

## 廖翰晖 / 应用安全负责人 / 豌豆思维



从高中在乌云上挖漏洞进入到网络安全行业。在大学中接触了各类CTF比赛积累网络安全经验。到工作以后接触了红蓝对抗应用安全建设等一系列工作。对应用安全体系化建设和红蓝对抗有丰富的兴趣。同时把对互联网应用安全建设的一部分经验转化出来分享给大家，增强自己的知识面。目前在一家大型互联网企业做应用安全负责人，主要负责devsecops化体系的建设。与内部应用安全运营相关的事宜。

演讲主题：甲方互联网RASP应用实践

**安世加**



# 甲方互联网RASP应用实践

## 广州站（城市沙龙）

12月10日 / 周五下午

**安世加**



# RASP是什么？

## RASP 背景介绍

Gartner 在2014年应用安全报告里将 RASP 列为应用安全领域的关键趋势，原文引用如下

『Applications should not be delegating most of their runtime protection to the external devices. Applications should be capable of self-protection (i.e., have protection features built into the application runtime environment)』

即 "应用程序不应该依赖外部组件进行运行时保护，而应该具备自我保护的能力，也即建立应用运行时环境保护机制"

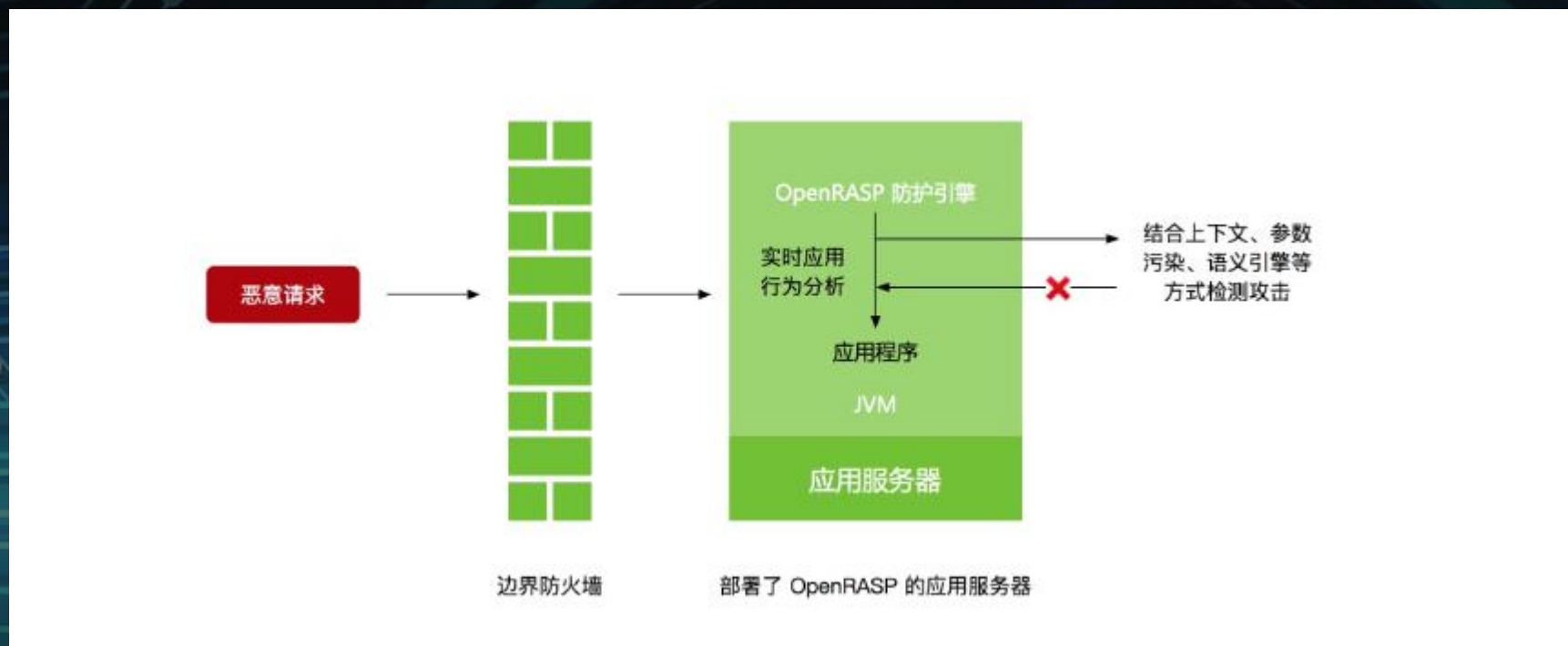
那么，为什么应用程序需要进行自我保护呢？

举个例子，当应用受到 struts 漏洞攻击，诸如WAF、IDS这样的外部防护设备，只能知道这个请求包含攻击特征，应该拦截；而应用程序却知道，自己运行了一段OGNL代码，然后莫名其妙的运行了系统命令。

当然，这是针对已知漏洞的讨论。如果是新型漏洞，应用则更加需要进行自我保护。

新的漏洞通常意味着新的请求格式、新的请求参数，外部防护设备需要及时增添规则，才能够进行防护；而应用程序则可以根据自身行为进行保护，通常不依赖于规则

# RASP如何在应用当中实现防护？



以java举例，RASP借助JVM提供的instrumentation技术，以Javaagent形式部署在java应用当中。agent对关键类和关键方法进行HOOK以增加针对性的防护算法，以防护攻击者的攻击行为。



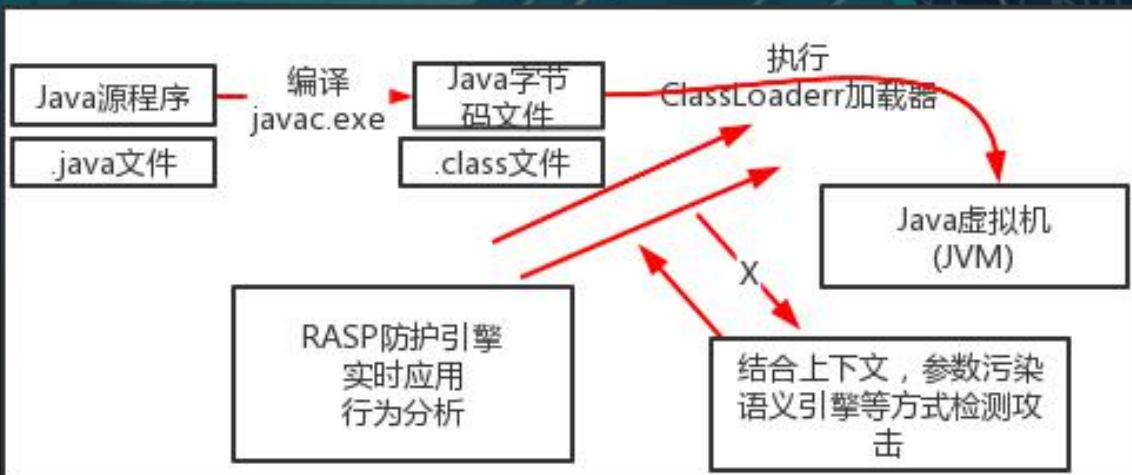
# RASP与WAF之间对比



部署WAF，实时分析流量。通过规则检测攻击

应用内部

WAF掌握的是应用的“入口流量”



RASP掌握的是应用内部所有动作的“上下文联系”

安世加

# RASP在互联网企业中的应用

10 结果, 显示 1 / 1 页

请求时间	URL	请求来源	拦截状态	攻击类型	报警消息	操作
2021-11-24 11:30:00	http://[redacted]et:88/uapws/rest/uapbd/getsupplier?pageindex=0&pagesize=200&code=KLZZHYL&org ...	106.15.12 1.223	记录日志	HTTP 响应采样检测	PII leak detected: 6212998605072095(Bank Card)	<a href="#">查看详情</a>
2021-11-09 14:49:06	http://[redacted]/ServiceDispatcherServlet/default	36.112.94. 82	记录日志	SQL 语句异常	oracle error 904 detected: ORA-00904: "ORG_ORGS"."ORG_ORGS": invalid identifier	<a href="#">查看详情</a>
2021-11-04 21:19:48	http://[redacted]/t00ms.jsp	120.242.18 0.128	拦截请求	命令执行	Command execution - Logging all command execution by default, command is /bin/sh -c cd "/product/nc65";ps;echo 32bcd66;pwd;echo a7586	<a href="#">查看详情</a>
2021-11-04 21:19:46	http://[redacted]/t00ms.jsp	120.242.18 0.128	拦截请求	命令执行	Command execution - Logging all command execution by default, command is /bin/sh -c cd "/product/nc65";ps;echo 32bcd66;pwd;echo a7586	<a href="#">查看详情</a>
2021-11-01 21:48:37	http://[redacted]service/~baseapp/UploadServlet	61.148.74. 134	拦截请求	反序列化攻击	Deserialization blacklist - blocked org.apache.commons.collections.functors.InvokerTransformer in resolveClass	<a href="#">查看详情</a>
2021-11-01 21:48:36	http://[redacted]service/~baseapp/UploadServlet	61.148.74. 134	拦截请求	反序列化攻击	Deserialization blacklist - blocked org.apache.commons.collections.functors.InvokerTransformer in resolveClass	<a href="#">查看详情</a>
2021-11-01 21:06:46	http://[redacted]/fs/dcupdateService/files	61.148.74. 134	拦截请求	反序列化攻击	Deserialization blacklist - blocked org.apache.commons.collections.functors.InvokerTransformer in resolveClass	<a href="#">查看详情</a>
2021-11-01 21:06:45	http://[redacted]/fs/dcupdateService/files	61.148.74. 134	拦截请求	反序列化攻击	Deserialization blacklist - blocked org.apache.commons.collections.functors.InvokerTransformer in resolveClass	<a href="#">查看详情</a>
2021-10-28 07:00:00	http://[redacted]apidemo/shqt/updpsndoc	81.71.98.1 35	记录日志	HTTP 响应采样检测	PII leak detected: 17621846810(Mobile Number)	<a href="#">查看详情</a>

RASP防护技术部署在互联网企业当中易忽视不易管理的供应链环境。

提升供应链系统的安全防护性。企业安全运营体系。

安世加



# RASP防护效果

报警详情

漏洞详情 请求信息 资产信息 修复建议

请求编号

127a74fd2dfd477eb0f280f9c82f1003

请求 URL

POST http://.jsp

请求来源

120.242.180.128 中国-

完整 Header 信息

accept-encoding: gzip, deflate

connection: close

content-length: 8929

content-type: application/x-www-form-urlencoded

host: 120.79.249.6:88

user-agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.137 Saf

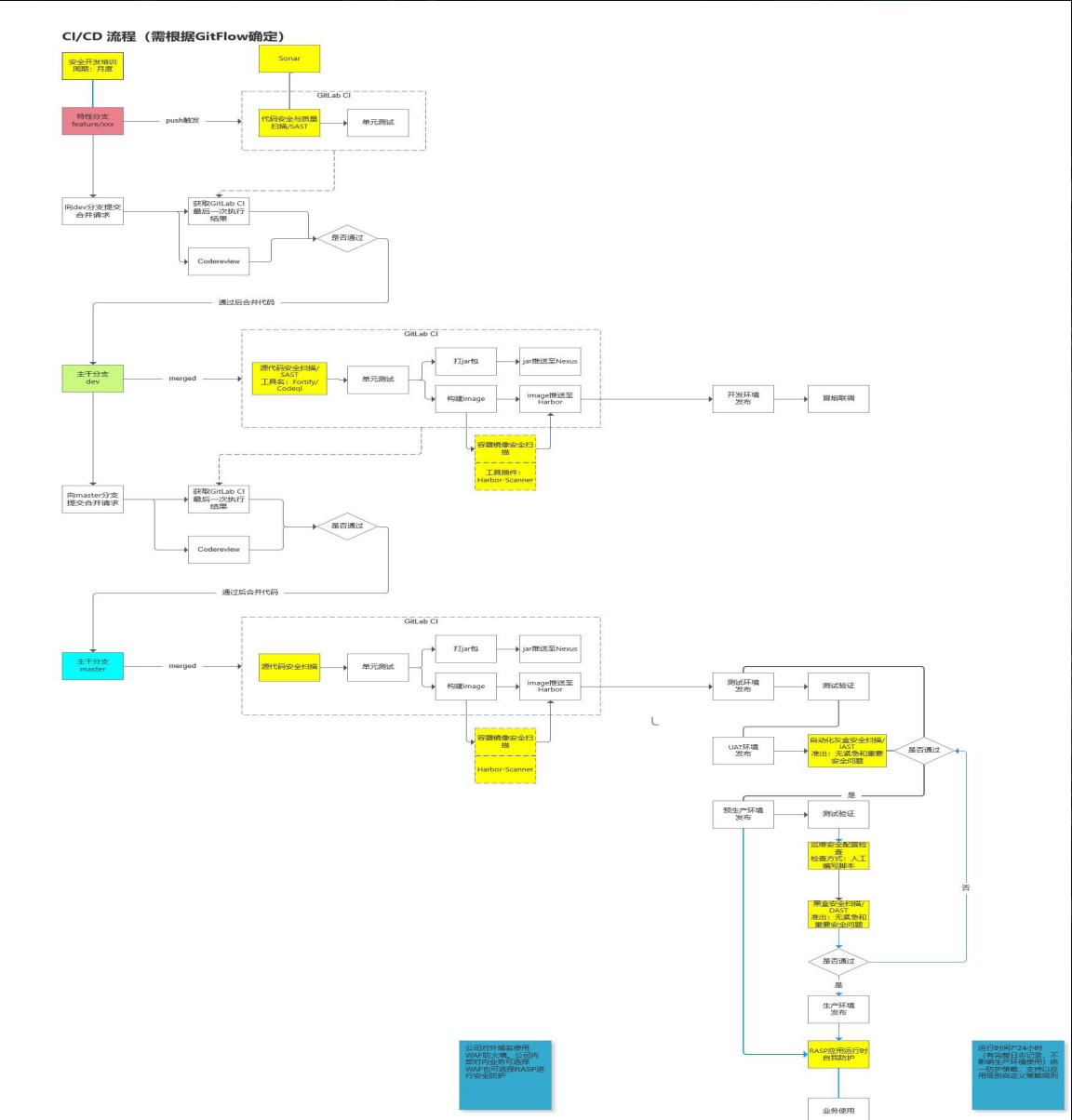
Form 参数

t=yv66vgAAADEBgwoAZgCiCQCCAKMJAIIPApAgApQkAggCmCACnQCCAKgIAKkJAIIPAggCrBwCsCgALAKIIAK0IAK4IAK8IALAIALEIA

供应链系统被上传webshell，攻击者利用webshell执行命令被RASP防护引擎实时拦截并推送相关告警

安世加

# 如何在企业开发过程当中融入RASP



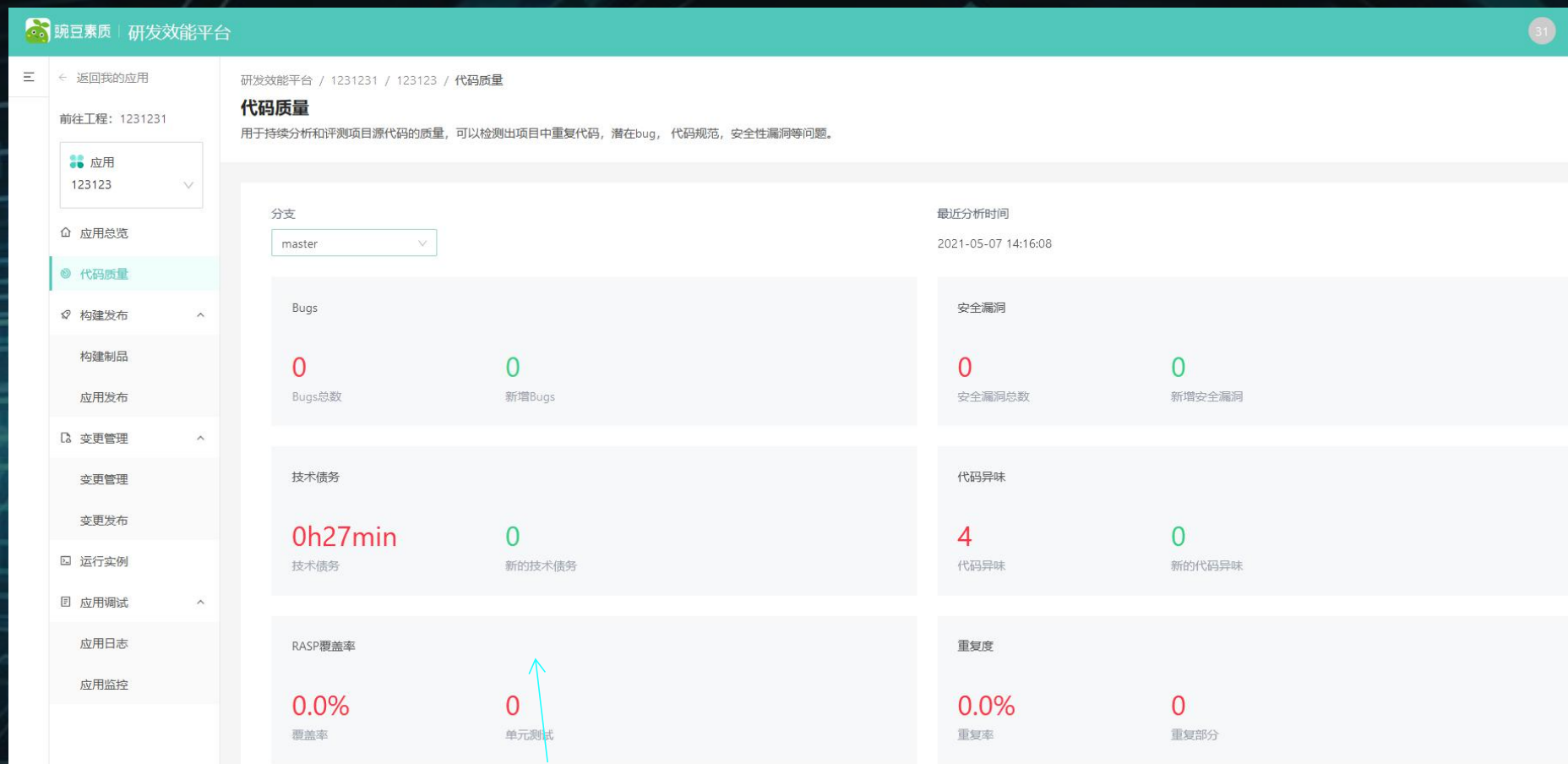
日常开发过程当中使用的devsecops流程架构图。

对核心应用上线时部署RASP进行安全防护

安世加



# 在devops快速迭代中自动化融入RASP



对业务快速频繁发版过程中把RASP接入到DevOps平台作为安全能力赋能。  
增强应用安全的防护性。

# RASP的一些玩法

在云控端后台设置相关防护算法。  
可实时下发防护算法到指定应用程序中实现**动态防护**

报警设置

黑白名单

防护设置

应用加固

登录认证

应用管理

后台设置

快速设置

☐

将所有算法设置为「记录日志」模式（"XXE 禁止外部实体加载" 算法除外）

☐

启动「研发模式」，开启一些消耗性能的检测算法

☐

打印「行为日志」，仅用于调试，请勿在线上开启 [帮助文档](#)

命令执行

拦截攻击

记录日志

完全忽略

算法1 - 通过反射执行命令，比如反序列化、加密后门

拦截攻击

记录日志

完全忽略

算法2 - 用户输入匹配算法，包括命令注入检测

拦截攻击

记录日志

完全忽略

算法3 - 识别常用渗透命令（探针） [\[高级选项\]](#)

拦截攻击

记录日志

完全忽略

算法4 - 查找语法错误和敏感操作 [\[高级选项\]](#)

拦截攻击

记录日志

完全忽略

算法5 - 记录或者拦截所有命令执行操作

拦截攻击

记录日志

完全忽略

算法6 - dnslog类命令

任意文件删除

拦截攻击

记录日志

完全忽略

算法1 - 用户输入匹配，禁止使用 ../ 删除文件

反序列化攻击

拦截攻击

记录日志

完全忽略

算法1 - 反序列化黑名单过滤 [\[高级选项\]](#)

目录遍历

拦截攻击

记录日志

完全忽略

算法1 - 用户输入匹配算法

拦截攻击

记录日志

完全忽略

算法2 - 通过反射调用，查看目录内容

拦截攻击

记录日志

完全忽略

算法3 - 尝试查看敏感目录

OGNL 代码执行

拦截攻击

记录日志

完全忽略

算法1 - OGNL语句黑名单 [\[高级选项\]](#)

任意文件读取

拦截攻击

记录日志

完全忽略

算法1 - 用户输入匹配算法

拦截攻击

记录日志

完全忽略

算法2 - 用户输入匹配算法 + http 协议

拦截攻击

记录日志

完全忽略

算法3 - 拦截 php:// 等异常协议

拦截攻击

记录日志

完全忽略

算法4 - 禁止使用 ../ 访问web目录以外的文件 [\[帮助文档\]](#)

拦截攻击

记录日志

完全忽略

算法5 - 文件探针算法

文件重命名

拦截攻击

记录日志

完全忽略

算法1 - 通过重命名方式获取 WebShell

HTTP 响应采样检测 (该检测点不支持阻断，拦截攻击等同于记录日志)

拦截攻击

记录日志

完全忽略

算法1 - 检查响应里是否有敏感信息 [\[高级选项\]](#)

SQL 注入

拦截攻击

记录日志

完全忽略

算法1 - 用户输入匹配算法 [\[高级选项\]](#)

拦截攻击

记录日志

完全忽略

算法2 - 拦截异常SQL语句 [\[高级选项\]](#)

拦截攻击

记录日志

完全忽略

算法3 - 记录数据库异常 [\[帮助文档\]](#) [\[高级选项\]](#)

拦截攻击

记录日志

完全忽略

算法4 - 正则表达式算法 [\[高级选项\]](#)

SSRF 请求伪造

拦截攻击

记录日志

完全忽略

算法1 - 用户输入匹配算法（支持 rebind 检测）

拦截攻击

记录日志

完全忽略

算法2 - 拦截 AWS/Aliyun/GCP metadata 访问

拦截攻击

记录日志

完全忽略

算法3 - 拦截常见 dnslog 地址



# RASP的展望

```
if (algorithmConfig.deserialization_blacklist.action != 'ignore')
{
    plugin.register('deserialization', function (params, context) {
        var clazz = params.clazz
        for (var index in algorithmConfig.deserialization_blacklist.clazz) {
            if (clazz === algorithmConfig.deserialization_blacklist.clazz[index]) {
                return {
                    action:    algorithmConfig.deserialization_blacklist.action,
                    message:   _("Deserialization blacklist - blocked " + clazz + " in resolveClass"),
                    confidence: 100,
                    algorithm:  'deserialization_blacklist'
                }
            }
        }
        if (clazz.startsWith('java.util.PriorityQueue')) {
            return {
                action:    algorithmConfig.deserialization_blacklist.action,
                message:   _("Deserialization blacklist - blocked " + clazz + " in resolveClass"),
                confidence: 100,
                algorithm:  'deserialization_blacklist'
            }
        }
    })
    return clean
}
```

通过增加自定义策略防护反序列化链  
并进一步收集反序列化类进行数据分析

```
    }
    if (clazz.startsWith('java.util.PriorityQueue')) {
        return {
            action:    algorithmConfig.deserialization_blacklist.action,
            message:   _("Deserialization blacklist - blocked " + clazz + " in resolveClass"),
            confidence: 100,
            algorithm:  'deserialization_blacklist'
        }
    }
}
```

**最终**达到对  
反序列化异常行为  
侦测的目的。

关注我们



---

安世加专注于网络安全行业领域，通过互联网平台、线下沙龙、峰会、人才招聘等多种形式，培养安全人才，提升行业的整体素质，助推安全生态圈的健康发展。

**安世加**