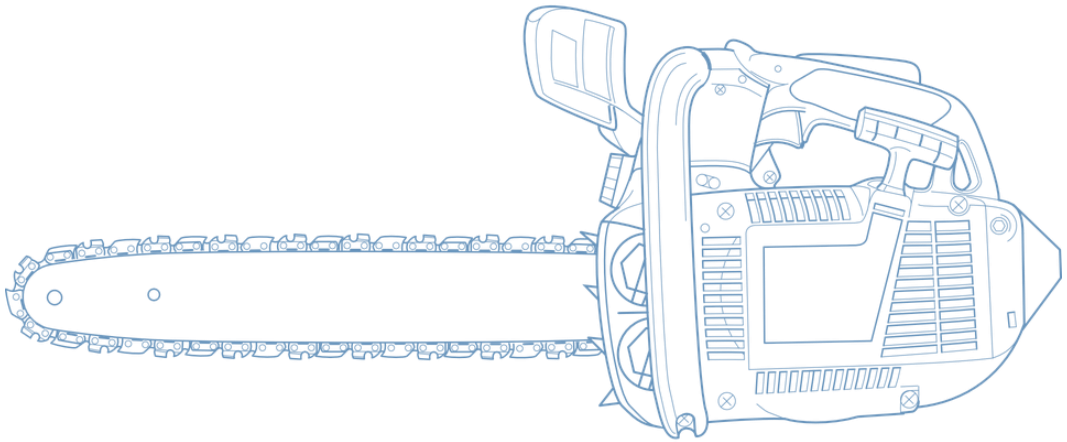
Chained Automated Workflow-based Exploit Generation
基于工作流的自动化链式漏洞利用生成工具

谭兴邦（PolluxAvenger）

2016 年 12 月 15 日



CONTENTS

1

RELATED PRESENTATIONS 相关工作

1 相关工作 RELATED PRESENTATIOS

二进制程序的攻击生成

- 基于补丁差异性的攻击生成技术
- 二进制程序的控制流劫持
- 与数据流关联的数据导向攻击

Web 应用程序的攻击生成

- Ardilla 用符号执行与污点跟踪来构建攻击向量
- CraxWeb 辅助约束求解器来构建攻击向量
- QED 利用静态分析来分析 Java Web 应用程序
- EKHunter 结合静态分析与约束求解

- 代码覆盖率低
- 攻击库简单
- 污点跟踪误报高
- 跨平台
- 无法生成二段攻击

脆弱性分析

- 静态分析结合动态分析的混合分析
- 修正过滤函数引起的错误
- MiMoSA 针对数据和工作流进行攻击
- 权限控制漏洞与数据库模型分析

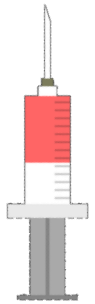
- 对源码要求高
- 不分析数据库
- 无法生成二段攻击

2

PROBLEM BACKGROUND 问题背景

2 问题背景 PROBLEM BACKGROUND

手动渗透测试



易部署

覆盖率低

漏报率高

基于路径的
自动程序分析

静态分析易误报

识别可能存在漏洞

复杂的语言特性

静态分析

- 路径选择
  - 选择哪条路径才能用最小代价发现最多漏洞
- 复杂的工作流
  - 预期内工作流的复杂跳转
  - 预期外任意请求对执行流的跳过与绕过

攻击生成

- 安全过滤
  - 内置过滤函数处的安全性判断
- 复杂的数据库状态
  - 数据库中存储的值影响执行路径
  - 应用与数据库之间数据的双向流动

2 问题背景 PROBLEM BACKGROUND

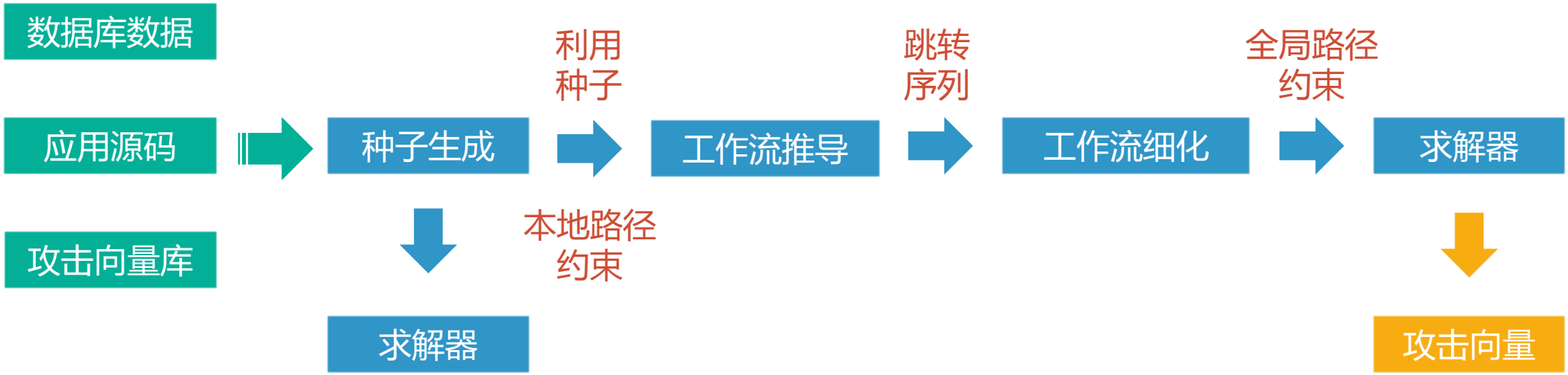
基于 PHP 的
Web 应用程序源代码

一系列可成功的
攻击向量的集合

3

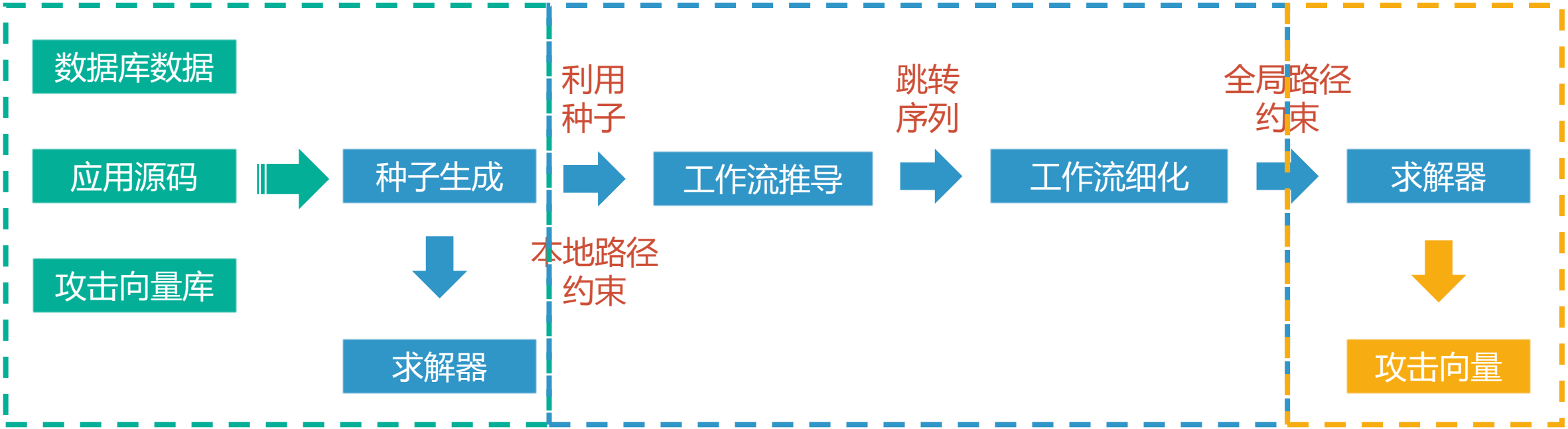
SOLUTION METHOD 解决方法

3 解决方法 SOLUTION METHOD

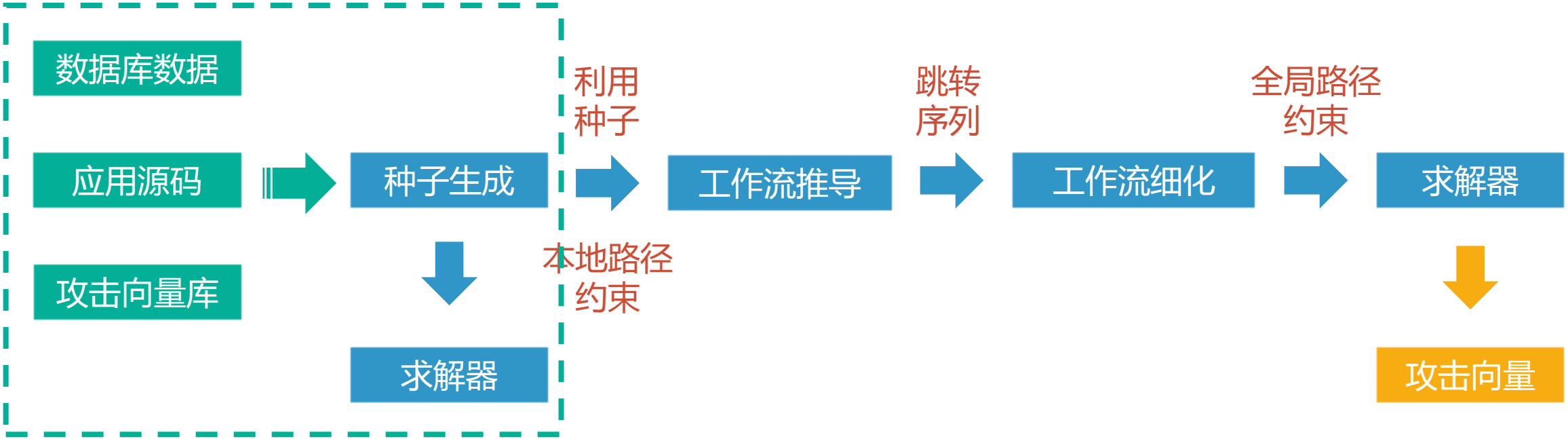


执行流程架构

3 解决方法 SOLUTION METHOD



3 解决方法 SOLUTION METHOD



种子生成难题

3 解决方法 SOLUTION METHOD

主要目的：确定可利用的敏感位置，排除无法利用的位置

exploit seed 是一个键值对（*S, I*），其中：

S 是敏感位置　I 是包含 S 的模块接收的键值对集合

$$I = \{(i_1, v_1), \ldots (i_n, v_n)\}$$

exploit seed 是执行可以直接触发漏洞的输入集合

3 解决方法 SOLUTION METHOD

模块内的每个敏感位置，Chainsaw 会尽量探索其全部执行路径

敏感位置： my_query( ) 与类似函数

针对每个敏感位置都会建立符号执行表达式与符号公式

符号执行表达式为静态脆弱点（常量）与动态脆弱点（变量）构建

符号公式包括： 路径条件
沿路径的输入转换
上下文引起的约束条件

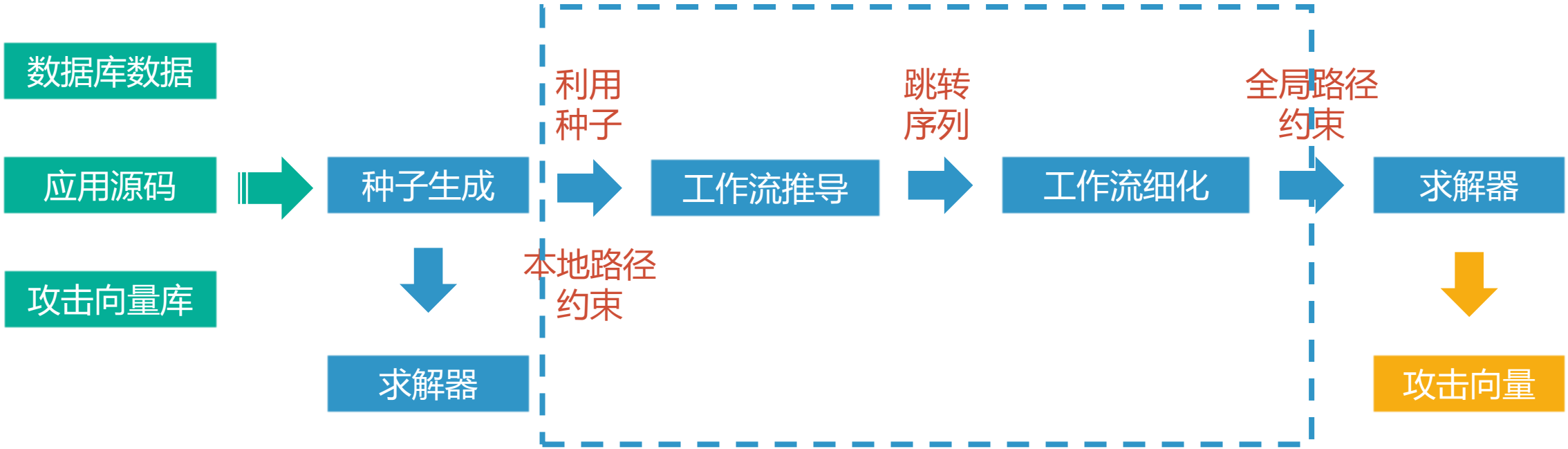
3 解决方法 SOLUTION METHOD

```
if(!isset($_SESSION['username']))
    header( "Location: ./login. php" );
else if(isset($_SESSION['username']) &&
isset($_SESSION['room_name']))
{
    $room_name = $_SESSION['room_name'];
    $sql = "SELECT room_name, level FROM ROOM_TABLE WHERE
room_name='$room_name'";
    $result = mysql_query($sql);
    $room_row = $db->sql_fetchrow($result);
    $accesslevel = $room_row['level'];
}
if($accesslevel == 1)
{
    if(isset($_GET['cat_desc']))
    {
        $cat_desc = htmlspecialchars($_GET['cat_desc']);
        $sql = "SELECT cat_desc FROM CAT_TABLE WHERE
cat_desc='$cat_desc'";
        $result = mysql_query($sql);
    }
}
```

F_p: `isset($_SESSION['username']) ^ isset($_SESSION['room_name']) ^ $room_name==$_SESSION['room_name] ^ $accesslevel== 1 ^ isset($_GET['cat_desc']) ^ $cat_desc==htmlspecialchars($_GET['cat_desc'])`

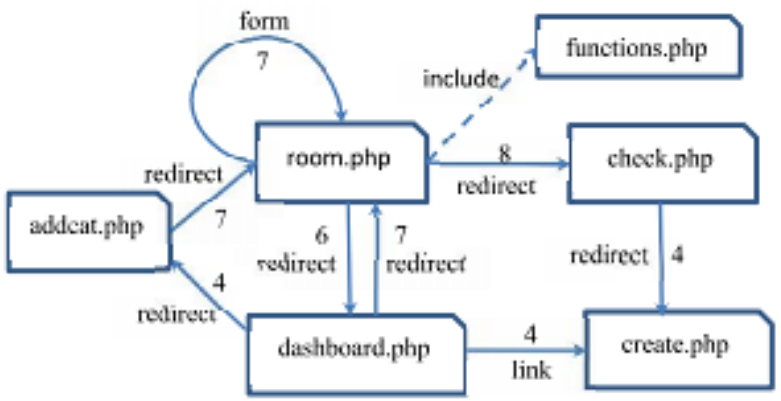
F_a: `$cat_desc=="foo' OR '1'=='1"`

3 解决方法 SOLUTION METHOD



工作流跳转难题

3 解决方法 SOLUTION METHOD



HTTP links, forms, meta tags
iframes, PHP redirection

工作流推导的输出是排序过的跳转序列

工作流细化的输出是满足约束的全局路径

room.php -> create.php

① room.php -> dashboard.php -> create.php

② room.php -> check.php -> create.php

① 6 + 4 = 10

② 8 + 4 = 12

10 < 12 ➡ 优选第一条路径

3 解决方法 SOLUTION METHOD

如何解决？

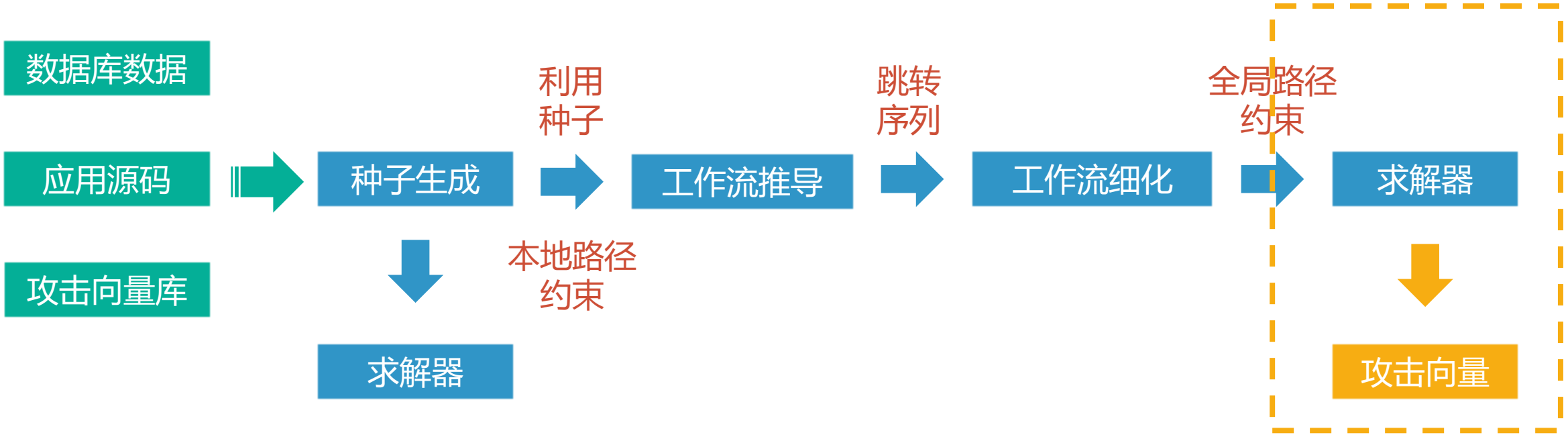
Chainsaw 最大的瓶颈在于对每个 exploit seed 都需要考虑全局执行路径

最大限度的减少 RWFG 边的数量

如果一个本地执行路径 A_i 不能与另一个模块的本地执行路径 L_i 建立联系
就将 RWFG 对应的边删掉

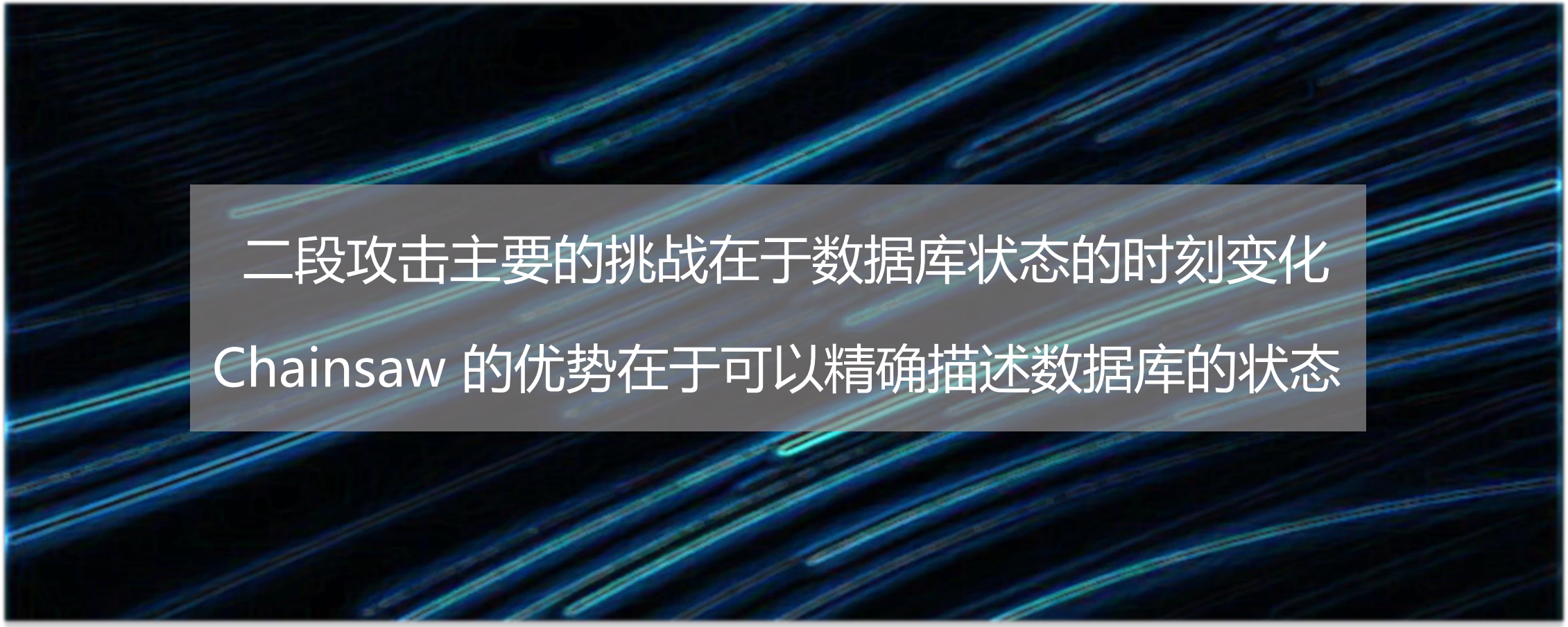
RWFG 中那些明显不可能的边的数量明显减少

3 解决方法 SOLUTION METHOD



二段攻击难题

3 解决方法 SOLUTION METHOD



3 解决方法 SOLUTION METHOD

静态输入生成

- 在分析期间建立写操作与读操作之间的映射
  - 写操作 F_p 关联
  - 约束 F_D 关联
  - F_p 与 F_D 交联送给求解器
- 没有的运行实体的条件限制

动态审计

- 可以使用现存系统中的数据库数据
  - DB-PHP 映射
  - 数据检索
  - 约束创建
- 不必重复多次数据来自数据库

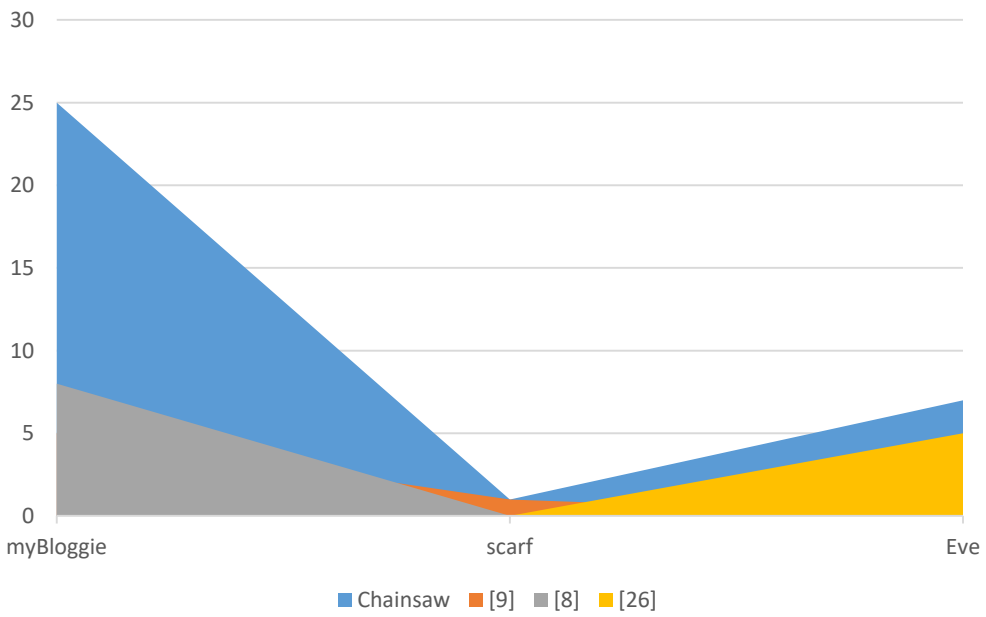
在数据非外部实体导入的情况下
静态与动态的区别只在开销上

4

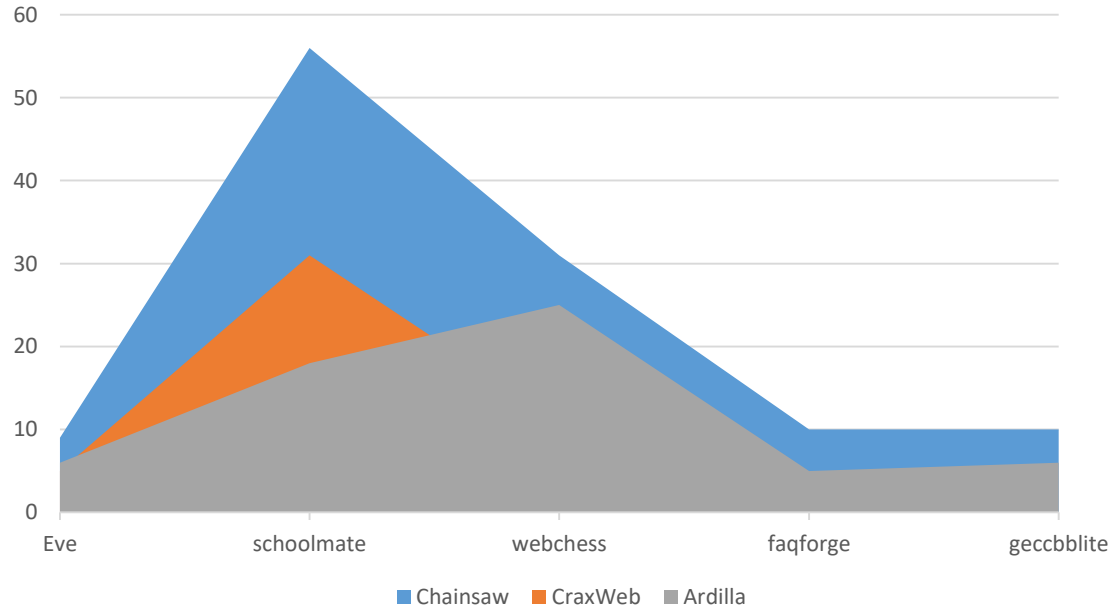
WORK PRODUCT 工作结果

4 工作结果 WORK PRODUCT

种子生成



攻击生成



4 工作结果 WORK PRODUCT

- 允许他人查看交流消息
- 允许向数据库中插入恶意域名
- 允许检索其他用户信息
- 动态审计模式效率更高
- 15% 的二阶占比
- 每次分析遍历上百万执行路径

5 EXISTENT SHORTAGE 存在不足

5 存在不足 EXISTENT SHORTAGE

1 静态分析的局限

- 应用中动态生成工作流无法分析
- SQL 语句非静态构建无法分析
- 少于 0.23% 的漏检率

2 不支持的 PHP 特性

- Pixy 不支持的控制流分析无法分析

3 求解失败

- 超时与返回未知结果无法分析
- 少于 1% 漏检率

6

LEARNING ADVANTAGES 创新之处

6 创新之处 LEARNING ADVANTAGES

源码分析来自 Pixy 和 TAPS　　规划求解来自 Z3 SMT 和 Z3-Str

1 上下文感知
- 评估每条敏感路径时创建两个语法分析器来对 SQL 和 HTML 构建抽象语法树

2 跳转结构分析
- 使用历史经验方案来解决遍历中的路径选择问题
- 从数据依赖与系统执行切片入手创建通用工作流图

3 数据库分析
- 加入对数据库额外限制的分析
- 存储过程因为样本集不存在就没有做

4 约束求解
- 解析 70 个最常用的内置 PHP 函数
- 增强了 Z3 对字符串的支持

THANKS