Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

面向网络应用程序的模糊测试

硕士研究生 邵思源

2024年5月26日

内容提要

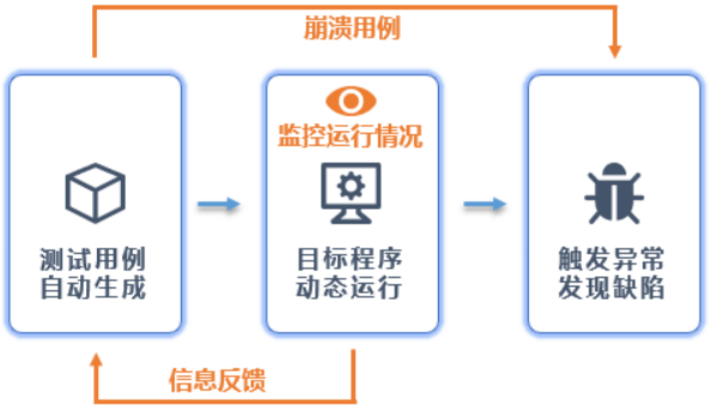
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - Witcher
  - Atropos
- 特点总结与工作展望
- 参考文献

预期收获

- 预期收获
  - 了解网络应用程序漏洞挖掘中的基本概念及分类
  - 理解网络应用程序模糊测试的特殊挑战
  - 理解前沿网络应用程序测试的技术原理与思路
  - 了解网络应用程序测试的技术发展前景

目标内涵

- 研究目标
  - 以网络应用程序为研究对象，面向OWASP TOP10等漏洞挖掘任务
  - 整合模糊测试、网络爬虫、虚拟化等相关技术
  - 提升对真实世界应用测试的覆盖率及缺陷触发能力
- 题目内涵解析
  - 网络应用程序：通过网络提供服务的软件。相较于传统的桌面应用程序，用户需通过网络浏览器或特定的网络客户端访问
  - 模糊测试：通过向测试对象提供大量畸形测试用例作为输入，并监视其错误响应，以揭示潜在的异常缺陷及安全漏洞



背景意义　网络应用程序生态面临的安全威胁与日俱增

- 研究背景
  - 发展规模：Web用户占比高达97.3% ⇨同时连续5年Web应用类漏洞占比第一
  - 技术演进：Web开发模式迭代更新，如框架、语言、组件等⇨攻击面扩大
  - 安全研究：依赖大量人力物力进行手工分析测试、缺陷分析⇨漏洞利用门槛高
- 研究意义
  - 安全生态：通过揭示并修复潜在漏洞可有效提升Web应用的可靠性与稳定性
  - 降低门槛：基于“一键测试”方案可大幅减少漏洞挖掘专业经验需求
  - 降本增效：降低安全服务成本，提高安全测试效率

LOG4J

通达OA
CVE-2024-1251

CVE-2024-5101 CVE-2024-4933
CVE-2024-5100 CVE-2024-4932
CVE-2024-5099 CVE-2024-4931
CVE-2024-5098 CVE-2024-4930
CVE-2024-5097 CVE-2024-1254

研究历史与现状　网络应用程序漏洞挖掘

2004 Huang等人首次提出静态匹配分析来挖掘Web应用代码缺陷

2011 Jovanovic 等人提出基于静态污点分析的静态Web测试

2016 Micheelsen等人提出利用静态控制流、数据流分析，挖掘Python应用中Sql注入、XSS等漏洞

2020 Google安全团队提出动态黑盒扫描工具Skipfish，基于爬虫和规则匹配技术构建应用地图进行漏洞匹配

2021 Eriksson等人提出了BlackWidow，基于网络爬虫动态构建应用逻辑有向图驱动模糊测试

2021 Rooij等人提出了Webfuzz，基于模型化图实现深度遍历网络接口引导模糊测试

2022 Zhao等人提出了Cefuzz，结合漏洞定位信息进行动态引导式变异测试，挖掘Php应用中的Rce漏洞

2023 Trickel等人提出了Witcher，基于灰盒覆盖率引导测试用例生成，挖掘Sql和命令注入漏洞

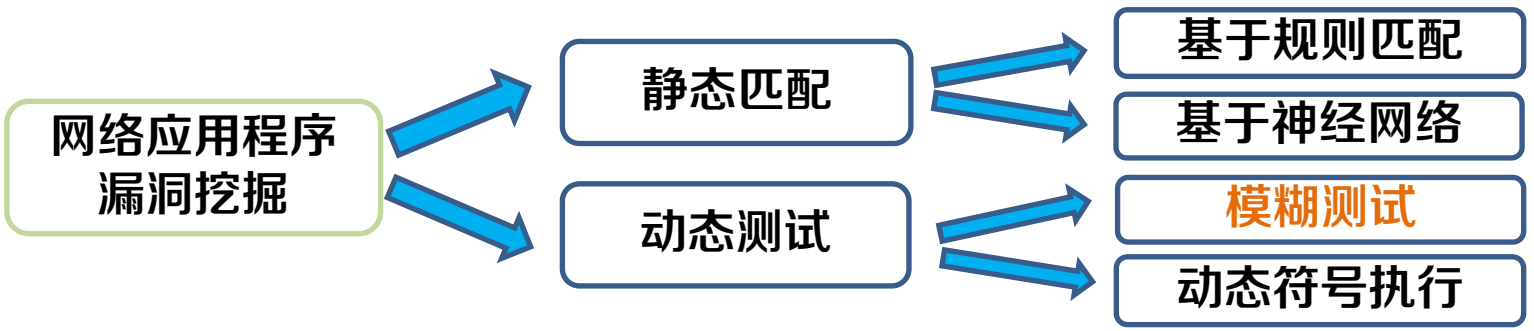
2024 Emre等提出Atropos，基于快照式、高级反馈驱动的Php应用模糊测试

2024 Kong 等人提出PFHE-MVD，结合HGCN实现针对源代码的多类型漏洞白盒分析检测

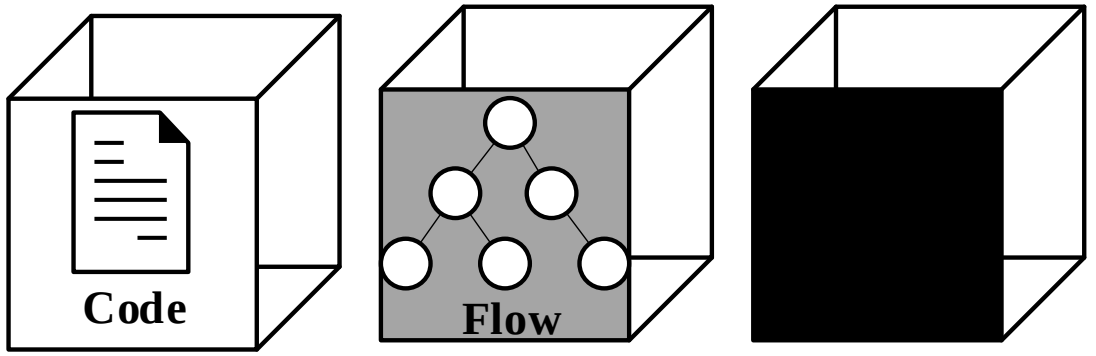
攻防演进环境下
Web应用安全性是否有保障

0day威胁带来安全损失巨大

实现Web漏洞发现的自动化、实战化刻不容缓！



- 基于知识依赖程度划分
  - 白盒：直接基于源代码分析理解语义和程序逻辑生成测试输入
    - 可推理程序的内部结构，覆盖率高；但使用场景受限、存在误报
  - 黑盒：测试仅关注应用程序的输入和输出
    - 测试效率高、应用场景适应性强；无法有效触发程序中的复杂逻辑，覆盖率较低
  - 灰盒：在对应用的访问受到限制的情况下运行(如无法访问源码)
    - 基于受限操作平衡测试的速度和深度，优于黑盒测试，但不及白盒测试的逻辑深度和全面性



PEACH FUZZER

AFL++

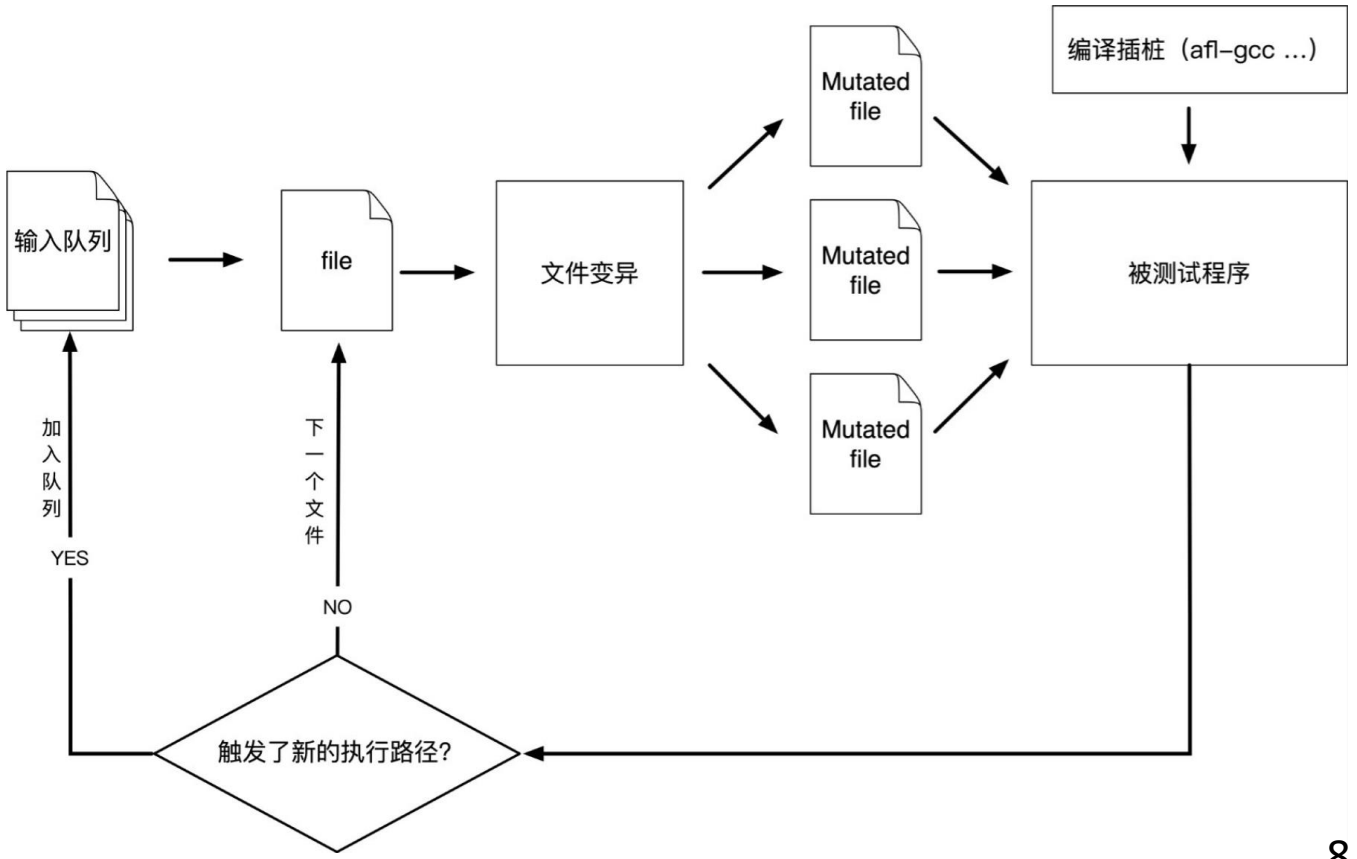
trailofbits/
deepstate

A unit test-like interface for fuzzing and symbolic execution

20 Contributors 83 Issues 809 Stars 97 Forks

灵通测

- 基于引导策略划分　　穷尽所有路径下的所有潜在缺陷
  - 覆盖引导：通过监控程序代码执行路径(即代码覆盖率)来引导测试用例生成，将缺陷探索问题转化为覆盖率的优化问题
    - 优势：自动化程度高、覆盖面广、缺陷遗漏较少
    - 劣势：大量的资源消耗
  - 定向测试：基于输入的先验知识，专注于特定的程序区域或功能定向生成用例，以提高缺陷发现效率
    - 优势：目标明确、效率高
    - 劣势：对先验依赖较大

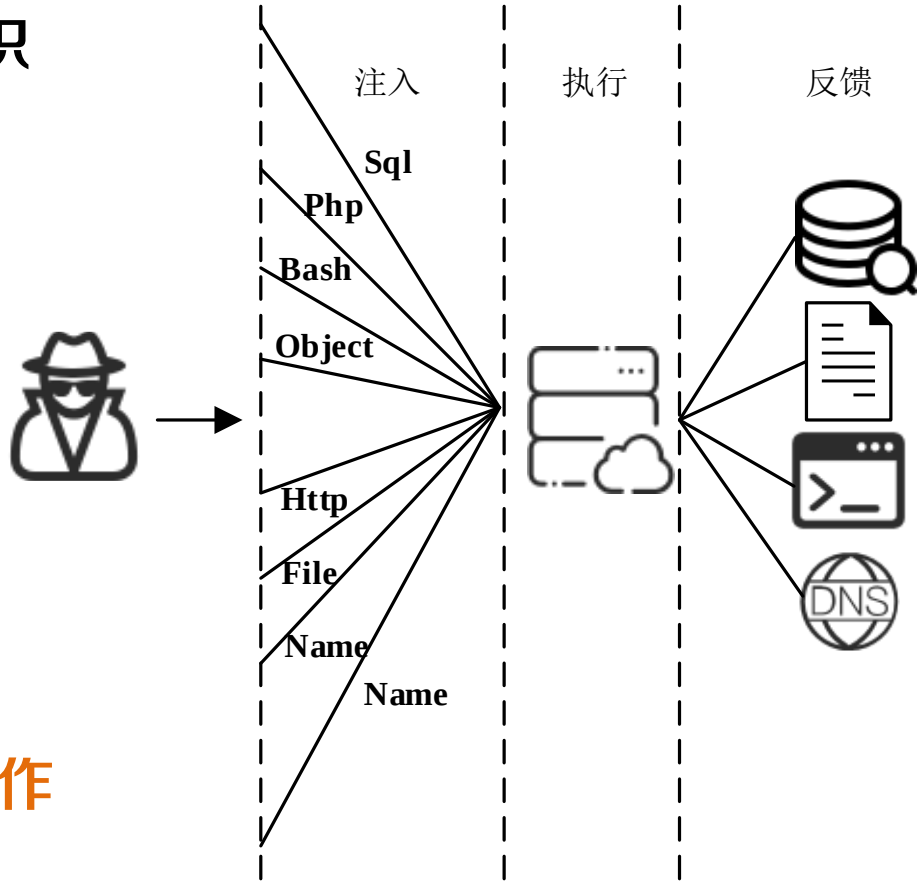


- 基本概念

恶意输入的反馈是缺陷判断的关键

  - OWASP Top 10 是针对开发人员和 Web 应用程序安全性的标准评估文档，代表了对 Web 应用程序最关键的安全风险的广泛共识

- 常见8类漏洞及特征
  - Sql注入：注入恶意的Sql语句实现数据库访问
  - 文件包含：引入外部文件，可执行恶意脚本
  - 文件上传：上传恶意脚本文件
  - 反序列化：注入序列化数据实现逻辑流控制
  - 任意文件读写：未授权读写系统上的任意文件
  - 代码执行：注入代码实现执行恶意代码动作
  - 命令执行：注入系统命令实现执行恶意指令动作



- 简单案例分析
  - 手工场景
    - 核心思路
      - 存在用户可控输入参数(危险函数)
      - 用户可构造恶意输入
      - 恶意动作执行、信息回显
    - 漏洞定位简单、Payload构造直观
  - 自动化场景
    - 核心思路
      - 收集可控输入、生成用例、监测异常
    - 提升覆盖率、触发更多程序分支

异常缺陷≠漏洞
Payload开发是额外工作

可控参数

拼接执行

反馈回显

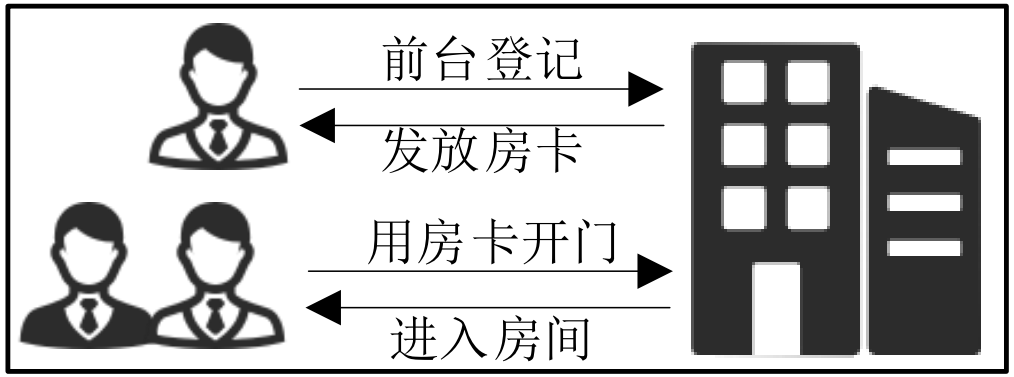
```php
// a) SQL injection vulnerability
$id = $_GET['id'];
$query = "SELECT name FROM users WHERE id='$id';";
$result = mysqli_query($mysql,  $query)

// b) remote code execution vulnerability
eval('$var = "'. $_COOKIE['a'] . '";');

// c) remote command execution vulnerability
shell_exec('ping  ' . $_POST['ip']);

// d) local / remote file inclusion vulnerabilities
include("lib/" . $_GET['page']); // LFI
include($_GET['page']); // RFI

// e) PHP object injection vulnerability
unserialize($_COOKIE['session']);

// f) SSRF vulnerability
file_get_contents("http://".$_POST['host']);

// g) arbitrary read vulnerability
$file = $_POST['file'];
echo file_get_contents($file);

// h) file upload vulnerability
if($_FILES['f']['type'] == "image/png")
  move_uploaded_file($_FILES['f']['tmp_name'],
                     $_FILES['f']['name']);
```

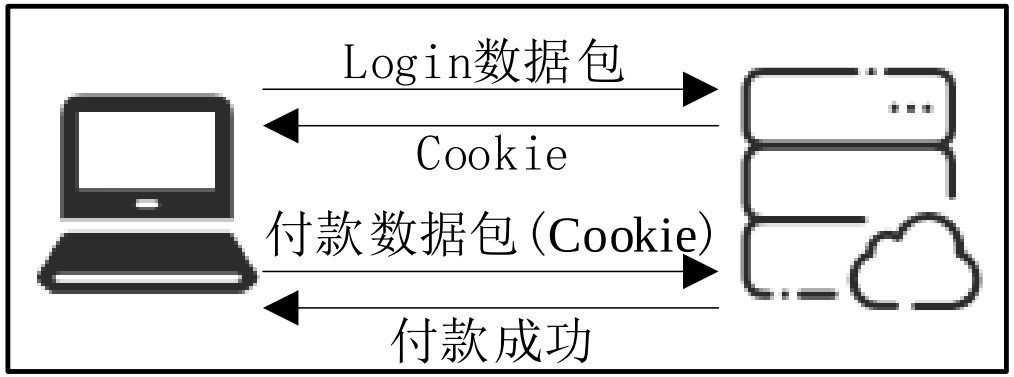
基础知识　网络应用程序测试的特殊挑战

- 接口约束
  - Web应用需基于媒介与Http服务器交互
  - 接口请求必须“是想要的”，用例生成细化变什么、怎么变
- 状态维持
  - Web应用存在大量高度依赖于状态前置条件和运行环境的程序分支
  - 明确存什么、怎么存、怎么恢复
- 异常缺陷监视与判定
  - Web应用漏洞不表现为内存安全违规，无法被常规基于崩溃的模糊测试识别

定性定量明确输入，科学合理认定输出

```
<form action="login.php">
    <input name="user">
    <input name="passwd">
<form>
```





Toss a Fault to Your Witcher: Applying Grey-box Coverage-GuidedMutational Fuzzing to Detect SQL and Command Injection Vulnerabilities

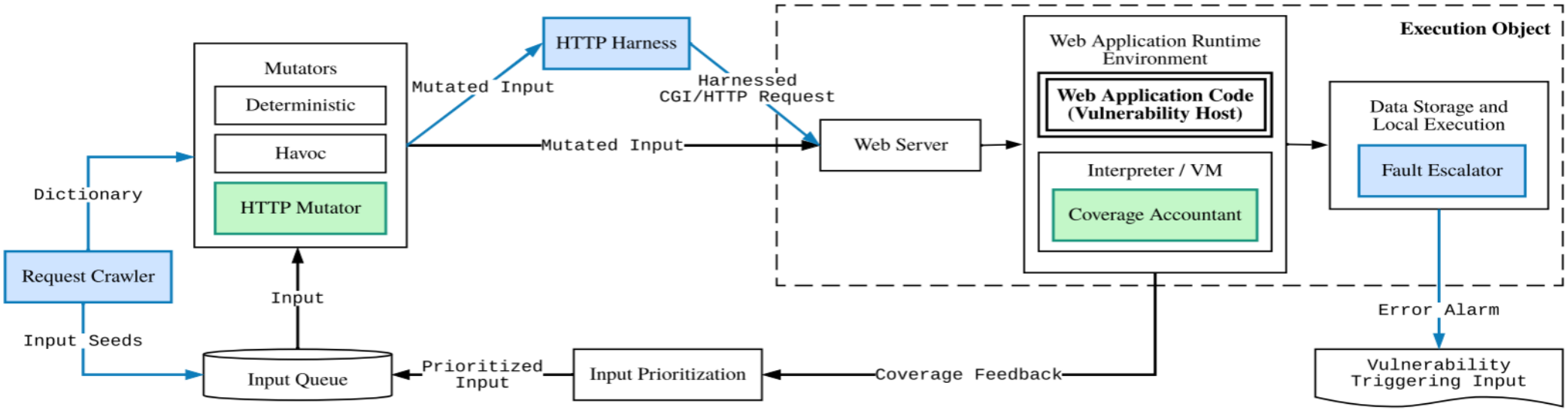
TIPO Witcher

T	目标	自动化测试Web应用程序中的Sql注入、命令注入缺陷
I	输入	1 个Web应用程序入口Url、1个初始化配置(Cookie)
P	处理	1.应用接口爬取与解析 2.测试用例生成/变异及请求封装 3.覆盖率计算反馈 4.异常监视与采集
O	输出	n个缺陷及对应触发测试用例

P	问题	Web应用测试数据不匹配半结构化格式，难以实现通用注入漏洞检测
C	条件	面向PHP开发的Web应用检测单次Sql注入、命令注入
D	难点	1.如何在符合接口约束情况下进行测试用例生成与变异 2.如何计算与Web应用程序的强关联的覆盖率
L	水平	IEEE Symposium on Security and Privacy (S&P), 2023(CCF A)

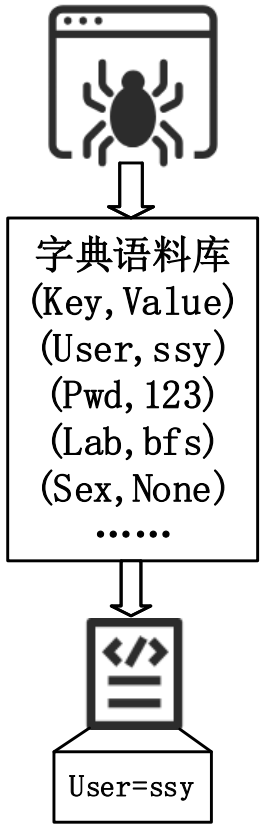
算法原理 Witcher

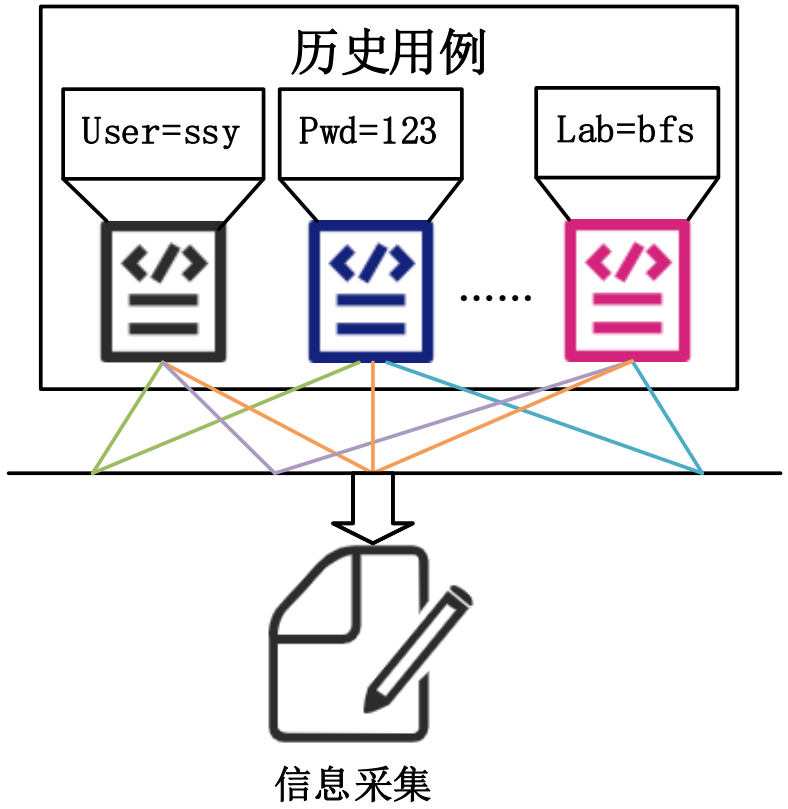
- 核心思想
  - 设计强关联覆盖率计算引导变异方向，接口约束下键值共享策略保障用例有效性
- 算法步骤
  - 加载配置信息并爬取起始Url可交互的页面元素，理解当前交互语义
  - 基于结构语义初始化种子/基于即定变异策略进行用例变异，并封装为结构化请求
  - 结合强关联覆盖率计算方法获取Web分支覆盖率并反馈，调整变异优先级
  - 对当前测试进行存储并进行异常缺陷捕捉与诊断



算法原理 变异策略

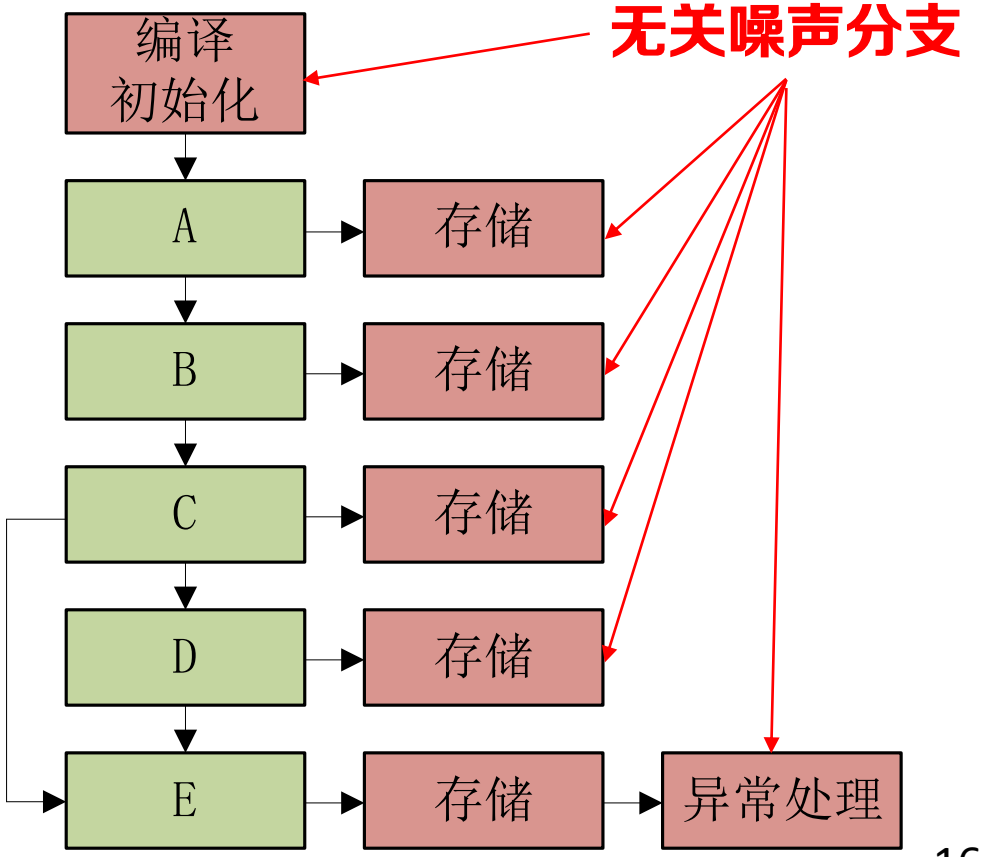
- 种子用例初始化　　种子用例的生成与变异，字典语料库的完善程度
  - 初始测试用例：随机触发用户交互事件至成功创建Http请求
  - 初始测试字典：爬取入口页面元素，正则匹配获取接口参数并构造字典
- 种子用例变异
  - 随机变异：基于AFL对Http请求中的参数值进行随机突变
  - Http参数变异：在一个接口的测试中，从有效历史测试用例中交叉共享参数名与参数值
  - Http字典变异：基于动态爬虫保持对字典的维护更新，从中随机选择1-10个变量添加生成测试用例





算法原理　覆盖率计算

- 覆盖统计降噪核心思想
  - 对编译后的字节码进行插桩，并存储字节码行号信息以计算覆盖率
  - 强化对Web应用程序关联分支的关注，减少覆盖率计算中的噪声
- 传统覆盖率计算缺陷
  - 传统方法：对解释器进行插桩统计覆盖率
    - 存在大量属于编译器动作的分支触发记录，会对Web测试的变异方向产生误导
  - Witcher：定制化解释器，存储解释器编译输出的字节码行号，每执行一条字节码指令时就会触发覆盖率统计
    - 借助解释器的Hook函数，若触发关键函数时会将此次变异字段重点标记反馈



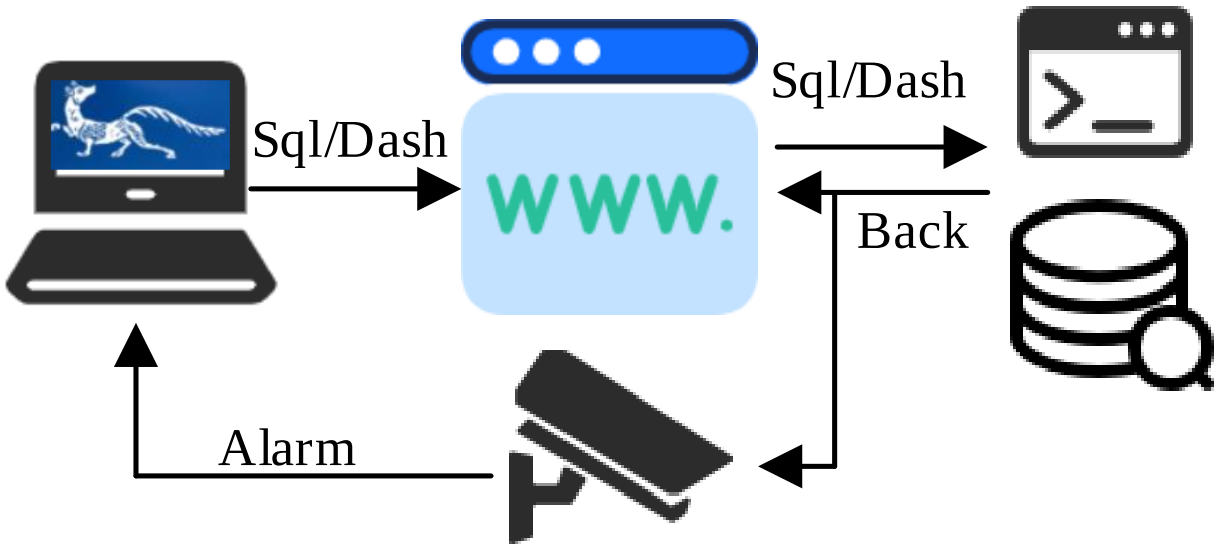
算法原理　异常缺陷捕捉与诊断

- 基于外部解析器支持的异常捕捉
  - 二进制模糊测试：若输入造成内存监视器抛出错误，则判定为缺陷异常
  - Witcher：定制化外部监视方法
    - Sql注入：利用Ld_Preload对函数recv()进行Hook，数据库抛出错误消息则反馈
    - 命令注入：采用Dash作为外部解析器，若Dash判断命令错误则反馈
    - 内存缺陷：基于AFL支持对内存错误的监测

异常缺陷≠有形漏洞

异常缺陷到漏利用
还剩一里地

模糊测试质量决定
漏洞利用高度



实验设计　数据源及对比方法

- 测试数据源
  - 覆盖6个类型应用
    - PHP、Java、Python、Ruby、Node、二进制
  - 包含2个类型漏洞
    - Sql注入、命令注入
- 评价指标
  - 漏洞挖掘数目、代码覆盖率
- 对比方法
  - Burpsuite(商用黑盒扫描器)
  - BlackWidow(灰盒Fuzzer，2021)
  - WebFuzz(灰盒Fuzzer，2021)

Application	Lang. or Platform	Release Date	Ver.	GitHub Stars	Google Results	Lines of Code
OpenEMR	PHP 7	2018	5.0.1.7	1.6k ★		9,443
WackoPicko	PHP 5	2018	1.0	265 ★		2,510
Doctor Appointment Booking System	PHP 7	2020	1.0	n/a	≈10 [65]	3,981
User Login Management System	PHP 7	2020	2.1	n/a	≈3 [66]	1,490
rConfig	PHP 7	2018	3.9.2	80 ★		48,405
Hospital Management System	PHP 7	2019	4.0	n/a	≈100 [67]	9,443
D-Link DIR-823G	C/MIPSEL	2018	1.0.3.B3	n/a		1,585,157
D-Link DIR-823G	C/MIPSEL	2018	1.0.2.B5	n/a		1,569,829
D-Link DIR-645	C/ARM	2014	1.0.4.B12	n/a		465,324
D-Link DIR-825	C/MIPSEB	2015	1.2.10.B1	n/a		542,992
Tenda AC9	C/ARM	2018	15.03.05.19	n/a		982,880
WebGoat	Java	2020	8.10	3.9k ★		14,761
FlaskBB	Python	2018	2.0.2	2.0k ★		14,534
Juice Shop	Node.js	2020	8.1.0	242 ★		26,221
Thredded	Ruby/Rails	2021	16.16	1.3k ★		4,426
phpBB	PHP 7	2021	3.3.3	1.4k ★		318,104
osCommerce	PHP 7	2017	2.3.4.1	272 ★		44,355
Wordpress	PHP 7	2021	5.7.1	15k ★		253,183

实验结果 性能对比

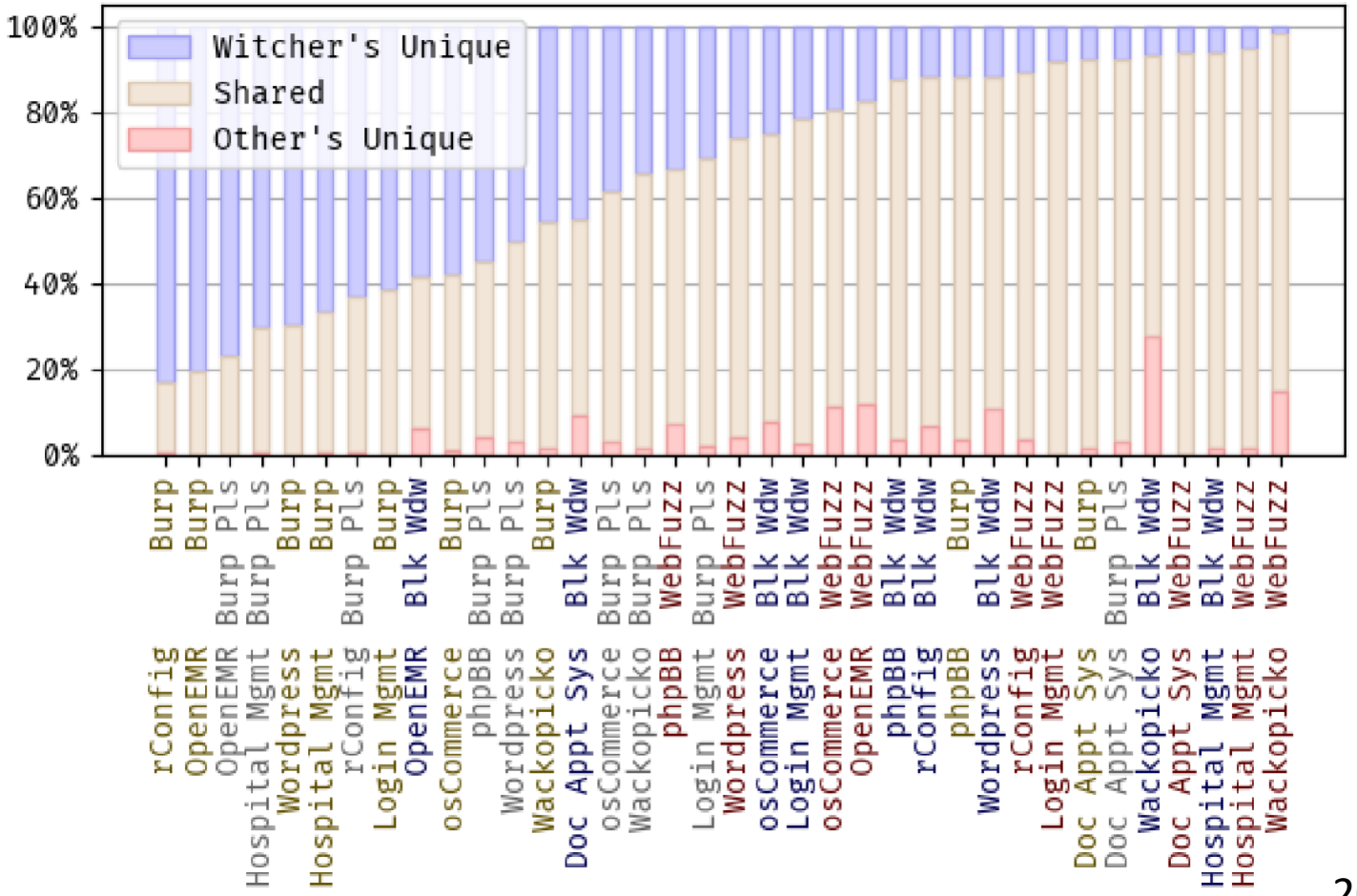
- 漏洞挖掘能力评估
  - Witcher VS 黑盒扫描器
    - Witcher 共发现 70 个漏洞，其中 52 个漏洞具有唯一性
    - 在9个应用程序中的的6个发现漏洞且数目最多
  - Witcher VS 灰盒Fuzzer
    - 在除 WackoPicko 应用之外的所有应用程序上发现了更多独特的代码行
    - WackoPicko是一个购物网站，对于状态维持要求较高

Application	Burp (solo)	BurpPlus Witcher	Witcher
Doctor Appt. Sys.	2 (0)	3 (0)	**3** (0)
Hosp. Mgmt.	13 (0)	13 (0)	**43 (30)**
Login Mgmt.	1 (0)	1 (0)	**6 (5)**
OpenEMR	0 (0)	0 (0)	**5 (5)**
osCommerce	0 (0)	0 (0)	0 (0)
phpBB	0 (0)	0 (0)	0 (0)
rConfig	0 (0)	0 (0)	**11 (11)**
WackoPicko	1 (0)	1 (0)	**2 (1)**
Wordpress	0 (0)	0 (0)	0 (0)
	17 (0)	18 (0)	**70 (52)**

Application	Black Widow			WebFuzz		
	$W \setminus B$	$W \cap B$	$B \setminus W$	$W \setminus B$	$W \cap B$	$B \setminus W$
Doctor Appt. Sys.	**209**	211	43	**74**	1,067	10
Hospital Mgmt	**92**	1,350	24	**164**	3,162	54
Login Mgmt	**37**	131	4	**27**	492	0
OpenEMR	**25,273**	15,464	2583	**25,237**	107,917	17,852
osCommerce	**2,657**	7,100	798	**4,147**	22,635	3,622
phpBB	**3,210**	22,121	879	**15,483**	35,480	6,544
rConfig	**458**	3,094	239	**301**	2,259	30
WackoPicko	72	670	**283**	50	2,415	**392**
Wordpress	**7,036**	46,259	6482	**41,239**	109,076	5,935

实验结果 性能对比

- 代码覆盖率评估
  - Witcher 在除WackoPicko之外的8个应用程序中均实现最高的代码覆盖率
    - Witcher的单次测试仅专注于穷尽一个接口的所有分支测试
    - Witcher测试中不存在接口交叉运行，接口间无依赖关联，也导致对页面关联动作较多的应用程序测试性能较弱



Fuzzer的状态维持能力一定程度影响代码覆盖

实验结果 性能对比

- 测试对象偏见测试
  - 所有应用程序的百分比增长（变化量/覆盖的总行数）均小于 3.3%
  - 对于所选的Web应用程序，目标覆盖漏洞偏差对代码覆盖结果的影响较小
- 测试速度比较
  - Witcher测试速度是第2名WebFuzz的6倍
  - 其他Fuzzer在生成测试用例前，需依赖于爬虫的结果

Application	Witcher v. Black Widow $W \setminus BW+$	$(BW+) \setminus W$	Inc.	Witcher v. WebFuzz $W \setminus WF+$	$(WF+) \setminus W$	Inc.
Doctor Appt.	206	43	0	74	10	0
Hosp. Mgmt.	88	24	0	71	60	6
Login	19	7	3	3	0	0
OpenEMR	25,117	2,606	23	22,473	19,915	2,063
OSCommerce	2,497	863	65	3,833	3,622	0
phpBB	2,967	1,159	280	15,133	7,508	964
rConfig	426	254	15	104	62	32
WackoPicko	66	285	2	50	407	15
Wordpress	6,966	6,733	251	37,401	10,365	4,430

Applications	Witcher Req/s	WebFuzz Req/s	Black Widow Req/s	Burp Req/s
Doctor Appt. Sys.	539.4	43.4	3.4	7.3
Hosp. Mgmt. Sys.	327.4	26.6	3.0	5.0
Login Mgmt.	180.9	112.2	0.9	13.3
OpenEMR	15.3	1.5	1.7	5.1
osCommerce	22.2	4.7	0.7	2.4
phpBB	14.7	1.5	0.7	1.1
rConfig	52.7	10.1	3.3	3.3
WackoPicko	101.9	2.2	0.2	23.5
Wordpress	24.4	0.1	0.5	0.1
Average	142.1	22.5	1.6	6.8

Atropos: Effective Fuzzing ofWeb Applications for Server-Side Vulnerabilities

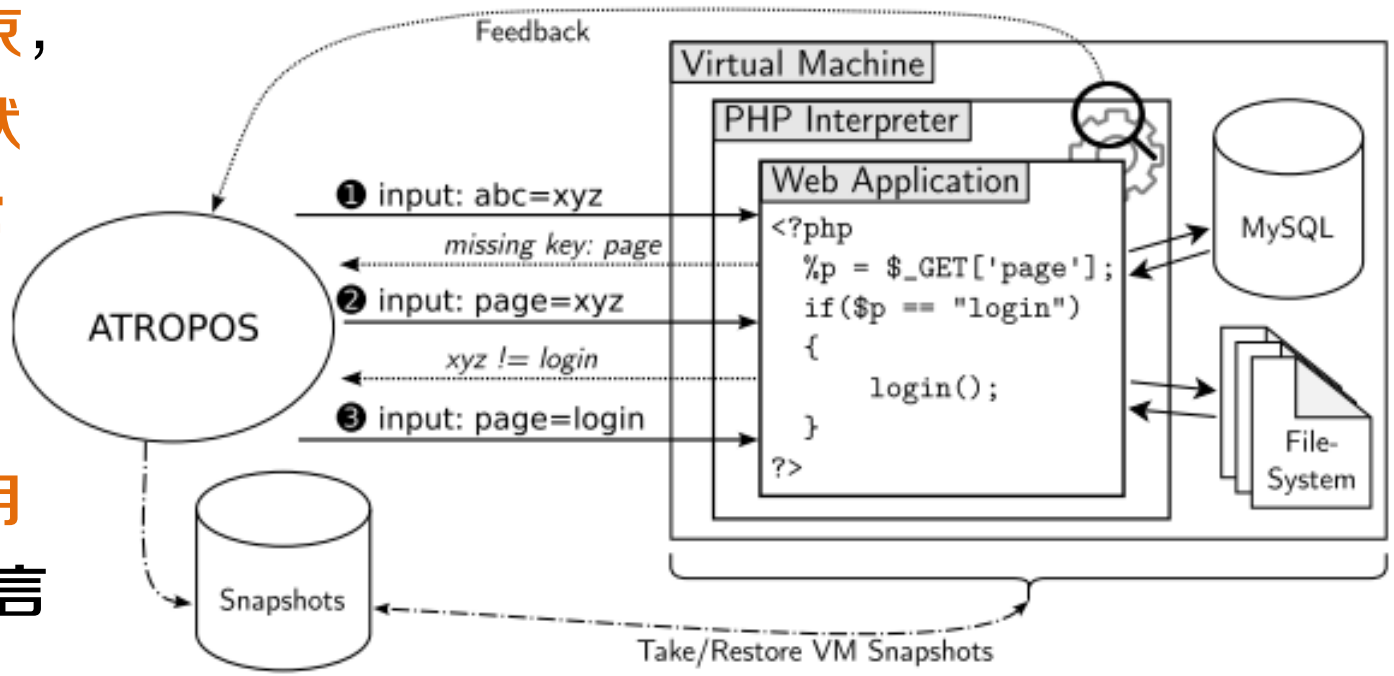
TIPO Atropos

T	目标	自动化测试Web应用程序中8类漏洞缺陷
I	输入	1个Web应用程序
P	处理	1.初始化测试环境 2.基于参数约束推导生成/变异测试用例 3.保存测试用例执行前后状态采集并错误和告警 4.基于反馈引导变异并进行状态恢复，迭代步骤2和3
O	输出	n个缺陷及其对应的触发输入

P	问题	Web测试中缺乏状态维持，且漏洞类型覆盖不足
C	条件	面向PHP开发的Web应用测试8类漏洞
D	难点	1.如何有效保存、恢复状态 2.如何设计多类型漏洞监视器
L	水平	USENIX Security Symposium, 2024(CCF A)

算法原理 Atropos

- 核心思想
  - 精细化推断键值对结构明确约束，基于阶段性快照恢复机制维持状态，多类型Bug预言机拓展异常捕捉能力
- 算法步骤
  - 在Http规范下随机初始化种子用例，开启FastCGI通信、Bug预言机监听

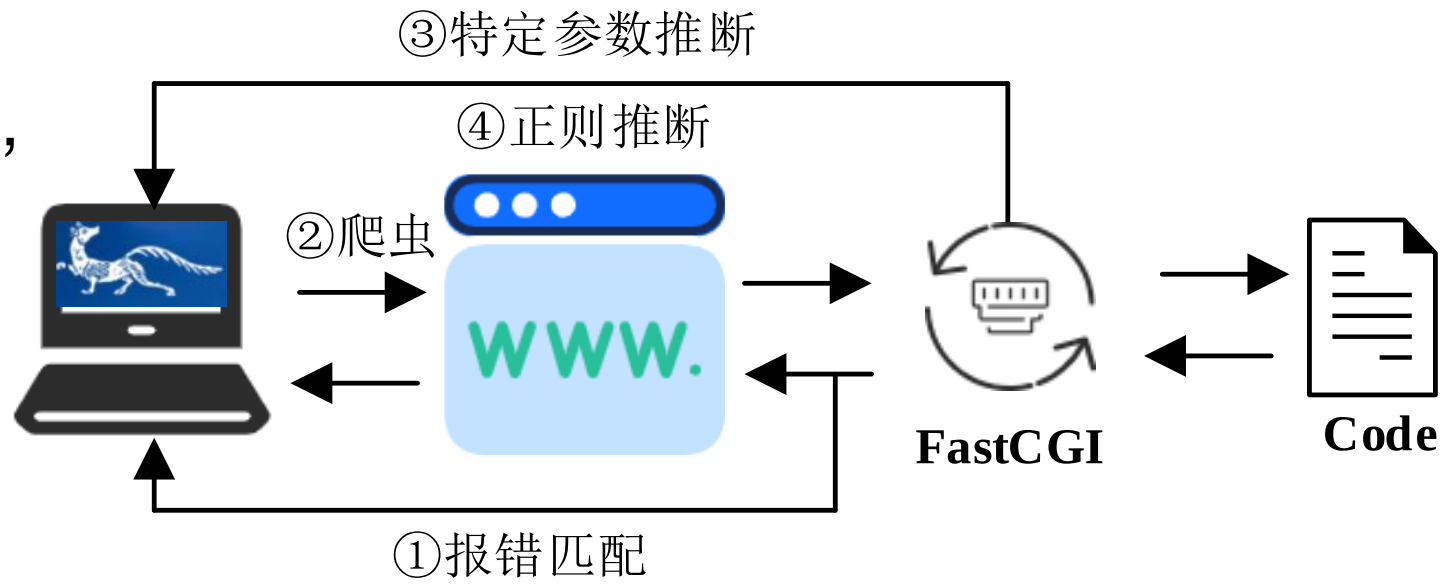


  - 基于高级反馈机制细粒度推断参数约束并进行测试用例生成和变异
  - 使用虚拟化快照保存执行前后状态（状态维持），Bug预言机进行异常监测
  - 基于程序反馈更新变异优先级并周期性恢复状态

算法原理　参数约束推断（高级反馈机制）

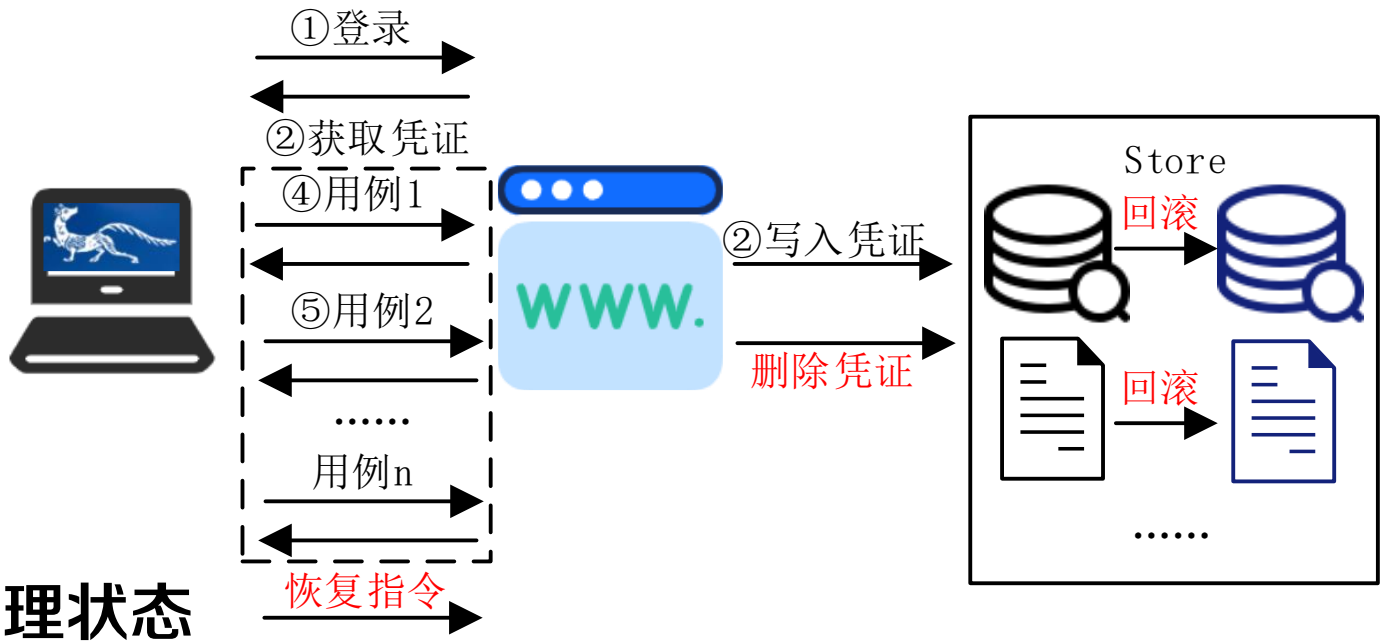
- 基于错误反馈的特定键推断
  - 内置Hook函数全周期监控应用程序的处理流，捕获代码运行错误并作为“缺失键”反馈
- 基于前端匹配的键推断（可选）
  - 基于动态网络爬虫，在前端JS中匹配键值对
- 特定值推断
  - 在解释器层Hook比较函数，在遇到值比较操作时，将比较的预期值反馈
- 正则值推断
  - 原理同上，Hook正则函数

可缓解冷启动



算法原理 状态维持

- 状态维持处理可行思路
  - 拉黑状态操作：人工介入将会引发状态改变的操作拉黑
  - 全面追踪并还原状态：追踪每一次状态变化，操作完成后还原状态



- Atropos基于快照机制追踪和管理状态
  - 运行在虚拟化环境中，并使用快照技术周期性来保存/恢复整个系统的状态
    - 周期性：若干个输入测试迭代结束后，系统执行快照恢复
  - 优势：
    - 确保了测试的可重复性并避免了状态累积的问题
    - 允许在保持应用功能不受限的同时，快速恢复环境，避免了昂贵的初始化过程

算法原理　多类型Bug预言机

- 漏洞预言条件
  - 不安全函数表现出异常或可疑的行为，通常会产生警告或错误
  - 这种行为是由攻击者控制的用户输入触发的
- 将错误反馈与数据流检查相结合，精准判定受控输入是否到达特定漏洞点

Bug预言机	Hook函数	捕捉错误类型
SQL Injection	Mysqli_query()、pdo_sqlite_errot()等	数据库语法错误
Remote Code Execution	Eval()、call_user_func()等	PHP解析错误
Remote Command Execution	Php_exec_ex()、shell_exec()等	不存在的指令错误
PHP Object Injection	Unserialize()	对象属性解析错误
Local and Remote File Inclusion	Php_verror()、php_error_cb()等	文件索引查找错误
Server-side Request Forgery	File_get_contents()、socket_connect()等	不存在的地址错误
Arbitrary File Read and Write	File_get_contents()、File_put_contents()等	文件操作函数返回
File Upload	Move_uploaded_file()	文件系统函数返回

实验设计　数据源及对比方法

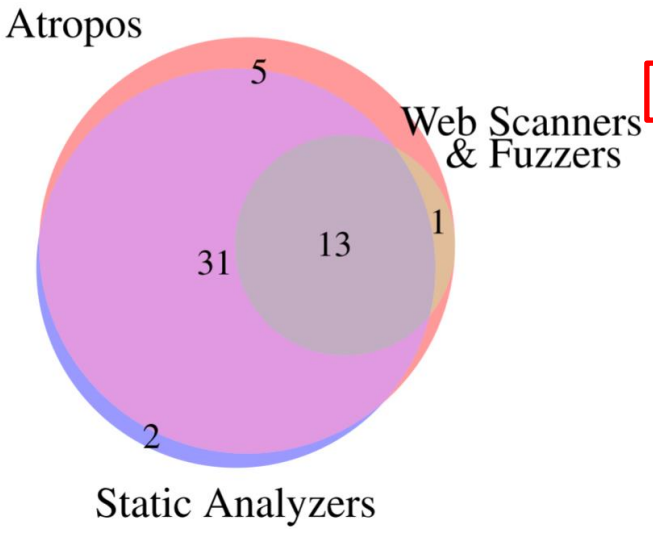
- 测试数据源
  - DVWA（4/16）、XVWA（6/9）、bWAPP（5/27）
  - 10款PHP应用，Github超过100Star
- 评价指标
  - TP、FP、TPR、Precision
  - Bug触发数目、漏洞挖掘数目（真实场景下）
  - 代码覆盖率
- 对比方法
  - 4款商业静态分析工具
  - 2款商业黑盒扫描器
  - 4个学术研究

Test Suite	Commit	Information
DVWA	#423ac71	7 x SQL injection, 3 x Remote Command/Code Execution, 3 x File Inclusion, 3 x File Upload
XVWA	#fb30fa5	3 x Remote Code/Command Execution, 1 x PHP Object Injection, 1 x SSRF, 2 x SQL Injection, 1 x File Inclusion, 1 x File Upload
bWAPP	#f3f423c	18 x SQL Injection, 5 x Remote Code/Command Execution, 2 x File Upload, 1 x Arbitrary Read, 1 x File Inclusion

静态	SONARQUBE、PROGPILOT、PSALM、PHPCS-SECURITY-AUDIT
动态	黑盒扫描器：WAPITI、ZAP 灰盒Fuzzer：Witcher(2023)、WFuzz(2023)、CEFuzz(2022)、WebFuzz(2021)

实验结果　漏洞挖掘能力评估

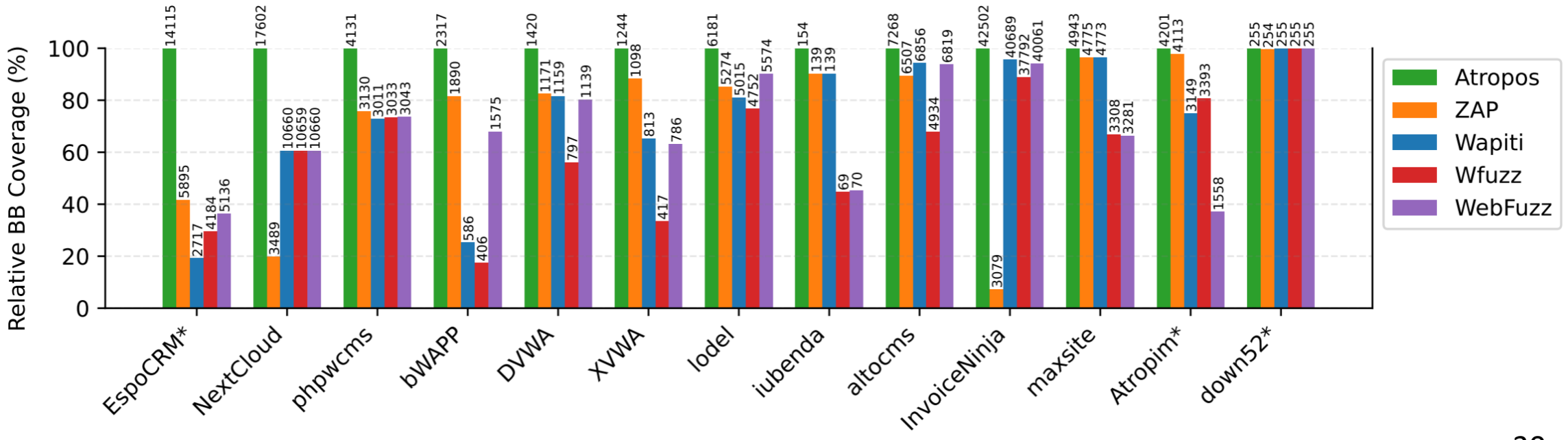
- **Bug触发实验**
  - 在40核心条件下，Atropos的误报、漏报和漏洞发现数目均达到最优
  - 在3个漏报原因分别为：测试时间不足未能覆盖相应文件、种子语料库缺乏漏洞触发特殊字段
  - 整体而言，相较于静态分析工具和黑盒扫描器，Fuzzer基本不会产生误报



	DVWA		XVWA		bWAPP		Precision	TPR
	TP	FP	TP	FP	TP	FP		
ATROPOS (40 cores)	**15**	0	**7**	0	**27**	0	**100% (49 / 49)**	**94% (49 / 52)**
ATROPOS (1 core)	14	0	6	0	19	0	**100% (39 / 39)**	75% (39 / 52)
SONARQUBE	5	0	4	0	20	0	**100% (29 / 29)**	56% (29 / 52)
PROGPILOT	12	2	4	0	18	5	83% (34 / 41)	65% (34 / 52)
PSALM	7	1	6	0	22	5	85% (35 / 41)	67% (35 / 52)
PHPCS-SECURITY-AUDIT	13	28	5	2	19	29	39% (37 / 96)	71% (37 / 52)[1]
WITCHER	0	0	3	0	0	0	**100% (3 / 3)**	6% (3 / 52)[2]
ZAP	2	4	2	1	3	6	39% (7 / 18)	13% (7 / 52)
WAPITI	0	17	4	15	0	2	11% (4 / 38)	8% (4 / 52)
WFUZZ	0	–	0	–	0	–	–	0% (0 / 52)
CEFUZZ	3	–	–	–	5	–	–	15% (8 / 52)[3]

实验结果　漏洞挖掘能力评估

- 代码覆盖率实验
  - ATROPOS比ZAP多达到63%的基本块，比Wapiti多46%，比Wfuzz多80%，比Webfuzz多50%
    - 高级反馈机制保障测试用例高度结构化、贴合参数约束从而触发更多的代码路径
    - 确保在多个连续请求中维持合法状态，有效触发状态强依赖的程序逻辑



实验结果　漏洞挖掘能力评估

- 真实世界应用程序测试分析
  - Atropos共计发现了7个独特漏洞
  - VS Fuzzer：Witcher仅支持Sql注入、命令注入但也无法有效检出
  - VS 静态分析工具：静态匹配模式平均只能发现大约 40% 的错误
  - VS 黑盒漏扫工具：黑盒高度依赖于即定漏洞范式，无法发现任何错误
  - 整体而言Atropos 是唯一能够发现所有漏洞的工具

Web Application	ATROPOS	SONARQUBE	PROGPILOT	PSALM	PHPCS-SECURITY-AUDIT	Witcher	wfuzz	ZAP	Wapiti
AltoCMS	✓	✗	✗	✗	✗	–	✗	✗	*to*
nextCloud	✓	✓	✗	✓	✗	–	✗	✗	✗
Invoice Ninja	✓	✓	✗	✓	✗	–	✗	✗	✗
Iubenda	✓	✗	✗	✓	✓	–	✗	✗	✗
lodel	✓	✗	✗	✗	✗	✗	✗	✗	✗
MaxSite	✓	✓	*oom*	✓	✗	–	✗	✗	✗
phpwcms	✓	✓	✗	✓	✓	–	✗	✗	✗

实验结果　漏洞挖掘能力评估

- 真实世界应用程序测试分析
  - 高级反馈机制能够运行时提取和识别在PHP文件中未直接引用的变量，并存储为字典供用例变异
  - 相较于其他爬虫、静态工具、扫描器能够更精细化捕捉危险参数
    - 静态分析：无法检测到此漏洞，静态字符串匹配提取会错过这种自定义模板格式
    - 黑盒扫描器：大多只进行单步漏洞探测，无法有效检出此类参数多层传递问题
    - 基于爬虫的Fuzzer：仅基于已渲染的页面爬取参数，无法捕捉该参数

与污点分析异曲同工
复杂度更胜一筹

```
<form action="<?php echo $PHP_SELF; ?>">
<IF CONDITION="[#CSSDATA]">
  <style type="text/css">[#CSSDATA]</style>
</IF>
```

```
foreach($_GET as $k=>$v)
    macro_vars[$k] = strip_tags($v);
foreach($_POST as $k=>&$v)
    macro_vars[$k] =& $v;
```

特点总结与工作展望

- 特点总结

算法	Witcher	Atropos
优势	1.支持多语言Web应用测试	1.状态可存储恢复 2.参数约束 3.用户配置信息更少
劣势	1.漏洞类型覆盖较少 2.无状态维持 3.复杂漏洞（如二次注入）无法覆盖 4.缺少缺陷自动化分析、漏洞Poc开发	1.语言仅支持PHP 2.环境复杂 3.复杂漏洞（如二次注入）无法覆盖 4.缺少缺陷自动化分析、漏洞Poc开发
特色	开创了针对Web应用强关联覆盖率计算方法	首个提出状态维持机制，精细化参数推导机制

- 未来发展展望
  - 复杂漏洞缺陷触发机制开发
  - 缺陷自动化分析研究
  - 改进覆盖率计算算法，优化扫描速度与资源开销
  - 动静结合，提升漏洞挖掘能力

参考文献

[1] Trickel E, Pagani F, Zhu C, et al. Toss a fault to your witcher: Applying grey-box coverage-guided mutational fuzzing to detect sql and command injection vulnerabilities[C]. 2023 IEEE symposium on security and privacy (S&P). IEEE, 2023: 2658-2675.

[2] Güler E, Schumilo S, Schloegel M, et al. Atropos: Effective Fuzzing of Web Applications for Server-Side Vulnerabilities[C]. USENIX Security Symposium. 2024.

[3] Eriksson B, Pellegrino G, Sabelfeld A. Black widow: Blackbox data-driven web scanning[C]. 2021 IEEE Symposium on Security and Privacy (S&P). IEEE, 2021: 1125-1142.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！