



# DevSecOps落地实践分享

李博 高尉峰 · 2022年12月26日





# 目录

## CONTENTS

1. 背景
2. 安全设计
3. 安全开发
4. 安全测试
5. 安全运营
6. 总结与思考







背景



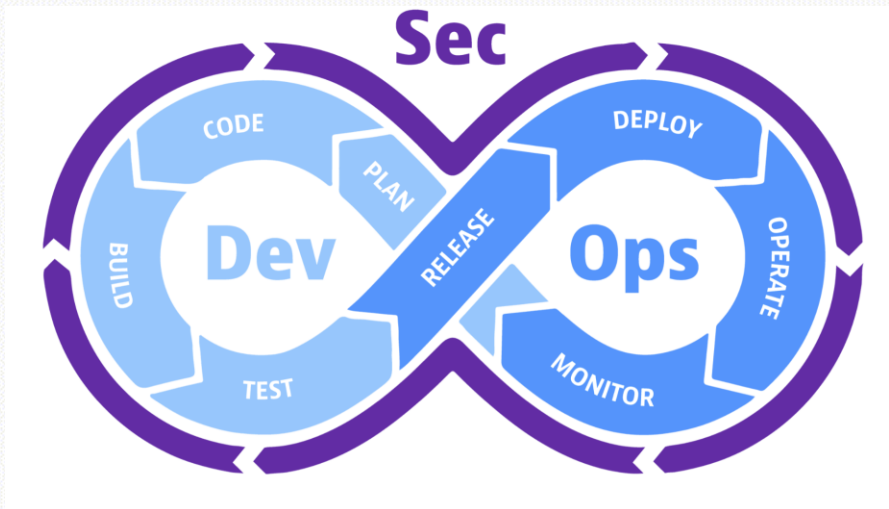


# 一、背景——行业背景

❖ 自2004年起，微软提出了**安全开发生命周期 (SDL)**，其核心理念就是将安全考虑集成在软件开发的每一个阶段:需求分析、设计、编码、测试和维护。从需求、设计到发布产品的每一个阶段每都增加了相应的安全活动与规范，以减少软件中漏洞的数量并将安全缺陷降低到最小程度。

❖ 2012年，Gartner首次提出DevSecOps，四年后，它发布了一份名为《DevSecOps: How to Seamlessly integrate Security into DevOps》的报告。这份报告的核心理念是：**安全是IT团队所有成员的责任，要贯穿到业务生命周期的每一个环节。**

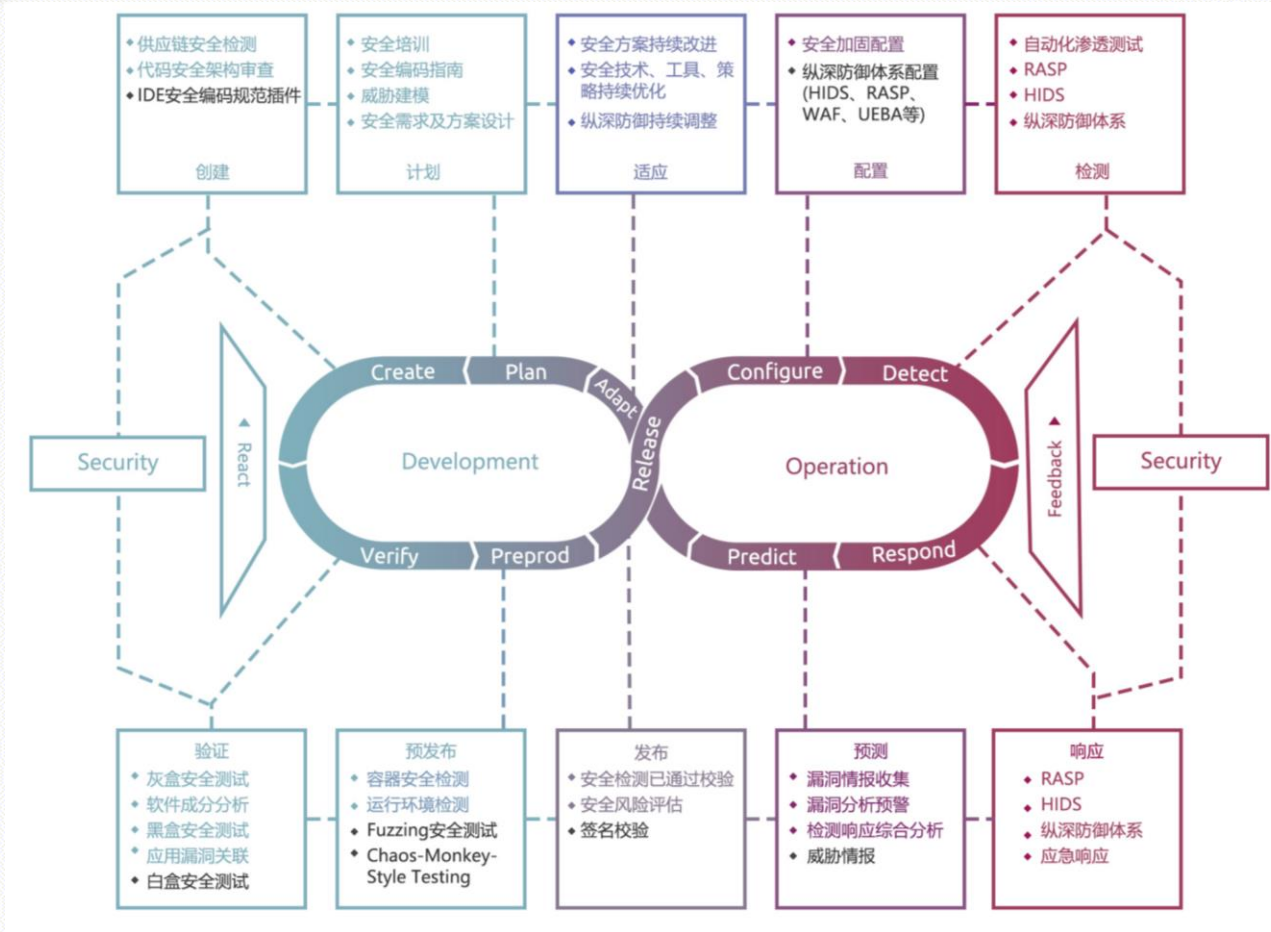
❖ DevSecOps的价值是在不牺牲安全性的前提下，快速落地和实施安全。DevSecOps由DevOps的概念延伸和演变而来。其核心理念是**安全是整个IT团队每个人的责任，需要贯穿从开发到运营整个业务生命周期每一个环节才能提供有效保障。**





# 一、背景——领域背景

DevSecOps依赖于DevOps流程工具链，将威胁建模工具、安全编码工具、安全测试工具、容器安全检测工具、基线加固工具、漏洞管理工具等自动化工具无缝集成到DevOps流程中，进而实现**开发**、**安全**、**运营**一体化。



DevSecOps工具链









## 二、安全设计——设计规范

🔗 **设计安全**是实现DevSecOps非常重要的一环，大量历史经验也表明，越早在架构设计阶段考虑到安全设计的系统，比那些在越晚的开发设计阶段才考虑安全设计的系统，要安全得多。根据美国国家标准与技术研究所（NIST）统计，在发布后执行代码修复，其修复成本相当于在设计阶段执行修复的30倍。

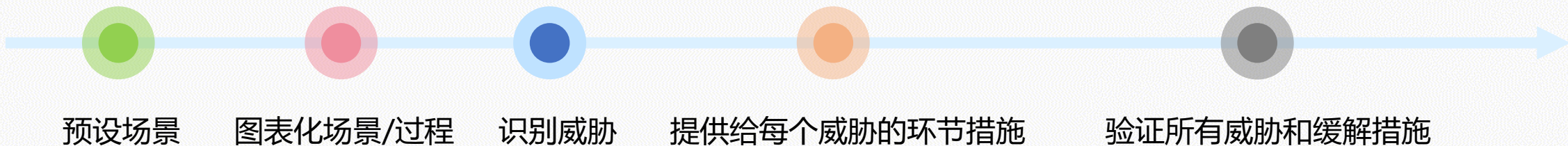
🔗 **DevSecOps 的目标**是在软件生命周期的全部阶段，可以更早、更快地发现并处理安全问题。因此，智慧城市平台部结合业界知识库和长期实践输出了《安全设计规范》，并且给开发人员进行了培训。





## 二、安全设计——威胁建模

### STRIDE威胁建模的过程



#### MICROSOFT THREAT MODELING TOOL

Version: 7.3.20120.2

##### Threat Model:

###### Create A Model

Model your system by drawing diagram(s). Make sure you capture important details.

###### Open A Model

Open an existing model file and analyze threats against your system.

###### Getting Started Guide

A step-by-step guide to help you get up and running now.

##### Template For New Models

中移系统集成威胁建模(4.1.0.80)

Browse...

##### Recently Opened Models

[oncity.tm7](#)

[onespace.tm7](#)

[威胁建模.tm7](#)

[Sample\\_Threat\\_Model.tm7](#)

##### Threat Modeling Workflow

1. Select your template.
2. Create your data flow diagram model.
3. Analyze the model for potential threats.
4. Determine mitigations.

##### Template:

###### Create New Template

Define stencils, threat types and custom threat properties for your threat model from scratch.

###### Open Template

Open an existing Template and make modifications to better suit your specific threat analysis.

##### Template Workflow

Use templates to define threats that applications should look for.

1. Define stencils
2. Define categories
3. Define threat properties
4. Define threat
5. Share your template

这我司在大量的实践经验基础上，构建了自己的安全威胁库和漏洞防御库，实现了**轻量级威胁建模过程**，通过安全评估调查问卷，从系统结构和使用场景去识别将要构建的应用类型，然后匹配对应的威胁库和漏洞防御库，确定安全需求基线，最终得出安全设计方案。









### 三、安全开发

- ❖ 开发人员编码要遵循《安全设计规范》、《安全编码规范》。
- ❖ 对引入的开源组件要做威胁分析，确保软件供应链安全。
- ❖ 代码仓库权限要做到最小化。
- ❖ 为了提升安全编码能力，开发通过代码编辑器插件调用源码扫描引擎，对代码进行实时的安全扫描，实时查看扫描报告。
- ❖ 我司还针对产品中常见的漏洞开发了安全sdk库，可以修复sql注入漏洞，xss漏洞，ssrf漏洞,反序列漏洞，url重定向漏洞。开发修复安全问题使用安全sdk，提高开发人员修复安全问题的效率。





## 三、安全开发

❧ **反序列化漏洞**：当输入的反序列化的数据可被用户控制，那么攻击者即可通过构造恶意输入，让反序列化产生非预期的对象，在此过程中执行构造的任意代码

```
public class UnsafeClass implements Serializable {
    public String name;

    //重写readObject()方法
    private void readObject(java.io.ObjectInputStream in) throws IOException, ClassNotFoundException {
        //执行默认的readObject()方法
        in.defaultReadObject();
        //执行命令
        Runtime.getRuntime().exec("open /Applications/Calculator.app/Contents/MacOS/Calculator");
    }
}
```

```
public class SafeClass implements Serializable {
    public String name;

    //重写readObject()方法
    private void readObject(java.io.ObjectInputStream in) throws IOException, ClassNotFoundException {
        //执行默认的readObject()方法
        in.defaultReadObject();
    }
}
```

```
try {
    UnsafeClass objectFromDisk = (UnsafeClass) ois.readObject();
} catch (Exception e) {
    //此处抛出异常，异常信息如下
    //2022-10-09 11:05:11 ERROR SecureObjectInputStream:91 - Unauthorized
    //deserialization class
}
```







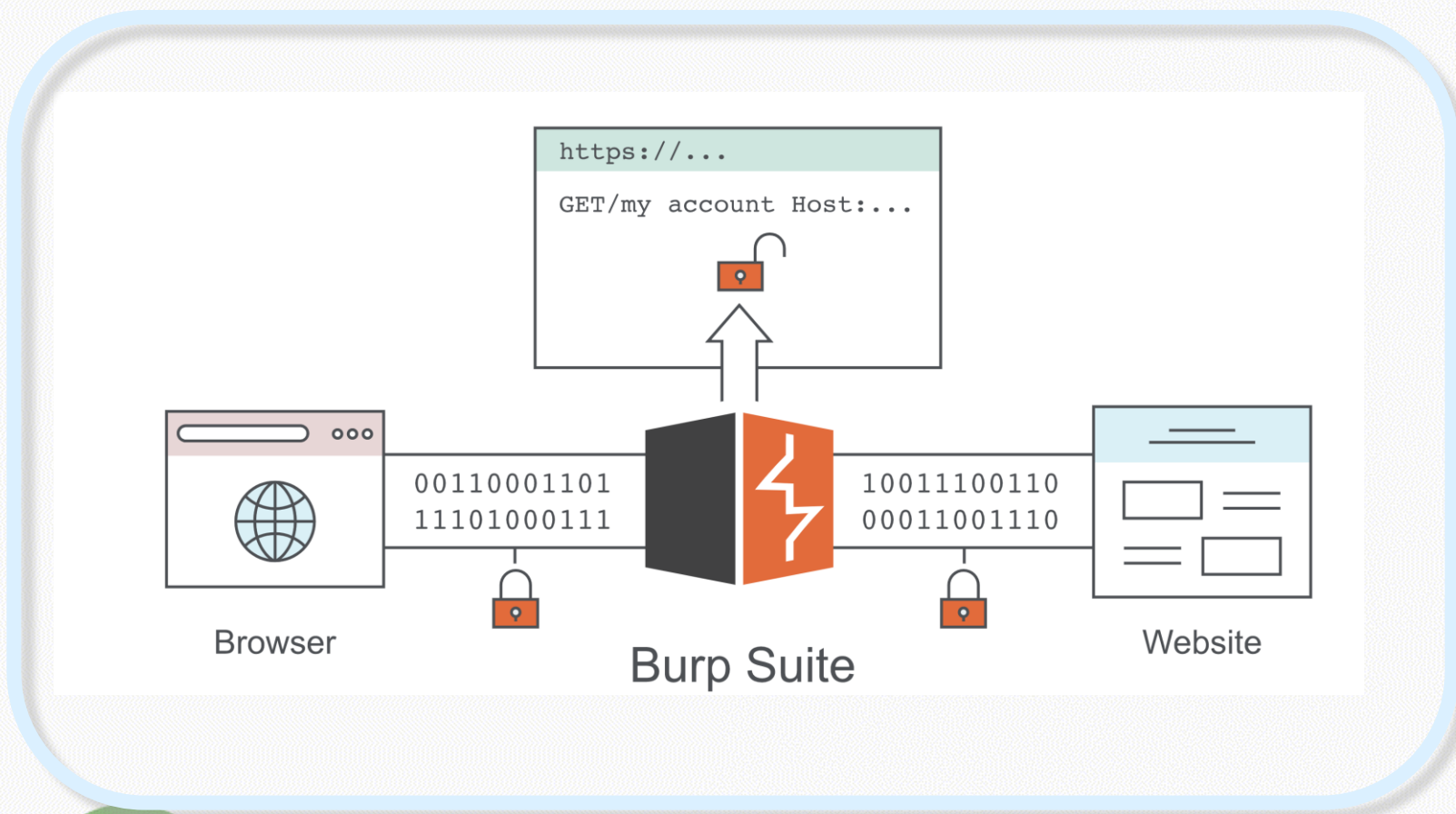
# 安全测试





## 四、安全测试——DAST

❖ **动态应用程序安全测试** (Dynamic Application Security Testing) 技术在测试或运行阶段分析应用程序的动态运行状态。它模拟黑客行为对应用程序进行动态攻击，分析应用程序的反应，从而确定该Web应用是否易受攻击。





## 四、安全测试——DAST

✎ 我司采用**burp suit, appscan, xray, awvs**等工具进行漏洞扫描,采用app store中的插件来增强扫描功能,对于鉴权、业务逻辑等漏洞采用人工渗透的方式。

### ✎ DAST 的优势

- 1、独立于应用程序
- 2、不需要查看源代码

### ✎ DAST 的劣势

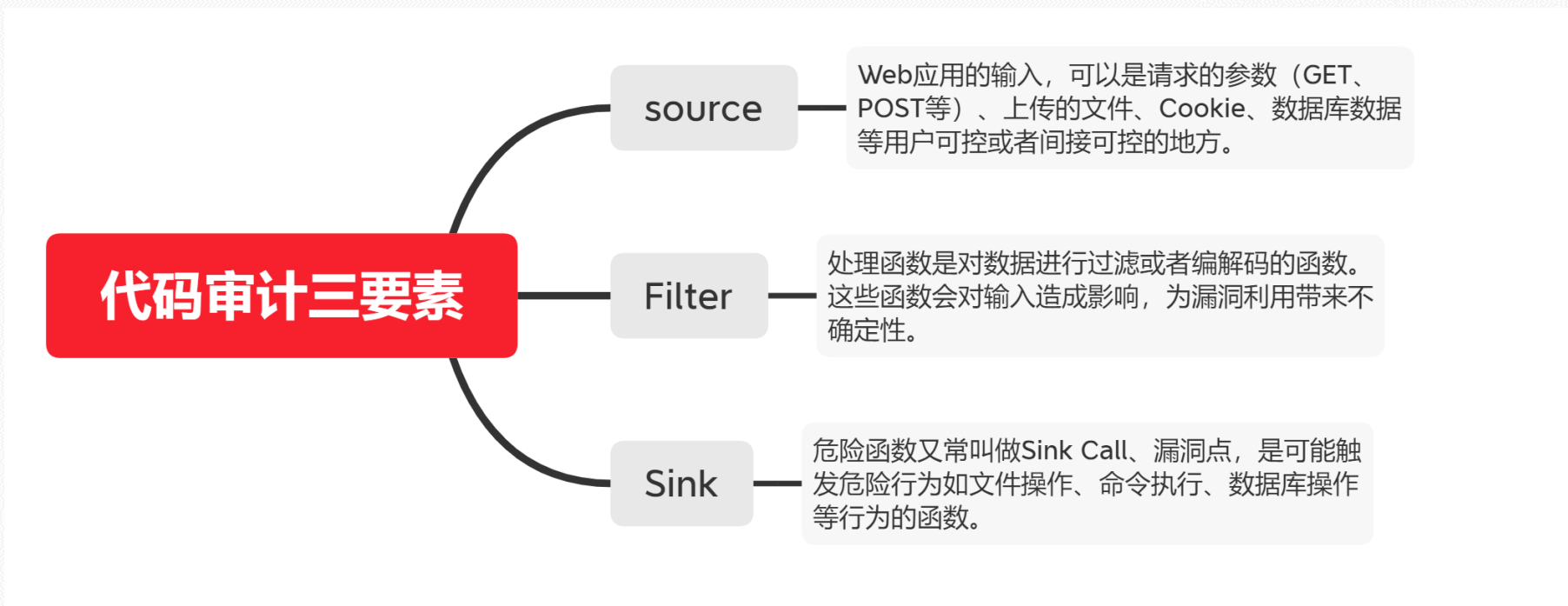
- 1、无法定位到代码中漏洞的确切位置
- 2、安全测试周期长





## 四、安全测试——SAST

✦ **静态应用程序安全测试** (Static Application Security Testing) 技术通常在编码阶段分析应用程序的源代码或二进制文件的语法、结构、过程、接口等来发现程序代码存在的安全漏洞。





## 四、安全测试——SAST

❏ 我司使用奇安信源码扫描系统进行源码安全扫描，通过流水线插件实现自动化扫描，给代码提交设置安全门禁，从而消减代码的安全风险。为了降低告警误报，部门根据业务自定义扫描模板，将代码扫描的**误报率降低了50%**。

### ❏ SAST 的优势

- 1、确保安全编码
- 2、能够实现业界大部分已知漏洞检测

### ❏ SAST 的劣势

- 1、**不能涵盖所有漏洞**：在源代码中很难自动检测出诸如身份验证，访问控制之类的问题。
- 2、**误报率很高**：SAST的结果包括大量误报，使开发和安全团队付出了很多时间和精力，来清除误报。
- 3、**效率低**：尽管SAST自动化可以连续扫描，但是对于大型代码库，一次扫描可能要花费几个小时。SAST只能扫出潜在的漏洞，还要让开发人员来验证是否存在安全风险。





## 四、安全测试——IAST

🔗 **交互式应用程序安全测试**（IAST）是2012年Gartner公司提出的一种新的应用程序安全测试方案，通过在服务端部署Agent程序，监控Web应用程序运行时函数执行、数据传输，并与扫描器端进行实时交互，高效、准确的识别安全缺陷及漏洞，同时可准确确定漏洞所在的代码文件、行数、函数及参数。IAST相当于是DAST和SAST结合的一种**互相关联运行时安全检测技术**。

IAST	主动式	被动式
检测原理	DAST+RASP	动态污点分析
误报	基本无	存在
漏报	存在	较少
加密场景	不支持	支持
覆盖场景	常规	更多
脏数据	可能	无
开发难度	中	高
分布式	支持	支持
复现难度	容易	一般



## 四、安全测试——IAST

### 被动式iast动态污点分析

- ❖ 污点分析可以抽象成一个三元组<sources,sinks,sanitizers>的形式,其中, source即污点源, 代表直接引入不受信任的数据到系统中;
- ❖ sink即污点汇聚点, 代表直接产生安全敏感操作或者泄露隐私数据到外界;
- ❖ sanitizer即无害处理,代表通过数据加密或者移除危害操作等手段使数据传播不再对软件系统的信息安全产生危害;
- ❖ 污点分析就是分析程序中由污点源引入的数据是否能够不经无害处理,而直接传播到污点汇聚点。如果不能,说明系统是信息流安全的; 否则,说明系统产生了隐私数据泄露或危险数据操作等安全问题。





## 四、安全测试——IAST

### SSRF举例:

```
String urlString = request.getParameter("url");  
URL url = new URL(urlString);  
URLConnection urlConnection = url.openConnection();
```

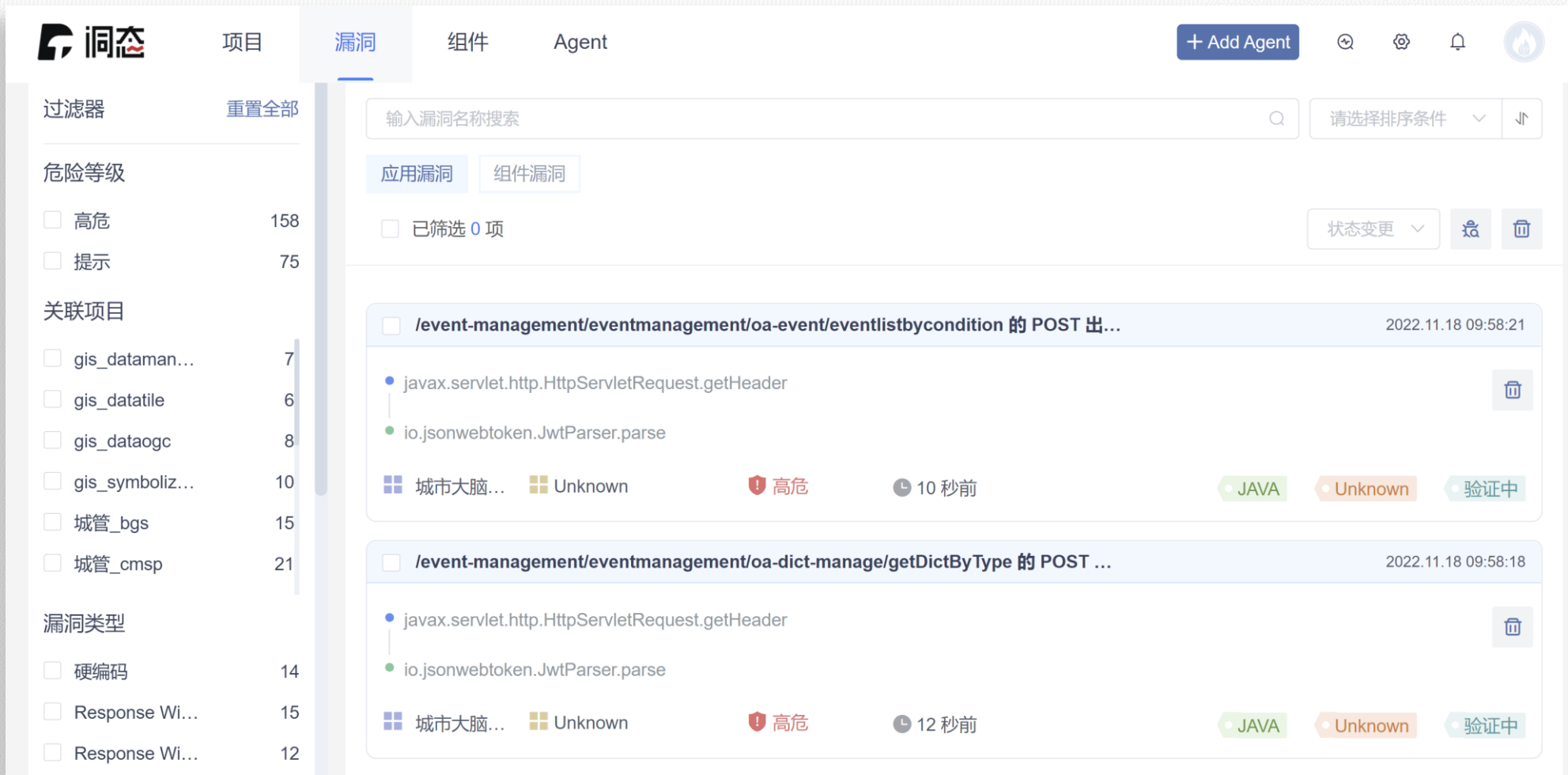
- 1、根据代码我们在IAST中设置RequestFacade.getParameter(java.lang.String)为我们设置的Source点;
- 2、设置Propagator污点传播: 如果URL<init>接受的构造参数为污点, 那么return值也标记为污点;
- 3、Sink点最后设置为URL的openConnection()方法, 检测接受的污点参数是否为自身。





# 四、安全测试——IAST

- 我司通过引入洞态iast系统，从流量层面和源码层面实现漏洞检测，具有**低误报**，**确认安全问题更方便**，**自动化程度高**的优点，成功弥补了DAST和SAST的缺点。





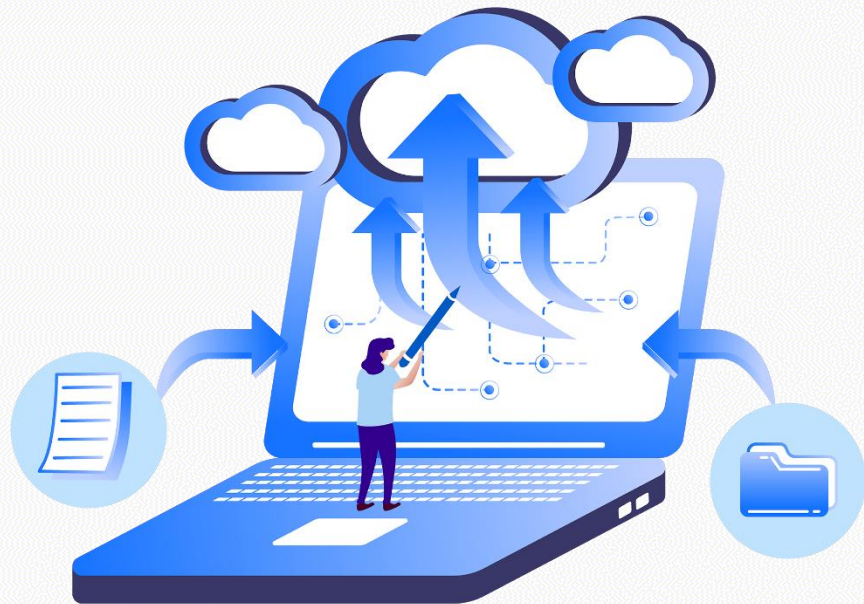
## 四、安全测试——IAST

### 优势

- 不会产生脏数据
- 准确性高，可以结合数据流以及请求上下文来分析
- 支持代码层面的分析
- 操作简单，方便漏洞定位与修复
- 解决应用系统安全测试与效率的矛盾

### 应用场景

- 适用于上线前的测试
- 安全人力缺乏状况





## 四、安全测试——SCA

**软件成分分析** (Software Composition Analysis , SCA) 是一种对二进制软件的组成部分进行识别、分析和追踪的技术。SCA通过分析开发人员所使用的各种源码、模块、框架和库。

- 1、软件成分分析分为两种模式，静态和动态。
- 2、静态模式是使用工具对目标工程文件进行解压，识别和分析各个组件的关系。
- 3、动态模式则是依赖于执行过程，在程序执行的同时收集必要的活动元数据信息，通过数据流跟踪的方式对目标组件的各个部分之间的关系进行标定。





## 四、安全测试——SCA

✎ 我司自研的onestone系统，可以**准确的查看系统使用的开源组件厂商和版本号**，提高了开发修复漏洞的效率。









# 五、安全运营





# 五、安全运营——配置核查

## ✖ k8s和docker基础设施扫描

- 我司部采用docker-bench实现容器环境扫描，采用kube-bench实现k8s基础环境扫描。
- 该方法能够覆盖安全业界的CIS测试标准，并且能够给出加固建议。然后使用shell脚本实现自动化加固。

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Scored)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Scored)
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Scored)
[PASS] 1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Scored)
[PASS] 1.1.5 Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Scored)
[PASS] 1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Scored)
[FAIL] 1.1.7 Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Scored)
[FAIL] 1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Scored)
[WARN] 1.1.9 Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Not Scored)
```

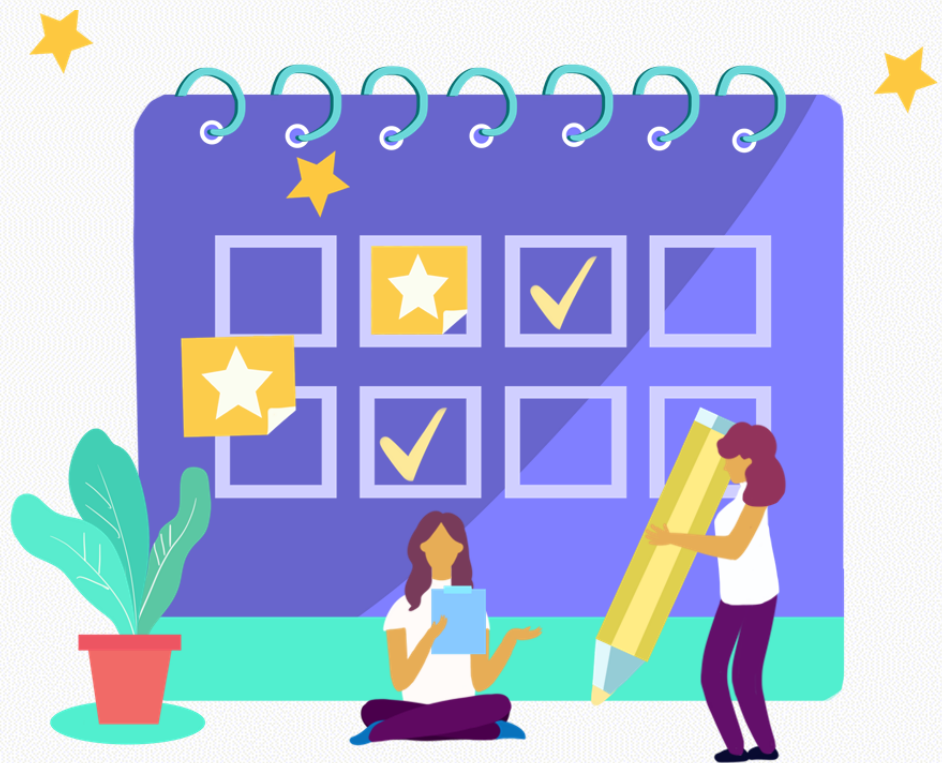




## 五、安全运营——RASP

❖ **WAF**作为第一道防线，WAF能够阻止基本攻击，但难以检测到 APT 等高级威胁，不仅如此，企业需要持续“调整”WAF 以适应不断变化的应用程序，这一过程消耗了安全管理人员大量的精力。此时，运行时应用程序自我保护技术（Runtime Application Self-protection, RASP）作为新一代运行时保护技术被引入，当应用程序遭受到实际攻击和伤害时，RASP可以实时检测和阻断安全攻击，无需人工干预，最终实现软件应用的自我保护，确保软件应用的安全运行。

❖ **RASP**在运营阶段可以应对无处不在的应用漏洞与网络威胁，为应用程序提供全生命周期的动态安全保护，可以精准识别应用运行时暴露出的各种安全漏洞，进行深度且更加有效的威胁分析，快速定位应用漏洞，大大提升修复效率，保障应用程序的安全性。





# 五、安全运营——RASP

## RASP优势

	waf	rasp
原理	基于原始的规则匹配对所有输入内容和流量进行检测	在可能发生攻击的关键函数调用处检查是否有恶意参数执行
防护应用种类	不区分应用	Web应用
误报	误报率较高	无误报
防御方式	给服务打补丁	不用打补丁
防御位置	网络边界	服务内部
对主机性能影响	高	低
攻击溯源	无法溯源	能够完整显示攻击路径，追溯到攻击ip，漏洞执行函数和恶意参数
防护0 day漏洞能力	低	高





## 五、安全运营——RASP

### RASP劣势

**性能损耗：**RASP实时拦截、深入检测用户数据流，这是对精确度和误报率都有很大的帮助，但是对用户性能有一些影响，这些性能消耗也必然影响到用户的体验，这也是影响企业客户部署 RASP 的很大一方面原因。

**部署成本：**RASP 是针对应用程序的，每个应用程序都必须有独立的探针，不能像防火墙一样只在入口放置一个设备就可以了，RASP还需要适配应用开发的技术。比如 PHP 应用与 Java 应用需要不同的 RASP 产品，增加了部署成本。





# 五、安全运营——RASP

✎ 我司采用onestone系统(RASP)实现入侵防御。主要解决以下**痛点**：

- 漏洞爆发后的修复时间少则两周，多则数月，存在攻击空窗期，无法实时拦截攻击。
- 开发人员打安全补丁时无法准确定位问题，导致安全问题未完全修复。
- 现有waf系统存在误报率高，无法拦截未知威胁，无法感知攻击，无法精准溯源的风险。







## 总结与思考





## 六、总结与思考

### Tips:

- DevSecOps只是一个方法指导，具体实施还是需要依赖于自己公司的业务情况。
- 安全工具自动化能力是DevSecOps的核心，安全工具自动化能力决定了DevSecOps能否落地。
- 整个体系流程的建设按不同的模块分别划分出来，这样才能保证每个模块都有人负责跟进，同时该模块负责人需要了解其他能力建设，才能对于分模块的计划和发展有更深的认识。
- 安全需要自上而下，对于流程的推进，部门间的合作，尤其是对业务的上线流程需要卡点、增加流程等都需要有上层的大力支持。
- 业务和安全总体来说还是对立的，安全需要和业务发展平衡好，需要让业务意识到安全是业务的一个属性，才能互相促进。
- 业务应该意识到安全不是给业务找麻烦的，我们的目标是一致的，都是为了业务的安全发展，应该意识到安全不应只是安全部门的事情，人人都应该为业务安全负责。







# 提问时间

谢谢大家







关注社区公众号  
了解更多活动

