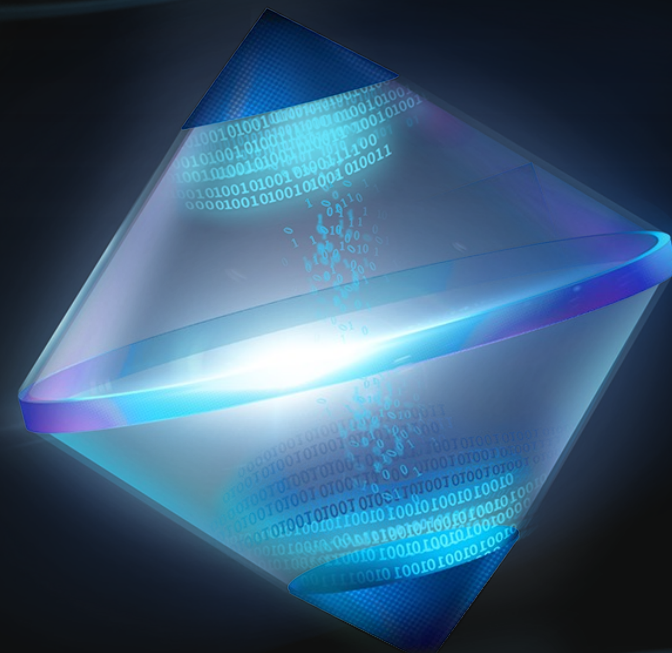


Tai-e: 基于程序分析的安全漏洞检测框架

演讲人：谭添 @ 南京大学



◆ 自我介绍

研究方向为程序分析与程序设计语言，对运用程序分析技术解决安全问题特别感兴趣。研究成果发表在TOPLAS、PLDI、OOPSLA、TOSEM、FSE、ISSTA等相关领域的CCF-A类国际顶级期刊与会议。

2019-至今

南京大学

助理/副研究员

2017-2019

奥胡斯大学
(丹麦)

博士后

2017

新南威尔士大学
(澳大利亚)

博士

2013

西北工业大学

本科

◆ 信息安全/网络安全/计算机安全



◆ 信息流安全

Top 10 Web Application Security Risks

1. **Injection.** Injection flaws, such as SQL, NoSQL, OS, and LDAP, occur when untrusted data is sent to an interpreter as part of a command or query. An attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication.** Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit implementation flaws to assume other users' identities temporarily or permanently.
3. **Sensitive Data Exposure.** Many web applications and APIs do not properly protect sensitive data such as credit card numbers, tax IDs, and other sensitive information.

2013-2019年间导致漏洞最多的原因^[2]

- Injection errors (No. 1), 11821, 4.6/day
- Information leaks (No. 4), 5086, 2.0/day



[1] The Open Web Application Security Project (OWASP), <https://owasp.org/>

[2] National Vulnerability Database, <https://nvd.nist.gov/>

报告内容

01

信息流安全

02

安全漏洞检测

03

Tai-e: 基于程序分析的
安全漏洞检测框架

报告内容

01

信息流安全

02

安全漏洞检测

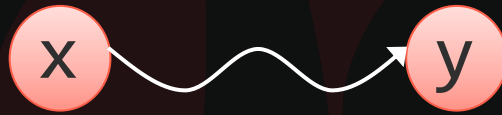
03

Tai-e: 基于程序分析的
安全漏洞检测框架

◆ 信息流* (Information Flow)

- 信息流：如果程序中变量 x 持有的信息被传播到变量 y ，则存在信息流 $x \rightarrow y$

- 一些信息流的例子



```
y = x;
```

```
a = x;  
b.f = a;  
c = b;  
y = c.f;
```

* Dorothy E. Denning and Peter J. Denning, “*Certification of Programs for Secure Information Flow*”. Communication of ACM 1977.

◆ 信息流安全 (Information Flow Security)

将信息流与安全联系起来

- 将程序中的变量划分成不同的安全等级
- 指定这些等级之间允许的信息流，即安全策略

◆ 安全等级 (Security Level)

最基础（可能也是最常用）的模型是二级分类，即一个变量可以被分为两种级别：

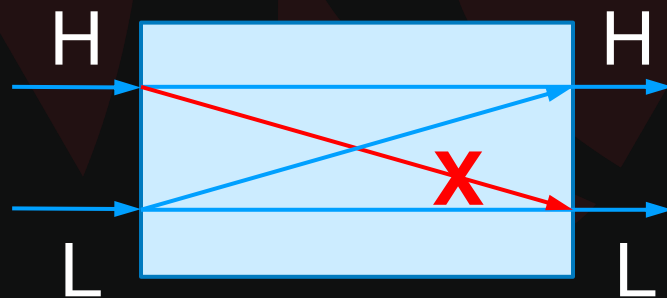
1. H: 表示**高**密级，如保密信息
2. L: 表示**低**密级，如公开信息

- `h = getPassword(); // h 为高密级`
- `broadcast(1); // 1 为低密级`

◆ 安全策略 (Security Policy)

- 限制不同安全等级之间的信息流
- 与二级分类搭配，最常见的策略：

禁止高密级变量的信息流入低密级变量



◆ 信息流安全 (Information Flow Security)

将信息流与安全联系起来

- 将程序中的变量划分成不同的安全等级
- 指定这些等级之间允许的信息流，即安全策略



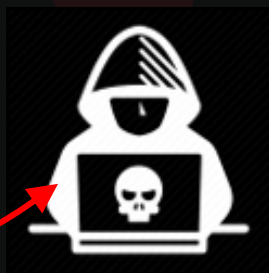
◆ 信息流安全

保密性 (Confidentiality)

- 阻止保密信息的泄露
- 安全等级
 - 高：保密信息 (secret)
 - 低：公开信息 (public)
- 安全策略



secret



public

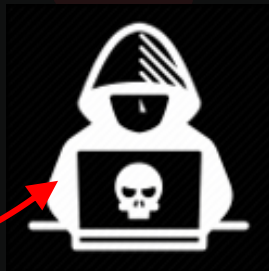
◆ 信息流安全：一体两面

保密性 (Confidentiality)

- 阻止保密信息的泄露
- 安全等级
 - 高：保密信息 (secret)
 - 低：公开信息 (public)
- 安全策略



secret



public

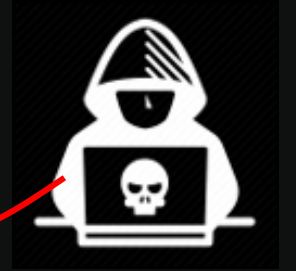


完整性 (Integrity)

- 阻止不可信的信息污染关键信息
- 安全等级
 - 高：关键信息 (critical)
 - 低：不可信信息 (untrusted)
- 安全策略



critical



untrusted



◆ 完整性 (Integrity)

- 阻止不可信的信息污染关键信息¹

```
x = readInput(); // untrusted
cmd = "... " + x;
execute(cmd); // critical
```

- 注入攻击 (2013-2019年间导致漏洞最多的原因²)
 - 命令注入攻击
 - SQL注入攻击
 - 跨站脚本攻击 (XSS)
 - ...

1. Ken Biba, “*Integrity Considerations for Secure Computer Systems*”. Technical Report, ESD-TR-76-372, USAF Electronic Systems Division, Bedford, MA, 1977.
2. National Vulnerability Database, <https://nvd.nist.gov/>

◆ 信息流安全：一体两面

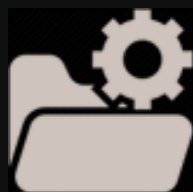
保密性 (Confidentiality)

对偶

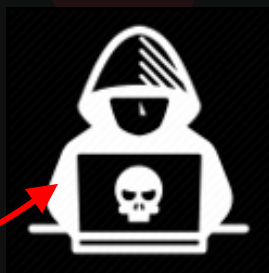
完整性 (Integrity)

- 阻止保密信息的泄露
- 安全等级
 - 高：保密信息 (secret)
 - 低：公开信息 (public)
- 安全策略

- 阻止不可信的信息污染关键信息
- 安全等级
 - 高：关键信息 (critical)
 - 低：不可信信息 (untrusted)
- 安全策略



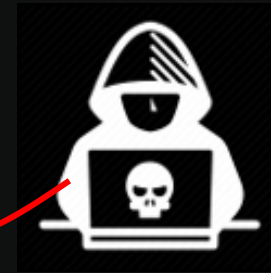
secret



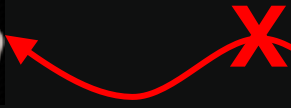
public



critical



untrusted



报告内容

01

信息流安全

02

安全漏洞检测

03

Tai-e: 基于程序分析的
安全漏洞检测框架

◆ 安全漏洞检测

- 目标：检测违反安全策略的信息流
- 代表性技术
 - 类型系统 (type system)
 - 程序分析 (program analysis)

◆ 基于类型系统的安全漏洞检测

- 扩展类型系统，在类型中加入安全相关的信息
- 使用类型推导技术检测安全漏洞
- 代表性工具：Jif*

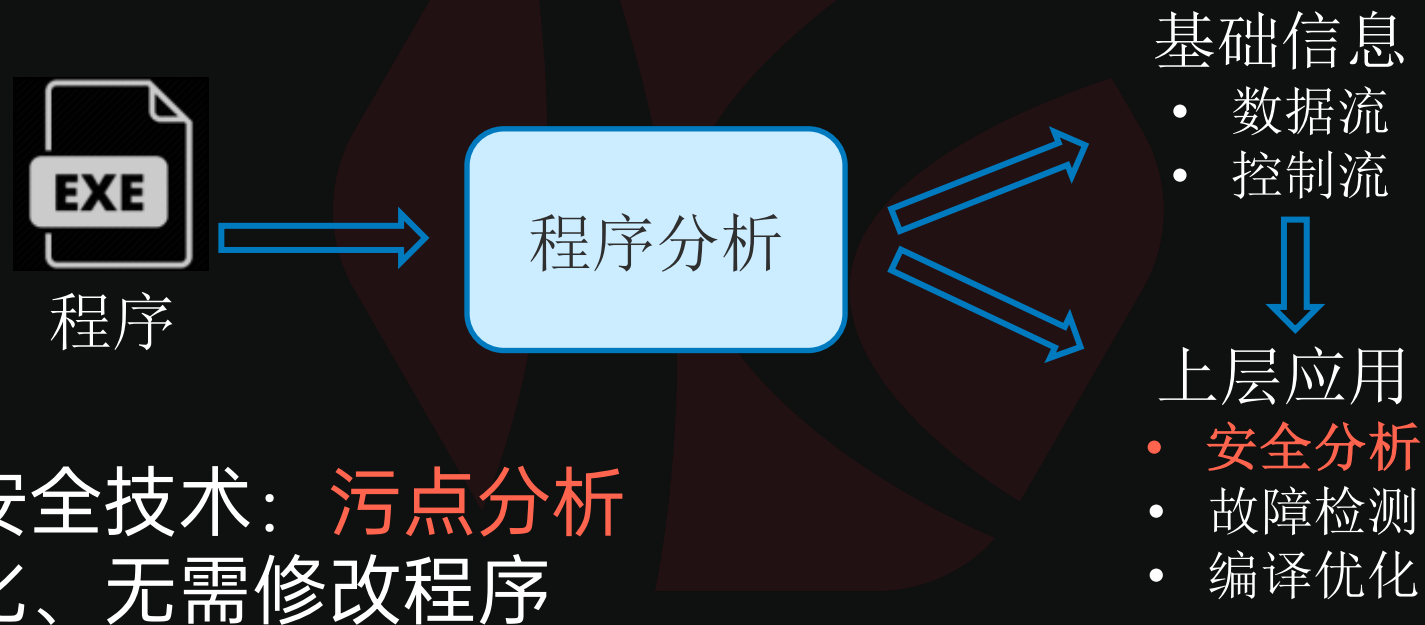
```
// Alice 控制 x 的信息
int {Alice->Bob} x;           // x 对 Bob 可见
int {Alice->Bob, Chuck} y;   // y 对 Bob、Chuck 可见
x = y; // OK: 对 x 的策略比 y 更严格
y = x; // BAD: 导致 x 对 Chuck 可见
```

- 缺点
 - 需要程序员编写大量类型标记 (label)
 - 无法用于已有软件

* <https://www.cs.cornell.edu/jif/>, Java information flow.

◆ 基于程序分析的安全漏洞检测

- 程序分析是自动分析程序运行时行为的技术，能够得出关于程序正确性、安全性、健壮性等属性的结论



- 代表性安全技术：污点分析
- 全自动化、无需修改程序

◆ 污点分析 (Taint Analysis)

- 最常用的信息流分析技术
- 污点分析将程序中的数据分为两类：
 - 感兴趣的数据，这些数据会被打上标记，称为污点数据
 - 其它数据
- 污点数据的来源称为sources。实际应用当中，污点数据通常来自某些方法（即sources）的返回值
- 污点分析追踪污点数据在程序中的流动，并观察它们是否会流入特定的程序位置（称为sinks）。实际应用当中，sinks通常为一些敏感的方法



◆ 污点分析：两种应用

污点分析可以检测两种信息流安全漏洞

• 保密性

- [高安全等级] Source: 保密数据的来源
- [低安全等级] Sink: 数据出口
- 对应信息泄露

```
x = getPassword(); // source
y = x;
log(y); // sink
```

• 完整性

- [低安全等级] Source: 不可信数据的来源
- [高安全等级] Sink: 关键计算
- 对应注入攻击

```
x = readInput(); // source
cmd = "... " + x;
execute(cmd); // sink
```

关键在于追踪污点数据在程序中的流动

◆ 信息流安全：一体两面

保密性 (Confidentiality)

对偶

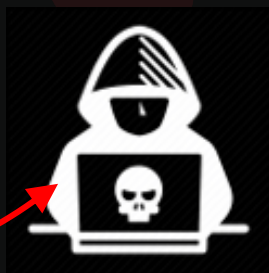
完整性 (Integrity)

- 阻止保密信息的泄露
- 安全等级
 - 高：保密信息 (secret)
 - 低：公开信息 (public)
- 安全策略

- 阻止不可信的信息污染关键信息
- 安全等级
 - 高：关键信息 (critical)
 - 低：不可信信息 (untrusted)
- 安全策略



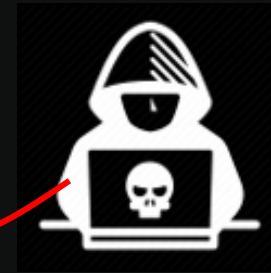
secret



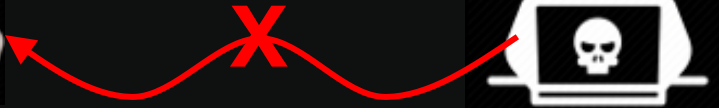
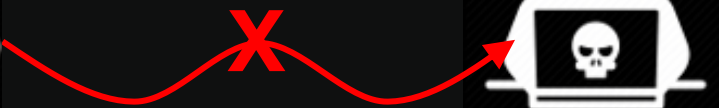
public



critical



untrusted



◆ 污点分析 (Taint Analysis)

- 动态污点分析
 - 运行时进行污点分析
 - 会造成显著的运行时开销
- 静态污点分析
 - 无需运行程序，无运行时开销
 - 尽早发现安全漏洞

报告内容

01

信息流安全

02

安全漏洞检测

03

Tai-e: 基于程序分析的
安全漏洞检测框架

◆ Tai-e: 通用型 Java 程序分析框架

The screenshot shows the GitHub repository page for 'Tai-e' by 'pascal-lab'. At the top, there is a navigation bar with a menu icon, the repository name 'pascal-lab / Tai-e', a yellow badge that says '开源刚满一年' (Open source just turned one year), a search icon, a plus icon with a dropdown arrow, a refresh icon, a pull requests icon, and a notification icon. Below the navigation bar, there is a secondary navigation bar with icons and labels for 'Code', 'Issues' (with a '2' badge), 'Pull requests', 'Actions', 'Wiki', 'Security', and 'Insights'. The main content area features the repository name 'Tai-e' with a 'Public' label, and several action buttons: 'Edit Pins', 'Unwatch' (with a '15' badge), 'Fork' (with a '93' badge), and 'Starred' (with a '688' badge). Below these buttons, there are more options: 'master' branch selector, 'Go to file', 'Add file', and a green 'Code' button. To the right, there is an 'About' section with a gear icon and the text 'An easy-to-learn/use static analysis framework for Java'. At the bottom, there is a list of recent pull requests, with the first one by 'zhangt2333' titled 'Fix key duplication wh...' (with a truncated title), marked as merged (green checkmark), from 'last week', and having '1,879' views. There are also three topic tags: 'java', 'security', and 'static-analysis'.

Navigation: pascal-lab / Tai-e

Badge: 开源刚满一年

Navigation: Code, Issues (2), Pull requests, Actions, Wiki, Security, Insights

Repository: Tai-e (Public)

Actions: Edit Pins, Unwatch (15), Fork (93), Starred (688)

Options: master, Go to file, Add file, Code

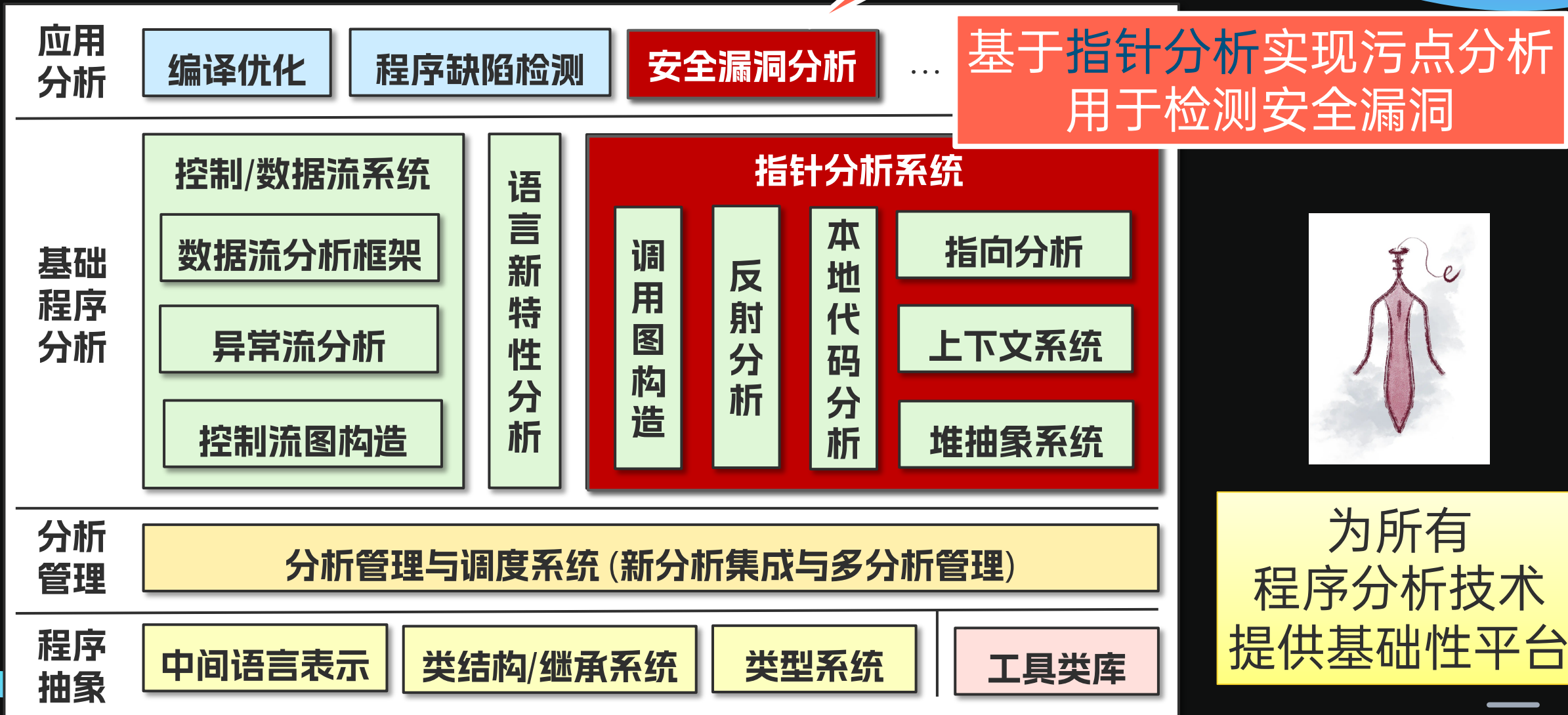
About: An easy-to-learn/use static analysis framework for Java

Recent Pull Request: zhangt2333 Fix key duplication wh... (merged, last week, 1,879 views)

Tags: java, security, static-analysis

◆ Tai-e: 通用型 Java 程序分析框架

Tai-e目前最重要的应用



为所有程序分析技术提供基础性平台

◆ 指针分析

- 最基础的程序分析技术之一（已有40+年历史）
 - 计算一个**指针**运行时能指向哪些**内存位置**
- 对于面向对象语言而言（如 Java）
 - 计算一个**变量/字段**运行时能指向哪些**对象**

◆ 一个指针分析的例子

“一个**指针**运行时能指向哪些**对象**？”

程序

input

指针分析

output

指向关系

```
void foo() {  
    A a = new A();  
    B x = new B();  
    a.setB(x);  
    B y = a.getB();  
}  
  
class A {  
    B b;  
    void setB(B b) { this.b = b; }  
    B getB() { return this.b; }  
}
```

Variable	Object
a	new A
x	new B
this	new A
b	new B
y	new B

Field	Object
new A.b	new B

◆ 指针分析 与 别名分析

- 两个紧密相关但并不一样的概念
 - 指针分析：一个指针指向哪些对象？
 - 别名分析：两个指针能否指向同一对象？

别名信息可从指针分析结果中导出

◆ Java 指针分析面临的挑战

- 速度
 - 现代程序规模日益庞大，显著影响指针分析的效率
- 精度
 - 复杂的程序导致结果中的指针会指向许多实际并不指向的对象
- 复杂语言特性
 - 反射、Lambda函数、本地代码调用等特性影响指针分析的完备性

◆ Java 指针分析的研究

- 我们深耕 Java 指针分析已有**十年**（本人博士毕业论文课题）
- 已在 **CCF-A** 类国际顶级会议/期刊发表多篇指针分析工作
- **PLDI'23**, **OOPLSA'21**, **TOPLAS'20**, **OOPLSA'18**, **FSE'18**, **PLDI'17**

国际顶级期刊，创刊41年来国内第三篇

国际顶级会议，2021年全球205篇投稿中的最高分

国际顶级会议，2023年全球录取83篇，国内唯一一篇独立单位完成

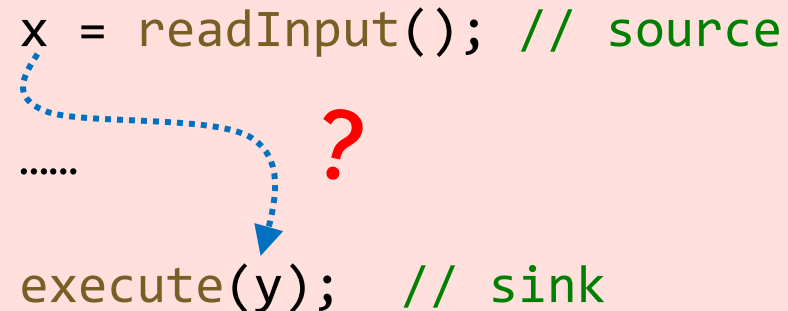
- 在该领域处于**国际领先**位置
- 目前**精度最高**与**速度最快**的 Java 指针分析技术都由我们提出

SO WHAT?

◆ 污点分析

- “污点数据能否流向sink？”

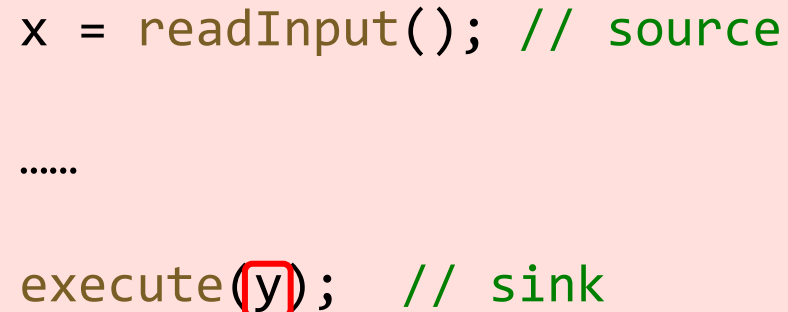
```
x = readInput(); // source
.....
execute(y); // sink
```

A diagram illustrating the flow of tainted data. A blue dotted line starts at the variable 'x' in the first line of code, moves down, then right, then down again, ending with a blue arrow pointing to the variable 'y' in the second line of code. A red question mark is placed above the arrow.

换一种问法

- “sink的指针能否指向污点数据？”

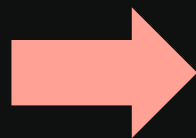
```
x = readInput(); // source
.....
execute(y); // sink
```

A diagram illustrating the pointer question. The code is the same as in the previous block. The variable 'y' in the second line of code is enclosed in a red square box.

y能否指向来自source
的污点数据？

◆ 污点/指针分析：手牵手，一起走

污点分析/指针分析 的本质是追踪
污点数据/对象 如何在程序中流动



污点分析与指针分析
一起运行

- 将污点数据视为 (artificial) 对象
- 将sources视为污点数据的创建点 (相当于 Java 中的 new 语句)
- 利用指针分析传播、追踪污点对象

指针分析

```
a = new A();  
x = a;  
c.f = x;  
d = c.f;  
...
```

Variable	Object
a	new A
x	new A
...	...

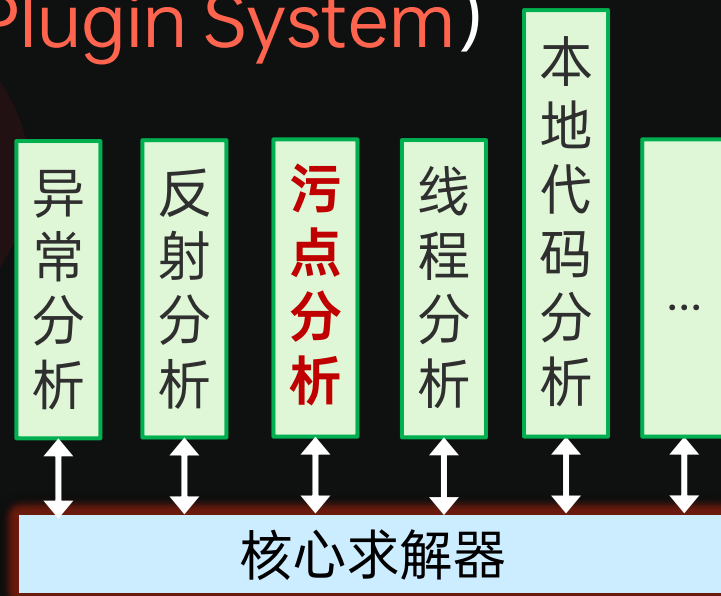
污点分析

```
a = source();  
x = a;  
c.f = x;  
d = c.f;  
...
```

Variable	Object
a	source()
x	source()
...	...

◆ Tai-e 指针分析框架

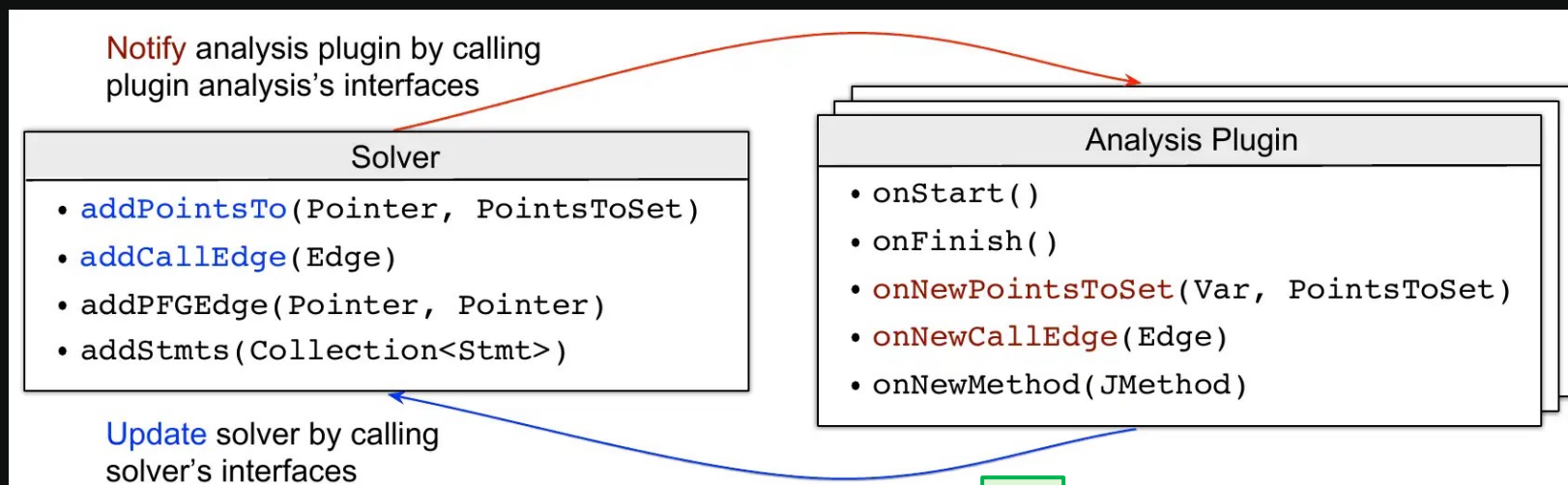
- 目前分析性能、分析完备性均为最优的指针分析框架*
- 为了辅助全程序分析的开发，我们在Tai-e中引入了一套简洁、有效的机制，称为**分析插件系统**（**Analysis Plugin System**）
- 我们将指针分析框架设计成
 - 一个**指针分析核心求解器**，以及
 - 一系列**分析**，与求解器进行交互
- 每个分析被称为一个**分析插件**（实现**Plugin**接口）



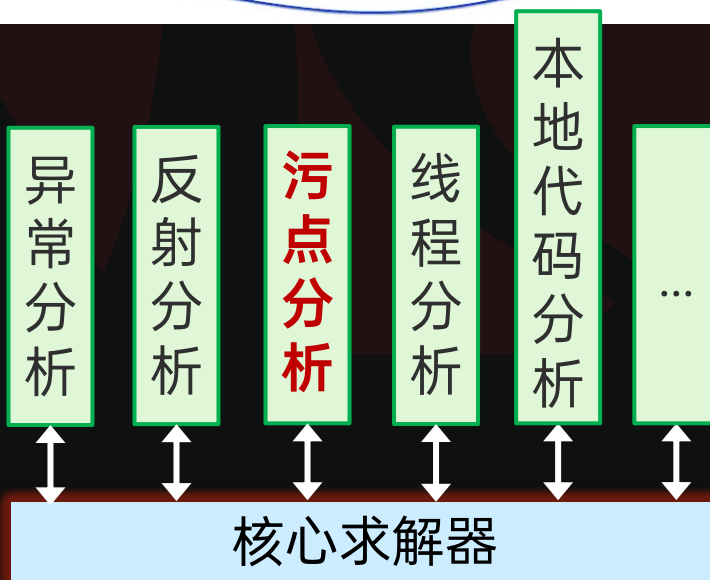
* Tian Tan and Yue Li, “*Tai-e: A Developer-Friendly Static Analysis Framework for Java by Harnessing the Good Designs of Classics*”. ISSTA 2023.

◆ Tai-e 指针分析框架

指针分析求解器与分析插件通过互相调用 **Plugin** 与 **Solver** 的 API 进行交互



- 好比“操作系统”
- 提供基础指针分析传播、追踪数据流动的功能



- 好比“应用程序”
- 基于核心求解器提供的功能实现各种分析

◆ 为什么基于指针分析实现污点分析？

• 污点分析面临的挑战

• 速度

- 现代程序规模日益庞大，显著影响指针污点分析的效率

• 精度

- 复杂的程序导致结果中的指针会指向许多实际并不指向的对象污点数据

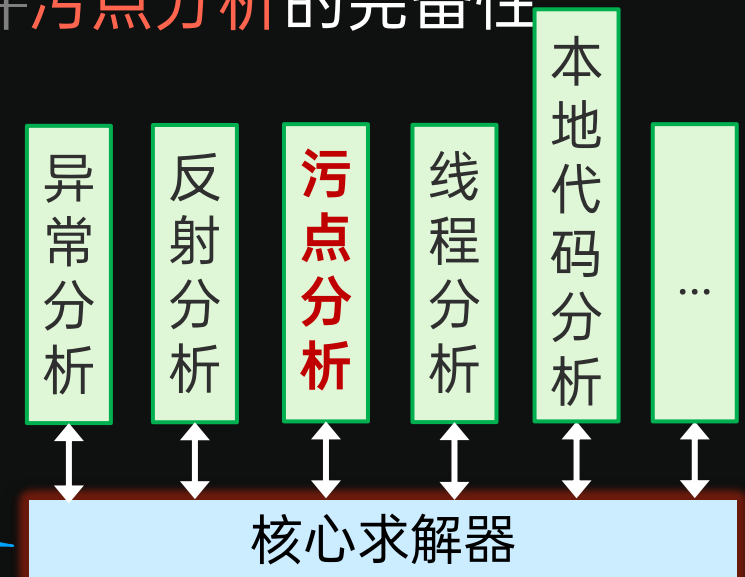
• 复杂语言特性

- 反射、Lambda函数、本地代码调用等特性影响指针污点分析的完备性

• 借助指针分析的技术，解决污点分析的困难

• 显著简化污点分析的实现复杂度与开发难度

内置追踪数据流的功能，以及
各类提升分析精度、速度的技术



◆ 为什么使用 Tai-e?

与其它通用型程序分析框架相比，Tai-e 有三大特色：

• 易于上手的教学平台

- 在南京大学开设《软件分析》课程（B站26万+播放）
- 专为程序分析教学精心设计“Tai-e教学版”及一系列实验

• 20+所高校教师（授课使用）

北大、上交、浙大、中科大、华中科大等

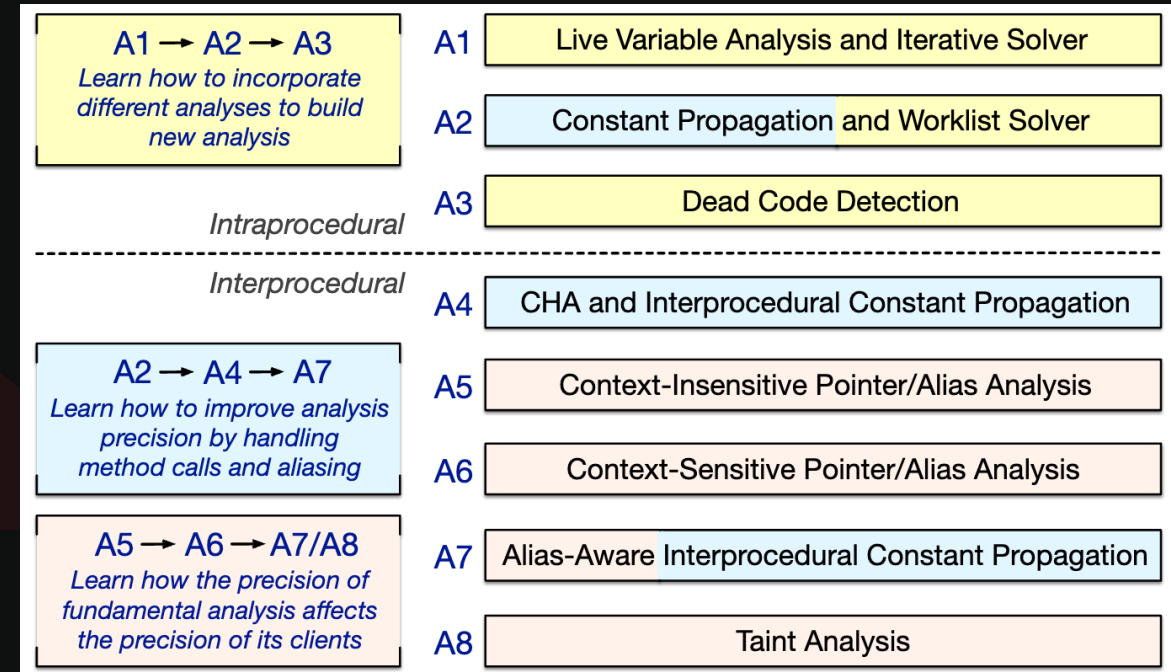
• 10+企业相关研发部门主管

（培训工程师使用）

华为、阿里、腾讯、字节、英伟达等

• 140+高校学生（学习使用）

清华、北大、华盛顿大学、宾西法尼亚大学、美国西北大学、普渡大学、多伦多大学、滑铁卢大学、东京大学、阿姆斯特丹大学等



◆ 为什么使用 Tai-e?

与其它通用型程序分析框架相比，Tai-e 有三大特色：

• 易于上手的教学平台

- 在南京大学开设《软件分析》课程（B站26万+播放）
- 专为程序分析教学精心设计“Tai-e教学版”及一系列实验
- 140+高校的学生、20+高校的学者、10+企业使用 Tai-e 学习/授课/培训

• 研发团队具备国际一流的研究能力（程序分析领域）

- PLDI'23, ISSTA'23, OOPSLA'21, TOPLAS'20, OOPSLA'18, FSE'18, PLDI'17, ...
- 把最先进的学术成果/技术融入 Tai-e

• 对开发者友好的设计

- Tai-e 的核心设计目标（参考对比多个国际上流行的 Java 分析框架）
- Tian Tan and Yue Li, “*Tai-e: A Developer-Friendly Static Analysis Framework for Java by Harnessing the Good Designs of Classics*”. ISSTA 2023.

◆ 实践案例 — Log4Shell

- Tai-e 污点分析可以快速检测出 Log4Shell 的完整注入路径
 - 跨越30+层方法调用，涉及50+节点



◆ Tai-e 未来计划



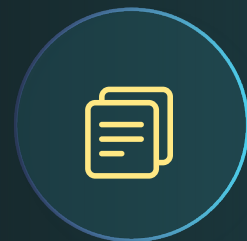
增强安全分析

开发工具集成



支持Android

丰富文档教程

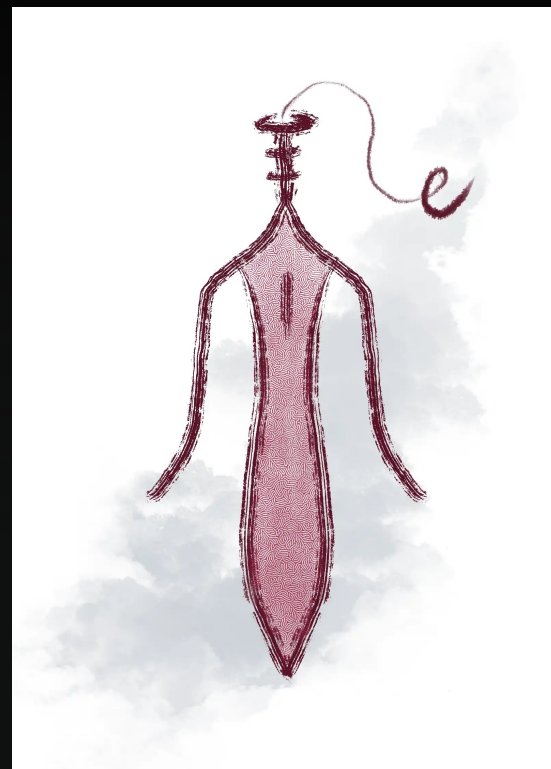


◆ 总结

- 信息流安全是安全领域最重要的问题之一
 - 每年有大量的信息泄露/注入攻击被发现
- 污点分析是检测信息流安全漏洞的有效手段
- 我们基于 Tai-e 框架及其指针分析系统实现了有效的污点分析
 - 污点分析可借助指针分析传播/追踪数据的能力
 - 直接获益于 Tai-e 中的高级指针分析技术带来的精度、速度、完备性
 - 简洁的污点分析实现，易于理解并进行二次开发

感谢您的观看！

THANK YOU FOR YOUR WATCHING



<https://tai-e.pascal-lab.net/>

谭添
南京大学

