·网络空间安全·　　文章编号：1000-3428(2022)11-0120-07　　文献标志码：A　　中图分类号：TP393.08

基于循环神经网络的Web应用防火墙加固方案

朱思猛[1,2]，杜瑞颖[1,2]，陈　晶[1,2]，何　琨[1,2]

（1.武汉大学 国家网络安全学院 空天信息安全与可信计算教育部重点实验室，武汉 430072；
2.武汉大学 日照信息技术研究院，山东 日照 276827）

摘　要：Web应用防火墙（WAF）基于一组规则检测和过滤进出Web应用程序的HTTP流量，鉴于恶意流量的复杂性，需要对WAF规则进行不断更新以抵御最新的攻击。然而，现有的WAF规则更新方法都需要专业知识来人工设计关于某种攻击的恶意测试流量，并针对该恶意流量生成防护规则，这种方法十分耗时且不能扩展到其他类型的攻击。提出一种基于循环神经网络(RNN)的Web应用防火墙加固方案，在不依赖任何专业知识的情况下自动化加固WAF。使用RNN模型生成恶意攻击样本，从中找到能够绕过WAF的恶意攻击，发现WAF规则存在的安全风险。在此基础上，通过设计评分函数找到恶意攻击样本的重要字符串来生成加固签名，阻止后续类似的攻击，并设计简化的正则表达式作为加固签名的表达形式。在4款WAF上针对SQL注入、跨站脚本攻击和命令注入这3种攻击进行测试，结果显示，该方案成功生成了大量绕过WAF的恶意样本，WAF针对这些样本的平均拦截率仅为52%，与传统突变方案和SQLMap工具相比能够生成更多绕过恶意攻击，在应用加固签名后，WAF的恶意攻击拦截率提升至90%以上且误报率维持为0，表明加固签名成功阻止了这些绕过攻击，验证了所提方案的有效性。

关键词：Web应用防火墙；循环神经网络；SQL注入；跨站脚本；命令注入

开放科学（资源服务）标志码（OSID）：

中文引用格式：朱思猛，杜瑞颖，陈晶，等. 基于循环神经网络的Web应用防火墙加固方案[J]. 计算机工程，2022，48(11)：120-126.

英文引用格式：ZHU S M，DU R Y，CHEN J，et al. Web application firewall reinforcement scheme based on recurrent neural network[J]. Computer Engineering，2022，48(11)：120-126.

Web Application Firewall Reinforcement Scheme Based on Recurrent Neural Network

ZHU Simeng[1,2]，DU Ruiying[1,2]，CHEN Jing[1,2]，HE Kun[1,2]

（1.Key Laboratory of Aerospace Information Security and Trusted Computing，Ministry of Education，School of Cyber Science and Engineering，Wuhan University，Wuhan 430072，China；
2.Rizhao Institute of Information Technology，Wuhan University，Rizhao，Shandong 276827，China）

【Abstract】Web Application Firewall（WAF）detects and filters HTTP traffic to and from a Web application via a set of rules. Owing to the complexity of malicious traffic，WAF rules must be constantly updated to defend against latest or advanced attacks. However，existing methods for updating WAF rules require high degree of human expertise to manually design malicious test traffic for a particular attack and generate protection rules for malicious traffic，which is time-consuming and cannot be adapted to other types of attacks. In this study，a WAF reinforcement scheme based on Recurrent Neural Network（RNN）is proposed to automate the reinforcement of the WAF without relying on any human expert knowledge. It generates malicious payloads through RNN and discovers bypassing malicious payloads against WAF from the payloads，that is，to discover the security risks of the WAF rules.Then it designs scoring functions to find the important strings of the malicious payloads to generate signatures and block subsequent similar attacks，and designs a simplified regular expression as the expression of the strengthened signature.We test four WAFs and examine three types of attacks：SQL injection，Cross-Site Scripting（XSS）and Command Injection（CI）.The results show that the proposed scheme successfully generates a large number of malicious payloads that bypass the WAF，and the average blocking rate of the WAF is only 52%.We also generate more bypassed malicious attacks compared with traditional mutation schemes and SQLMap. After applying the signatures，the WAF malicious attack blocking rate increased to over 90% and maintained a false positive rate of 0.This shows that the signatures successfully block these bypassed attacks，thereby validating the effectiveness of the proposed scheme.

基金项目：国家重点研发计划（2021YFB2700200）；国家自然科学基金（U1836202，61772383，61702379，62172303）。

作者简介：朱思猛（1997—），男，硕士研究生，主研方向为网络安全；杜瑞颖、陈　晶，教授、博士生导师；何　琨，副研究员。

收稿日期：2021-12-20　　**修回日期：**2022-02-18　**E-mail：**zhusim@whu.edu.cn

【Key words】Web Application Firewall（WAF）；Recurrent Neural Network（RNN）；SQL injection；Cross-Site Scripting（XSS）；Command Injection（CI）
DOI：10. 19678/j. issn. 1000-3428. 0063581

0　概述

Web应用程序部署在Web服务器上，并通过Web浏览器提供可访问的服务[1]。由于具有便利性，Web应用程序在互联网上得到了广泛的应用，如网络邮件、网上银行等，同时也出现了针对Web应用程序的各种网络攻击。研究报告显示[2]，针对Web应用程序的攻击占所有网络威胁的32%以上，是最活跃的一类网络攻击。为了防御这类攻击，Web应用防火墙（Web Application Firewall，WAF）被广泛应用[3-4]。WAF通过检查传入的HTTP流量来确定是拦截还是将流量转发到目标Web应用程序。多数WAF根据定义的规则来做出决策，这些规则可以具体表现为匹配恶意Payload特征的正则表达式。随着针对Web应用程序的攻击不断发展演变，WAF的被动防御很难拦截新出现的攻击，因此，需要及时测试WAF的安全防护能力，并更新其规则来进行加固，以此拦截新的攻击，并进一步阻止后续类似的攻击。

WAF安全能力测试的关键步骤是生成有效的测试输入，其中可分为两类：白盒测试和黑盒测试。白盒测试需要访问相关源码[5]，这在现实环境下很难实现。黑盒测试在不了解目标内部机制的情况下工作，可进一步分为基于突变的黑盒测试和基于生成的黑盒测试。基于突变的方法通过应用精心设计的突变操作组合来修改现有Payload，以获得测试输入[6-8]，基于生成的方法则根据设计的攻击语法生成测试输入[9]。然而，基于突变和基于生成的方法都依赖对某种攻击的专业知识来设计突变操作或生成语法，且仅针对特定目标，效果不够理想，只适合已知的攻击检测，很难检测到新的或未知的攻击[10-11]。

本文基于循环神经网络（Recurrent Neural Network，RNN）设计一个自动化加固WAF的方案，测试给定的WAF并生成用于规则更新的加固签名。针对现有方案的局限性，以恶意攻击数据集为基础，利用RNN模型生成更多的恶意攻击Payload，并将这些Payload发送至被WAF保护的应用程序，以发现WAF存在的安全防护问题。在此基础上，对成功绕过WAF的恶意Payload生成加固签名，并设计简化的正则表达式作为表示加固签名的方式。

1　基础知识

1.1　Web应用程序攻击

Web应用程序是目前通过Internet提供信息和服务最流行的平台之一。随着Web应用程序越来越多地被用于关键服务，出现了很多针对Web应用程序的攻击，如SQL注入、跨站脚本（Cross-Site Scripting，XSS）、命令注入（Command Injection，CI）、cookie盗窃、浏览器劫持等。由于许多Web应用程序与后端数据库系统交互，因此针对Web应用程序的攻击会导致严重的后果。本文主要研究SQL注入、跨站脚本和命令注入这3种典型的攻击[12-14]。

SQL注入攻击用于攻击数据驱动的应用程序。在这些应用程序中，恶意SQL语句被插入到字段中执行，数据库信任Web应用程序并执行应用程序发出的查询。使用这类攻击，攻击者可以操纵SQL查询并导致意外执行。

跨站脚本攻击是一种注入类型的攻击。攻击者可以利用这类攻击在网站上注入恶意的客户端代码，当被攻击者登陆网站时就会自动运行这些恶意代码，从而攻击者可以突破网站的访问权限，冒充被攻击者。

命令注入攻击的目标是通过易受攻击的应用程序在主机操作系统上执行任意命令。当应用程序将不安全用户提供的数据传递给系统shell时，就会发生这类攻击。

1.2　循环神经网络

循环神经网络（RNN）用于对序列数据建模[15]，如用于文本处理、程序分析等[16-18]，一般由输入层、隐藏层和输出层组成。RNN使用隐藏状态作为携带时间步之间信息的短期记忆，能够很好地处理序列信息。

然而，RNN存在梯度消失和梯度爆炸的问题，会导致消耗大量的时间并难以做进一步优化。对此，HOCHREITER等提出了长短时记忆（Long Short Term Memory，LSTM）网络。LSTM网络通过引入单元状态弥补了原始RNN的缺陷，利用遗忘门、输入门和输出门，分别控制上一时刻的单元状态保留到当前时刻、当前时刻网络的输入保存到单元状态以及单元状态输出保留到LSTM当前输出的数量。另一种流行的RNN结构是门控循环单元（Gated Neural Unit，GRU）[19]，其仅使用2个门，即更新门和重置门。GRU在LSTM网络的基础上做了很多简化，同时能够保持与LSTM网络相同的效果。

2　相关研究

关于提升WAF安全防护能力的研究，目前主要集中在检测恶意攻击[20-21]方面，对WAF的安全测试和评估则不够深入，无法解决许多实际问题。相关研究较少的原因在于WAF是一个to-B产品，主要影响企业用户而不是消费者，导致其受众面相对狭窄，同时研究者也很难找到专业的WAF进行测试。

对于WAF的安全测试，文献[9，22]结合决策树模型和进化算法将Web攻击的Payload分解为切片，并使用进化算法进一步生成恶意Payload，但具体攻

击的语法规范需要人工总结，在测试过程中也存在一些限制，如生成的Payload仅源于自己总结的语法规范，不能涵盖新出现的攻击，此外，方案仅对单个WAF有用。文献[7]提出了突变的方案，但加固目标是基于机器学习的WAF而不是真正的WAF。当前，多数WAF都是黑盒的，不仅使用机器学习方法进行防御，而且混合使用多种方法，而上述方案都需要预先设计突变操作或语法，复杂耗时并且只对特定的WAF有效。因此，本文提出一种可应用于各种类型攻击的WAF加固方案。

3 WAF自动化加固方案

本文提出一种WAF自动化加固方案，其可发现绕过给定WAF的恶意Payload，并生成加固签名来修复相应安全防护漏洞。

如图1所示，本文方案的系统框图主要由Payload生成器和签名生成器2个部分组成。首先，发送收集到的数据集到被WAF保护的应用程序，得到被WAF拦截的恶意攻击Payload作为Payload生成器的输入，通过RNN生成更多恶意攻击Payload。然后，签名生成器利用其中可以绕过WAF的Payload，通过评分函数找到对WAF判别有重要影响的字符串，生成可以匹配绕过攻击的加固签名。在这其中一个重要前提是，系统从恶意Payload中生成Payload，这些生成的Payload应视为具有恶意的。

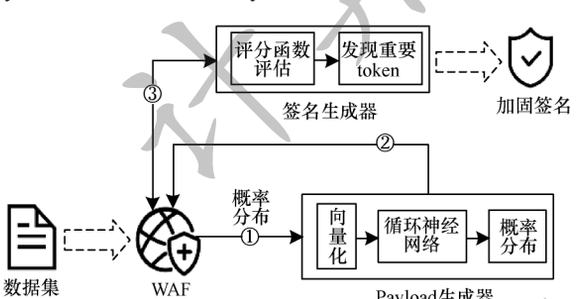

图1 本文方案的系统框图

Fig.1 System block diagram of the proposed scheme

3.1 Payload生成器

Payload生成器负责生成针对WAF的恶意攻击Payload。该模块接收恶意的Payload，并在原始Payload上产生微小扰动，输出更多恶意Payload。笔者认为，由于Payload是否恶意取决于上下文，因此其中应选用能够学习序列结构的神经网络。由于传统基于n-gram的方法受有限长度上下文的限制，因此本文选择使用RNN。RNN允许学习任意长度的上下文来预测下一个字符序列，给定恶意Payload的语料库，RNN模型能够以无监督的方式进行训练来生成新的恶意Payload。由于RNN每个时刻隐藏层的输出都会传递给下一时刻，因此每个时刻的网络都会保留一些来自之前时刻的历史信息，并结合当前时刻的网络状态一并传给下一时刻，这一过程可以表示为：

$$\boldsymbol{h}_t=f_h(\boldsymbol{i}_t,\boldsymbol{h}_{t-1}),\hat{\boldsymbol{y}}_t=f_o(\boldsymbol{h}_t) \tag{1}$$

其中：$\boldsymbol{i}_t$是输入；$\boldsymbol{h}_t$和$\boldsymbol{h}_{t-1}$是隐藏状态向量；$\hat{\boldsymbol{y}}_t$是时刻t的输出；f_h是激活函数；f_o用于计算当前隐藏层输出的概率分布。

为了使输入能够被神经网络处理，Payload生成器要将输入进行向量化，转换成适合处理的结构。首先，需要将字符串映射为数字。设$X=\{x_1,x_2,\cdots,x_N\}$为输入序列，其中$x_t\in\mathbb{N}|1\leqslant t\leqslant N$表示输入序列中位置$t$处字符的自然数。然后，将$x_t$转换为$N$维的one-hot向量$\hat{\boldsymbol{x}}_t=(\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2,\cdots,\hat{\boldsymbol{x}}_N)^{\mathrm{T}}$，定义为：

$$\forall 1\leqslant j\leqslant N,\hat{\boldsymbol{x}}_j=\begin{cases}\boldsymbol{1},j=x_t\\ \boldsymbol{0},\text{其他}\end{cases} \tag{2}$$

通过上述向量化方法得到一个高维的向量，从而获得训练所需的特征。在此基础上，使用GRU神经网络模型解决RNN面临的梯度消失或梯度爆炸的问题。模型的输出层包含了一个前馈网络，其输入为循环神经网络的输出。本文应用softmax函数为预测输入序列的下一个值提供概率分布。最后，使用交叉熵误差函数作为优化目标对GRU模型进行优化，此部分采用ADAM优化算法[23]，其为一种基于梯度的优化算法，与其他算法相比只需要一阶梯度，并且内存占用较少。

3.2 签名生成器

签名生成器的作用是生成匹配恶意攻击的加固签名。该模块接收绕过WAF的恶意Payload作为输入。签名被广泛用于防御攻击，签名系统都有相同的基本假设：Payload存在一个子字符串，该子串在过程中保持不变，并且具有足够的唯一性，可以用作签名而不会导致误报。由于人工生成加固签名通常需要花费大量的时间，因此本文选择使用绕过WAF的Payload来自动生成WAF加固签名。

由于可能会有巨大的搜索空间，因此应找到对WAF判别有最重要影响的字符串。使用4个能够正确反映字符串对判别重要性且计算效率较高的评分函数，来评估对目标WAF做出决策最重要的token，如图2所示，其中，WAF判别结果可以为0或1，结果为1表明实线部分和虚线部分的判别结果不同。

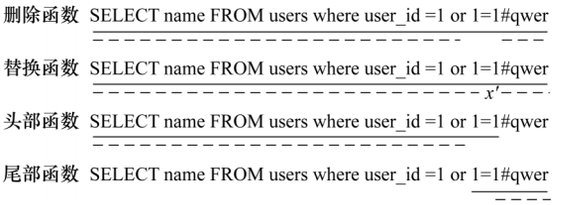

图2 以SQL为例的评分函数说明

Fig.2 Description of scoring function taking SQL as an example

假设输入序列$X=\{x_1,x_2,\cdots,x_N\}$，其中x_i表示第i个位置的token。为根据重要性进行排序，需要衡量第

i个位置的token对判别的影响。使用4个评分函数来评估Payload中每个token对WAF判别的重要性，以f代表WAF的判别结果。下面将分别对4个评分函数进行解释说明。

删除函数比较删除token前后的Payload语句，以评估x_i对WAF的影响，便于直观地理解其重要性。删除函数表达式如下：

$$\mathrm{DS}(x_i)=f(x_1,x_2,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)-f(x_1,x_2,\cdots,x_{i-1},x_{i+1},\cdots,x_n) \tag{3}$$

替换函数将x_i替换为另一个字符串x_i'，通过评估替换的影响来衡量token的重要性。替换函数表达式如下：

$$\mathrm{RS}(x_i)=f(x_1,x_2,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)-f(x_1,x_2,\cdots,x_{i-1},x_i',x_{i+1},\cdots,x_n) \tag{4}$$

由于token前后的字符串是相关的，可以按顺序衡量token的重要性，因此定义头部函数比较x_1到x_i之间的字符串与x_1到x_{i-1}之间的字符串，定义尾部函数比较x_i到x_n之间的字符串与x_{i+1}到x_n之间的字符串。这2种函数的表达式如下：

$$\mathrm{HS}(x_i)=f(x_1,x_2,\cdots,x_{i-1},x_i)-f(x_1,x_2,\cdots,x_{i-1}) \tag{5}$$

$$\mathrm{TS}(x_i)=f(x_i,x_{i+1},\cdots,x_n)-f(x_{i+1},x_{i+2},\cdots,x_n) \tag{6}$$

上述4个函数的结果以0和1表示，表明token发生变化时是否对判别结果产生影响。综合4个评分函数的结果，评估所有token，选择若干个频繁出现且对判别影响较大的字符串token，然后使用正则表达式生成签名，每个签名由一个或多个这样的token组成。正则表达式在灵活性、准确性和效率方面对于攻击检测具有显著的优势。但是，完整的正则表达式过于复杂，而签名不需要其众多的语法规则。因此，本文引入简化正则表达式作为表示签名的方式。简化正则表达式签名是正则表达式的简化形式，其中只包含一个限定符“\S*”，表示任意非空白字符。

通过编辑距离对token进行聚类，将相似的token划分为一个组，在一组中随机查找2个字符串的公共子序列。2个字符串的公共子序列是2个字符串中相同顺序的字节序列，虽然它不一定是连续的，如在字符串“xxabxxxcx”和“abyyyyycyy”中，最长的公共子序列是“abc”。然后，使用此子序列来不断地匹配和产生新的子序列。当某组的子序列不再匹配时，迭代查找不同组子序列的公共子序列。对于结果，在每2个字符之间添加“\S*”，生成一个简化的正则表达式，即加固签名。

4　实验

通过检测SQL注入、XSS攻击和命令注入这3种典型网络攻击来评估方案的有效性，并将本文方案与现有基于突变的解决方案和主流测试工具进行比较。此外，实验还将评估签名生成器的有效性。

4.1　实验设置

在ModSecurity、Naxsi、Safedog、Xwaf这4款WAF上进行实验。根据以下标准选择WAF：首先，WAF来自商业公司或者是GitHub上的高评分项目，并且在安装过程中不会报告错误；其次，WAF可以禁用对挑战黑洞（Challenge Collapsar，CC）攻击的保护，因为这种保护会阻止系统有效地测试WAF。

在实验中，Web应用系统部署在带有Intel® Xeon® CPU E5-2680和32 GB RAM的Linux工作站上运行。所选WAF部署在相同配置的Linux工作站或Windows工作站上。分别在Linux工作站和Windows工作站上安装Ubuntu 18.04和Windows 7。深度学习模型是使用Google的TensorFlow框架[24]实现的，该框架提供了必要的优化算法和损失函数。

4.2　数据集

本文选择自行生成部分数据集，以及在WAF服务器上收集到的数据作为实验所用的恶意数据集，主要是基于以下原因：一方面，这类数据集大多都是内部数据集，由于隐私问题而无法公开共享；另一方面，很多数据集高度匿名或无法满足测试需求。此外，“恶意”和“良性”的定义依赖于具体的应用程序，不存在通用的定义。

在生成的这部分数据集中，对于SQL注入攻击，选择使用随机查询生成器生成数据并运行相应脚本，使其变为恶意Payload。随机查询生成器是SQL Server的官方测试工具，给定语法，该工具返回一组该语法所表示的Payload，然后从现有的渗透测试工具SQLMap中提取恶意脚本来生成SQL注入的Payload。对于命令注入攻击，修改了Commix来根据实验需要生成恶意Payload。Commix是一个开源渗透测试工具，能够自动检测和利用命令注入漏洞。对于XSS攻击，使用XSStrike来生成恶意Payload. XSStrike是一个流行的跨站脚本检测套件，在Github上拥有9 000多个Star。

为获得WAF自身以及加固后的误报率，本文使用来自公共数据集HTTP CSIC 2010[25]的数据作为良性数据集。数据集HTTP CSIC 2010包含针对电子商务Web应用程序生成的流量，其中有36 000个正常请求。

4.3　漏洞检测

绕过目标WAF的Payload数量是衡量WAF安全防护能力的重要指标。当恶意攻击Payload绕过WAF时，表明WAF中存在安全防护问题。

表1列出了WAF安全测试结果。本文并没有针对3种攻击测试所有4款WAF，这是因为某些WAF不能保护应用程序免受特定攻击，当同时满足低拦截率和高通过率时，笔者认为WAF不提供针对此种攻击的保护。由表1可以看出，在测试的4款WAF中，只有ModSecurity和Naxsi可以防止命令注入，这可能是因为命令注入攻击在Web应用程序中相对较少，通常需要与其他攻击结合使用。由表1还可以看出，一些WAF有过于严格的规则，不仅阻止恶意攻击，而且阻止良性请求，这在现实条件下是不可行的，Naxsi就是这种WAF，其提供了拦截阈值选项，因此，需要修改参数使其拦截能力达到较为合理的水平来进行测试。

表1 WAF安全测试结果

Table 1 WAF security test results %

WAF	攻击类型	原始恶意数据拦截率	原始良性数据绕过率	生成恶意Payload拦截率
ModSecurity	SQL注入	83	98	60
	XSS	96	98	41
	CI	84	98	67
Naxsi	SQL注入	95	67	89
	XSS	95	67	70
	CI	72	67	55
Safedog	SQL注入	88	97	47
	XSS	93	97	26
Xwaf	SQL注入	77	81	36
	XSS	95	81	32

表1中的第3列描述了使用原始数据集中恶意Payload测试WAF时的拦截率，其中被拦截的恶意Payload将作为Payload生成器的输入，第3列这一指标显示了WAF的拦截能力。可以看出，4款WAF应对一般的恶意攻击，均展现出了较高的拦截率。

表1中的第4列描述了使用良性Payload测试WAF的结果。可以看出绕过率很高，这表明WAF的误报率很低。然而，并非所有良性数据绕过率都达到了高水平，以Naxsi为例，这款WAF选择牺牲误报率来换取拦截率，因此对于恶意数据和良性数据的拦截率都达到了较高的水平，但对于被WAF拦截的良性Payload，也选择其作为Payload生成器的输入。

表1中的第5列描述了WAF对生成的恶意Payload的拦截率。在实验中，对每类攻击都生成数千个恶意Payload来进行测试。图3~图5直观显示了WAF对生成的恶意攻击Payload的拦截情况，其中每个直方图对应一种攻击，直方图长度表示拦截率。可以看出，所有3种攻击的拦截率都大幅降低，这意味着生成了成功的绕过攻击。由图4可以看出，所有受测试WAF对原始恶意XSS攻击的拦截率都高达90%。然而，以Safedog为例，对于生成的恶意Payload，WAF仅拦截到其中的26%，而未防护到其中大多数的攻击，说明本文提出的方案成功生成了大量恶意绕过攻击。

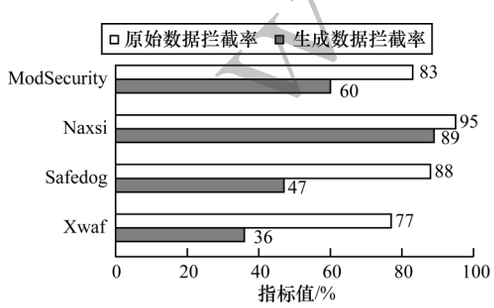

图3 SQL注入攻击Payload的拦截率变化

Fig.3 The change of Payload interception rate during SQL injection attack

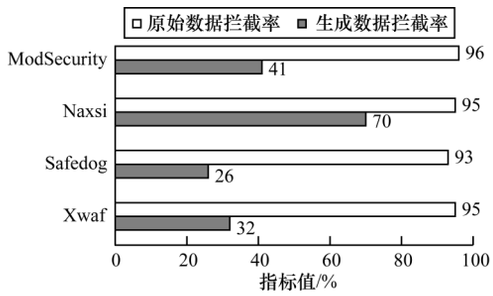

图4 XSS攻击Payload的拦截率变化

Fig.4 The change of Payload interception rate during XSS attack

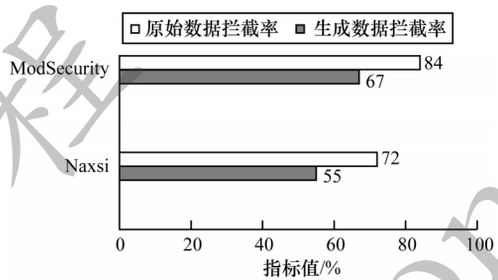

图5 命令注入攻击Payload的拦截率变化

Fig.5 The change of Payload interception rate during command injection attack

4.4 与现有方案的对比

SQL注入攻击是一种常见的攻击，目前已经有许多成熟的应对方案和工具。因此，以SQL注入攻击为基准，将本文提出的方案与基于突变的方案[7]（下文简称突变方案）以及SQLMap工具在ModSecurity和Safedog这2款WAF上进行实验比较。突变方案通过反复随机应用一些预定义的突变操作来生成恶意攻击Payload。SQLMap是一个面向SQL的开源渗透测试工具，利用其绕过脚本来生成Payload。选择SQLMap是因为此工具目前非常流行，被知名机构和安全专家广泛使用。

1）与突变方案的比较。突变操作是一个函数，其改变Payload并保留原始语义。表2中定义了多种突变操作，包括大小写变换、空格替换、注释插入、注释重写等。以大小写交换操作为例，其随机更改Payload中关键字的大小写，由于SQL不区分大小写，因此语义不受影响。将这些突变操作应用于原始数据集以生成绕过Payload。为了防止随机应用突变操作造成的随机性，生成10组数据，并计算最终绕过率的平均值。

2）与SQLMap的比较。SQLMap是一种广泛使用的开源渗透测试工具，能够自动检测和利用SQL注入漏洞。SQLMap可以生成大量不同的SQL注入攻击，涵盖了所有常见的SQL注入攻击技术。本文选用的工具版本1.4.2.38包含57个绕过脚本。在实验中，舍弃了其中一些与编码相关的脚本，因为这些脚本是为特定的后端设计的，将对WAF检测的准确性产生较大影响。本文将其他绕过脚本应用于原始数据集，同时，执行与突变方案相同的操作以减少随机性干扰。

表2　突变操作定义

Table 2　Definition of mutation operation

突变操作	定义	解释
大小写变换	CS(…a…B)→ …A…b	CS(admin' OR 1 = 1#)→ ADmIn' oR 1 = 1#
空格替换	WS(…k_1k_2…)→ …k_1…k_2…	WS(admin' OR 1 = 1#)→ admin' \n or \t 1 = 1#
注释插入	CI(…k_1k_2…)→ …k_1/**/k_2…	CI(admin' OR 1 = 1#)→ admin'/**/OR 1 = 1#
注释重写	CR(…/*s_0*/…#s_1)→ …/*s_0'*/…#s_1'	CR(admin'/**/OR 1 = 1#) → admin'/*abc*/OR 1 = 1#xyz
十六进制编码	IE(…n…)→ …0x[n]$_{16}$	IE(admin'/**/OR 1 = 1#)®admin' OR 0x1 = 1#
符号变换	OS(…⊕…)→ …田…(⊕≡田)	OS(admin' OR 1 = 1#)®admin' OR 1 LIKE 1#

在ModSecurity和Safedog上使用本文方案、突变方案和SQLMap进行实验。在相同的环境中，测试3种方案并对结果与原始绕过率进行比较，如表3所示。由表3可以看出：所有方案都可以产生绕过WAF的Payload，这表明WAF普遍没有提供足够的保护；对于ModSecurity，突变方案和SQLMap都能有效提高WAF的绕过率，但本文方案能够得到最大的提高幅度，且远高于其他2个方案；对于Safedog情况类似，但可以看到，使用突变方案后绕过率出现了降低，这可能是因为WAF提供了针对突变的专门保护，因为突变样本与良性SQL样本的差异很大，最初能够绕过的样本在突变后反而被拦截，导致绕过率降低，这也表明突变方案对于WAF不是通用的；整体上，本文方案效果优于其他2种方案，生成的数据具有较高的绕过率，且生成了更多绕过WAF的恶意攻击。

表3　3种方案生成Payload的绕过率比较

Table 3　Bypassing rate comparison of Payload generated by three schemes　%

WAF	原始数据	SQLMap	突变方案	本文方案
ModSecurity	17	22	24	40
Safedog	12	25	10	53

4.5　生成签名验证

在本节中，将评估输出加固签名的效率。为了生成加固签名，使WAF能够有效拦截绕过的恶意攻击，需要实现以下2个目标：最大化拦截的恶意攻击数量和最小化拦截的良性请求数量。在实验中，将使用拦截率和误报率这2个指标作为检测结果的评价标准，拦截率表示恶意Payload被判别为恶意Payload的比例，误报率表示良性Payload被判别为恶意Payload的比例。本文使用原始数据集中的绕过样本以及良性Payload来评估生成的加固签名。在实验中，需要检验这些签名的效率，即应用加固签名后的误报率和绕过率。

表4体现了应用加固签名后WAF拦截率的变化。签名需要在误报率和拦截率之间保持平衡，如果误报率过高，那么无论拦截率有多高都是没有意义的，因为这将导致良性和恶意Payload的无差别拦截。从表3的最后2列可以看出，本文方案得到的加固签名实现了较高的拦截率。在应用加固签名后，拦截率高达90%甚至100%，表明签名阻止了多数恶意绕过攻击。此外，使用良性Payload对其进行误报率评估，没有一条良性Payload被拦截，表明加固后的WAF对正常流量不会产生影响。笔者对误报率都为0的情况也进行了分析，认为可能原因是恶意Payload与良性Payload区别很大，因此加固签名很难匹配到良性Payload。而现实情况也确实如此，与恶意Payload相似或与良性Payload不同的流量更有可能来自潜在攻击者，正常网络流量中不会出现与恶意Payload相似的情况。总体而言，与应用加固签名之前的情况相比，应用加固签名后WAF的防护能力有了较大的提高，拦截率保持在90%以上。

表4　加固签名评估结果

Table 4　Evaluation result of reinforcement signature　%

WAF	攻击类型	加固前拦截率	加固后拦截率	加固后误报率
ModSecurity	SQL注入	60	99	0
	XSS	41	96	0
	CI	67	95	0
Naxsi	SQL注入	89	100	0
	XSS	70	100	0
	CI	55	98	0
Safedog	SQL注入	47	99	0
	XSS	26	97	0
Xwaf	SQL注入	36	91	0
	XSS	32	98	0

5　结束语

本文提出的WAF加固方案通过RNN模型生成恶意Payload，并用于测试WAF，以发现WAF存在的安全问题，同时还设计了一个签名生成器，利用评分函数找到关键字符串来生成加固签名，保护WAF免受后续类似攻击，解决了传统方案依赖专业知识的问题。实验针对3种攻击类型评估4款WAF，结果表明本文方案发现了大量恶意绕过Payload，生成的加固签名也有效地防止了这些攻击，并且误报率为0。后续将基于强化学习方法优化本文方案，通过从WAF获得反馈来指导神经网络的学习，进一步改善Payload的质量并提高绕过率。

参考文献

[1] LI X, XUE Y. A survey on Web application security [EB/OL]. [2021-12-05]. https://www.isis.vanderbilt.edu/sites/default/files/main_0.pdf.

[2] NTTsecurity. 2019 NTT global threat intelligence report [EB/OL]. [2021-12-05]. https://ishare.iask.sina.com.cn/f/8vy3VW5Iyt.html.

[3] 马月,侯雪城,吴佳帅,等. Web应用防火墙(WAF)技术的综述[J]. 计算机时代,2020(3):13-15,19.
MA Y, HOU X C, WU J S, et al. Research on technologies of Web application firewall[J]. Computer Era, 2020(3): 13-15, 19. (in Chinese)

[4] 刘志光. Web应用防火墙技术分析[J]. 情报探索,2014(3):103-105,129.
LIU Z G. Analysis on firewall technologies of Web-based application[J]. Information Research, 2014(3): 103-105, 129. (in Chinese)

[5] LI Y F, DAS P K, DOWE D L. Two decades of Web application testing—a survey of recent advances [J]. Information Systems, 2014, 43: 20-54.

[6] LEE T, WI S, LEE S, et al. FUSE: finding file upload bugs via penetration testing[C]//Proceedings of 2020 Network and Distributed System Security Symposium. Reston, USA: Internet Society, 2020: 1-10.

[7] DEMETRIO L, VALENZA A, COSTA G, et al. WAF-A-MoLE: evading Web application firewalls through adversarial machine learning[C]//Proceedings of the 35th Annual ACM Symposium on Applied Computing. New York, USA: ACM Press, 2020: 1745-1752.

[8] LYU C, JI S, ZHANG C, et al. MOPT: optimized mutation scheduling for fuzzers [C]//Proceedings of the 28th USENIX Security Symposium. [S. l.]: USENIX, 2019: 1949-1966.

[9] APPELT D, NGUYEN C D, PANICHELLA A, et al. A machine-learning-driven evolutionary approach for testing Web application firewalls [J]. IEEE Transactions on Reliability, 2018, 67(3): 733-757.

[10] BAU J, BURSZTEIN E, GUPTA D, et al. State of the art: automated black-box Web application vulnerability testing [C]//Proceedings of IEEE Symposium on Security and Privacy. Washington D. C., USA: IEEE Press, 2010: 332-345.

[11] DOUPÉ A, COVA M, VIGNA G. Why Johnny can't pentest: an analysis of black-box Web vulnerability scanners [C]//Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Germany: Springer, 2010: 111-131.

[12] BOYD S W, KEROMYTIS A D. SQLrand: preventing SQL injection attacks[C]//Proceedings of International Conference on Applied Cryptography and Network Security. Berlin, Germany: Springer, 2004: 292-302.

[13] 谷家腾,辛阳. 基于动态分析的XSS漏洞检测模型[J]. 计算机工程,2018,44(10):34-41.
GU J T, XIN Y. XSS vulnerability detection model based on dynamic analysis[J]. Computer Engineering, 2018, 44(10): 34-41. (in Chinese)

[14] 朱辉,沈明星,李善平. Web应用中代码注入漏洞的测试方法[J]. 计算机工程,2010,36(10):173-175,178.
ZHU H, SHEN M X, LI S P. Test method on code injection vulnerabilities of Web application[J]. Computer Engineering, 2010, 36(10): 173-175, 178. (in Chinese)

[15] 刘博,王明烁,李永,等. 深度学习在时空序列预测中的应用综述[J]. 北京工业大学学报,2021,47(8):925-941.
LIU B, WANG M L, LI Y, et al. Deep learning for spatio-temporal sequence forecasting: a survey[J]. Journal of Beijing University of Technology, 2021, 47(8): 925-941. (in Chinese)

[16] 何力,郑灶贤,项凤涛,等. 基于深度学习的文本分类技术研究进展[J]. 计算机工程,2021,47(2):1-11.
HE L, ZHENG Z X, XIANG F T, et al. Research progress of text classification technology based on deep learning[J]. Computer Engineering, 2021, 47(2): 1-11. (in Chinese)

[17] LI Z, ZOU D Q, XU S H, et al. VulDeePecker: a deep learning-based system for vulnerability detection [C]//Proceedings of 2018 Network and Distributed System Security Symposium. Reston, USA: Internet Society, 2018: 1-10.

[18] 郭可翔,王衡军,白祉旭. 基于多通道CNN和BiGRU的字词级文本错误检测模型[J]. 计算机工程:2022,48(9):63-70.
GUO K X, WANG H J, BAI Z X. A word-level text error detection model based on multi-channel CNN and BiGRU[J]. Computer Engineering: 2022, 48(9): 63-70. (in Chinese)

[19] CHO K, VAN MERRIENBOER B, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, USA: Association for Computational Linguistics, 2014: 1-10.

[20] ITO M, IYATOMI H. Web application firewall using character-level convolutional neural network [C]//Proceedings of the 14th International Colloquium on Signal Processing & Its Applications. Washington D. C., USA: IEEE Press: 103-106.

[21] 姚琳琳,何倩,王勇,等. 基于分布式对等架构的Web应用防火墙[J]. 计算机工程,2012,38(22):114-118.
YAO L L, HE Q, WANG Y, et al. Web application firewall based on distributed P2P architecture [J]. Computer Engineering, 2012, 38(22): 114-118. (in Chinese)

[22] APPELT D, NGUYEN C D, BRIAND L. Behind an application firewall, are we safe from SQL injection attacks? [C]//Proceedings of the 8th International Conference on Software Testing, Verification and Validation. Washington D. C., USA: IEEE Press, 2015: 1-10.

[23] KINGMA D P, BA J. Adam: a method for stochastic optimization[EB/OL]. [2021-12-05]. https://arxiv.org/abs/1412.6980.

[24] ABADI M, AGARWAL A, BARHAM P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems[EB/OL]. [2021-12-05]. https://arxiv.org/abs/1603.04467.

[25] CSIC. HTTP CSIC 2010[EB/OL]. [2021-12-05]. https://www.isi.csic.es/dataset/.

编辑 金胡考