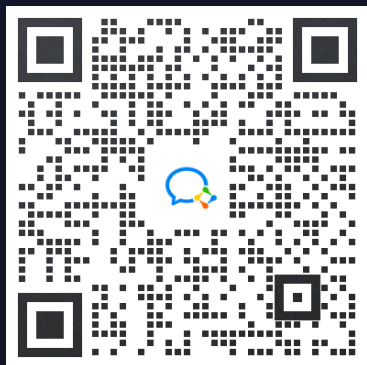




# 腾讯云代码分析

Tencent Cloud Code Analysis

Version : 20240418



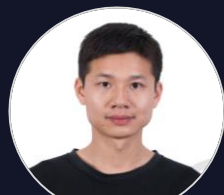
↑扫码加入代码分析交流群



# 团队成员介绍



张勇军



覃奋翔



林桂鸿



陈越威



唐晨



罗春兰



叶琪霖



贺毅楠



王惠



胡京迪



查茂鸿



江剑君



黄嘉怡



赵璐



李友杰



刘铭康



刘安静



甘伟



黄淇

# 目录

- 01 产品介绍
- 02 应用场景
- 03 产品特性及优势
- 04 页面效果展示
- 05 典型案例展示





# 01 产品介绍

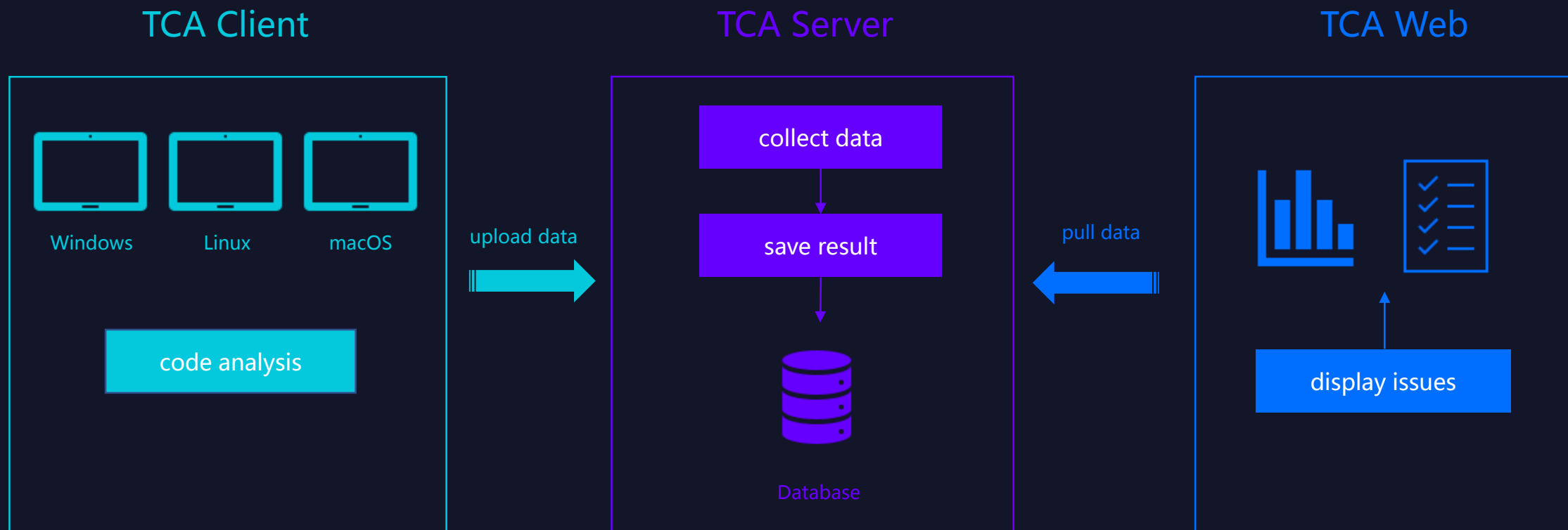
01 Product description

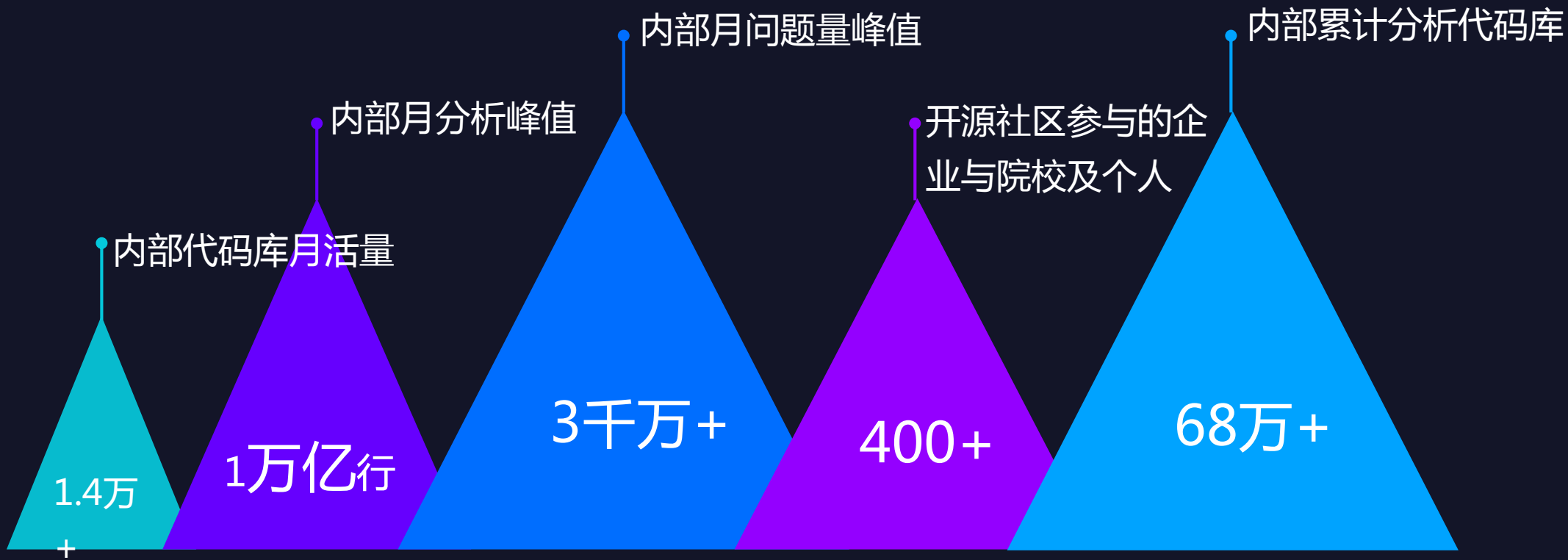


- 腾讯云代码分析于 2013 年从个别独立代码分析工具开始，持续逐步迭代强化，至今发展成集众多分析工具的云原生、分布式、高性能的代码综合分析跟踪子系统。
- 代码分析旨在运用词法分析、语法分析、控制流、数据流分析等的技术，对代码进行综合分析，查找代码中的规范性、结构性、安全漏洞等问题，进而输出代码的全方位质量报告，帮助项目持续监控项目代码质量。
- 腾讯云代码分析帮助项目实现“测试左移”，从而尽早以低成本、高效率发现代码问题，减少修复成本，缩短修复时间。

“在实施环节中修复错误，比在设计环节中修复错误的成本高出六倍。测试成本会增加**15倍**，部署成本会高至**100倍**。”

“用心关注每行代码迭代，助力维护卓越代码文化”。





- 腾讯云代码分析自立项以来，在腾讯内部已迭代至 6.0 版本，实践了 68W+ 代码库分析工作，为腾讯云、腾讯会议、手机 QQ、微视、QQ 音乐等明星产品迭代提供了有力支持。
- 对外与 CODING DevOps、腾讯云官网、腾讯云移动金融开放平台保持战略合作，助力更多外部企业提升研发质量。
- 在 2021 年底，腾讯云代码分析正式开源至 GitHub，至今已有 400+ 企业/院校/组织/个人加入开源社区（开源地址：<https://github.com/Tencent/CodeAnalysis>）。



## 02 应用场景

02 Application scenarios



开发全流程可接入，代码缺陷无处可藏。



### 代码安全

注入型漏洞  
URL重定向漏洞

### 代码质量

数组越界  
空指针解引用

### 代码异味

圈复杂度  
重复代码  
过长参数

### 专项提升

iOS减包  
权限调用扫描

# 代码安全

- 支持对OWASP Top10 中常见的漏洞进行分析，包括SQL注入、XML注入、外部实体注入攻击、敏感信息泄漏、URL重定向漏洞等，并结合CWE中常见漏洞，比如服务端请求伪造漏洞、服务器模板注入漏洞等，进行专项安全漏洞分析，准确识别漏洞所在位置并提供修复建议。

【Python】基础安全

安全

python安全扫描规则包

适用于 1 种语言  
python

【Go】强化Go安全规则

安全

Go安全规则，需要申请license使用

适用于 1 种语言  
Go

【PHP】强化ThinkPHP框架安全规则

安全

ThinkPHP框架安全规则，需要申请license使用

适用于 1 种语言  
php

查看详细规则 >

【Python】强化Djang框架安全规则

安全

Djang框架安全规则，需要申请license

适用于 1 种语言  
python

查看详细规则 >



规则名称	规则概要
xss	跨站脚本攻击
ssrf	服务端请求伪造
sql	sql注入
xml	xml注入
reflectioni	反射型注入
ldap_injection	LDAP注入

# 代码质量

- 数组越界（AOB）和空指针引用（NPD）等这类问题对软件稳定性、代码可靠性影响巨大，但在编码期间很难被检测到。而普通的代码走查方式成本高、有效性差，且不易跟踪管理。
- 腾讯云代码分析支持识别潜在漏洞，帮助开发分析和解决代码缺陷，减少代码走查测试成本，提高软件可靠性、健壮性。

数组越界

空指针引用

## func\_ret\_null

func\_ret\_null 函数返回值可能为nullpointer，但是调用该函数时指针未经判空便进行使用  
在选用func\_ret\_null\_full 时，检查器会在项目内全局搜索空指针函数的调用情况，否则只会在相关文件内进行检查。

### 代码示例

以下提供一个或多个func\_ret\_null代码案例

在下面代码中 test 函数中调用 get\_name 可能返回空指针，在后续使用 name 指针前应该判断是否为空指针

```
// name.hpp

char* get_name(int id) {
    char* name = 0;
    if (id == 1) {
        name = "Zeus";
    } else if (id == 2) {
        name = "Hades"
    } else {
        return nullpointer;
    }
    return name;
}

void test(int i) {
    char* name = get_name(i);
```

## array\_overflow

array\_overflow 检查数组越界的情况。不正确的缓存区访问可能损坏内存，导致程序崩溃或读取到权限外的内存。

### 代码示例

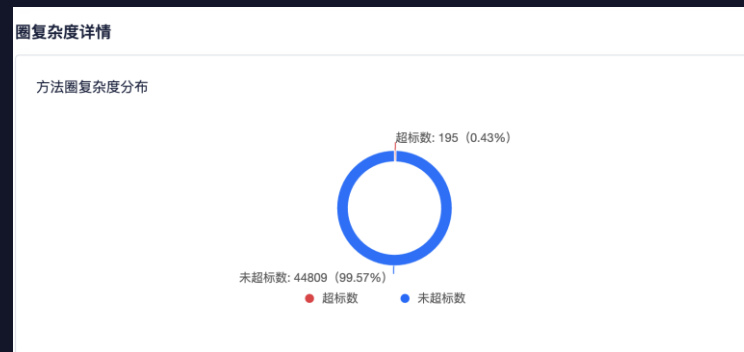
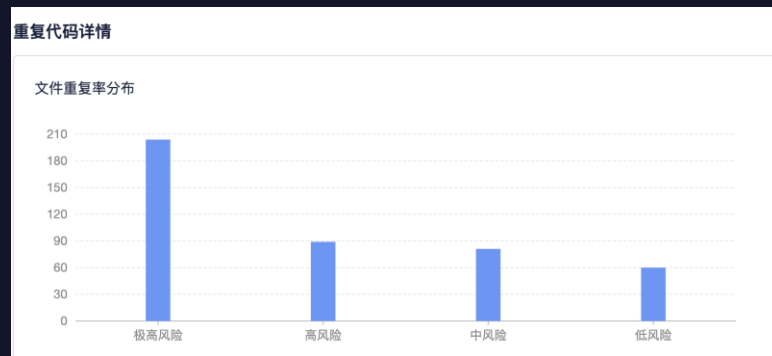
以下提供一个或多个array\_overflow案例

```
void foo() {
    int array[10];
    int i = get();
    // i = 9;
    if (i > 8 && i <= length(array)) { // Shoud be i < length(array)
        array[i] = 1; // defect: array[10] overflow
    }
    array[i] = 1; // defect: array[10] overflow
}

void test(int i) {
    int n= 10;
    char *p = malloc(sizeof(int) * 10);
    int y = n;
    p[y] = 'a'; // defect: writing to buffer[y] overflow
}
```

# 代码异味

- “代码异味是一种表象，它通常对应于系统中更深层次的问题。” 如果程序没有用一种好的表达方式来表现，那程序会很难阅读，难维护，难修改。
- 通过针对圈复杂度、重复代码、过长方法、过长参数列表等多类型代码异味扫描，将代码异味可视化，协助开发者更便捷地重构代码，提升代码的可读性、可维护性。写取悦自己、让他人仰慕的代码。



# 专项提升

- iOS审核、iOS减包、Android动态权限调用扫描、Android危险权限扫描、Android减包等多类专项扫描，有针对性进行代码问题分析。助力快速通过审核，避免公关危机等。

## 【Java】Android动态权限调用扫描

动态权限API检测 –  
TwilightManager

动态权限API检测 –  
WallpaperManager

动态权限API检测 –  
TelephonyManager

.....

## 【Object-C】减包扫描

扫描未使用的变量

扫描未使用的方法

发现未使用的图片文件

.....

## 【Java】Android危险权限扫描

文件共享权限扫描

文件读写模式扫描

通讯录API扫描

.....

## iOS隐私合规检查

扫描疑似私有API

苹果审核-不得强制安装其他软件

苹果审核-不得占位未实现功能

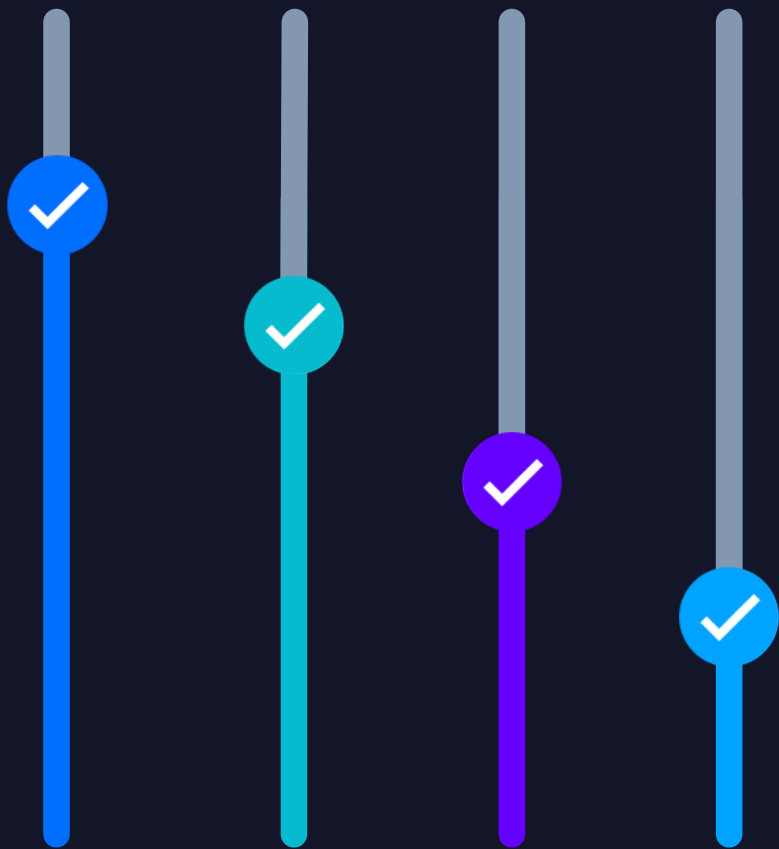
.....



## 03 产品特性及优势

03 Product features and benefits

## 稳定可靠的架构



- 云原生微服务架构，支持资源弹性调度
- 分布式客户端模式，自适应优化分析效率
- 国产化ARM64适配，支持信创
- 服务分层设计，支持灵活扩展适配
- 数据高效存储，支持大规模并发分析





# 标准化API接口，灵活融入DevOps

- 支持对接业内常见Git/SVN仓库。
- 标准化API全开放，可快速对接主流CI、DevOps平台。
- 支持 GitHub action , Jenkins plugin , 快速接入。



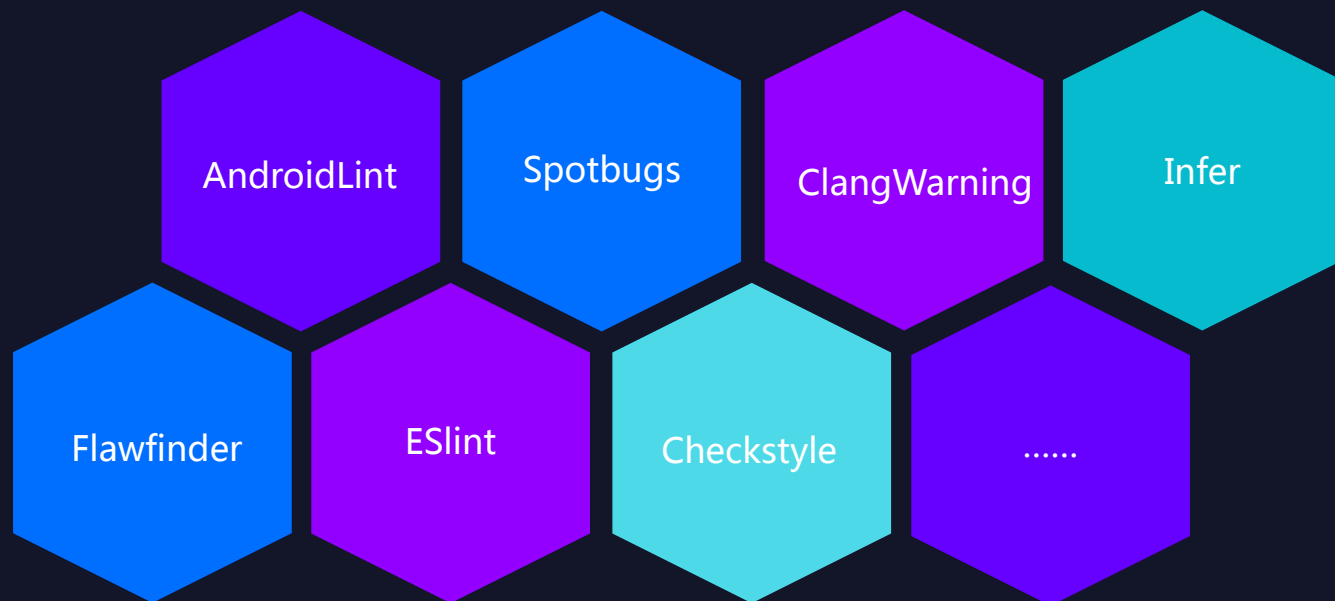
# 分布式客户端

- 客户端可以作为常驻进程，启动一个分析节点。
- 多个分析节点可以分布式并行执行服务端下发的分析任务，提高扫描效率。
- 分布式的客户端支持的环境含 Linux、Mac、Windows、ARM64，满足用户高频分析场景。



## 多工具支持，工具规则可扩展

- 已支持 248 款工具，2500+分析规则，让各种类型代码都可以轻松扫描。
- 可扩展的用户自定义工具、规则引擎：支持用户自定义工具规则，可以针对自身业务定义业务逻辑规则；可集成自研工具、商业工具，满足项目需要。



# 多种语言支持

- 覆盖业内主流 33 门语言（部分工具不受语种限制）。
- 支持自动识别语言。

语言	C/C++	C#	Css	Go	Html	JavaScript	T-SQL	Protocol Buffers
	Java	Kotlin	Lua	Objective-C	PHP	Python	XML	Rust
	Ruby	Scala	Swift	TypeScript	Visual Basic	ABAP	Dart	SQL
	Apex	COBOL	Flex	PL/I	PL/SQL	RPG	Shell	WebAssembly
框架	ThinkPHP	Django	Testing	Testify	React	Bottle	Vue	.....

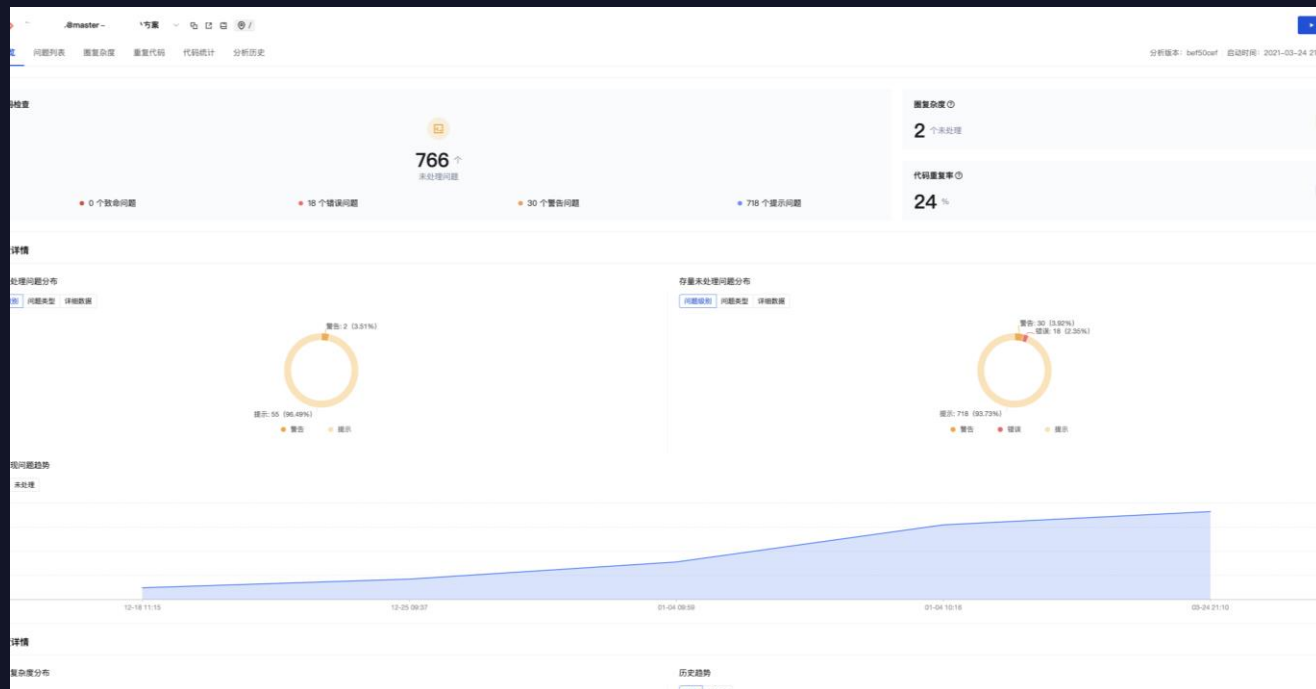
# 增量全量分析

- 增量全量分析互相配合，满足研发全流程各环节的不同要求，兼顾研发效能和研发质量。
- 增量分析配合高性能工具执行引擎和高效任务调度逻辑，助力快速获得扫描结果，无需漫长等待，提升研发效能。



# 全方位质量报告

- 图形化问题分布报告，多维度展示代码库最新健康状态。
- 历史趋势分析视图，轻松监管代码质量趋势。
- 本地离线报告，便捷查阅代码质量概览。



# 持续跟踪问题

- 问题跟踪和处理：
  - 持续跟踪管理问题：根据最新扫描结果自动关闭已修复问题。
  - 支持主动标记处理问题，如无需处理的问题、误报的问题。
  - everything as code ( XaC )，支持.code.yml设置文件责任人。
- 问题过滤：
  - 支持按照指定分支过滤，轻松配合合流等常见研发场景。
  - 支持使用代码注释过滤问题
  - .code.yml过滤文件
  - 可选过滤LFS文件或submodule相关问题。
  - 支持设置质量门禁，为开发过程设置锚点。



# AI修复建议

- 基于腾讯混元大模型的 AI 能力，支持多种主流编程语言。
- AI助手能够为格式不规范、安全漏洞、逻辑错误等各类代码问题提供建议，帮助提升代码质量。

```
String query = "SELECT * FROM access_log WHERE action LIKE '%" + action + "%'";

try (Connection connection = dataSource.getConnection()){
    try {
        Statement statement = connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY);
    }
}

// 修复后的代码
// 使用 PreparedStatement 替代 Statement
PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM access_log
WHERE action LIKE ?");
preparedStatement.setString(1, "%" + userInput + "%");
ResultSet results = preparedStatement.executeQuery();

ResultSet results = statement.executeQuery(query);
```

**【SQL 注入】规则描述: SQL注入漏洞**  
错误原因: 变量 'SQL value: SELECT \* FROM access\_log WHERE action LIKE '%<script>%'' 是用户可控的 (因为变量 'query' 的影响)

**AI 修复建议** 来自腾讯混元助手

**代码问题**  
在这段代码中, query 变量是由用户输入的数据构成的。这可能导致 SQL 注入漏洞。当用户输入包含恶意 SQL 语句的数据时, 这些语句可能会被执行, 从而导致数据泄露或者其他安全问题。

**修复建议**  
为了修复这个问题, 我们可以使用预编译的 SQL 语句 (PreparedStatement) 来避免 SQL 注入。预编译的 SQL 语句可以将参数与 SQL 语句分开, 从而避免恶意数据被插入到 SQL 语句中执行。

在这个修复后的代码中, 我们使用了 PreparedStatement 和 setString 方法来设置查询参数。这样可以确保用户输入的数据不会被解释为 SQL 语句的一部分, 从而避免了 SQL 注入漏洞。

**【len-as-condition 问题】规则描述:**  
错误原因: Do not use 'len(SEQUENCE)' without comparison to determine if a sequence is empty

**AI 修复建议** 来自腾讯混元助手

**代码问题**  
在代码中, 使用了 len(lint\_detail.get('issues', [])) 来判断 lint\_detail.get('issues', []) 是否为空。这种写法可能会导致性能问题, 因为 len() 函数会遍历整个序列来计算长度, 而我们只需要判断序列是否为空。

**修复建议**  
为了避免不必要的性能损失, 我们可以直接使用 if lint\_detail.get('issues', []) 来判断序列是否为空。如果序列为空, get() 方法会返回一个空列表, 而空列表在布尔上下文中被视为 False, 因此不会进入 if 语句。如果序列非空, get() 方法会返回实际的序列, 在布尔上下文中被视为 True, 因此会进入 if 语句。

**修复后的代码**

```
if lint_detail.get('issues', []):
    if lint_detail.get('issues') and len(lint_detail.get('issues', [])):
```

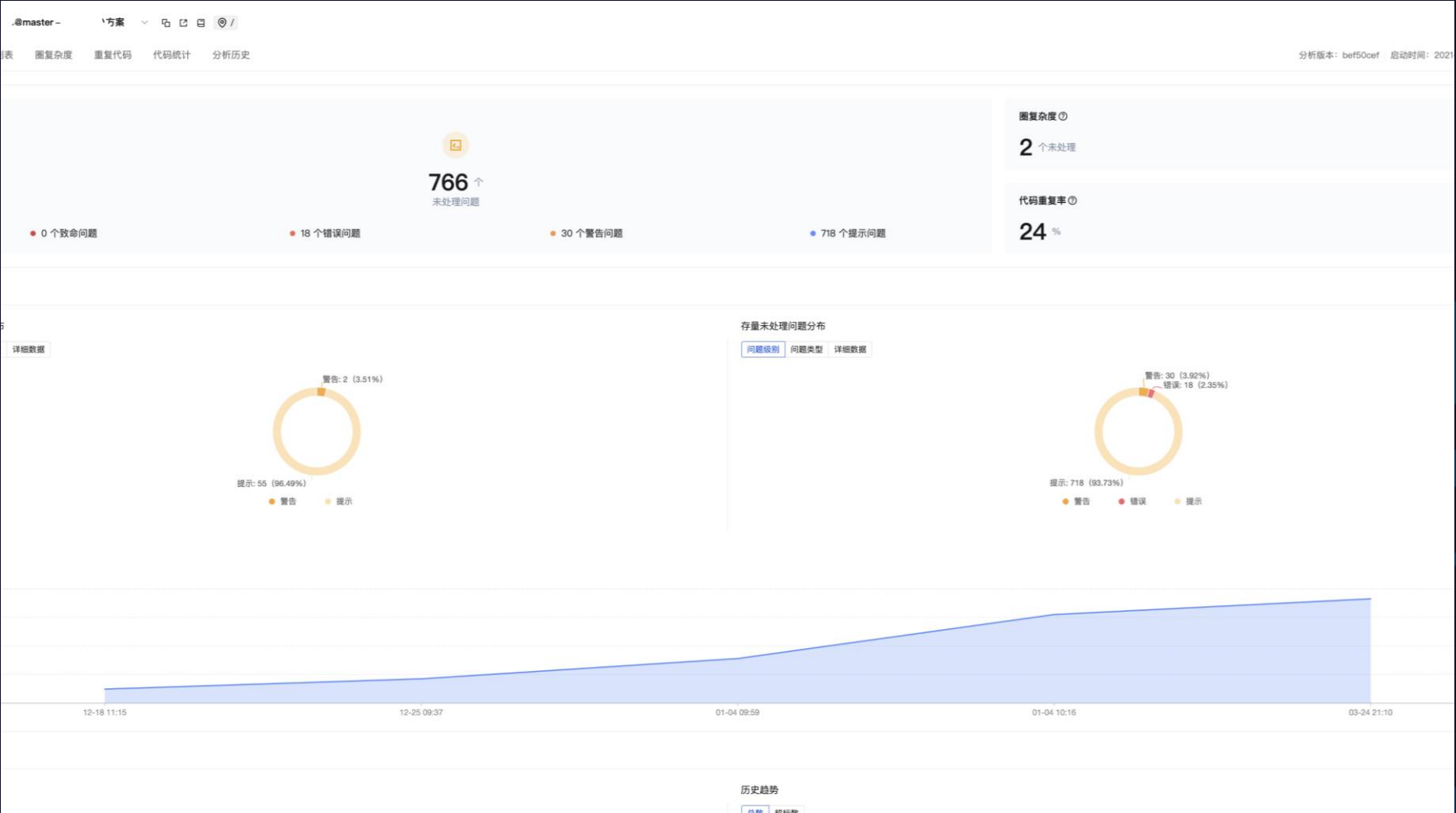




## 04 页面效果展示

04 UI pages

# 分析结果概览



问题列表

master

检查

启动分析

分析概览

问题列表

圈复杂度

重复代码

代码统计

分析历史

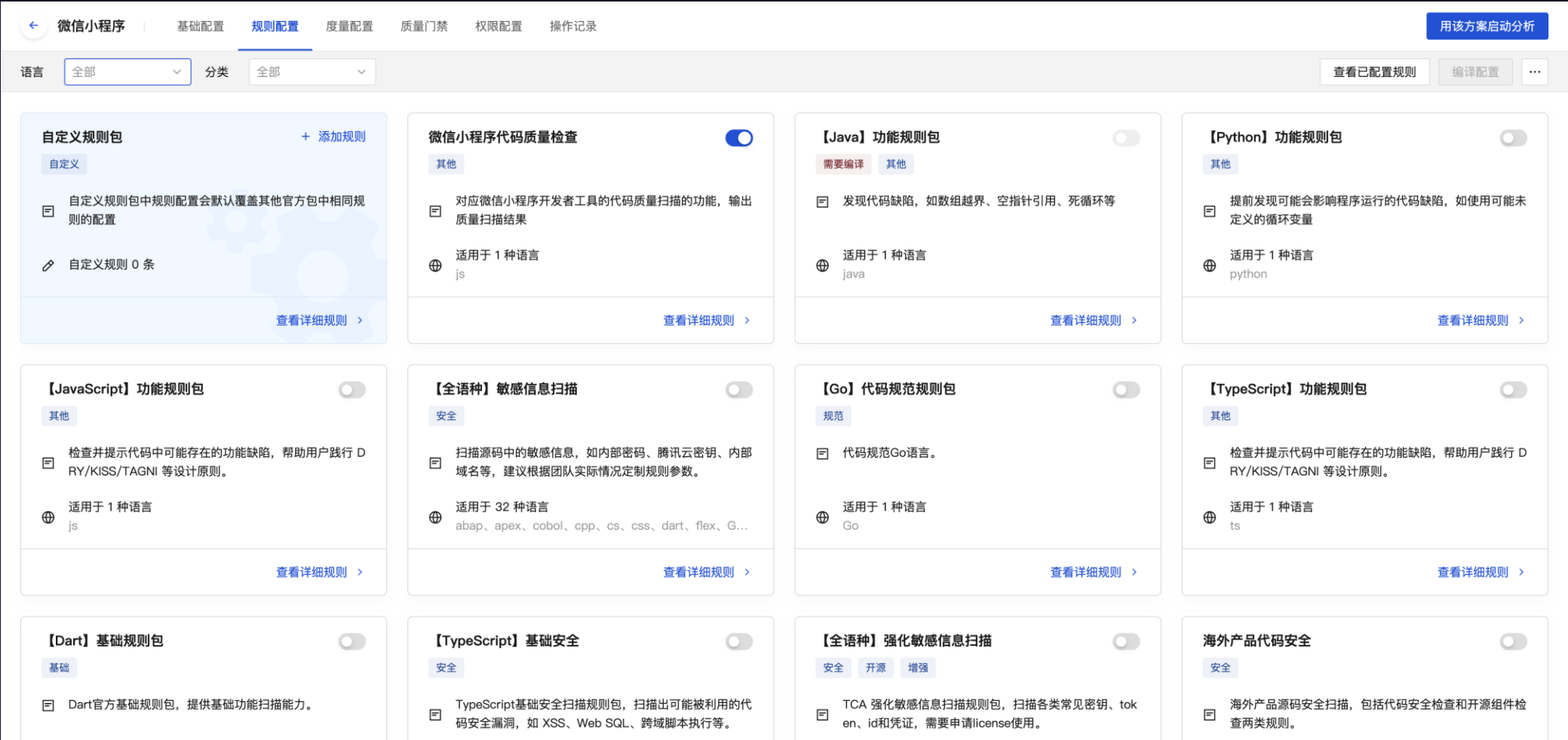
分析版本: 371fb78e 启动时间: 2023-09-11 15:15:46 成功

问题级别	全部	状态	未处理	解决方式	全部	责任人	请输入	发现问题分析ID	请输入	高级搜索	清空过滤					
所属文件	规则	引入版本	引入时间	问题级别	状态	责任人	<input type="checkbox"/>	dsvw.py	dsvw.py	subprocess-shell-true	错误信息: Found 'subprocess' function 'check_output' with 'shell=True'. This is dangerous because this call will spawn the command...	e31b59e9	2019-11-04 22:12	错误	未处理	
<input type="checkbox"/>	dsvw.py	dsvw.py	formatted-sql-query	错误信息: Detected possible formatted SQL query. Use parameterized queries instead.	5652fcfc	2015-11-23 18:45	警告	未处理								
<input type="checkbox"/>	dsvw.py	dsvw.py	use-defused-xml	错误信息: The Python documentation recommends using 'defusedxml' instead of 'xml' because the native Python 'xml' library is vulnerable...	b8459c7b	2020-05-11 00:13	错误	未处理								
<input type="checkbox"/>	dsvw.py	dsvw.py	sqlalchemy-execute-raw-query	错误信息: Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL injection, in order...	1634587e	2015-11-24 05:10	错误	未处理								

共 4 项数据

10 条/页 1

# 代码检查规则配置





## 05 典型案例展示

05 Typical cases

## 强化安全漏洞分析

Java  
强化安全规则包

### CmdInject

☆☆☆☆☆

- 检查代码中是否存在[命令行注入漏洞](#)。
- 当使用 `childprocess` 等模块执行命令时，拼接了用户可控的输入，会导致命令执行漏洞。攻击者利用漏洞可以控制目标主机或者容器。
- 修复建议：需要评估 `childprocess` 等模块执行命令的使用，应限定或校验命令和参数的内容。

### PathTraversal

☆☆☆☆

- 检查代码中是否存在[路径穿越漏洞](#)。
- 操作文件时，应该限定文件的路径范围，如果拼接用户输入到文件路径，可能导致路径穿越漏洞。攻击者利用漏洞可以访问到文件系统上的任意文件，这可能导致信息泄漏等问题。
- 修复建议：按业务需求，使用白名单限定后缀范围，校验并限定文件路径范围。

### SQLInject

☆☆☆☆☆

- `SQLInject` 规则用于检查代码中是否存在[SQL注入漏洞](#)。
- 错误的拼接用户可控的值到 SQL 语句，可能导致 SQL 注入漏洞。攻击者可以修改 sql 语法来更改查询的目标或结果，泄露数据库敏感信息，也可以使用 SQL 文件操作攻击底层 Web 服务器。如果使用该 SQL 查询进行授权认证，攻击者还可以用于提权。
- 修复建议：SQL 语句默认使用预编译并绑定变量，使用安全的 ORM 操作。

### SSRF

☆☆☆☆

- 检查代码中是否存在[服务端请求伪造漏洞 SSRF\(Server-side request forgery\)](#)。
- 攻击者在未能取得服务器所有权限时，利用服务器漏洞以服务器的身份发送一条构造好的请求给服务器所在内网。
- 修复建议：限定访问网络资源地址范围，请求网络资源应加密传输。

### XSS

☆☆☆☆

- 检查代码中是否存在[跨站脚本攻击漏洞 XSS\(Cross-site scripting\)](#)。
- 如果 WEB 页面在动态展示数据时使用了用户的输入内容，没有对输入的内容过滤或者进行转义，黑客可以通过参数传入恶意代码，当用户浏览该页面时恶意代码会被执行。
- 修复建议：在输出所有用户可控的数据时，对数据做转义或者编码。

更多详情参考：[【Java】强化安全规则包](#)

```
1 void bad(HttpServletRequest req, HttpServletResponse resp){
2     String cmd = req.getParameter("cmd");
3     Runtime rt = Runtime.getRuntime();
4     rt.exec(cmd); // 触发规则
5 }
```

```
1 void bad(HttpServletRequest req, HttpServletResponse resp){
2     String image = req.getParameter("image");
3     File file = new File("resources/images/", image); // 触发规则
4
5     if (!file.exists()) {
6         return Response.status(Status.NOT_FOUND).build();
7     }
8
9     return Response.ok().entity(new FileInputStream(file)).build();
10 }
```

```
1 void bad(HttpServletRequest req, HttpServletResponse resp){
2     String id = req.getParameter("id");
3     Connection conn = null;
4     Statement statement = null;
5     ResultSet rs = null;
6
7     Class.forName("com.mysql.cj.jdbc.Driver");
8     conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sec_sql", "root", "root");
9     String sql = "select * from userinfo where id = " + id;
10    statement = conn.createStatement();
11    statement.executeUpdate(sql); // 触发规则
12 }
```

```
1 import org.springframework.context.annotation.Configuration;
2 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
3 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
4 import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
5
6 @EnableWebSecurity
7 @Configuration
8 public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
9     @Override
10    protected void configure(HttpSecurity http) throws Exception {
11        http
12            .csrf(csrf ->
13                csrf.disable() // 触发规则
14            );
15    }
16 }
```

```
1 void bad(HttpServletRequest req, HttpServletResponse resp){
2     String id = request.getParameter("id") != null ? request.getParameter("id") : null;
3     Doc doc = getDetailsById(id);
4     byte[] b = doc.getUploaded();
5     try {
6         response.setContentType("APPLICATION/OCTET-STREAM");
7         String disHeader = "Attachment;Filename=" + doc.getName();
8         response.setHeader("Content-Disposition", disHeader);
9         ServletOutputStream out = response.getOutputStream();
10        out.print(b); // 触发规则
11    }
12 }
```

# 代码质量缺陷分析



- 检查数组越界的情况。不正确的缓存区访问可能损坏内存，导致程序崩溃或读取到权限外的内存。
- 检查 strcpy, strcat 字符串复制/拼接过程中长度不当导致的溢出，同样 gets scanf 函数也视为不安全。
- 如果发现多线程中某个全局变量在未持有锁便更新时，则会上报错误。
- 如果发现文件内存在 mtx1 -> mtx2 的上锁顺序时，另存在 mtx2 -> mtx1 的上锁顺序，视为死锁或存在死锁的可能，则会上报错误。死锁发生时程序将会卡死无法正常执行。
- 在程序申请了资源但并未按时释放时上报错误 目前场景包括：句柄打开时未关闭，指针分配内存后没有及时释放。
- .....

# 云应用接入扫描

### 使用场景

- 公共节点资源池无法满足日常分析需要
- 需要使用编译型工具进行代码分析
- 对代码安全有更高要求

### 使用优势

- 打通了云官网与云资源，支持一键购买云服务器并安装启动客户端
- 无需额外部署，无需登录机器，大幅降低接入成本
- 方便团队专机资源一键接入，自主管控机器资源和编译环境

### 使用步骤

1. TCA 中创建标签与节点
2. 安装云应用：<https://console.cloud.tencent.com/cloudapp/install/detail/pkg-a0lpnk8s>
  - 选择购买的机器配置：可自由灵活配置
  - 填写客户端节点信息和启动命令：从节点的启动命令中获取
  - 安装应用，打开应用并等待节点接入
3. TCA 中确认节点状态已在线
4. 云应用已接入，正式开始代码分析



# 本地快速扫描

### 使用场景

本地开发过程中，可以对本地代码目录下的临时代码（未关联scm仓库或未提交到scm仓库的本地代码）进行扫描，对某个目录或某些文件进行快速扫描，产出本地扫描结果。

注：该模式不与代码仓库关联，只对给定的目录或文件进行扫描，不依据版本号做增量分析，也不定位问题责任人。

### 使用步骤

1. 在页面上创建分析方案模板，获取分析方案模板ID

该模式不与代码仓库绑定，因此不能直接使用分析方案，需使用分析方案模板，根据模板链接获取模板ID。

2. 初始化扫描需要的工具

进入客户端client目录，执行 quickinit 指令：

```
python3 codepuppy.py quickinit -t TOKEN --scheme-template-id SCHEME_TEMPLATE_ID --org-sid ORG_SID
```

3. 执行快速扫描

进入客户端client目录，执行 quickscan 指令：

```
python3 codepuppy.py quickscan -t TOKEN --scheme-template-id SCHEME_TEMPLATE_ID --org-sid ORG_SID -s SOURCE_DIR --file FILE
```

# 在Jenkins中使用代码分析

1. 获取代码分析的Jenkins插件
2. 在TCA中创建团队和项目
3. 在Jenkins中上传插件文件，以安装代码分析插件
4. 创建Jenkins任务，配置需要代码库地址和下载凭证
5. 配置Jenkins插件相关参数
  - 在构建任务配置中选择【TCA】插件，配置项目代码路径、TCA团队ID、TCA项目名称、个人token、分支名称、分析语言、方案名称等参数。
6. 启动构建并查看分析结果
  - 点击【Build Now】启动构建任务，在【Console Output】中可以查看控制台执行过程，执行完成后，可在下方看到分析结果的链接。



# 欢迎您的咨询

技术支持 : [tca@tencent.com](mailto:tca@tencent.com)

商务合作 : [tca@tencent.com](mailto:tca@tencent.com)