起这些攻击，窃取、篡改数据库数据或执行未授权的代码。在设计HP-RASP系统时，本文假设攻击者具有以下3种能力。1）Web攻击能力：攻击者可以通过发送精心构造的恶意请求，利用Web应用中的漏洞发起攻击。这些攻击包括但不限于SQL注入、XSS攻击和Java反序列化攻击。攻击者能够通过调整攻击载荷的形式或内容，绕过基于规则的防护机制。2）变种攻击能力：攻击者可以对传统攻击方式进行修改，生成新型或变种攻击。例如，攻击者可能使用稍微变换的SQL语句格式或特殊字符绕过静态规则检测系统。3）未知漏洞利用能力：攻击者可能利用尚未公开或未被系统收录的漏洞进行攻击。这意味着攻击载荷可能不包含在现有的黑名单或签名库中，从而难以被静态规则检测。

基于上述假设，本文的防御目标为以下3类。1）不影响应用运行的前提下，实时监控和检测恶意请求，尤其是SQL注入、XSS攻击和反序列化攻击。2）通过引入BERT模型，提升对变种和未知攻击的识别能力，避免静态规则的局限性。3）确保系统在高并发环境下保持良好的性能表现，尽量减少额外的资源消耗和延迟。

2 相关工作

2.1 RASP 技术

Gartner[23]在2012年提出RASP（Runtime Application Self-Protection）技术，该技术通过嵌入或链接到应用程序或其运行环境中，实时监测并防护攻击。YUAN[12]等人同年对自保护系统进行了分类和综述。2016年，ČISAR[16]等人和LANE[24]提出了更详细的RASP框架。近年来，YIN[25]等人基于数据流分析，提出了一种针对脚本注入攻击的RASP方案，通过自动在关键位置插入过滤器实现防护；SALEMI[14]则结合RASP技术改进WAF，提高了安全性。国内同样也有一些研究，邱

若男[26]等人总结了4类漏洞通用利用模式，提出一种基于RASP技术的Java Web框架漏洞通用检测与定位方案；LI[27]等人通过对公开的Java反序列化漏洞进行研究，提出一种基于运行时检测的Java反序列化防御方案；余航[28]等人结合RASP和数据流监控方法风险分析，提出一种基于RASP的Web安全监测方法。此外，RASP技术还被应用于区块链安全研究领域[29,30]，利用传统安全技术解决区块链中的安全问题。

2.2 SQL 注入检测技术

SQL注入攻击发生在攻击者尝试向Web应用程序的数据库插入恶意代码时。对于本文RASP系统主要防护的盲注，已有不少研究和工具提出了对应的解决方案。AMNESIA工具[31]通过分析与监控来防范SQL注入攻击。KOMIYA[32]等人采用机器学习算法改进了检测性能。为应对同时利用XSS与SQL注入的攻击，PUTTHACHAROEN[33]等人研究了通过JavaScript利用cookie表的联合攻击，ZHANG[34]等人则使用执行流机制来防护基于JavaScript的XSS攻击和SQL注入攻击。郭春[35]等人提出一种基于特征词对的关键载荷截取方法，有效解决了关键载荷提取的问题。黄恺杰[36]等人提出一种基于大语言模型的SQL注入攻击检测方法，显著提高了SQL注入检测的准确率和检测性能。

2.3 XSS 攻击检测技术

自本世纪初以来，XSS攻击始终威胁着Web应用程序的安全。XSS攻击检测技术主要分为服务端检测和客户端检测，本文的RASP系统属于客户端检测技术。MITROPOULOS[37]等人尝试通过收集浏览器JavaScript引擎中的脚本元素并生成其指纹来防御恶意JavaScript驱动的XSS攻击，最终验证指纹的有效性。WEISSBACHER[8]等人提出了ZigZag，它通过异常检测技术自动强化基于JavaScript的Web应用程序，监控客户端代码的执行轨迹以发现异常。谭宇辰[38]等人利用深度学习中的卷积神经网络、时间循环神经网络、多层神经网络3种算法来检测XSS攻击。马征[39]等人提出一种基于遗传算法和支持向量机的XSS攻击检测模型，利用遗传算法搜索特征空间，迭代生成最优测试用例，从而扩充数据集、丰富XSS攻击向量库。

2.4 Java 反序列化攻击检测技术

在Java中，反序列化是将某种格式的数据结构化为对象的逆向过程。当前常用的序列化数据格式是JSON。KOUTROUMPOUCHOS[40]等人提出了ObjectMap工具，用于检测基于Web应用的Java反序列化和对象注入漏洞。CRISTALLI[41]等人提出一种新的沙箱方法，通过定义可信的执行路径来保护Java应用的反序列化行为。LI[42]等人提出基于运行时检测的Java反序列化漏洞防御技术方案。通过在上下文信息中匹配漏洞利用中的反序列化入口函数，来判断当前行为是否为反序列化漏洞的利用行为。郑鹏[43]等人基于符号执行和污点分析提出一种自动检测方法，实现调用链检测工具Taint Gadget。

2.5 BERT 模型及其预训练

2018年，DEVLIN[17]等人提出了BERT模型，该模型通过联合考虑上下文预训练深层双向表示，用于无标签文本的语言表征。在此基础之上，项慧[44]等人提出一种基于语言特征集成学习的大语言模型生成文本检测方法，利用判决机制有效提高了文本的检测率，验证了BERT模型检出特征语句的有效性。

3 HP–RASP 方法设计

为了应对现有RASP系统在防御深度和性能上的不足，本文设计并实现了HP-RASP系统，旨在通过结合启发式规则和深度学习模型来提高对Web攻击的检测能力，同时减少性能开销。HP-RASP的架构如图2所示，当未知流量输入时，HP-RASP可以阻断恶意流量穿行，放行良性流量流通。HP-RASP的核心思路是将传统的规则匹配方法与基于BERT的深度学习模型结合，利用模型对攻击载荷的理解能力，提升对复杂和变种攻击的检测效果。

从图2中可以看到，在传统RASP部分，本文基于

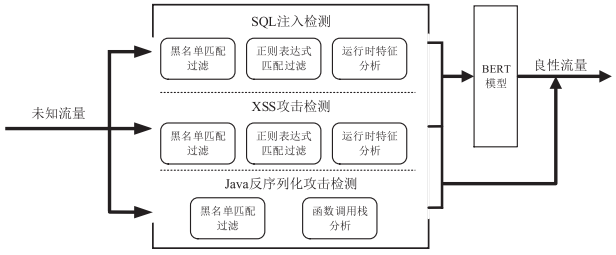


图 2 HP–RASP 系统架构图

3种不同的网络攻击实现了不同的检测模块。对于SQL注入和XSS攻击，本文使用黑名单和正则表达式来识别有效载荷，并使用运行时特征来分析请求的行为。对于Java反序列化攻击，本文通过黑名单匹配类并分析调用栈，以捕捉隐藏的风险点。BERT模型是该系统中一个精心设计的深度学习模块，帮助系统更好地区分输入流量的有效载荷，特别是一些混淆的和未知的攻击。

接下来，本文将详细解答以下问题：1）HP-RASP如何监控应用程序行为；2）HP-RASP如何查找代码钩点并进行插桩；3）HP-RASP如何定义静态检测规则；4）HP-RASP如何针对性训练BERT模型。

3.1 应用程序监控

本文使用Java实现了HP-RASP系统，并专注于保护Java网络应用程序。为了确保系统能够在应用程序执行过程中实时感知并监控应用程序的内部行为，HP-RASP首先需要插入代码钩点以截获关键函数的调用。如图3所示，HP-RASP会在程序装载执行前首先扫描整个应用程序，获取一些关键方法的位置以构建合适的钩点。然后将这些钩点通过代码插桩的方式插入到应用程序中。当插桩后的应用程序运行后，HP-RASP系统就通过钩点反馈运行时的信息，实时监测应用程序运行状态，当检测到异常执行时，就会终止执行流，并记录相关日志信息。

3.2 代码插桩

代码插桩是RASP系统的基础，本文将字节码注入到受保护的应用程序中以实现应用程序的实时防护。HP-RASP系统利用Java的字节码操作技术，通

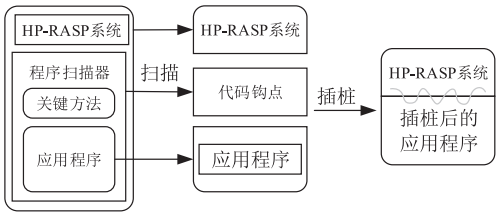


图 3 应用程序监控方法

过Java Instrumentation接口[45]动态修改目标应用程序的字节码，从而在不修改源代码的情况下，实现在目标应用程序运行时在目标程序进程空间插入安全监控逻辑。

在初始化阶段，本文通过修改premain方法，加载Instrumentation代理，该方法会在主应用的main方法执行之前被调用，如图4所示。本文使用java.lang.instrument包来挂钩目标方法。当JVM加载Java类时，类的字节码会通过Transformer模块进行处理，该模块判断当前类是否需要插入钩点。如果类符合条件，系统将借助ASM框架[46]对其字节码进行逐步分析。

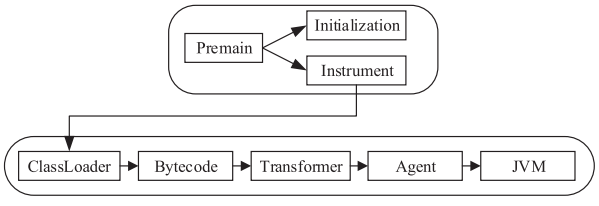


图 4 初始化启动过程

在字节码分析过程中，HP-RASP系统基于插桩算法对目标方法进行插桩。插桩算法的核心逻辑如算法1所示，首先，HP-RASP通过getInvokeStaticSrc方法获取目标类和方法的静态源代码；然后，使用getMethod函数检索目标类中的所有方法；最后，HP-RASP逐一遍历这些方法，并在方法的开头或结尾处使用insert函数插入检测代码。插入的检测逻辑将负责在每次目标方法调用时进行入侵检测和分析。若目标类不包含方法，则系统会返回错误并跳过该类的注入。

算法1 插桩算法的核心逻辑

输入：class, method, parameters
输出：none
1. src = getInvokeStaticSrc(class, method, parameters)

```
2. methods = getMethod(class)
3. if methods != null && methods.size() > 0 then
4.      for method : methods do
5.              insert(method, src)// 在该方法的结尾或开头处插桩
6.      end for
7. else
8.      error: 该类中不存在方法
9. end if
```

3.2.1 SQL 注入的插桩点

为了应对Java Web应用中的多种SQL注入，本文为以下风险类构建了库，如表1所示。表1展示了与SQL注入相关的典型类。从表1可知，对于Java Web应用中SQL请求的常规调用路径，本文在调用路径的每个关键节点上插入钩点，如SQLDriverManager、SQLConnection、SQLStatement等，这些钩点是MySQL、Oracle等流行数据库软件的重要接口。此外，本文添加了支持SQL多部分上传的钩点FileUploadHook。这些钩点不仅用于防范SQL注入共计，有时也对其他攻击的防范有所帮助。

表 1 部分 SQL 危险类

类	钩点
com/mysql/jdbc/NonRegisteringDriver	ConnectionHook
com/mysql/jdbc/NonRegisteringDriver	DriverManagerHook
org/sqlite/JDBC	DriverManagerHook
org/postgresql/Driver	DriverManagerHook
com/mysql/jdbc/PreparedStatement	StatementHook
com/mysql/jdbc/StatementImpl	StatementHook
org/sqlite/jdbc3/JDBC3Statement	StatementHook
...	...
rg/postgresql/jdbc/PgStatement	StatementHook
org/hsqldb/jdbc/JDBCStatement	StatementHook
com/mysql/jdbc/ResultSetImpl	SQLResultSetHook
org/sqlite/jdbc3/JDBC3ResultSet	SQLResultSetHook
oracle/jdbc/.../OracleResultSetImpl	SQLResultSetHook
org/sqlite/Conn	ConnectionPreparedHook
oracle/jdbc/.../PhysicalConnection	ConnectionPreparedHook
com/ibm/db2/jcc/am/Connectio	ConnectionPreparedHook
org/apache/../fileupload/FileUploadBase	MultipleHook

3.2.2 XSS 攻击的插桩点

本文为服务器端和应用程序前端设置了特殊的钩点，确保系统能够获取足够的参数以确定请求是否存在XSS攻击。表2展示了与XSS攻击相关的典型类。本文通过使用BESResponseBodyHook分析来自应用前端的参数以阻止恶意XSS请求。而ServerRequestHook则专注于服务器部分，生成输入脚本的特征并进行识别。此外，本文还为一些框架（如Catalina和Spring）构建了特殊的钩点。

表 2 部分 XSS 风险类

类	钩点
com/bes/.../OutputBuffer	BESResponseBodyHook
io/undertow/.../ServletRequestContext	ServerRequestHook
io/undertow/.../AttachmentKey	ServerRequestHook
org/apache/coyote/Response	ServerResponseBodyHook
...	...
org/apache/catalina/connector/Request	ServerParamHook
apache/catalina/.../FilterChain	ServerPreRequestHook
apache/catalina/.../CoyoteAdapter	ServerPreRequestHook
org/springframework/.../RequestWrapper	SpringHook

3.2.3 Java 反序列化攻击的插桩点

为应对Java反序列化攻击，本文实现了两种钩点：原生Java反序列化类钩点和FastJSON钩点。本文重点关注java/io/ObjectInputStream类，该类用于反序列化先前使用ObjectOutputStream写入的原始数据和对象，并用于恢复这些先前序列化的对象。大多数反序列化的数据可以从这个钩点获得。FastJSON钩点则相对特殊，它是一个用于将Java对象转换为JSON表示的Java库，广泛应用于各种场景。本文专注于FastJSON的一些入口类来获取钩挂数据，如com/alibaba/fastjson/parser/ParserConfig。

3.3 静态检测规则

在传统RASP部分，本文通过预定义的静态规则来检测SQL注入和XSS等常见的Web攻击。这些规则主要依赖于黑名单和正则表达式匹配，适用于处理已知的、结构相对固定的攻击模式。

3.3.1 SQL 注入的检测规则

对于SQL注入攻击，HP-RASP首先使用黑名单匹配对SQL负载中的关键字、特定语法特征及敏感操作符进行快速过滤，以识别可能包含恶意特征的请求。然后，HP-RASP进一步对请求负载进行语法结构验证以提取恶意SQL语法模式，包括多语句执行、十六进制字符串、版本注释等特征。最后，系统通过运行时

特征分析进一步评估请求的威胁性。该方法结合了数据库运行时的状态信息（如error_code）以及请求中可能导致敏感操作的动态特征，如文件读取或写入操作、对information_schema等系统表的访问频率、敏感函数的调用次数。通过将这些特征综合成评分体系，HP-RASP可有效判别请求的威胁级别，从而将其分类为“安全”“恶意”或“不确定”。

3.3.2 XSS 攻击的检测规则

在HP-RASP系统中，XSS攻击的检测规则同样结合了黑名单匹配、正则表达式过滤和运行时特征分析。首先，通过黑名单匹配过滤明显的恶意字符和标签，如<script>、onerror等已知的XSS攻击特征。其次，使用正则表达式对输入内容的长度、结构和字符组合进行过滤，对于超长的用户输入或具有特定字符组合的请求进行深入检查。此外，通过运行时特征分析，结合钩点收集的输入频率、参数长度等信息，判断请求中是否包含大量恶意内容。例如，对反射型XSS，系统会检测是否存在输入输出直接关联的行为；而对于存储型XSS，系统会监控潜在的数据持久化操作，结合数据库相关的钩点检测危险的持久化操作，防止恶意脚本长期存储在服务器中。

3.3.3 Java 反序列化攻击的检测规则

在HP-RASP系统中，Java反序列化攻击的检测规则主要依赖于黑名单匹配过滤和函数调用栈分析。系统首先通过黑名单识别已知的危险类，如com.sun.rowset.JdbcRowSetImpl、java.rmi和javax.xml等潜在危险的类。然后，通过分析反序列化过程中捕获的函数调用栈，判断是否存在高风险的方法调用链。当调用栈中出现黑名单中的危险方法并满足特定的调用关系时，系统将此请求判定为可能的反序列化攻击，从而及时拦截可能的威胁。

3.4 BERT 模型模块

传统规则匹配方法依赖于关键字和统计特征，难以准确捕获攻击负载的上下文语义特征，而BERT模型的引入可以弥补这一缺陷。BERT通过深度语义特征提取，使系统不仅能够检测到常见的恶意特征，还能够应对复杂变种攻击。由于XSS攻击和Java反序列化攻击相关数据集的缺乏，本文暂时针对SQL注入进行了模型训练。具体使用的数据集见4.1.1节。

在训练过程中，本文主要解决了两大挑战：1）SQL存在一定的的语法结构，需要适应其独特的上下文语义；2）恶意负载通常仅占整个请求的一小部分，这就要求模型的注意力机制能够更准确地定位潜在的恶意部分。为此，本文以bert-base-uncased[17]模型为基础，进行多轮微调优化，具体模型架构如图5，其步骤如下。

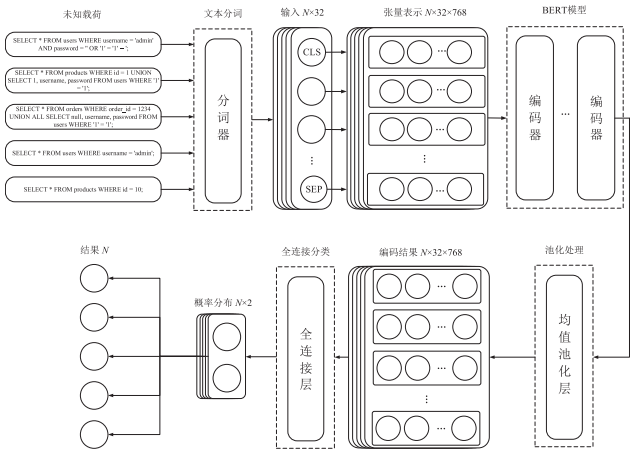


图 5 BERT 模型架构

1）提取训练数据的句子特征。HP-RASP系统使用改进的BERT分词器对输入负载进行预处理。分词器加载预定义的词汇表文件，建立词与索引的双向映射。分词处理包括基础分词和子词分词两个阶段，其中基础分词负责对输入文本进行清理和预处理，而子词分词采用贪婪最长匹配算法，将词进一步拆分为共享的子词嵌入。

2）将负载转化为高维向量。分词结果经过映射后，生成适配BERT模型的输入张量，包括input_ids、attention_mask和token_type_ids。通过填充或截断操作，所有输入张量被统一为固定长度（32个Token）并添加特殊标记（如[CLS]和[SEP]）。

3）将高维向量传入BERT模型。预处理后的输入向量被统一转换为ONNX格式的张量以适配 BERT 模

型的推理需求。输入张量的形状固定为[N, 32, 768]，表示多样本输入和固定的序列长度。输入张量随后被传入BERT模型进行推理。在推理阶段，HP-RASP采用ONNX Runtime加速器执行模型计算，大幅降低了推理延迟和资源消耗。

4）将输出特征向量进行均值池化处理。均值池化首先利用attention_mask过滤掉填充值对应的特征向量，仅保留实际输入Token的特征。然后，对有效Token的特征向量逐维求平均，生成长度为hidden_size的全局特征向量。

5）将池化特征连接至全连接层。全连接层接受长度为hidden_size的输入特征向量，并通过线性变换实现特征映射过程。输出的分类分数用于后续的分类概率计算。

6）归一化输出向量以给出概率分布。HP-RASP系统通过Softmax函数将全连接层输出的分类分数转换为概率分布。Softmax函数对分类分数进行指数变换后归一化，确保输出值范围在[0,1]。这一过程能够明确每个分类标签的可能性。

7）根据最高概率的标签判定分类结果。HP-RASP系统在Softmax函数生成的概率分布中选择最大概率对应的标签作为预测结果。

8）计算梯度，反向传播更新参数矩阵。HP-RASP系统的BERT模型通过以交叉熵损失函数为目标，并通过自动微分技术计算损失函数对模型参数的梯度。参数更新采用Adam优化器，结合批量梯度下降策略，动态调整学习率以加快模型的收敛速度。

此外，为在保持检测精度的同时降低性能开销，模型在训练后被转换至ONNX格式。这样，HP-RASP系统在检测流量时，能有效利用BERT的高准确率并保持较短的响应时间。

4 实验评估

4.1 评估设置

4.1.1 评估基准

针对SQL注入攻击，评估采用了两个数据集：GitHub开源项目payloadbox[47]中的sql-injection-payload-list数据集和Kaggle的SQLi数据集[48]。其中，payloadbox是github上的一个开源项目，它包含了完整类型的SQL注入，如盲目的SQL注入攻击、条件响应、时间延迟等。Kaggle的SQLi数据集也是常用的SQLi数据集，本文将其用作良性数据补充。本文最终使用了3060条样本数据作为SQL注入系统的测试数据，其中1939条是良性数据，其他1121条数据是恶意数据。实验中，本文将数据集按照8:1:1的比例划分为训练集、测试集和验证集，确保了模型的泛化能力和性能的客观性。

在XSS攻击的评估中，选用了GitHub开源项目payloadbox[47]的xss-payload-list数据集。该数据集包含超过6000条有效载荷，涵盖反射型XSS和存储型XSS攻击。数据集中包括了历史上多种复杂的XSS攻击模式，代表了实际Web应用场景中常见的XSS攻击特征。本文最终使用了6606条反射型XSS和6606条存储型XSS作为测试数据。

对于Java反序列化攻击，本文使用ysoserial[49]工具生成评估数据集。该工具提供了一系列在Java库中发现的面向属性编程工具链，能够生成多种有效的反序列化攻击负载，用于模拟常见Java反序列化漏洞。

4.1.2 实验基线

目前学术研究和生产环境中针对Java的RASP开源产品较少，同类型研究均未开源[26-28]，无法作为实验对比对象，因此，本文实验基线选择的对比系统是开源项目OpenRASP[22]。OpenRASP是目前被广泛应用的开源RASP系统，对比其性能有助于展示HP-RASP的创新效果。

4.1.3 评估标准

为了更加科学地评估结果，本文使用了机器学习中常用的评价指标，实验评估主要从精确率、召回率和*F*1分数3项指标进行衡量。其中，精确率表示检测到的攻击中实际为攻击的比例，以减少误报率；召回率表示实际存在的攻击中成功检测到的比例，以体现系统的防护能力；*F*1分数是Precision和Recall的调和平均数，用于平衡检测精度和召回率。由于XSS攻击

和Java反序列化攻击数据集缺少良性样本，因此这两者仅测试召回率。

4.1.4 实验设置

实验在Ubuntu 16.04 LTS服务器上进行，配置为2 vCPU、2.9 GHz，主要环境包括Java 1.8用于RASP实现，Tomcat 5.8模拟攻击请求。BERT模型的训练和推理通过Python 3.7和PyTorch 1.12实现，系统的输出结果以JSON格式记录。

4.2 实验结果

在评估中，本文比较了HP-RASP与OpenRASP系统在SQL注入、XSS攻击和Java反序列化攻击3种防护类型上的表现。

4.2.1 SQL 注入防护

在SQL注入防护方面，本文使用payloadbox数据集[47]和Kaggle SQLi数据集[48]进行测试，实验结果如表3所示。HP-RASP结合BERT模型表现效果显著优于OpenRASP，在召回率保持优异的同时，精确率和*F*1分数均显著高于对比系统。这表明在SQL注入的检测中，BERT模型能够有效提升恶意请求的检测准确率。与HP-RASP相比，OpenRASP系统由于缺乏对语义的理解，一些经过精心构造的恶意载荷可绕过OpenRASP的检测，从而导致精确率偏低。

表 3 SQL 注入防护实验结果

系统	精确率	召回率	*F*1 分数
HP-RASP	81.9%	98.3%	89.3%
OpenRASP	44.3%	97.9%	61.0%

4.2.2 XSS 攻击防护

在XSS攻击防护上，HP-RASP使用了payloadbox[47]的xss-payload-list数据集进行测试。该数据集包括了反射型XSS和存储型XSS的多种负载。实验结果如表4所示，相较于OpenRASP仅支持反射型XSS检测，HP-RASP成功实现了对存储型XSS的检测，并且对两种XSS攻击的召回率均达到了99%以上。

4.2.3 Java 反序列化攻击防护

对于Java反序列化攻击，HP-RASP使用ysoserial[49]生成的模拟数据集进行验证。HP-RASP和OpenRASP的召回率均为84.62%，表明在Java反序列化攻击防护方面，两者的检测效果基本相当，但HP-RASP在扩展检测范围和提升检测准确性上展现出进一步的潜力。

表 4 XSS 攻击防护实验结果

攻击类型	实验结果	HP-RASP	OpenRASP
反射型 XSS	攻击次数 / 次	6606	6606
	拦截次数 / 次	6598	6408
	召回率	99.9%	97.0%
存储性 XSS	攻击次数 / 次	6606	6606
	拦截次数 / 次	6598	0
	召回率	99.9%	0

4.3 时间消耗

在时间消耗评估中，本文测试了RASP系统对应用响应时间的影响。由于RASP的部署不可避免地会带来一定的时间开销，因此在工程应用中，耗时指标是评价系统性能的重要参数。实验中，本文在每种防护方案下使用200个线程并发发送了共计20000次请求，并设置了多种指标来衡量不同方案的耗时情况。表5展示了不同解决方案时间消耗的实验结果，本文对比了多个解决方案，包括原始应用程序、OpenRASP的应用程序、使用 HP-RASP 的应用程序以及使用去掉BERT模型的HP-RASP 的应用程序。其中指标50% Line、90% Line和95%Line分别表示50%、90%和95%的请求耗时未超过该值，仅分别有50%、10%和5%的请求耗时超过该值。

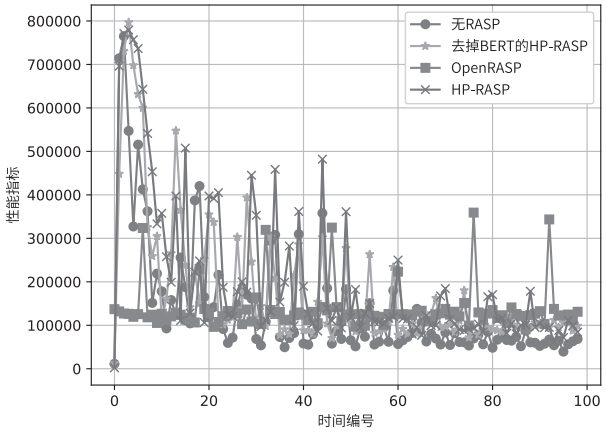
实验结果表明，RASP的部署对应用的吞吐量几乎没有影响。与OpenRASP相比，HP-RASP系统展现出更佳的响应速度，在恶意和正常请求的响应时间上均表现更佳。具体而言，与未加RASP系统的应用相比，HP-RASP的90%响应时间线仅比原应用增加了不到20%的延迟，在绝大多数请求的响应时间上表现为可接受的延迟。此外，加入BERT模型对系统的时间开销影响较小，表明模型在提升检测准确性的同时，并未显著增加系统的响应时间。实验表明，HP-RASP在不显著增加响应时间的前提下，能够有效提升应用的安全性，从而在实际应用中具备较高的工程适用性。

表 5 不同解决方案的耗时比较

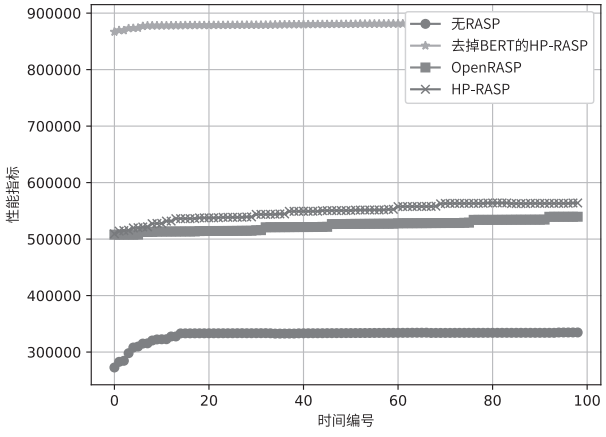
解决方案	50% Line /ms	90% Line /ms	95% Line /ms	吞吐率 /QPS	接受速率 /KB·s^{-1}	发送速率 /KB·s^{-1}
原始应用程序（良性）	4	6	6	200.1	1016.11	29.90
原始应用程序（恶意）	5	5	5	200.0	1025.18	33.10
OpenRASP（良性）	5	6	7	199.8	1029.22	29.86
OpenRASP（恶意）	5	9	30	199.8	1062.84	36.20
去掉 BERT 的 HP-RASP（良性）	5	7	10	199.9	1028.87	29.87
去掉 BERT 的 HP-RASP（恶意）	5	7	9	199.9	1059.99	34.77
HP-RASP（良性）	5	7	10	200.0	1028.93	29.87
HP-RASP（恶意）	5	7	9	200.1	1069.78	34.79

4.4 系统资源消耗

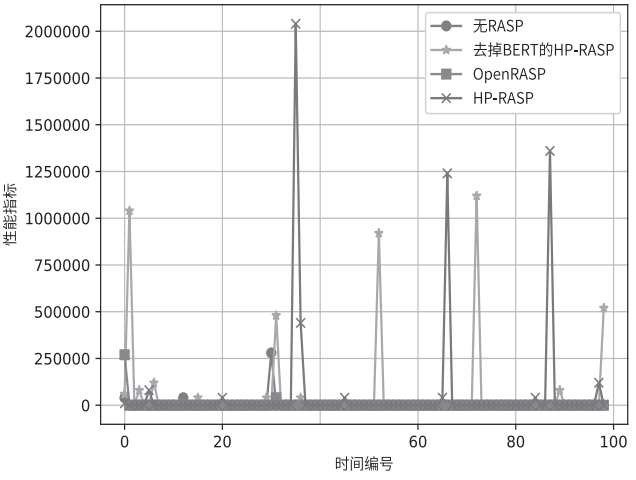
本文使用脚本监控每个解决方案在运行时处理良性和恶意样本的资源使用情况。详细结果如图6和图7所示。



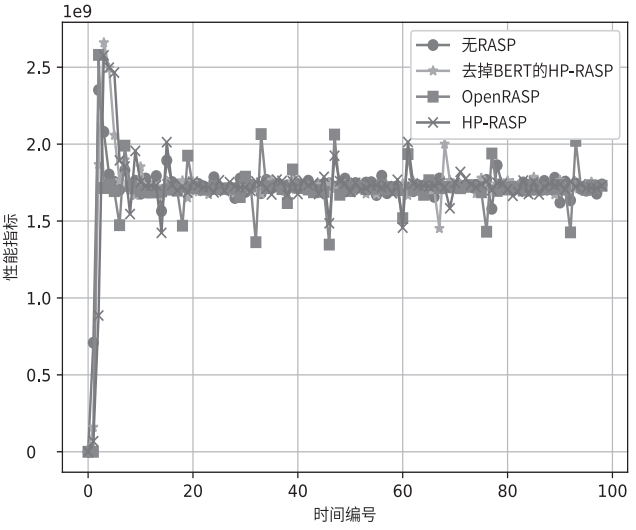
a）良性样本 CPU 指标



b）良性样本内存指标

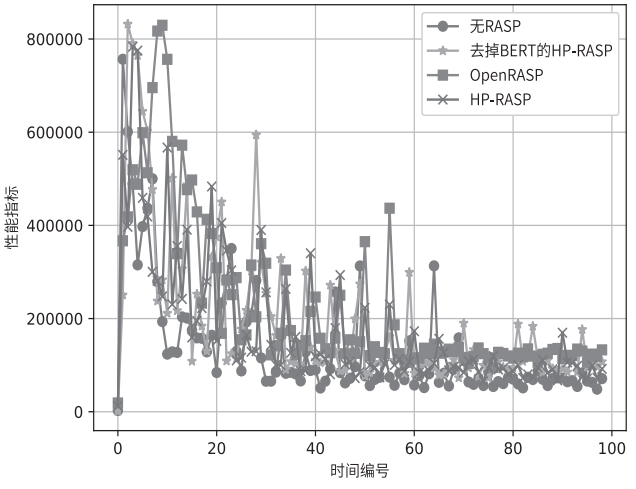


c）良性样本磁盘 IO 指标

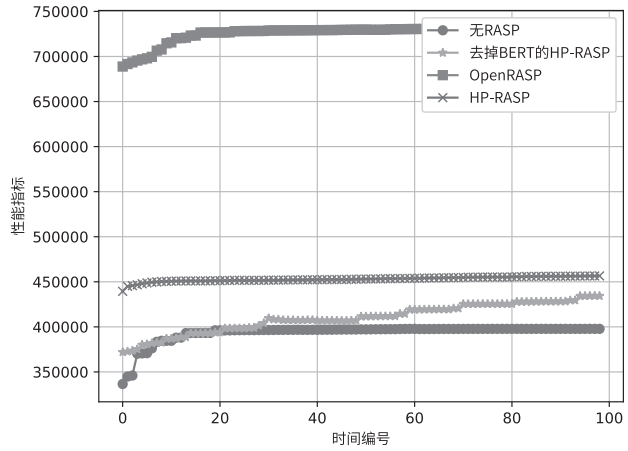


d）良性样本网络 IO 指标

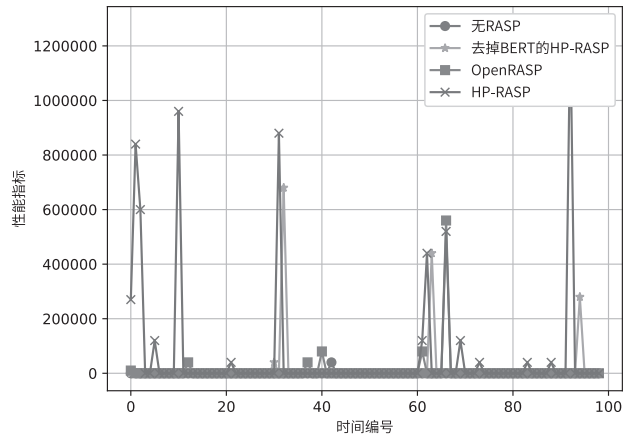
图 6 不同解决方案在处理良性样本时系统资源比较



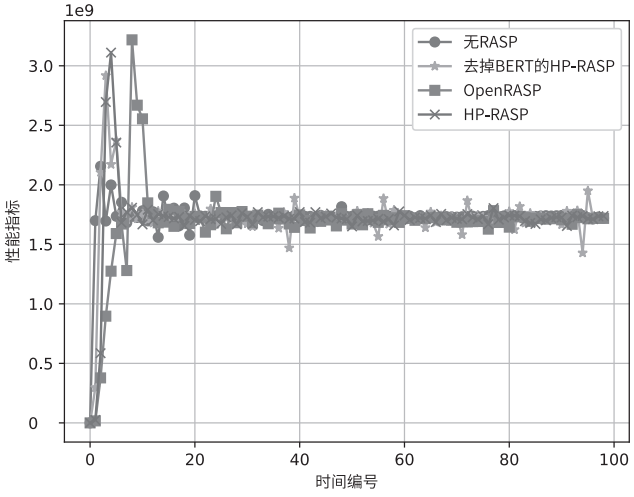
a）恶意样本 CPU 指标



b）恶意样本内存指标



c）恶意样本磁盘 IO 指标



d）恶意样本网络 IO 指标

图 7 不同解决方案的处理恶意样本时系统资源比较

从图6和图7可以看出，各解决方案在面对良性和恶意请求时，系统的网络IO指标相当，这表明HP-RASP系统在面对多样化外部请求时，发送和接收保持稳定。两个名为CPU指标的图表显示，本文系统在计算资源消耗方面显著低于OpenRASP系统，对原始应用程序影响较小。在系统存储资源消耗方面，BERT模型占用了一定量的内存资源，这是可以预见的。此外，除了一些例外，不同解决方案的磁盘资源消耗保持在稳定范围内。总体而言，HP-RASP系统在资源消耗实验中表现良好，与原始应用程序相比，增加是可控的，并且相对于原型系统有了更大的改进。

5 未来展望

本研究的主要局限在于XSS和Java反序列化攻击的训练数据。由于缺少高质量、可泛化的攻击样本，本文在这两个攻击类型的模型训练上遇到一定挑战。尽管本文使用现有开源数据集并生成了一定量的样本，但是这些数据尚不足以实现精细的模型优化。因此，当前的系统对SQL注入的检测效果最佳，而在XSS和Java反序列化攻击的识别上表现尚待进一步提升。未来的研究可以通过构建更为全面的训练集来完善模型的防护能力。

6 结束语

本文提出一种基于RASP的混合防护系统HP-RASP，结合规则匹配和深度学习模型，显著增强了对恶意SQL注入、XSS攻击和Java反序列化攻击的检测效果，并在不显著增加系统响应时间的前提下实现了对应用更全面的保护。在系统架构设计上，HP-RASP通过将深度学习模型融入传统的运行时应用自我保护系统，使系统具备更强的恶意行为识别能力。实验结果表明，HP-RASP在检测率和响应时间上均有较好的表现，验证了该方法在实际工程应用中的可行性。

参考文献：

[1] JOHN P, TERRY A H. SANS 2022 Top New Attacks and Threat Report[EB/OL]. (2022-09-12)[2024-10-22]. https://www.sans.org/white-papers/sans-2022-top-new-attacks-threat-report/.

[2] ALWAN Z S, YOUNIS M F. Detection and Prevention of SQL Injection Attack: A Survey[J]. International Journal of Computer Science and

Mobile Computing, 2017, 6(8): 5–17.
[3] GUNAWAN T S, KASIM L M, KARTIWI M, et al. Penetration Testing Using Kali Linux: SQL Injection, XSS, Wordpres, and WPA2 Attacks[J]. Indonesian Journal of Electrical Engineering and Computer Science, 2018, 12(2): 729–737.
[4] APPLEBAUM S, GABER T, AHMED A. Signature–Based and Machine–Learning–Based Web Application Firewalls: A Short Survey[J]. Procedia Computer Science, 2021, 189: 359–367.
[5] JUNIOR M D, EBECKEN N F. A New WAF Architecture with Machine Learning for Resource–Efficient Use[J]. Computers & Security, 2021, 106: 102290.
[6] RAZZAQ A, HUR A, SHAHBAZ S, et al. Critical Analysis on Web Application Firewall Solutions[C]//IEEE. 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS). New York: IEEE, 2013: 1–6.
[7] NGUYEN H T, TORRANO–GIMENEZ C, ALVAREZ G, et al. Application of the Generic Feature Selection Measure in Detection of Web Attacks[C]//Springer. Computational Intelligence in Security for Information Systems. Heidelberg: Springer, 2011: 25–32.
[8] WEISSBACHER M, ROBERTSON W, KIRDA E, et al. ZigZag: Automatically Hardening Web Applications against Client–Side Validation Vulnerabilities[C]//USENIX. The 24th USENIX Security Symposium (USENIX Security 15). Berkeley: USENIX, 2015: 737–752.
[9] PRABHUDESAI P, BHALERAO A A, PRABHUDESAI R. Web Application Firewall: Artificial Intelligence Arc[J]. International Research Journal of Engineering and Technology, 2019, 6(8): 3706–3708.
[10] TEKEREK A, BAY O F. Design and Implementation of an Artificial Intelligence–Based Web Application Firewall Model[J]. Neural Network World, 2019, 29(4): 189–206.
[11] DEMETRIO L, VALENZA A, COSTA G, et al. WAF–A–MoLE: Evading Web Application Firewalls through Adversarial Machine Learning[C]//ACM. Proceedings of the 35th Annual ACM Symposium on Applied Computing. New York: ACM, 2020: 1745–1752.
[12] YUAN E, MALEK S. A Taxonomy and Survey of Self–Protecting Software Systems[C]//IEEE. 2012 7th International Symposium on Software Engineering for Adaptive and Self–Managing Systems (SEAMS). New York: IEEE, 2012: 109–118.
[13] CLINCY V, SHAHRIAR H. Web Application Firewall: Network Security Models and Configuration[C]//IEEE. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). New York: IEEE, 2018: 835–836.
[14] SALEMI M. Automated Rules Generation into Web Application Firewall Using Runtime Application Self–Protection[D]. Louvain–La–Neuve: Ecole Polytechnique De Louvain, 2020.
[15] SETH A. Comparing Effectiveness and Efficiency of Interactive Application Security Testing (IAST) and Runtime Application Self–Protection (RASP) Tools[M]. Raleigh: North Carolina State University, 2022.
[16] ČISAR P, ČISAR S M. The Framework of Runtime Application Self–Protection Technology[C]//IEEE. 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI). New York: IEEE, 2016: 81–86.
[17] DEVLIN J, CHANG Mingwei, LEE K, et al. BERT: Pre–Training of Deep Bidirectional Transformers for Language Understanding[EB/OL]. [2024–10–20]. https://arxiv.org/abs/1810.04805v2.
[18] ROGERS A, KOVALEVA O, RUMSHISKY A. A Primer in BERTology: What We Know about How BERT Works[J]. Transactions of the Association for Computational Linguistics, 2020, 8: 842–866.
[19] WHANG T, LEE D, LEE C, et al. An Effective Domain Adaptive Post–Training Method for BERT in Response Selection[EB/OL]. [2024–10–20]. https://arxiv.org/abs/1908.04812v2.
[20] LI Junlong, ZHANG Zhuosheng, ZHAO Hai, et al. Task–Specific Objectives of Pre–Trained Language Models for Dialogue Adaptation[EB/OL]. (2020–09–10)[2024–10–20]. https://www.semanticscholar.org/paper/Task–specific–Objectives–of–Pre–trained–Language–Li–Zhang/1cad933afc55f1a562e27ebd4f65c5d0f5a6c26a.
[21] SUN Yi, ZHENG Yu, HAO Chao, et al. NSP–BERT: A Prompt–Based Few–Shot Learner through an Original Pre–Training Task: Next Sentence Prediction[EB/OL]. [2024–10–20]. https://arxiv.org/abs/2109.03564v2.
[22] Baidu. OpenRASP[EB/OL]. (2017–8–10)[2024–10–20]. https://github.com/baidu/openrasp.
[23] Gartner. IT Glossary[EB/OL]. (2012–06–07)[2024–10–20]. https://www.gartner.com/en/information–technology/glossary/runtime–application–self–protection–rasp.
[24] LANE A. Understanding and Selecting RASP: Technology Overview[EB/OL]. (2016–05–17)[2024–10–20]. https://securosis.com/blog/understanding–and–selecting–rasp–technology–overview.
[25] YIN Zhongxu, LI Zhufeng, CAO Yan. A Web Application Runtime Application Self–Protection Scheme against Script Injection Attacks[C]//Springer. Cloud Computing and Security. Heidelberg: Springer, 2018: 566–577.
[26] QIU Ruonan, HU Anqi, PENG Guojun, et al. A General Detection and Location Scheme for Java Web Framework Vulnerability Based on RASP Technology[J]. Journal of Wuhan University (Natural Science Edition), 2020, 66(3): 285–296.
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.
[27] LI Yulin, CHEN Libo, LIU Yujiang, et al. Java Deserialization Vulnerability Defense Technology Based on Run–Time Detection[J]. Chinese Journal of Network and Information Security, 2024, 10(2): 154–164.
[28] YU Hang, WANG Shuai, JIN Huamin. RASP Based Web Security Detection Method[J]. Telecommunications Science, 2020, 36(11): 113–120.
余航，王帅，金华敏 . 基于RASP的Web安全检测方法 [J]. 电信科学，2020，36（11）：113–120.
[29] LIEM C, ABDALLAH E, OKOYE C, et al. Runtime Self–Protection in a Trusted Blockchain–Inspired Ledger[EB/OL]. [2024–10–20]. https://www.academia.edu/38279827/Runtime_Self_Protection_in_a_Trusted_Blockchain_inspired_Ledger.
[30] YANG Wenchuan, PENG Jing. Research on EVM–Based Smart

Contract Runtime Self-Protection Technology Framework[C]// Springer. Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications (WAINA-2020). Heidelberg: Springer, 2020: 617-627.

[31] HALFOND W G J, ORSO A. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks[C]//ACM. Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. New York: ACM, 2005: 174-183.

[32] KOMIYA R, PAIK I, HISADA M. Classification of Malicious Web Code by Machine Learning[C]//IEEE. 2011 3rd International Conference on Awareness Science and Technology (iCAST). New York: IEEE, 2011: 406-411.

[33] PUTTHACHAROEN R, BUNYATNOPARAT P. Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique[C]//IEEE. The 13th International Conference on Advanced Communication Technology (ICACT2011). New York: IEEE, 2011: 1090-1094.

[34] ZHANG Qianjie, CHEN Hao, SUN Jianhua. An Execution-Flow Based Method for Detecting Cross-Site Scripting Attacks[C]//IEEE. The 2nd International Conference on Software Engineering and Data Mining. New York: IEEE, 2010: 160-165.

[35] GUO Chun, CAI Wenyan, SHEN Guowei, et al. Research on SQL Injection Attacks Detection Method Based on the Truncated Key Payload[J]. Netinfo Security, 2021, 21(7): 43-53.

郭春，蔡文艳，申国伟，等 . 基于关键载荷截取的 SQL 注入攻击检测方法 [J]. 信息网络安全，2021，21（7）：43-53.

[36] HUANG Kaijie, WANG Jian, CHEN Jiongyi. A Large Language Model Based SQL Injection Attack Detection Method[J]. Netinfo Security, 2023, 23(11): 84-93.

黄恺杰，王剑，陈炯峄 . 一种基于大语言模型的 SQL 注入攻击检测方法 [J]. 信息网络安全，2023，23（11）：84-93.

[37] MITROPOULOS D, STROGGYLOS K, SPINELLIS D, et al. How to Train Your Browser[J]. ACM Transactions on Privacy and Security, 2016, 19(1): 1-31.

[38] TAN Yuchen, CAI Jingjing, NI Chen. Research on Web Attack Detection Technology Based on Deep Learning[J]. Netinfo Security, 2020, 20(S2): 122-126.

谭宇辰，蔡晶晶，倪辰 . 基于深度学习的 Web 攻击检测技术研究 [J]. 信息网络安全，2020，20(S2): 122-126.

[39] MA Zheng, CHEN Xuebin, ZHANG Guopeng, et al. XSS Attack Detection Method Based on Genetic Algorithm and Support Vector Machine[J]. Journal of Jiangsu University (Natural Science Edition), 2024, 45(6): 686-693.

马征，陈学斌，张国鹏，等 . 基于遗传算法和支持向量机的 XSS 攻击检测方法 [J]. 江苏大学学报（自然科学版），2024，45（6）：686-693.

[40] KOUTROUMPOUCHOS N, LAVDANIS G, VERONI E, et al. ObjectMap: Detecting Insecure Object Deserialization[C]//ACM. Proceedings of the 23rd Pan-Hellenic Conference on Informatics. New York: ACM, 2019: 67-72.

[41] CRISTALLI S, VIGNATI E, BRUSCHI D, et al. Trusted Execution Path for Protecting Java Applications against Deserialization of Untrusted Data[C]//Springer. International Symposium on Recent Advances in Intrusion Detection. Heidelberg: Springer, 2018: 445-464.

[42] LI Yulin, CHEN Libo, LIU Yujiang, et al. Java Deserialization Vulnerability Defense Technology Based on Run-Time Detection[J]. Chinese Journal of Network and Information Security, 2024, 10(2): 154-164.

[43] ZHENG Peng, SHA Letian. Java Deserialization Vulnerability Detection Method Based on Hybrid Analysis[J]. Computer Engineering, 2023, 49(12): 136-145.

郑鹏，沙乐天 . 基于混合分析的 Java 反序列化漏洞检测方法 [J]. 计算机工程，2023，49（12）：136-145.

[44] XIANG Hui, XUE Yunhao, HAO Lingxin. Large Language Model-Generated Text Detection Based on Linguistic Feature Ensemble Learning[J]. Netinfo Security, 2024, 24(7): 1098-1109.

项慧，薛鋆豪，郝玲昕 . 基于语言特征集成学习的大语言模型生成文本检测 [J]. 信息网络安全，2024，24（7）：1098-1109.

[45] Oracle. Interface Instrumentation[EB/OL]. [2024-10-20]. https://docs.oracle.com/javase/7/docs/api/java/lang/instrument/Instrumentation.html.

[46] HEUSER S, NADKARNI A, ENCK W, et al. ASM: A Programmable Interface for Extending Android Security[C]//USENIX. The 23rd USENIX Security Symposium (USENIX Security 14). Berkeley: USENIX. 2014: 1005-1019.

[47] İsmail T. Payloadbox[EB/OL]. (2024-06-18)[2024-10-20]. https://github.com/payloadbox/.

[48] Syed S. Sql Injection Dataset[EB/OL]. (2021-09-09)[2024-10-20]. https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset.

[49] Chris F. Ysoseria[EB/OL]. (2024-05-31)[2024-10-20]. https://github.com/frohoff/ysoserial.