2018

OPENRASP INTERNALS

V2019.04.17

百度安全
Baidu Security

OpenRASP Internals v2019.04.17

前言 3

1. 背景介绍 4

    1.1 OASES 简介 - https://oases.io 4

    1.2 OpenRASP 简介 - https://rasp.baidu.com 4

    1.3 应用安全主要威胁 4

2. RASP 技术介绍 5

    2.1 RASP 背景介绍 5

    2.2 RASP 技术原理 5

    2.3 RASP 技术和现有方案主要区别 6

3. OpenRASP 功能简介 7

    3.1 Web 2.0 攻击检测 7

    3.2 服务器安全基线检查 7

    3.3 应用加固 8

    3.4 统一管理后台 8

    3.5 SIEM/SOC 集成 8

4. 大规模部署 9

    4.1 理清架构 9

    4.2 性能测试 9

    4.3 灰度发布 9

    4.4 正式部署 9

    4.5 线上运维 10

    4.6 和 Nginx/Apache/Lighttpd 一起使用 10

5. 定制你的 OpenRASP 11

    5.1 技术定制 11

    5.2 后续规划 11

    5.3 OEM 支持 11

6. 常见问题 12

前言

本电子书讲解 OpenRASP 在企业安全里的最佳实践，它主要为以下人员编写，

- 安全运维人员
- 后端研发人员

当你看到本电子书时，你可能已经听说过RASP技术，也可能对这项新技术一无所知。无论是哪种情况，我们建议你带着如下问题，浏览下面的章节:

- RASP 技术是什么?
- OpenRASP 解决哪些问题、哪些技术难点?
- OpenRASP 是否适合我的公司，我的安全团队?
  - 当我决定部署 OpenRASP，我如何跟研发团队讲清楚它的架构?

本电子书将会为你一一解惑。如果你还有其他问题，或者对本电子书有任何疑问，请访问 https://rasp.baidu.com/#section-support，加入QQ技术讨论群，联系群主

另外，本电子书还在持续更新中，具体请留意QQ技术讨论群的公告

1. 背景介绍

在开始之前，我们先来了解下项目背景

1.1 OASES 简介 - https://oases.io

OASES（Open AI System Security Alliance）智能终端安全生态联盟是国内首个致力于提升智能终端生态安全的联合组织，由百度、华为、中国信通院联合发起成立，成员包括安全厂商、行业安全专家、设备厂商、高校和各行业安全专家等。OASES生态安全联盟希望能够引导一个开放、共享、合作、共建的安全生态链，促进手机厂商、智能终端厂商与安全厂商及安全专家之间建立良性的互动与合作，共同推进智能终端安全生态的建设。

OASES 目前包含 KARMA、OASP、OpenRASP、MesaLink、MesaLock Linux 等几个子项目。 其中，OpenRASP 主要应用于云端应用安全防护，以及上线前的应用安全测试。

1.2 OpenRASP 简介 - https://rasp.baidu.com

Gartner 在2014年提出了『运行时应用自我保护』技术的概念，即 对应用服务的保护，不应该依赖于外部系统；应用应该具备自我保护的能力。OpenRASP 是该技术的开源实现，它改变了防火墙依赖请求特征来拦截攻击的模式。对于注入类的漏洞，我们可以识别用户输入的部分，并检查程序逻辑是否被修改。由于不依赖请求特征，我们每条报警都是成功的攻击。

目前，OpenRASP 已经集成在多个商业主机安全软件里，也有大量客户将它部署至生产环境。如果你在使用过程中遇到任何问题，请加入我们的技术讨论QQ群，联系我们处理；如果你想关注项目进度，以及后面的规划，请查看 baidu/openrasp - 里程碑。

1.3 应用安全主要威胁

2001年的时候，大部分网站都没有对SQL注入的防护，处在裸奔的状态。到现在，应用安全攻防技术已经发展了十几个年头，我们却依然面临着大量威胁。

- 新型漏洞不断爆发，比如反序列化漏洞
- 网络边界越来越模糊，研发不知道什么时候上线了新服务
- 来自内部的数据泄露越来越多了
- 服务器上发现了遗留后门，黑客做了什么事情不知道
- …

对于大部分中小企业来说，对应用安全的理解可能还停留在WAF阶段。它们通常没有足够的人力去做整体的安全规划，对SDL的管理也有限。

在未来，百度安全将会逐渐开放各项安全能力、最佳实践，帮助中小型企业规划、建设自己的安全体系，并将SDL落地。

2. RASP 技术介绍

下面，我们来详细了解下RASP技术。作为一个安全运维人员，你可能会提出如下问题:

- RASP 技术如何工作?
- RASP 技术解决哪些安全问题?
- 和传统的应用安全防护方案相比，它的优势在哪里?

2.1 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。

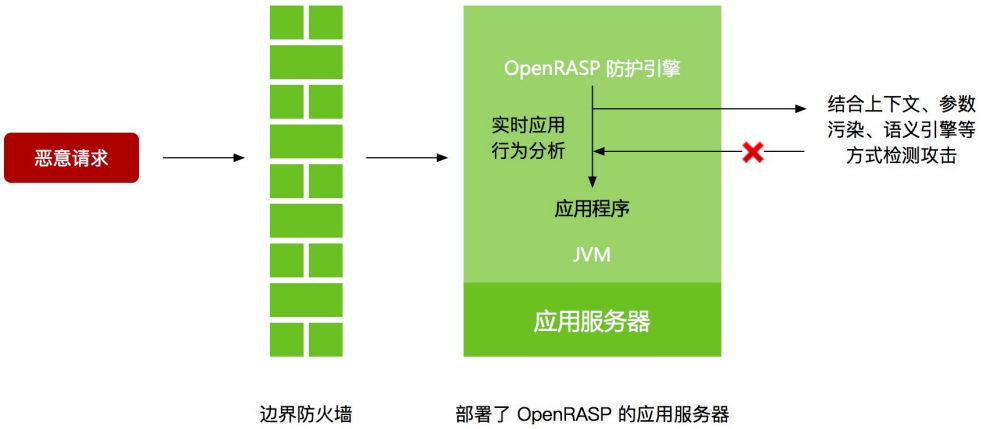
当然，这是针对已知漏洞的讨论。如果是新型漏洞，应用则更加需要进行自我保护。

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

2.2 RASP 技术原理

现在，我们对RASP技术有了一些初步的认识 -- RASP 是一种新型的应用防护技术，它部署在 tomcat/php/nodejs/python.. 等应用服务器上，并和他们紧密结合。下面，我们以Java服务器为例，来了解下它的架构。

在 Java 技术栈下，RASP 引擎以 javaagent 的形式实现，并运行在 JVM 之上。在应用服务器启动的时候，RASP 引擎借助 JVM 自身提供的 instrumentation 技术，通过替换字节码的方式对关键类方法进行挂钩:



如果你使用过APM产品，对 javaagent 技术应该不会陌生。RASP 的技术原理与APM是一样的，只是挂钩的函数要少很多。以 OpenRASP 为例，我们挂钩了SQL查询、文件读写、反序列化对象、命令执行等关键操作，具体列表可参考 二次开发 - 架构说明 - Hook 函数列表。

2.3 RASP 技术和现有方案主要区别

首先，RASP 几乎没有误报情况。边界设备基于请求特征检测攻击，通常无法得知攻击是否成功。对于扫描器的踩点行为、nday 扫描，一般会产生大量报警。RASP 运行在应用内部，失败的攻击不会触发检测逻辑，所以每条攻击都是成功的报警。

其次，RASP 可以发现更多攻击。以SQL注入为例，边界设备只能看到请求信息。RASP 不但能够看到请求信息，还能看到完整的SQL语句，并进行关联。如果SQL注入让服务器产生了语法错误或者其他异常，RASP引擎也能够识别和处理。

最后，RASP 可以对抗未知漏洞。发生攻击时，边界防护设备无法掌握应用下一步的动向。RASP 技术可以识别出异常的程序逻辑，比如反序列化漏洞导致的命令执行，因此可以对抗未知漏洞。

3. OpenRASP 功能简介

3.1 Web 2.0 攻击检测

首先，我们定义了25种攻击手法，并参考 OWASP TOP 10 2017 对检测能力进行了分类。针对每个漏洞，我们详细的说明了攻击场景、当前检测能力的覆盖程度。具体请参考 功能说明 - 检测能力说明。

- A1 - 注入
- A2 - 失效的身份认证和会话管理
- A3 - 敏感数据泄露
- A4 - XML 外部实体（XXE）
- A5 - 失效的访问控制
- A6 - 安全配置错误
- A7 - 跨站脚本（XSS）
- A8 - 不安全的反序列化
- A9 - 使用含有已知漏洞的组件
- A10 - 不足的日志记录和监控

其次，针对已经公开的 CVE 漏洞，我们也进行了大量测试。具体覆盖状况请参考 功能说明 - CVE 漏洞覆盖说明。

最后，如果你想要了解 OpenRASP 检测算法，你可以查看百度安全实验室公众号发表的文章，或者直接查看检测插件源代码。

3.2 服务器安全基线检查

OpenRASP 会在应用服务器启动时，进行安全配置规范检查。做好安全基线，可以减少应用服务器被入侵的风险。目前我们支持的策略如下，

- 3001 - 关键 cookie 是否开启 httpOnly
- 3002 - 进程启动账号检查
- 3003 - 后台密强度检查
- 3004 - 不安全的默认应用检查
- 3005 - Directory Listing 检查
- 3006 - 数据库连接账号审计
- 3007 - JBoss HTMLAdaptor 认证检查
- 4001 - PHP allow_url_include 配置审计
- 4002 - PHP expose_php 配置审计
- 4003 - PHP display_errors 配置审计
- 4004 - PHP yaml.decode_php 配置审计

不同的策略，支持的服务器也不同，具体请参考 功能说明 - 服务器安全基线检查。

3.3 应用加固

当应用收到请求，我们会通过输出响应头的方式，实现对应用的加固。目前支持的配置如下:

内容	响应头	可选配置
点击劫持防护	X-Frame-Options	不开启/deny/sameorigin
MIME 嗅探防护	X-Content-Type-Options	不开启/nosniff
XSS Auditor 防护	X-XSS-Protection	不开启/1; mode=block
文件自动运行防护	X-Download-Options	不开启/noopen

3.4 统一管理后台

管理后台支持攻击事件查看、基线日志查看，也支持主机管理、检测插件升级等功能。具体部署和使用方法请参考 服务配置 - 管理后台 文档。

3.5 SIEM/SOC 集成

无论是自研的，开源的还是商业SIEM产品，OpenRASP 都可以无缝集成。目前我们支持三种集成方式，分别是文件日志、Syslog TCP 方式、管理后台推送。在官方文档里，我们给出了 ELK、Splunk 两种SIEM的配置方法，具体请查看 安装部署 - SIEM系统集成。

4. 大规模部署

4.1 理清架构

通常，安全团队需要先跟研发团队、运维团队进行沟通，讲清楚RASP技术的架构，对服务器的影响会有哪些，以及部署RASP的必要性。

因此，我们编写了 二次开发 - 架构说明，供大家参考。如果你还有其他疑问，请加入QQ技术讨论群进行讨论。

另外，对于Java服务器，如果你在使用APM产品，还需要关注 APM 兼容性说明 这个FAQ。

4.2 性能测试

在线上部署之前，请先联系QA在测试环境进行压力测试，并需要关注下面这些问题:

- 如果CPU打满，平均请求响应时间下降多少?
- 如果CPU打满，QPS会下降多少?

在百度内部，我们测试了大量的业务系统和开源应用；QQ群里的用户也帮助我们测试了很多线上业务。在CPU打满的情况下，QPS影响通常在 1%~4% 之间，响应时间延迟通常在 3~10ms，基本可以忽略。

当然，如果你发现实际的性能损耗超过了 5%，请参考 二次开发 - 代码调试 - 性能调试 文档，采集性能数据。然后加入QQ群联系群主，我们会在第一时间进行分析，并尽快解决问题。

4.3 灰度发布

在上线前，我们建议你针对不同的业务，先灰度一些机器。在百度内部，我们会先跟业务线沟通，至少观察**一周**再决定下一步部署哪些机器。

为了保证代码质量，我们使用 travis 实现自动化测试，在每次提交代码时都会运行；在每次发版前还会有QA进行功能和压力测试。我们的程序捕获了所有的异常，即使引擎内部出现错误，通常也不会影响到服务器。虽然从第一个小版本到现在，我们从未发现稳定性问题，但我们依然需要保持谨慎，并和业务线的紧密沟通。

4.4 正式部署

百度内部的IDC环境较为复杂，除了常规的物理机，我们还有多种类似docker、虚拟主机的运行平台。为了支持这些环境，我们进行了大量的适配工作。目前，我们已经公开了大规模部署脚本和安装方案，具体请参考 安装部署 - 大规模部署 文档。

4.5 线上运维

运维工作通常包含如下内容，

- 机器存活性监控，包括磁盘空间、内存占用、是否在线
- 进程存活性监控，包括数据库、管理后台、客户端失联
- 数据库定期备份，包括MongoDB、ElasticSearch

考虑到大部分公司都有自己的监控平台和运维团队，所以这里不再赘述。

4.6 和 Nginx/Apache/Lighttpd 一起使用

以 nginx + php-fpm 架构为例，静态资源的请求通常不会经过RASP，所以RASP无法防护。比如敏感文件下载漏洞，就属于静态资源:

```
https://www.example.com/wwwroot.zip
```

在这种情况下，你需要通过配置web服务器来提升安全级别。若要禁止下载压缩包、SQL备份、git信息等敏感文件，你可以在 nginx 里增加如下配置:

```
location ~* \.(7z|tar|zip|rar|bz2|gz|sql)$ {
    deny all;
}

location /.git {
    deny all;
}
```

修改后使用 nginx -s reload 生效

5. 定制你的 OpenRASP

不同的企业使用不同的技术栈，安全防护的需求可能也不同。比如有的客户需要我们支持 DB2 数据库，有的客户需要允许在服务器上执行 python 命令。对于通用的需求，你可以加入QQ群联系我们处理，大部分合理的需求我们都会实现。

5.1 技术定制

目前已经有多家安全厂商对OpenRASP进行封装，他们通常会对进行如下改造:

- 规则加密和混淆
- 自定义远程管理
- 自定义检测算法

若要进行技术定制，首先要搞清楚我们的系统设计和底层架构，具体可参考我们的 二次开发 和 插件开发 文档。

如果你还有其他问题，请加入QQ技术讨论群进行讨论。

5.2 后续规划

根据QQ群里的用户反馈，我们计划在 2019 年加入对 golang/python/nodejs 三种语言的支持。具体的版本开发计划，请参考 baidu/openrasp - 里程碑 文档，以及 baidu/openrasp - Issues。

5.3 OEM 支持

作为开源生态的一个环节，我们欢迎安全厂商基于 OpenRASP 进行二次开发。为了更好的支持商用，OpenRASP 采用了 Apache License 2.0 协议，更多细节请参考聊聊 Apache 开源协议这篇文章。

如果你打算OEM我们的开源产品，请务必关注我们的QQ技术讨论群公告、OpenRASP 里程碑、百度安全实验室公众号，以便及时得知新的特性和bug修复状况。

6. 常见问题

具体请参考 首页 - FAQ，目前我们解答了如下问题:

- 目前支持哪些应用服务器?
- OpenRASP 可以检测哪些攻击类型?
- OpenRASP 是否会影响服务器的性能?
- 如何集成到现有的 SIEM/SOC 平台中?
- 如何开发一个检测插件 / 定制安全检测能力?
- 对于一个新插件，如何避免产生太多误报，影响业务?
- 为什么选择用 JavaScript 插件实现检测逻辑?
- 检测插件是否支持实时更新?
- OpenRASP和商业RASP产品的区别在哪里?

如果你还有其他疑问，请加入QQ技术讨论群，联系群主。对于好的问题，我们会在第一时间增加到FAQ列表里。