

**长亭科技**

**雷池(S20)嵌入式测试方案**

**版本1.1**

北京长亭科技有限公司

2020年3月27日

**目录**

[1 概述 5](#_Toc36303648)

[1.1 测试目的 5](#_Toc36303649)

[1.2 测试要求 5](#_Toc36303650)

[1.3 评判标准 5](#_Toc36303651)

[1.4 测试说明 5](#_Toc36303652)

[2 环境准备 6](#_Toc36303653)

[3 网络拓扑 6](#_Toc36303654)

[4 站点管理 7](#_Toc36303655)

[4.1 HTTP站点 7](#_Toc36303656)

[4.2 站点攻击检测策略 8](#_Toc36303657)

[4.2.1 默认策略 8](#_Toc36303658)

[4.2.2 自定义策略 9](#_Toc36303659)

[5. WAF攻击检测功能 10](#_Toc36303660)

[5.1 WEB攻击检测测试 10](#_Toc36303661)

[5.1.1 SQL注入攻击 10](#_Toc36303662)

[5.1.2 XSS攻击 11](#_Toc36303663)

[5.1.3 Java反序列化攻击 11](#_Toc36303664)

[5.1.4 文件上传攻击 12](#_Toc36303665)

[5.1.5 文件包含攻击 13](#_Toc36303666)

[5.1.6 PHP代码注入攻击 14](#_Toc36303667)

[5.1.7 CSRF攻击 15](#_Toc36303668)

[5.1.8 SSRF攻击 16](#_Toc36303669)

[5.1.9 PHP反序列化攻击 17](#_Toc36303670)

[5.1.10 ASP代码注入攻击 17](#_Toc36303671)

[5.1.11 JAVA代码注入攻击 19](#_Toc36303672)

[5.1.12 命令注入攻击 20](#_Toc36303673)

[5.1.13 情报模块 20](#_Toc36303674)

[5.1.14 服务器响应检测模块 21](#_Toc36303675)

[5.1.15 机器人检测模块 22](#_Toc36303676)

[5.2 常用编码解码测试 23](#_Toc36303677)

[5.2.1 BASE64解码 23](#_Toc36303678)

[5.2.2 URL解码 24](#_Toc36303679)

[5.2.3 JSON 解析 25](#_Toc36303680)

[5.2.4 十六进制转换 25](#_Toc36303681)

[5.2.5 斜杠反转义 26](#_Toc36303682)

[5.2.6 XML解析 27](#_Toc36303683)

[5.2.7 PHP反序列化对象解析 27](#_Toc36303684)

[5.2.8 UTF-7解码 28](#_Toc36303685)

[6 自定义规则 29](#_Toc36303686)

[6.1 URL匹配 29](#_Toc36303687)

[6.2 路径匹配 30](#_Toc36303688)

[6.3 Query匹配 31](#_Toc36303689)

[6.4 请求方法匹配 32](#_Toc36303690)

[6.5 Host匹配 33](#_Toc36303691)

[6.6 Cookie匹配 34](#_Toc36303692)

[6.7 User-Agent匹配 35](#_Toc36303693)

[6.8 Referrer匹配 36](#_Toc36303694)

[6.9 Content-Type匹配 37](#_Toc36303695)

[6.10 IP匹配 38](#_Toc36303696)

[6.11 X-Forwarded-For匹配 39](#_Toc36303697)

[6.12 Origin匹配 40](#_Toc36303698)

[6.13 Session匹配 41](#_Toc36303699)

[6.14 Header匹配 42](#_Toc36303700)

[6.15 请求头长度匹配 43](#_Toc36303701)

[6.16 Body匹配 44](#_Toc36303702)

[6.17 POST参数匹配 45](#_Toc36303703)

[6.18 上传文件名匹配 46](#_Toc36303704)

[6.19 HTTP状态码匹配 48](#_Toc36303705)

[6.20 响应内容匹配 49](#_Toc36303706)

[6.21 Request匹配 50](#_Toc36303707)

[7 访问频率控制 51](#_Toc36303708)

[7.1 基于IP限制满足条件的用户 51](#_Toc36303709)

[7.2 基于IP限制指定用户 52](#_Toc36303710)

[7.3 基于Session限制满足条件的用户 53](#_Toc36303711)

[7.4 基于Session限制指定用户 54](#_Toc36303712)

[7.5 基于IP添加白名单 55](#_Toc36303713)

[7.6 基于Session添加白名单 56](#_Toc36303714)

[8 日志和告警 57](#_Toc36303715)

[8.1 攻击详情日志 57](#_Toc36303716)

[8.2 系统操作日志 58](#_Toc36303717)

[8.3 日志归档设置 59](#_Toc36303718)

[8.4 日志下载 60](#_Toc36303719)

[8.5 Syslog日志告警 60](#_Toc36303720)

[8.6 邮件告警 61](#_Toc36303721)

[9 系统设置 62](#_Toc36303722)

[9.1 三权分立管理 62](#_Toc36303723)

[9.2 配置备份与还原 63](#_Toc36303724)

[9.3 用户自定义密码 64](#_Toc36303725)

[9.4 基于IP限制管理页面访问 64](#_Toc36303726)

[9.5 Open API接口 65](#_Toc36303727)

[9.6 攻击检测统计 66](#_Toc36303728)

[9.7 用户管理模块 67](#_Toc36303729)

[10 报告功能 68](#_Toc36303730)

[10.1 定时生成报告 68](#_Toc36303731)

[10.2 自定义时间报告生成 68](#_Toc36303732)

[11 系统信息 69](#_Toc36303733)

[11.1 系统运行实时数据 69](#_Toc36303734)

[11.2 系统运行历史数据 70](#_Toc36303735)

# 概述

## 测试目的

按照测试方案要求，对厂商提供的WAF设备进行可行性测试，以明确WAF设备功能的可实现性，为WAF设备的选型提供测试依据。

## 测试要求

测试结果必须提供截图作为辅助证明。

## 评判标准

在测试方案的“测试结果”一栏中填入“通过”、“部分通过”、“未通过”、“未测试”、“无此功能”或者性能指标值。

## 测试说明

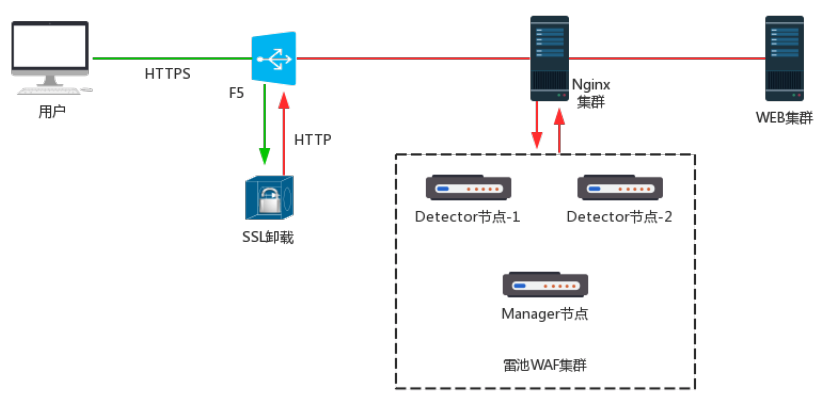
在以下测试用例中，防护站点均采用[www.site.com](http://www.site.com)为例进行说明，测试时需替换为真实的防护站点。

# 环境准备

|  |  |  |
| --- | --- | --- |
| **硬件要求** | | |
|  | 推荐配置 | 最低配置要求 |
| **CPU** | 4核 | 2核 |
| **内存** | 8G | 4G |
| **硬盘容量** | 200G | 100G |

|  |  |
| --- | --- |
| **系统及Docker支持版本** | |
| **系统** | CentOS 7.4 、 Centos 7.5 Ubuntu 16.04 、 Ubuntu 18.04 、  Ubuntu 18.10 |
| **Docker** | 18.03.1-ce 、 18.06.0-ce 、  18.06.1-ce 、 18.09.3 |

# 网络拓扑



# 站点管理

## HTTP站点

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | HTTP站点 | | |
| **测试项目描述** | WAF嵌入式模式在没有添加防护站点时是默认的防护策略，会对高危攻击进行拦截同时也会记录攻击检测日志，添加HTTP站点目的时为了让该站点设置其他的防护策略。 | | |
| **执行步骤** | 1. 在「网站防护」->「防护站点管理」中->「添加站点」； 2. 填写被防护站点的配置; 3. 浏览器访问：http://www.site.com/shell.php查看是否成功添加站点进行防护并记录日志。 | | |
| **预期结果** | 成功添加防护站点进行防护并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 4.2 站点攻击检测策略

### 4.2.1 默认策略

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 默认防护策略修改 | | |
| **测试项目描述** | 对http站点进行防护策略修改，未配置站点前所有的检测策略均为默认策略，配置站点后，可以将检测策略修改为如高防模式，观察模式等其他默认策略。 | | |
| **执行步骤** | 1. 在「网站防护」-> 「防护站点管理」 中选择要修改防护策略的站点；  2. 在「攻击防护策略」中防护策略改为默认存在的五种策略之一，如修改防护策略为观察模式；  3. 浏览器访问：http://www.site.com/shell.php进行测试，查看WAF的攻击日志中的攻击流量是否是放行状态。 | | |
| **预期结果** | 对站点攻击流量的检测为修改的观察防护策略，在攻击检测日志中可看到该防护站点的攻击流量均为放行状态。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 4.2.2 自定义策略

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义策略 | | |
| **测试项目描述** | 添加自定义防护策略，修改其中的检测模块，并在站点中使用自定义的防御策略。 | | |
| **执行步骤** | 1. 在「网站防护」-> 「防护策略管理」中 ->「添加防护策略」；  2. 在站点管理中将想要修改防护策略的站点的防护策略修改为自定义的防护策略。  3. 浏览器访问：http://www.site.com/shell.php查看是否符合自定义的防护策略。 | | |
| **预期结果** | 对站点攻击流量的检测为自定义的防护策略，且流量的拦截与否符合自定义的防护策略。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 5. WAF攻击检测功能

## WEB攻击检测测试

### SQL注入攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | SQL注入攻击检测 | | |
| **测试项目描述** | SQL注入攻击是WEB应用系统最为严重的攻击之一，通过SQL注入可能导致信息泄漏等恶性事件的产生，SQL注入的本质是数据库没有将输入的数据片段与可执行的代码加以区分，导致输入可执行代码对数据库进行有危险的操作。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. URL短接SQL payload进行测试，可参考“/?id"and(select\*from(select sleep(0))a/\*\*/union/\*\*/select 1)”,观察是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### XSS攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | XSS攻击检测 | | |
| **测试项目描述** | XSS攻击同样也是WEB应用系统最为严重的攻击之一，主要利用网站交互和电子邮件等特性发起攻击，常见的有反射型XSS、存储型XSS，其原理是诱导用户打开有危险行为的链接。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php?url=<script>alert(document.cookie)</script> 3. 观察WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### Java反序列化攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | Java反序列化攻击检测 | | |
| **测试项目描述** | 如果Java应用对用户输入，即不可信数据做了反序列化处理，那么攻击者可以通过构造恶意输入，让反序列化产生非预期的对象，非预期对象在产生的过程中就有可能带来任意代码的执行，Java反序列化漏洞也是近些年出现比较严重的漏洞。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，如用浏览器访问如下链接：   http://www.site.com/index.php?\xac\xed\x00\x05sr\x00\x11java.lang.Boolean\xcd%20r\x80\xd5\x9c\xfa\xee\x02\x00\x01Z\x00\x05valuexp\x01   1. 观察WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 文件上传攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 文件上传攻击检测 | | |
| **测试项目描述** | 文件上传是WEB应用系统很常见的一种功能，本身是一项正常的业务需求，但如果在上传时没有对文件进行正确过滤，则可能会发生安全问题，例如一段PHP脚本伪装成图片文件，经解释器处理后变为可执行的脚本，WAF能够对上传的文件内容进行有效的过滤。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 新建“ file.php;jpg ”文件，文件内容可任意修改，通过站点的上传文件入口处上传该文件； 3. 观察WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别文件上传攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 5.1.5 文件包含攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 文件包含攻击检测 | | |
| **测试项目描述** | 文件包含漏洞是一种典型的高危漏洞。由于程序员未对用户可控的变量进行输入检查，导致用户可以控制被包含的文件，成功利用时可以使web server将特定文件当成脚本执行，从而导致用户可获取一定的服务器权限。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php?page=/etc/passwd 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### PHP代码注入攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | PHP代码注入攻击检测 | | |
| **测试项目描述** | PHP注入是泛指其他种种允许攻击者直接对服务器脚本引擎提供代码的注入攻击。在PHP注入实例里，服务端脚本引擎是PHP，实际上PHP注入是“动态赋值安全隐患”、“包含文件注入”等的安全隐患。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php?<?php echo $a; ?> 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### CSRF攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | CSRF攻击检测 | | |
| **测试项目描述** | 跨站请求伪造(Cross-Site Request Forgery，CSRF)是一种使已登录用户在不知情 的情况下执行某种动作的攻击。因为攻击者看不到伪造请求的响应结果，所以CSRF攻击主要 用来执行动作，而非窃取用户数据。当受害者是一个普通用户时，CSRF可以在其不知情的情 况下实现转移资金、发送邮件等操作;但当如果受害者是一个具有管理员权限的用户时， CSRF就可能威胁到整个Web系统的安全。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/add?content=hello&user=helloReferer:%20http://www.qq.com/abc.html 3. 使用BurpSuite拦截请求并修改或添加Referer字段：   Referer: http://www.qq.com/abc.html   1. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### SSRF攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | SSRF攻击检测 | | |
| **测试项目描述** | 服务器端请求伪造(Server-Side Request Forgery，SSRF)允许攻击者使用目标服 务器来代表攻击者自己执行HTTP请求，这和CSRF类似，因为两个漏洞都在不被受害者察觉 的情况下执行了HTTP请求。只不过在SSRF中，受害者是漏洞服务器，而在CSRF中，它是 用户的浏览器。一般情况下，SSRF攻击的对象是从外网无法访问的内部系统。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php?bslotes=dict://127.0.0.1:80 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### PHP反序列化攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | PHP反序列化攻击检测 | | |
| **测试项目描述** | PHP应用中对数据做了反序列化处理，攻击者通过构造恶意输入，让反序列化产生非预期行为，导致攻击者控制反序列化后的对象属性从而实现最终的攻击。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php?a:1:{i:0;O:3:"Foo":1:{s:10:"aMemberVar";s:7:"sdssdsd";}} 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### ASP代码注入攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | ASP代码注入攻击检测 | | |
| **测试项目描述** | ASP注入是泛指其他种种允许攻击者直接对服务器脚本引擎提供代码的注入攻击。在ASP注入实例里，服务端脚本引擎是ASP，实际上ASP注入是“动态赋值安全隐患”、“包含文件注入”等的安全隐患，与PHP代码注入相似。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 新建文本文件，内容填写为：   <%@codepage=65000%>  <%response.codepage=65001:eval(request("xxx"))%>  并修改文件名为file.png，通过站点上传该文件   1. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### JAVA代码注入攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | JAVA代码注入攻击检测 | | |
| **测试项目描述** | Java代码注入是泛指其他种种允许攻击者直接对服务器脚本引擎提供代码的注入攻击。在JAVA代码注入实例里，服务端脚本引擎是Java，不安全的代码编写会导致攻击者获取到敏感信息或权限。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/index.php 3. 并使用BurpSuite添加Content-Type字段：   Content-Type:  %{#context['com.opensymphony.xwork.dispatcher.HttpServletResponse']}Multipart   1. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 命令注入攻击

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 命令注入攻击检测 | | |
| **测试项目描述** | 命令注入是指涉及动态生成内容的一类关键应用程序漏洞。攻击者使用易受攻击的应用程序在主机操作系统上执行任意命令。它们通过表单，cookie和HTTP头将错误数据传递给系统shell。这允许攻击者获得对网站的控制并执行底层应用程序所适应的任何操作或过程。获得对这些系统的访问权限的攻击者可以修改，操纵或读取数据; 注入窃取数据或攻击基础设施的命令; 并参与其他破坏性活动。命令注入高度依赖于特权。它发生在输入验证不足时。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/env x=’() { :;}; echo vulnerable’ bash -c “test” 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 情报模块

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 情报模块检测 | | |
| **测试项目描述** | 情报模块根据请求中是否含有测试文件、备份文件、代码仓库、敏感文件来判断是否进行记录，上述文件的泄漏可以为攻击者提供一些信息来进行针对性的攻击。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/etc/shadow 3. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 服务器响应检测模块

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 服务器响应检测模 | | |
| **测试项目描述** | 当攻击者发送一些具有攻击行为的请求时，服务端会返回一些信息给攻击者，WAF会根据服务器端返回的信息来判断攻击是否有效从而判断是否进行拦截。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/phpinfo.php 3. 并且会返回phpinfo的信息的时候； 4. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 机器人检测模块

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 机器人检测模块 | | |
| **测试项目描述** | 攻击者在进行攻击时，会优先进行信息探测，使用脚本或者扫描器对网站进行大规模的扫描探测行为，机器人检测模块会根据bot检测算法来判断是否是正常的浏览器发出的正常的请求，大量的探测行为会导致网站的大量信息被攻击者熟知并进行针对性攻击，会暴露更多的攻击面。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在客户端构造请求，比如用浏览器访问如下链接：http://www.site.com/ 3. 使用BurpSuite修改User-Agent字段为 sqlmap； 4. 查看WAF是否对攻击进行拦截并记录攻击日志。 | | |
| **预期结果** | WAF有效识别攻击，拦截攻击流量并记录攻击日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 常用编码解码测试

### BASE64解码

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | BASE64解码 | | |
| **测试项目描述** | 构造base 64编码后的攻击请求发送至WAF，WAF应成功解码并产生攻击告警，攻击的payload被编码后，会加大WAF的防护压力。 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行base 64编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/?value=MSUyMG9yJTIwMSUyMD0lMjAx 3. 查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别编码攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### URL解码

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | URL Encoding解码 | | |
| **测试项目描述** | 构造URL编码后的攻击请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行URL编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/?value=1%2520or%25201%2520%3D%25201 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别编码攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### JSON 解析

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | HTML Entities解码 | | |
| **测试项目描述** | 构造HTML Entities编码后的攻击请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行HTML Entities编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/show.php?id={"id":"31%20and%201=1"} 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别编码攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 十六进制转换

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | Hex Encoding解码 | | |
| **测试项目描述** | 构造HEX编码后的攻击请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行Hex编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/?value=0x3c696d67207372633d78206f6e6572726f723d616c65727428646f63756d656e742e636f6f6b6965293e 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别编码攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### 5.2.5 斜杠反转义

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | Unicode Escape解码 | | |
| **测试项目描述** | 构造Unicode Escape编码后的攻击请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行Unicode编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/?value=%u0073%u0065%u006c%u0065%u0063%u0074+1+from+t%23 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别编码攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### XML解析

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 混合编码 | | |
| **测试项目描述** | 构造URL多层编码后的请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行混合编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/?value= JTI2JTIzeDNDJTNCYSUyMGhyZWYlM0QlMjYlMjN4MjclM0IlMjAlNUN0JTIwamF2YXNjcmlwdCUzQWFsZXJ0JTI4MSUyOSUyMCUyNiUyM3gyNyUzQiUyMCUyZiUyNiUyM3gzRSUzQg== 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别多层编码的攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### PHP反序列化对象解析

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 混合编码 | | |
| **测试项目描述** | 构造URL多层编码后的请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行混合编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/index.php?answer= O:7:"chybeta":1:{s:4:"test";O:7:"ph0en2x":1:{s:5:"test2";s:10:"phpinfo();";}} 3. 观察请求是否被记录，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别多层编码的攻击请求，拦截攻击流量并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

### UTF-7解码

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 混合编码 | | |
| **测试项目描述** | 构造URL多层编码后的请求发送至WAF，WAF应成功解码并产生攻击告警 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 对攻击payload进行混合编码（编码时可使用火狐的hackbar插件或者其他网络工具），然后在对应的测试站点后直接添加“/”和被编码的字符串，比如用浏览器访问如下链接：http://www.site.com/index.php?aswer=%2B%2Fv8%20%2BADw-SCRIPT%2BAD4-alert(%27XSS%27)%3B%2BADw-%2FSCRIPT%2BAD4- 3. 观察请求是否被拦截，查看WAF是否对攻击进行拦截并记录攻击日志并观察payload是否被解码为编码之前的状态。 | | |
| **预期结果** | WAF成功识别多层编码的攻击请求，拦截并记录日志 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 自定义规则

## URL匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义URL匹配规则 | | |
| **测试项目描述** | 添加匹配URL规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点url中包含wp-login时进行拦截； 3. 浏览器访问：http://www.site.com/wp-login/adminMangage.php; 4. 查看WAF是否对该流量进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 路径匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义路径匹配规则 | | |
| **测试项目描述** | 添加匹配路径规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点解码后路径中包含adminLogin时进行拦截； 3. 浏览器访问：http://www.site.com/adminLogin.php; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Query匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Query匹配规则 | | |
| **测试项目描述** | 添加匹配Query规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的Query参数中包含or 1232=1321时进行拦截； 3. 浏览器访问：http://www.site.com/search?tid=1 or 1232=1321; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 请求方法匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义请求方法匹配规则 | | |
| **测试项目描述** | 添加匹配请求方法规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点时请求方式为GET时进行拦截； 3. 浏览器访问：http://www.site.com/shario/thinkPHPLogin; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Host匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Host匹配规则 | | |
| **测试项目描述** | 添加匹配Host规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点Host字段中包含site.com时进行拦截； 3. 浏览器访问：http://www.site.com/; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Cookie匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Cookie匹配规则 | | |
| **测试项目描述** | 添加匹配Cookie规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的Cookie字段中包含fs2tk6o6m4sre1gopbh1qasnl5时进行拦截； 3. 浏览器访问：http://www.site.com/adminLogin.php或curl http://www.site.com/apache2/conf.d -H “Cookie: PHPSESSID= s2tk6o6m4sre1gopbh1qasnl5” 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## User-Agent匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义User-Agent匹配规则 | | |
| **测试项目描述** | 添加匹配User-Agent规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点User Agent字段中包含sqlmap时进行拦截； 3. curl -vv http://www.site.com/UA-Content.php -H “User-Agent: sqlmap”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Referrer匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Referrer匹配规则 | | |
| **测试项目描述** | 添加匹配Referrer规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点Referrer字段中包含 / 时进行拦截； 3. curl -vv http://www.site.com/ReferrerText.php -H “Referer: http://www.qq.com/abc.html”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Content-Type匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Content-Type匹配规则 | | |
| **测试项目描述** | 添加匹配Content-Type规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点Content-Type字段中包含Multipart时进行拦截； 3. curl -vv http://www.site.com/index.php -H “Content-Type: %{#context['com.opensymphony.xwork.dispatcher.HttpServletResponse']}Multipart”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## IP匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义IP匹配规则 | | |
| **测试项目描述** | 添加匹配IP规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点时的ip为172.16.160.1 时进行拦截； 3. 让ip为172.16.160.1机器访问一下防护站点即可； 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。  除了可以匹配特定IP，还支持阻断来自特定地域的请求，国内精确到省，国外精确到国家 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## X-Forwarded-For匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义X-Forwarded-For匹配规则 | | |
| **测试项目描述** | 添加匹配X-Forwarded-For规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点X-Forwarded-For字段中包含40.73.32.238 时进行拦截； 3. curl -vv http://www.site.com/XFF.html -H “X-Forwarded-For: 40.73.32.238”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Origin匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Origin匹配规则 | | |
| **测试项目描述** | 添加匹配Origin规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点Origin字段中不包含www.site.com 时进行拦截； 3. curl -vv http://www.site.com/ -H “Origin: www.chaitin.com”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Session匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Session匹配规则 | | |
| **测试项目描述** | 添加匹配Session规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览站点的Session字段中完全匹配fs2tk6i6m4sre1gopbh1qasnl5时进行拦截； 3. 在「网站防护」->「防护站点管理」中选择防护站点，并且在「用户识别」中添加用户识别位置，从Cookie字段的PHPSESSION变量中提取用户的ID； 4. curl -vv http://www.site.com/dvwa/css.main.css -H “Cookie: PHPSESSID= fs2tk6i6m4sre1gopbh1qasnl5” 5. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Header匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Header匹配规则 | | |
| **测试项目描述** | 添加匹配Header规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点HTTP header中包含Upgrade-Insecure-Requests:1时进行拦截； 3. curl -vv http://www.site.com/apache2/session.html -H “Upgrade-Insecure-Requests: 1”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 请求头长度匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义请求头长度匹配规则 | | |
| **测试项目描述** | 添加匹配请求头长度规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的HTTP请求头长度小于200 时进行拦截； 3. curl -vv http://www.site.com/body.html -H “Content-Length: 17”; 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Body匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义Body匹配规则 | | |
| **测试项目描述** | 添加匹配Body规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点完整的Body字段中包含body=siljxijslnoa 时进行拦截； 3. curl http://www.site.com/body.html -d "body=siljxijslnoa" 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## POST参数匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义POST参数匹配规则 | | |
| **测试项目描述** | 添加匹配POST参数规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的POST参数中包含level=low时进行拦截； 3. curl http://www.site.com/body.html -d "level=1" 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 上传文件名匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义上传文件名匹配规则 | | |
| **测试项目描述** | 添加匹配上传文件名规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的上传文件名中包含php字段时进行拦截； 3. 浏览器上传后缀名包含php的文件； 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## HTTP状态码匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义HTTP状态码匹配规则 | | |
| **测试项目描述** | 添加匹配HTTP状态码规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点的HTTP状态码包含404 时进行拦截； 3. 浏览器访问一个不存在的页面，如http://www.site.com/shario/thinkPHPLogin 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 响应内容匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义响应内容匹配规则 | | |
| **测试项目描述** | 添加匹配响应内容规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点，响应内容中包含Not Found 时进行拦截； 3. 浏览器访问一个可以返回Not Found内容的页面 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Request匹配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义站点匹配规则 | | |
| **测试项目描述** | 添加匹配站点规则来检测流量是否匹配并进行日志记录 | | |
| **执行步骤** | 1. 将WEB流量牵引至有加载WAF nginx so模块的Nginx节点； 2. 在「网站防护」->「自定义规则」->「全局自定义规则」中->「添加规则」，当用户浏览防护站点，HTTP请求头中有字段Pragma并且值为no-cache时进行拦截； 3. curl http://www.site.com/body.html -H "Pragma=no-cache" 4. 查看WAF是否对攻击进行拦截并记录匹配该条自定义规则的流量日志。 | | |
| **预期结果** | WAF成功匹配自定义规则，拦截流量并记录日志。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 访问频率控制

## 基于IP限制满足条件的用户

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于IP限制满足条件的用户 | | |
| **测试项目描述** | WAF能够设置触发条件来限制满足条件的IP对防护网站的访问频率或者是禁止访问，当IP对防护网站的访问满足了触发条件时，WAF会对触发限制条件的IP进行限制，返回403页面，根据设置的访问频率规则来决定什么时候对触发规则的IP进行放行。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「访问频率限制规则」中->「添加规则」；  2. 配置访问频率限制规则；  3. 多次访问防护站点以便触发访问频率限制规则 | | |
| **预期结果** | 当用户触发访问频率规则时进行封禁，成功返回403页面。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于IP限制指定用户

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于IP限制指定用户 | | |
| **测试项目描述** | WAF能够通过指定IP即黑名单的方式来限制指定用户访问防护站点。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「访问频率限制规则」中->「添加规则」；  2. 配置访问频率限制规则；  3. 多次访问防护站点以便触发访问频率限制规则 | | |
| **预期结果** | 指定IP用户访问防护站点，返回403页面。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于Session限制满足条件的用户

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于Session限制满足条件的用户 | | |
| **测试项目描述** | WAF通过设置触发限制条件，当用户满足限制条件，通过匹配SESSION值来对用户进行限制行为。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「访问频率限制规则」中->「添加规则」；  2. 配置访问频率限制规则；  3. 同时在「网站防护」->「防护站点管理」->「站点防护方式」中设置->「用户识别」；  4. 设置防护站点的用户识别方式，Safeline用户系统、从请求中提取用户ID两种方式来获取用户的SESSION值；  5. 多次访问防护站点以便触发访问频率限制规则 | | |
| **预期结果** | 满足限制条件的用户访问防护站点时页面返回403。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于Session限制指定用户

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于Session限制指定用户 | | |
| **测试项目描述** | 基于Session限制指定用户即为设置Session黑名单，当Session值匹配到黑名单中时该用户访问防护站点会显示403页面。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「访问频率限制规则」中->「添加规则」；  2. 配置访问频率限制规则；  3. 同时在「网站防护」->「防护站点管理」->「站点防护方式」中->「用户识别」；  4. 设置防护站点的用户识别方式，Safeline用户系统、从请求中提取用户ID两种方式来获取用户的SESSION值；  5. 多次访问防护站点以便触发访问频率限制规则 | | |
| **预期结果** | 当用户Session值匹配到黑名单时访问防护站点返回页面403。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于IP添加白名单

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于IP添加白名单 | | |
| **测试项目描述** | 基于IP白名单是与访问频率控制的黑名单相对应，当设置了访问频率控制黑名单后，触发限制条件的用户会禁止访问防护站点，在这之后当添加了IP白名单后，此时若是该IP还在黑名单中的话，依旧是拒绝访问，只有将该IP从黑名单中去除后，当该IP触发限制条件后才不会再次进入黑名单中，才能够继续访问；当该IP的请求中含有攻击流量时，依旧会被waf进行拦截，想要对该IP的攻击流量也放行的话需要在自定义规则里面进行设置。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「不限制这些用户」中->「添加一个不受限制的用户」；  2. 配置IP白名单；  3. 多次访问防护站点以便测试是否封禁 | | |
| **预期结果** | 访问能够触发限制条件的请求，该IP能正常访问，当请求中有攻击请求时依旧会被WAF进行拦截。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于Session添加白名单

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 基于Session添加白名单 | | |
| **测试项目描述** | 与基于IP添加白名单相应的即是以Session为匹配条件的白名单，当用户的Session进行匹配时，不会触发黑名单中的规则，有一点需要强调的是只有当黑名单是以Session为受限制目标的类型时白名单就会生效，若是黑名单时IP，白名单时Session时WAF依旧会进行拦截，因此只有当黑白名单的受限制目标的类型相同时才会生效。 | | |
| **执行步骤** | 1. 在「网站防护」->「访问频率控制」->「不限制这些用户」中->「添加一个不受限制的用户」；  2. 添加不受限制的Session白名单  3. 多次访问防护站点以便触发访问频率限制规则 | | |
| **预期结果** | Session值匹配到白名单的用户不会触发黑名单中的规则从而被限制访问 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 日志和告警

## 攻击详情日志

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 详细告警信息 | | |
| **测试项目描述** | WAF设备应该不仅能识别并拦截攻击，而且还能够记录详细的日志，系统提供的告警信息应该可以包括分析攻击事件的所有相关信息，并提供足够的相关知识参考和快速策略编辑的便利性。 | | |
| **执行步骤** | 1. 添加防护站点，绑定至后端真实server； 2. 将WEB流量牵引至WAF检测节点； 3. 发起任意一条攻击测试； 4. 访问被拦截后，查看入侵检测日志是否记录日志。 | | |
| **预期结果** | 告警信息应该具备以下内容：  详细的攻击信息，包括攻击检测引擎名称、源地址、时间、风险等级、执行动作、攻击类型、请求域名、源IP、地理位置、触发原因、payload、cookie、user-agent、原始请求头。并且与下载之后的日志相同。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 系统操作日志

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 详细系统操作日志 | | |
| **测试项目描述** | WAF系统操作日志中记录了WAF用户所做的系统操作以及WAF启动情况，当WAF启动时会记录组件状况；当用户在WAF上做一些配置等操作时，会记录下操作日志。 | | |
| **执行步骤** | 1. 删除一条攻击检测日志 | | |
| **预期结果** | 在系统操作日志中会看到哪个作用做了删除攻击检测日志的操作。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 日志归档设置

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 日志归档测试 | | |
| **测试项目描述** | 日志在经过一段时间会因为攻击流量的积累会变的庞大，会导致日志查询变慢，所以会设置一个日期来使得日志进行归档，减少日志查询规模， | | |
| **执行步骤** | 1. 在「日志管理」->「日志归档管理」中；  2. 调整归档方式，可以自定义归档周期；  3. 可以将已归档的日志进行恢复，以便在攻击日志中再次查找到攻击详情。 | | |
| **预期结果** |  | | |
| **截图证明** | 可以设置归档方式 | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 日志下载

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 日志下载功能测试 | | |
| **测试项目描述** | WAF有多种日志，攻击检测日志、频率控制日志、扩展插件日志、系统操作日志，提供下载到本地便于查看整理及转发。 | | |
| **执行步骤** | 1. 在「日志管理」->「攻击检测日志」中；  2. 勾选想要下载的攻击流量日志，更多操作中选择下载；  3. 在「日志管理」->「日志下载管理」中；  4. 选择刚才创建的任务点击下载即可下载流量详情到本地。 | | |
| **预期结果** | 下载日志到本地。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Syslog日志告警

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | Syslog日志告警 | | |
| **测试项目描述** | WAF可以将产生的日志以Json格式发送到Syslog服务器，便于Syslog服务器集中管理各类设备日志。 | | |
| **执行步骤** | 1. 在「系统设置」->「告警收信配置」->「告警方式配置」中；  2. 添加新的告警服务，填写Syslog服务器，选择要发送的日志类型。 | | |
| **预期结果** | WAF发送Json格式日志到Syslog服务器。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 邮件告警

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 邮件告警 | | |
| **测试项目描述** | WAF配置邮件告警方式，当WAF配置发生变化时便会发送邮件告警到指定邮箱。 | | |
| **执行步骤** | 1. 在「系统设置」->「告警收信配置」->「邮件发信配置」中；  2. 配置SMTP服务；  3. 同时在「系统设置」->「告警收信配置」->「告警方式配置」中；  4. 添加新的告警服务，填写接收邮箱，选择要发送的日志类型。 | | |
| **预期结果** | 当WAF系统配置发送变化时，便会发送邮件告警到指定收信邮箱。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 系统设置

## 三权分立管理

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 三权分立管理 | | |
| **测试项目描述** | 三权分立即系统管理员、安全管理员、审计管理员，三者互不干预，便于分权管理。分权管理，避免操作不当导致危险。 | | |
| **执行步骤** | 1. 设置多个不同身份的账号，系统管理员、安全管理员与审计管理员； 2. 分别登录不同身份的账号进行验证，比较操作权限的不同。 | | |
| **预期结果** | 可以成功设置操作权限不同的用户。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 配置备份与还原

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 配置备份与还原 | | |
| **测试项目描述** | 对WAF的配置进行备份，以便日后还原；将备份文件上传WAF即可还原配置。 | | |
| **执行步骤** | 1. 在「系统设置」->「配置备份还原」中；  2. 创建一个新的备份，刷新即可点击下载备份文件；  3. 将备份文件拖至文件还原处即可还原配置。 | | |
| **预期结果** | 可以对WAF配置进行备份；将备份文件拖至文件上传处即可还原配置。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 用户自定义密码

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 用户自定义密码 | | |
| **测试项目描述** | WAF在创建用户的时候会自动生成一个随机密码，创建用户的时候是无法自定义密码，只有创建完成用户之后才能自定义修改密码。 | | |
| **执行步骤** | 1. 登陆想要修改密码的用户，到个人中心，选择使用偏好； 2. 修改认证方式处便可以自定义修改密码或者修改认证方式。 | | |
| **预期结果** | 可以自定义修改用户密码。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 基于IP限制管理页面访问

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 设置IP白名单限制管理页面的访问 | | |
| **测试项目描述** | 因为安全原因，管理页面要做白名单限制，只有在白名单中的IP才能访问管理页面。 | | |
| **执行步骤** | 1. 在「系统设置」->「系统用户」中；  2. 选择要做IP限制访问的用户，进入用户设置，修改访问来源限制；  3. 添加IP白名单；  4. 使用未再白名单中的IP来使用上面的用户进行登录。 | | |
| **预期结果** | 在白名单中的IP才能访问WAF管理页面。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## Open API接口

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | Open API | | |
| **测试项目描述** | WAF提供部分功能开放的API，通过API可以帮助管理员进行自动化操作，使其可以与已有的数据系统集成，以方便管理、统计和数据收集。 | | |
| **执行步骤** | 1. 在「个人中心」->「OPENAPI」中；  2. 填写配置，设置API接口权限；  3. 生成token之后便可以在脚本中携带token值请求WAF的API接口实现配置。 | | |
| **预期结果** | WAF管理界面能够生成策略，并且策略可以生效 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 攻击检测统计

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 攻击检测统计 | | |
| **测试项目描述** | WAF提供攻击检测统计，可以从攻击者和攻击目标两个维度进行溯源和统计分析。 | | |
| **执行步骤** | 1. 在「系统设置」->「其他系统设置」中；  2. 编辑攻击检测统计，选择开启；  3. 浏览器访问：<http://www.test.com/?id=1%20and%201=2>  4. 在「统计信息」->「攻击检测统计」->「攻击检测溯源」中，有预期结果1  5. 在「统计信息」->「攻击检测统计」->「攻击目标分析」中，有预期结果2 | | |
| **预期结果** | 预期结果1：可以看到当前时间一分钟内攻击者的源地址，1小时内攻击次数，最后攻击时间。  预期结果2： 可以看到攻击目标的地址，1小时内被攻击次数，最后被攻击时间。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 用户管理模块

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 用户管理模块 | | |
| **测试项目描述** | 为了满足等保需求，WAF提供用户管理模块，可以对密码复杂度、登陆限制进行配置。 | | |
| **执行步骤** | 1. 在「系统设置」->「系统用户配置」中；  2. 编辑密码配置，设置最小密码长度为8；  3. 在「个人中心」->「账户安全」->「认证方式」中，点击修改密码，输入新密码123456，有预期结果1；  5. 在「系统设置」->「系统用户配置」中，编辑登陆限制，在“登陆错误限制方式”中输入5，在登陆错误限制时长中输入30，有预期结果2 | | |
| **预期结果** | 预期结果1：可以看到提示密码至少包含8个字符。  预期结果2： 连续输入5次错误密码后，提示“操作过于频繁，请等待30分钟”。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 10 报告功能

## 10.1 定时生成报告

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 定时生成报告 | | |
| **测试项目描述** | WAF会定时生成日报、周报、月报，以供管理人员对每日、每周、每月防护站点所受到的攻击有整体的报告 | | |
| **执行步骤** | 1. 进入「统计信息」->「防护报告导出」->「防护报告」。 | | |
| **预期结果** | 每日、每周、每月都会定时生成日报、周报、月报。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 自定义时间报告生成

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 自定义时间报告生成 | | |
| **测试项目描述** | 由于不同的需求，可能会自定义时间段来生成报告。 | | |
| **执行步骤** | 1. 进入「统计信息」->「防护报告导出」->「防护报告」；  2. 选择手动生成一个报告，并选择时间段。 | | |
| **预期结果** | 可以下载自定义时间段的报告。 | | |
| **截图证明** |  | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

# 系统信息

## 系统运行实时数据

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 系统运行实时数据查询 | | |
| **测试项目描述** | 查询WAF系统的运行实时数据信息 | | |
| **执行步骤** | 1. 进入「系统信息」->「节点状态」；  2. 选择负载状态，有预期结果1；  3. 选择网络状态，有预期结果2；  4. 选择检测状态，有预期结果3；  5. 选择转发状态，有预期结果4；  6. 选择磁盘状态，有预期结果5； | | |
| **预期结果** | 预期结果1：可以看到当前时间一分钟内的CPU使用率和内存使用率；  预期结果2：可以看到当前时间一分钟内的网络读写速率；  预期结果3：可以看到当前时间一分钟内检测引擎的每秒检测请求数和平均检测耗时；  预期结果4：可以看到当前时间一分钟内每秒新建连接数和平均处理延迟；  预期结果5：可以看到当前时间一分钟内磁盘读写速率； | | |
| **截图证明** | CPU使用率和内存使用率    网络读写速率    每秒检测请求数和平均检测耗时    每秒新建连接数和平均处理延迟    磁盘读写速率 | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |

## 系统运行历史数据

|  |  |  |  |
| --- | --- | --- | --- |
| **测试项目** | 系统运行历史数据 | | |
| **测试项目描述** | 由于不同的需求，可能会查询系统运行的历史数据。 | | |
| **执行步骤** | 1. 进入「系统信息」->「节点状态」->「历史数据查询」；  2. 编辑查询信息，选择需要查询的节点，选择时间段，如3月2号到3月3号，点击「查询」； | | |
| **预期结果** | 可以查询到所选节点在所选时间段内的CPU使用率、内存使用率、磁盘使用率、读网络、写网络、每秒新建连接数、每秒检测请求数的信息。 | | |
| **截图证明** | 历史CPU使用率和内存使用率    历史网络读写速率和每秒新建连接数、每秒检测请求数 | | |
| **判定结果** | ☑通过 □部分通过 □未通过 □未测试 □无此功能 | | |
| **测试人员** |  | **日期** |  |