BeijingForestStudio

北京理工大学信息系统及安全对抗实验中心

基于黑盒扫描的Web漏洞挖掘技术

硕士研究生　邵羊飞

2026 年 04 月 26 日

问题回溯

- 总结反思
  - 读ppt痕迹较重
  - 没有面向观众
- 相关内容
  - 2022.12.19 杨宗源：《web项目开发方法》
  - 2021.05.27 李班：《Web前端框架对比》
  - 2021.01.24 崔成钢：《python Web编程-Django》
  - 2020.10.19 魏继勋：《设计模式在Web开发中的实践》
  - 2020.1.13 门元昊：《Web快速开发方法简介》

内容提要

- 预期收获
- 内涵解析与研究目标
- 研究背景与研究意义
- 研究历史
- 知识基础
- 算法原理
  - Yurascanner
  - BACscan
- 特点总结与工作展望
- 参考文献

预期收获

- 预期收获
  - 了解基于黑盒扫描的Web应用漏洞挖掘技术的基本方法
  - 理解基于黑盒扫描的Web应用漏洞挖掘技术的技术原理
  - 了解大模型在Web应用漏洞挖掘技术的应用

内涵解析与研究目标 Web应用漏洞挖掘技术

- 研究目的
  - 在攻击者利用之前，自动发现并验证Web应用中的安全缺陷，从而提升系统防护能力，保障数据、业务与用户安全
- 内涵解析
  - Web漏洞挖掘：Web漏洞挖掘是针对Web应用的页面、接口和业务流程，系统发现并验证其安全缺陷的过程
  - 和Web攻击的区别：

	Web网站漏洞扫描工具	渗透攻击工具
目的	自动发现Web应用中的潜在漏洞并生成报告	模拟真实攻击行为，验证系统安全性
方法	以静态、动态分析为主，自动化检测	以手动或半自动方式进行定向攻击测试
范围	关注SQL注入、XSS等广泛漏洞类型	关注特定攻击场景，如密码破解、数据窃取

2022年8月5日

Twitter 上发生的一起事件影响了部分账户和私人信息。

该漏洞导致，如果有人向Twitter系统提交电子邮件地址或电话号码，系统会告知提交的电子邮件地址或电话号码关联Twitter帐户

谷歌修复了一个可能泄露用户私人电话号码的漏洞

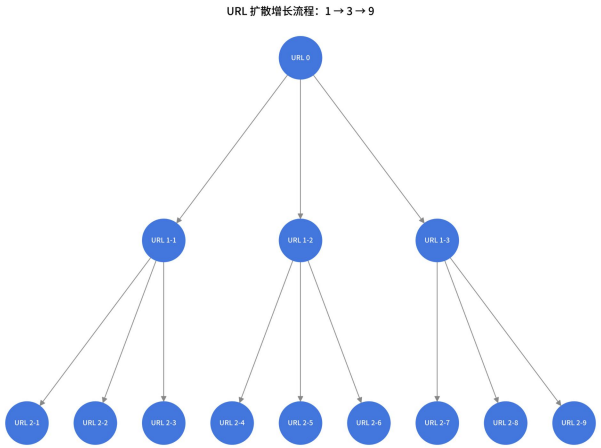
研究人员表示，通过脚本自动执行攻击链，可以在 20 分钟或更短时间内暴力破解 Google 帐户所有者的恢复电话号码，具体时间取决于电话号码的长度。

Brutecat 表示，谷歌为他们发现该漏洞支付了 5000 美元的漏洞赏金。

扎克·惠特克 — 美国太平洋时间上午 7:00 · 2025 年 6 月 9 日

研究背景与研究意义

- 研究背景
  - 随着Web应用规模和复杂度不断上升，其安全漏洞带来的数据泄露与业务风险日益突出
  - Web应用越复杂，扫描器越容易陷入状态爆炸、流程理解不足和逻辑漏洞发现困难等问题
  - 大模型推动Web漏洞扫描从规则驱动向语义理解与自主推理驱动演进，为复杂业务漏洞挖掘提供了新的技术路径



- 研究意义
  - 现实上提升复杂Web漏洞的发现与防护能力
  - 理论上推动漏洞扫描方法向智能化演进

研究历史 Web应用漏洞挖掘技术

Crawler+attack

DetLogic将应用的预期行为建模为带注释的有限状态机，推导出输入参数、访问控制和业务流程的约束。通过违反这些约束构造攻击向量，成功模拟并识别出Web应用中的逻辑漏洞

Crawler+attack

AuthZee采用了创新的进化爬虫和三元组测试技术，结合三个用户账户的登录凭证，自动发现Web应用中的资源并检测是否存在授权漏洞

Crawler+attack

EvoCrawl采用了进化搜索算法，EvoCrawl能够有效地发现不同的Web交互序列，探索Web应用的更多代码和服务器端状态。

attack

BACScan采用了创新的反馈驱动式预言机，BACScan能通过推断操作依赖的网页并分析操作反馈，检测Web应用中的BAC漏洞

2018 2021 2023 2025

2021

BlackWidow采用了黑箱数据驱动的方法，BlackWidow通过结合导航建模、遍历和追踪状态依赖，提升了Web爬虫和扫描的深度

Crawler

2023

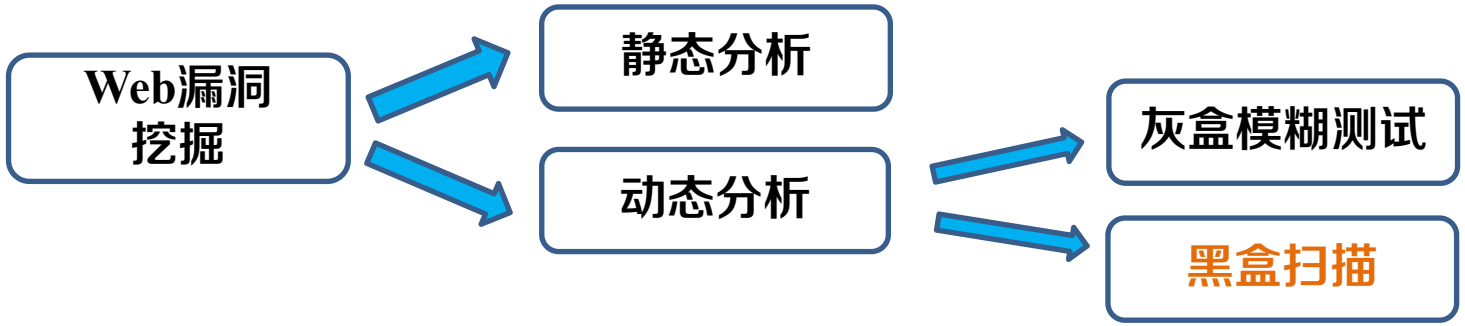
BlackOstrich采用了结合字符串约束求解能力的深度Web爬虫和扫描方法，BlackOstrich通过动态推断适当输入并通过输入验证检查

Crawler

2025

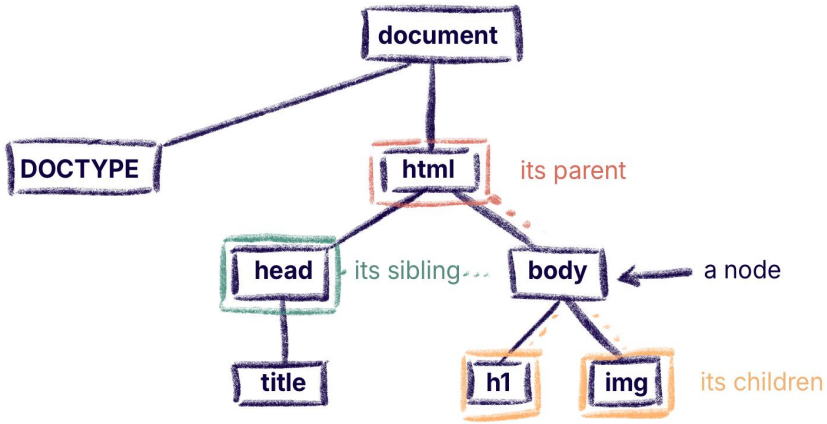
YuraScanner采用了大语言模型（LLM）驱动的方法，YuraScanner通过自主执行任务和工作流，克服了传统Web应用扫描器在探索深层状态时的局限性

Crawler



知识基础 Web基本构成

- 客户端/前端
  - 用户通过浏览器访问Web应用并进行交互
  - 前端负责页面展示、表单输入和事件响应
- DOM（文档对象模型）
  - 浏览器会将网页解析为一棵DOM树，页面中的按钮、输入框、链接等元素都会变成可操作节点
  - 前端脚本和自动化工具通常基于DOM定位和操作页面元素
- 网络通信
  - 浏览器通过HTTP/HTTPS与服务器通信
  - 请求中可包含URL、参数、请求头、Cookie等信息
  - 服务器处理后返回页面或数据响应
- 服务端与数据库

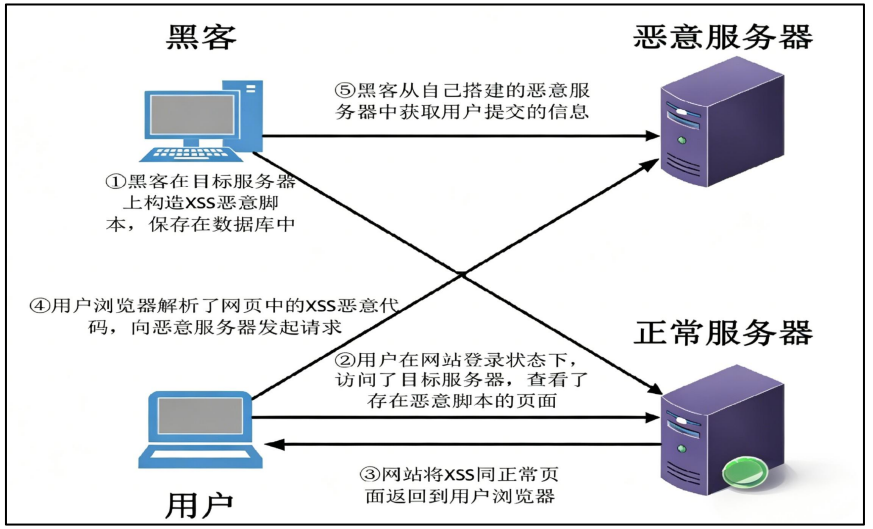


知识基础 Web漏洞挖掘

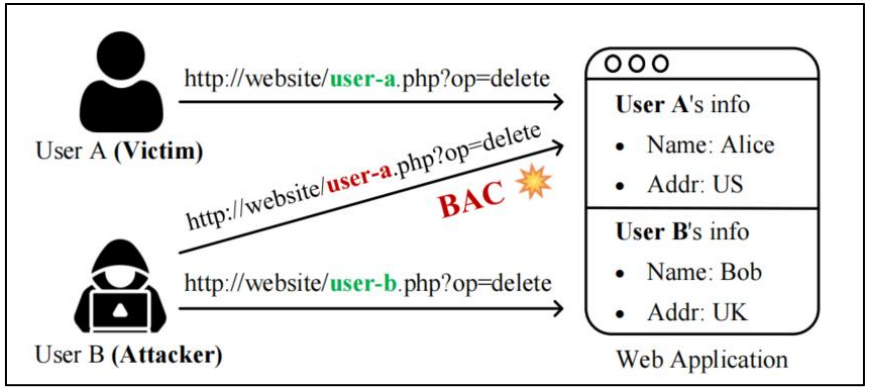
- 定义
  - Web漏洞挖掘，是指通过人工或自动化方式，分析并发现Web应用在输入处理、权限控制、业务流程和数据交互中的安全缺陷，从而识别攻击者可能利用的风险点
- 挖掘目标
  - 是否能访问不该访问的数据
  - 是否能执行不该执行的操作
  - 是否能通过异常输入影响系统行为
  - 是否能绕过正常业务流程
- 典型漏洞类型
  - SQL注入、XSS、越权访问、逻辑漏洞

知识基础 Web漏洞

- XSS（跨站脚本）
  - 恶意脚本被注入网页并在浏览器中执行
  - 成因：输入过滤或输出转义不足
  - 危害：窃取Cookie、劫持会话、篡改页面
  - 例子：评论区脚本被其他用户打开时触发
- BAC（访问控制漏洞）
  - 系统未正确限制用户对资源或功能的访问
  - 本质：未授权读取或未授权操作
  - 表现：查看他人数据、调用管理员接口
  - 例子：修改订单编号后查看他人订单



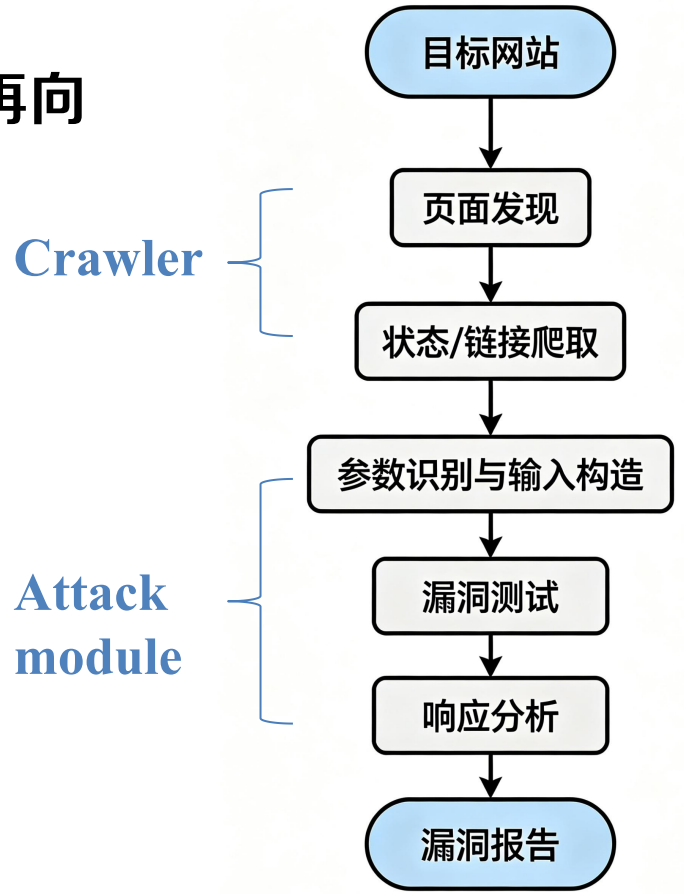
存储型XSS漏洞



BAC漏洞

知识基础 Web扫描

- Web扫描定义
  - Web扫描器是一类自动化黑盒安全测试工具
  - 它通常从初始页面出发，不断发现URL、表单和输入点，再向这些位置注入测试输入，根据返回结果判断是否存在漏洞
- 扫描流程
  - 扫描器 = crawler + attack module
- 挑战
  - 缺乏工作流理解：难以按正确顺序执行多步骤操作
  - 难以进入深层状态：很多漏洞位于多步操作之后
  - 导航策略机械：BFS、随机导航难以走通复杂流程
- 传统扫描器的问题不是不会测，而是到不了该测的地方



YURASCANNER: Leveraging LLMs for Task-driven Web App Scanning

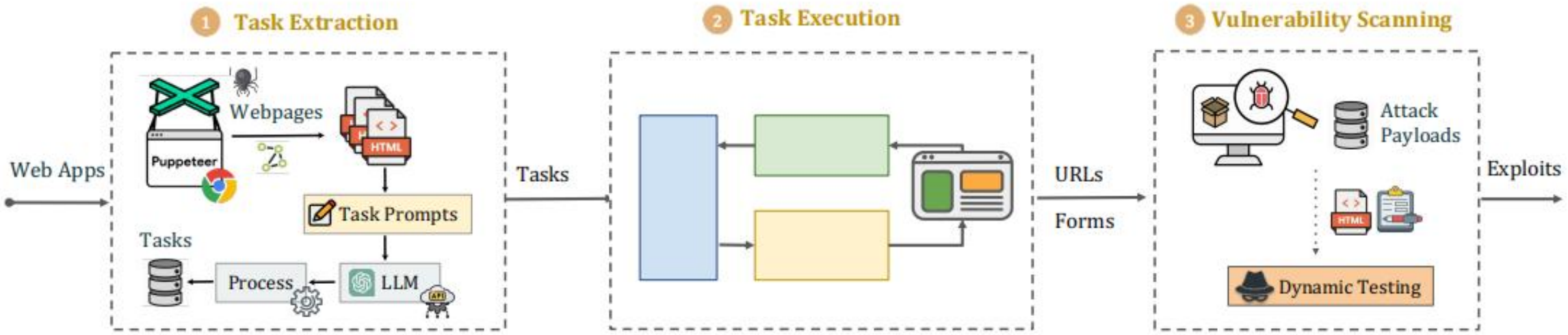
算法原理 TIPO

T	目标	在黑盒条件下，利用LLM自动生成并执行任务驱动的Web工作流
I	输入	1个待测Web应用种子URL，可选0~T条任务提示
P	处理	1.利用LLM生成站点任务列表 2.由LLM决策，执行任务并实现多步交互 3.漏洞测试，生成扫描结果
O	输出	输出站点任务列表、每个任务对应的多步执行轨迹、发现的URL与表单集合，以及最终漏洞报告

P	问题	1.扫描器基于广度优先和随机遍历，难以深入多步骤操作流程 2.基于模型的方法难以跨网络应用迁移
C	条件	待测网页具有较复杂的工作流
D	难点	理解网页任务与业务流程，自主完成多步交互并进入传统方法难以到达的深层状态
L	水平	NDSS（CCF A）

算法原理 YURASCANNER

- 核心创新
  - 将扫描器建模为目标驱动智能体
    - 不是看到当前页就反应式决策，而是围绕当前任务目标持续推进
  - 利用LLM理解任务与页面语义
    - 结合任务目标、当前页面和历史动作，让模型预测下一步更合理的操作



算法原理 任务抽取

- 目标：先理解网站能做什么，再决定扫描器要做什么
- 流程
  - 输入一个种子URL，观察网站的主要功能区
  - 浅层爬取一级页面
    - 提取菜单、链接、按钮等页面线索，并定位可交互元素
  - LLM理解页面语义，生成任务目录并排序
    - 按CRUD依赖关系安排执行顺序
    - Create-Read-Update-Delete

```
Use the following list of button labels on the website to generate a list of tasks to complete on the website. Adding items, Editing items and, finally, Deleting items. Keep tasks simple and straight to point. Like, "Add a product", "Send E-mail", "Delete a comment".
```

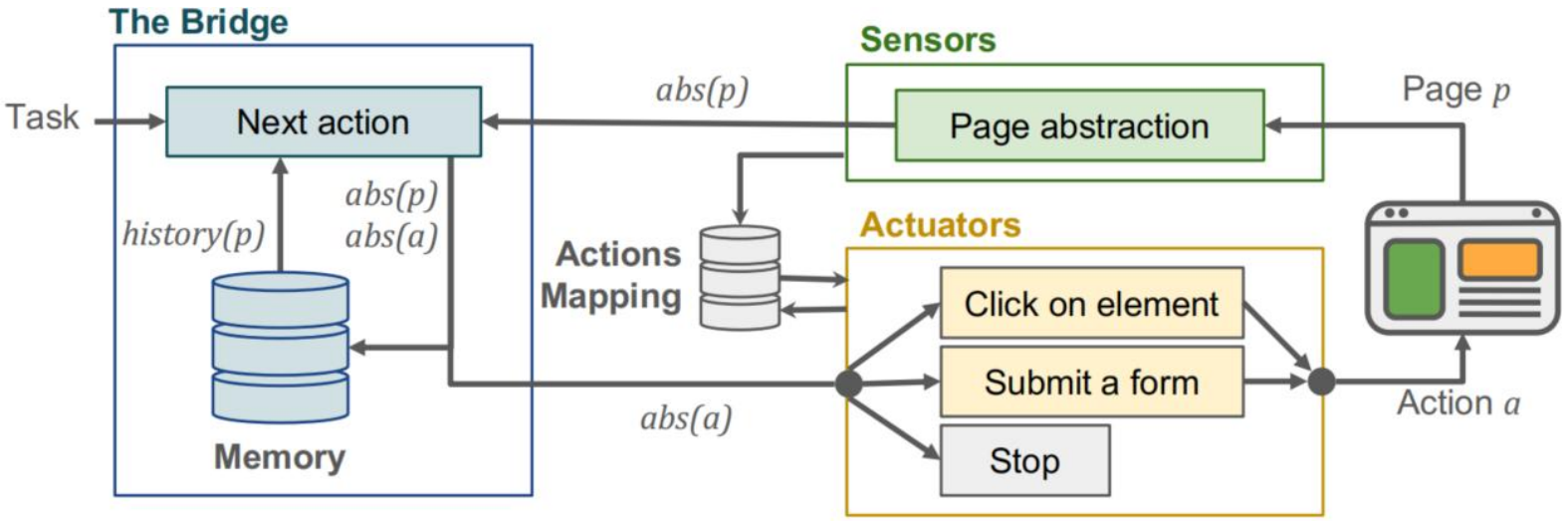
```
{page}
The list of tasks:
```

页面线索：
“Products/Orders/Customers/Add/Edit/Delete”

抽取出的任务：
“Add a new product”
“Edit an existing order”
“Delete a customer record”

算法原理 任务执行

- Sensors：把真实DOM压缩成可供模型理解的页面表示
  - 从DOM和JavaScript事件中提取所有可交互元素
  - 语义提取与抽象化：为每个动作生成一个简短的、富含语义的文本表示，并将整个页面压缩为abs(p)格式
    - 抽象动作格式：<标签名id=数字>语义文本</标签名>
    - 每个动作被分配一个本页唯一整数ID
  - 优点：降低Token消耗、聚焦任务相关动作

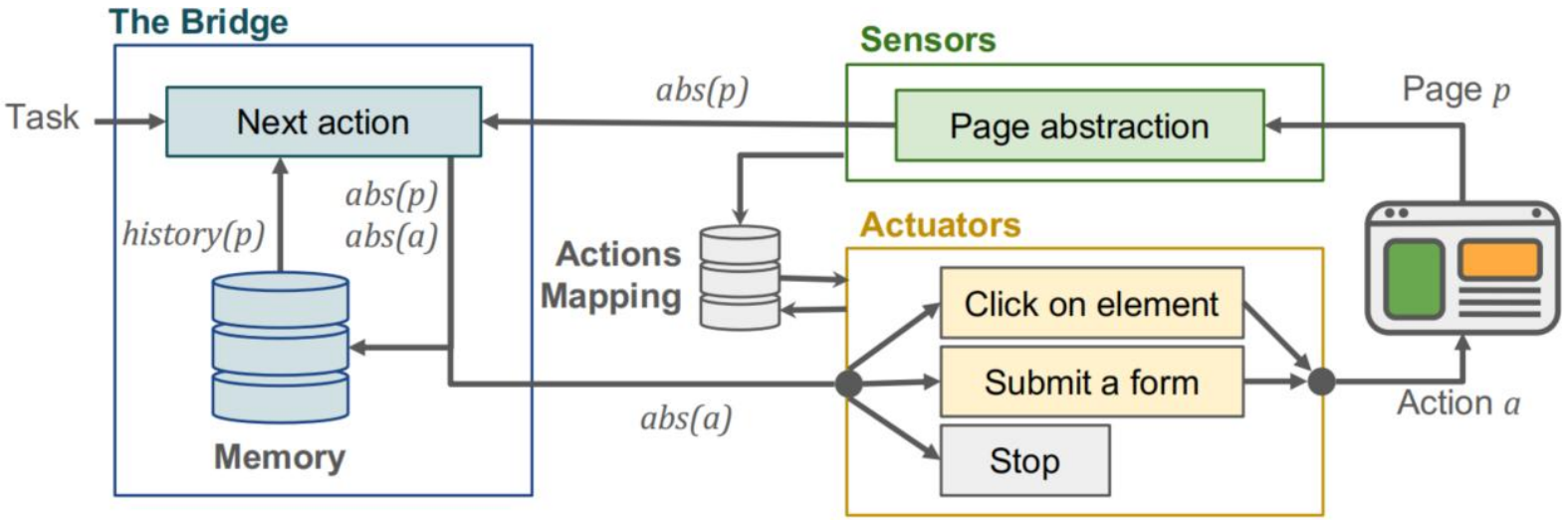


URL:https://example.com/admin
Title:Dashboard-ExampleApp

<buttonid=0>CreateNewUser</button>
<formid=1name="search"action="/search"method="GET">...</form>

算法原理 任务执行

- Bridge：决定当前任务下最合理的下一步
  - 将浏览器状态（abs(p)+历史）、任务描述组织成LLM能理解的提示词
  - 调用大型语言模型（LLM）生成标准指令
    - CLICK X / FILL & SUBMIT FORM X / STOP
  - 维护一个滑动窗口式记忆，记录最近6个已执行的动作



提示词

1角色设定与指令部分
角色分配：网络操作助手
可用命令说明
输入信息清单

2示例部分
示例抽象页面
示例任务描述
示例历史步骤
期望的示例命令

3实际查询部分
角色分配：网络操作助手
插入真实的当前抽象页面abs(p)。
插入真实的当前任务描述task_descr
插入最近六步的历史动作记录last_steps
YOURCOMMAND：

算法原理 任务执行

- Actuators：把高层决策变成真实浏览器操作
  - 通过ID定位真实DOM
  - 操控Puppeteer执行动作
    - STOP：终止任务
    - CLICK：执行点击
    - FILL & SUBMIT FORM
    - 文本类字段：调用FormGPT生成值
    - 下拉/复选框：确定性算法，例如永远选第二个
- 让LLM只负责决策，不直接负责全部执行细节

```
You are FormGPT, which is an agent for ↩
automatically filling out forms on a web page.
As an input, you are given a form in HTML ↩
representation. Your task is to fill out the ↩
form with fitting values.

You can issue this command multiple times (each ↩
command in a new line):
TYPE X "text" - type the provided text into the ↩
input with id X.
```

```
Here is an example:
FORM TO FILL OUT:
{example_form}
YOUR LIST OF COMMANDS: {example_commands}
```

```
The actual form follows now. Please provide ↩
your list of commands as an answer.
FORM TO FILL OUT:
{current_form}
YOUR LIST OF COMMANDS:
```

实验流程

- 实验对象
  - TE数据集：10个应用，用来人工核验任务生成与执行效果
  - TE+VD数据集：20个应用，用来评估攻击面覆盖和漏洞检测
- 对比方法
  - BlackWidow（2021）
  - BFS
  - Random BFS
- 实验参数：
  - 每个扫描器运行上限4小时
  - 登录/注册单独处理，统一提供凭据或session cookie
  - 作者还对比了GPT-4、Llama3-8B、Gemma2-9B、Gemma2-27B，最终选用GPT-4，因为它在随机抽取的20个任务上执行成功率最高

Dataset	Name	Version	Category
TE+VD	*Redacted*	*Redacted*	*Redacted*
TE+VD	GitLab	16.11.2-ce.0	VCS
TE+VD	OpenCart	4.0.2-3	eCommerce
TE+VD	Redmine	5.1.2	Bug tracker
TE+VD	Dolibarr	19.0.2	CRM
TE+VD	Moodle	4.4.0	LMS
TE+VD	MediaWiki	1.41.1	Wiki
TE+VD	OwnCloud	10.14.0	Media sharing
TE+VD	LimeSurvey	6.5.3	Polls
TE+VD	phpBB	3.3.11	Forum
VD	NextCloud	29.0.1	Media sharing
VD	Joomla	5.1.1	CMS
VD	Leantime	3.1.4	Projects
VD	EspoCRM	8.2.5	CRM
VD	iTop	3.1.1	IT
VD	MintHCM	4.0.4	HCM
VD	Mautic	5.0.4	Marketing
VD	GLPI	10.0.15	Assets
VD	Silverpeas	6.3.5	Web Portal
VD	Monica	4.1.2	CMS
	WordPress	6.5.3	CMS

实验结果 任务执行效果

- 一共生成2361个任务，其中1818个是有效任务，有效率约77%
- 在这1818个有效任务里：
  - 667个完全成功
  - 448个部分成功
  - 合计1115个成功或部分成功，占61.3%
- 失败主要来自两类问题：
  - 状态缺失/no action占主要部分
  - 步骤偏离预期约占25%左右
  - 很多任务有前置依赖，前一步没完成，后续任务就会连锁失败

App	Success	Partial	Failed			Total
			Missing state	*Deviated*	*No action*	
Redacted	76	51	8	27	58	220
OpenCart	50	45	15	4	2	116
GitLab	73	31	77	0	2	183
Moodle	54	42	38	6	3	143
MediaWiki	63	26	18	30	59	196
OwnCloud	30	18	31	25	22	126
phpBB	99	79	20	28	20	246
LimeSurvey	30	38	4	9	35	116
Dolibarr	75	78	24	21	11	209
RedMine	117	40	52	22	32	263
Total	667	448	287	172	244	1,818

实验结果 攻击面覆盖收益

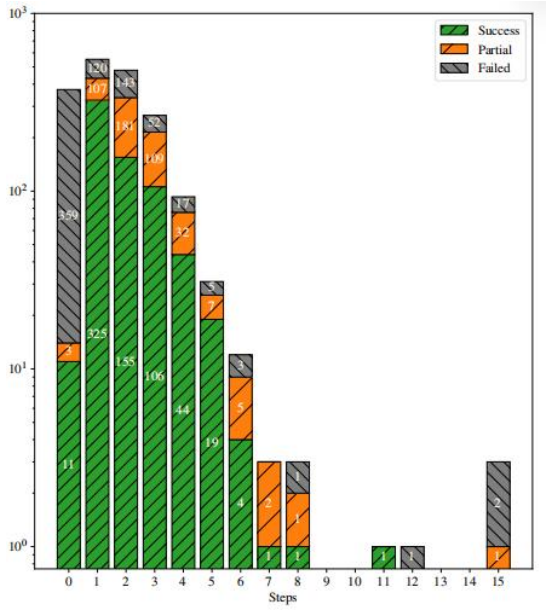
- 新发现表单占联合攻击面的35.46%
- 新发现URL占联合攻击面的25.7%
- 相比基线，平均增幅约为：
  - +55.19%（Forms）
  - +35.16%（URLs）
- 与其他工具的重叠区域并不大，平均只有：
  - 18.3%（Forms）
  - 16.9%（URLs）
- 这说明它和传统扫描器发现的是不同类型的攻击面

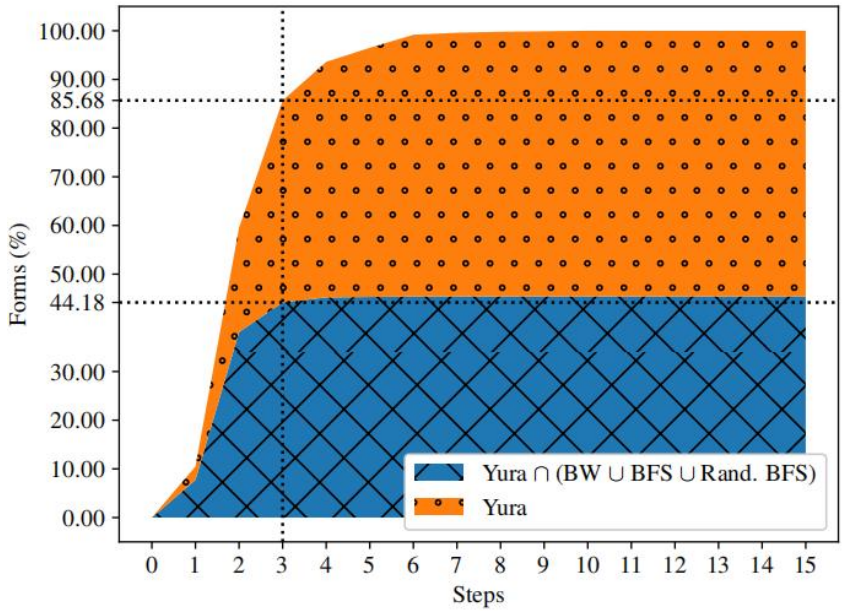
App	Metric	Black Widow \B	Black Widow ∩	Black Widow B\	BFS \B	BFS ∩	BFS B\	Rand. BFS \B	Rand. BFS ∩	Rand. BFS B\
Dolibarr	Forms	28	39	37	39	28	60	37	30	104
	URL	589	219	1,119	635	173	2,776	616	192	2,128
GitLab	Forms	20	43	60	28	35	63	25	38	73
	URL	116	317	334	107	326	218	97	336	260
LimeSurvey	Forms	14	30	36	21	23	9	36	8	2
	URL	403	227	374	420	210	82	520	110	15
MediaWiki	Forms	25	11	77	19	17	70	22	14	96
	URL	284	251	1,481	284	251	998	273	262	2,292
Moodle	Forms	132	123	101	112	143	273	119	136	207
	URL	1,229	643	2,040	1,072	800	5,723	1,115	757	4,879
OpenCart	Forms	82	1	184	78	5	52	71	12	72
	URL	394	90	217	378	106	399	266	218	901
Redacted	Forms	117	15	101	122	10	31	107	25	108
	URL	346	260	256	373	233	170	199	407	476
OwnCloud	Forms	2	10	52	2	10	29	1	11	47
	URL	33	125	134	27	131	148	25	133	161
phpBB	Forms	168	12	114	164	16	102	163	17	119
	URL	796	378	1,017	718	456	315	765	409	489
RedMine	Forms	44	55	29	50	49	50	50	49	80
	URL	286	189	54	287	188	73	284	191	120

Metric	Black Widow \B Tot.	Black Widow \B %	Black Widow ∩ Tot.	Black Widow ∩ %	Black Widow B\ Tot.	Black Widow B\ %	BFS \B Tot.	BFS \B %	BFS ∩ Tot.	BFS ∩ %	BFS B\ Tot.	BFS B\ %	Rand. BFS \B Tot.	Rand. BFS \B %	Rand. BFS ∩ Tot.	Rand. BFS ∩ %	Rand. BFS B\ Tot.	Rand. BFS B\ %
Forms	632	35.8	339	19.2	791	44.8	635	37.1	336	19.6	739	43.2	631	33.5	340	18.0	908	48.3
URLs	4,476	31.5	2,699	19.0	7,026	49.4	4,301	23.7	2,874	15.9	10,902	60.3	4,160	22.0	3,015	15.9	11,721	62.0
Forms / URL	0.14	-	0.13	-	0.11	-	0.15	-	0.12	-	0.07	-	0.15	-	0.11	-	0.08	-

实验结果 更深的状态，更深的页面

- 任务大多是短流程，但最长可达到15步
- 在YuraScanner发现的表单中：
  - 约44.18%也能被其他工具发现，这些大多位于3步以内的浅层区域
  - 到3步时，YuraScanner已累计覆盖自己发现表单的85.7%
  - 还有14.3%的表单位于3步以上的更深状态，作者认为这些基本超出了现有工具可达范围
  - 传统方法更擅长浅层覆盖，而任务驱动方法更擅长深层覆盖，两者并非完全替代关系





实验结果 漏洞检测结果

- 在20个应用上，作者将YuraScanner与BlackWidow进行XSS检测对比
  - 共得到15个漏洞报告
  - 去重后共有13个唯一漏洞
  - 这些漏洞全部为zero-day
  - 其中12个由YuraScanner发掘
  - 报告主要出现在Redacted、Moodle、Leantime三个应用中
- 这些漏洞大多位于距主页2到4次点击的位置，其中更深的位置正是传统扫描更难进入的区域

Name	Tot.	Uniq.	Yura		BW	
			Stored	Reflected	Stored	Reflected
Redacted	12	11	4	7	-	1
GitLab	-	-	-	-	-	-
OpenCart	-	-	-	-	-	-
Redmine	-	-	-	-	-	-
Dolibarr	-	-	-	-	-	-
Moodle	2	1	1	-	1	-
MediaWiki	-	-	-	-	-	-
OwnCloud	-	-	-	-	-	-
LimeSurvey	-	-	-	-	-	-
phpBB	-	-	-	-	-	-
NextCloud	-	-	-	-	-	-
Joomla	-	-	-	-	-	-
Leantime	1	1	-	-	1	-
EspoCRM	-	-	-	-	-	-
iTop	-	-	-	-	-	-
MintHCM	-	-	-	-	-	-
Mautic	-	-	-	-	-	-
GLPI	-	-	-	-	-	-
Silverpeas	-	-	-	-	-	-
Monica	-	-	-	-	-	-

Black-Box Detection of Broken-Access-Control Vulnerabilities in Web Applications

算法原理 TIPO

T	目标	提升对修改类MBAC漏洞的检测准确性与覆盖能力
I	输入	一个目标Web应用、目标URL，以及至少3个测试账号
P	处理	1.IDDG构建：构建页面间数据依赖图 2.MBAC检测：在数据依赖图基础上进行漏洞检测 3.根据token变化判断漏洞类型
O	输出	BAC漏洞集合V，共包含K个漏洞条目；每个条目可给出对应URL、HTTP方法、参数、漏洞类型（RBAC/MBAC）及可复现的请求线索

P	问题	传统黑盒BAC扫描主要依赖响应相似度，对RBAC较有效，但对MBAC常因修改结果不在当前响应中而产生较高误报与漏报
C	条件	黑盒Web应用、多账号可登录测试环境、可被爬取的页面导航关系
D	难点	从大量请求/响应中高效定位反馈页
L	水平	ACM CCS 2025，CCF A

知识基础 传统BAC黑盒检测

- BAC
  - RBAC（读取）：攻击者读到了本不该访问的数据
  - MBAC（修改）：攻击者修改、删除或新增了本不该操作的数据
  - MBAC更容易破坏业务完整性
- 漏洞判定方法：比较响应是否相似
  - 传统BAC黑盒扫描通常会配置受害者账号和攻击者账号
  - 将受害者请求替换为攻击者身份重放，若攻击者响应与受害者高度相似，则判为可疑漏洞
  - 擅长发现RBAC，而不擅长发现MBAC

订单提交页/订单列表页

OWASP Top 10 2025

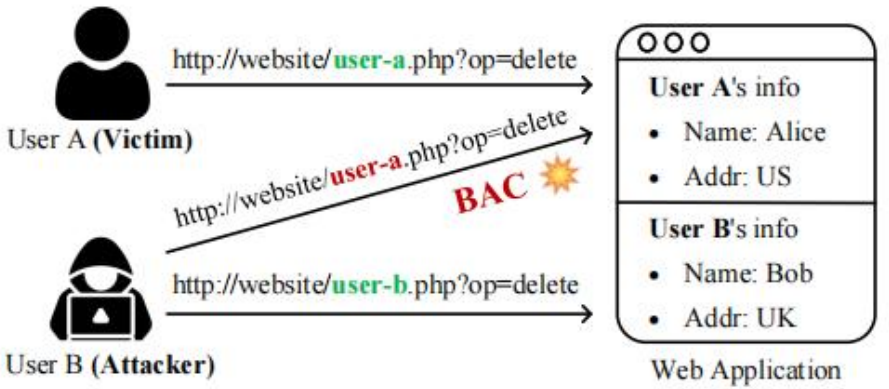
Top 10:2025 List

1. A01:2025 - Broken Access Control
2. A02:2025 - Security Misconfiguration
3. A03:2025 - Software Supply Chain Failures
4. A04:2025 - Cryptographic Failures
5. A05:2025 - Injection
6. A06:2025 - Insecure Design
7. A07:2025 - Authentication Failures
8. A08:2025 - Software or Data Integrity Failures
9. A09:2025 - Security Logging and Alerting Failures
10. A10:2025 - Mishandling of Exceptional Conditions

Web应用程序最关键十大安全风险

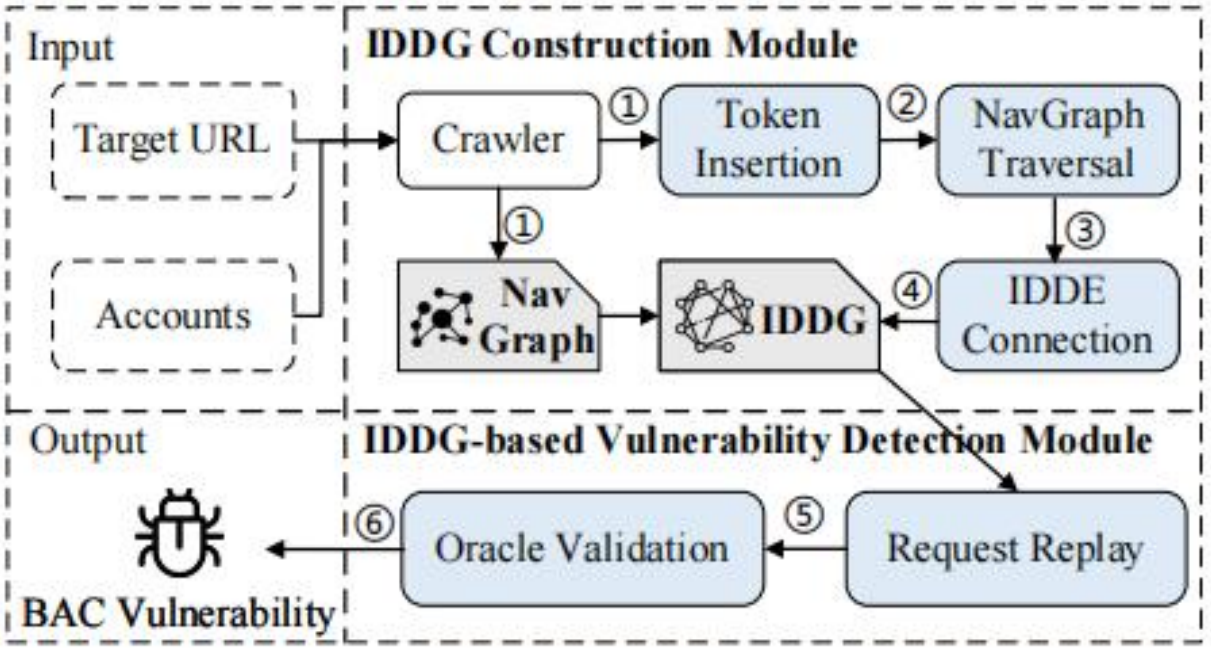
知识基础 传统BAC黑盒检测

- 多账号会话（Victim/ Attacker）
  - 受害者负责探索，攻击者负责攻击
  - 通过不同的Cookie/Session，模拟不同身份对同一业务的访问
- 请求重放（Request Replay）
  - 将受害者原本发出的业务请求，用攻击者身份重新发送
  - 若攻击者也能让受害者数据发生变化，就可能存在BAC漏洞
- 页面导航图
  - 由BlackWidow构建，本质仍是浅层随机遍历爬虫
  - 记录页面之间的跳转关系，帮助回到正确业务状态



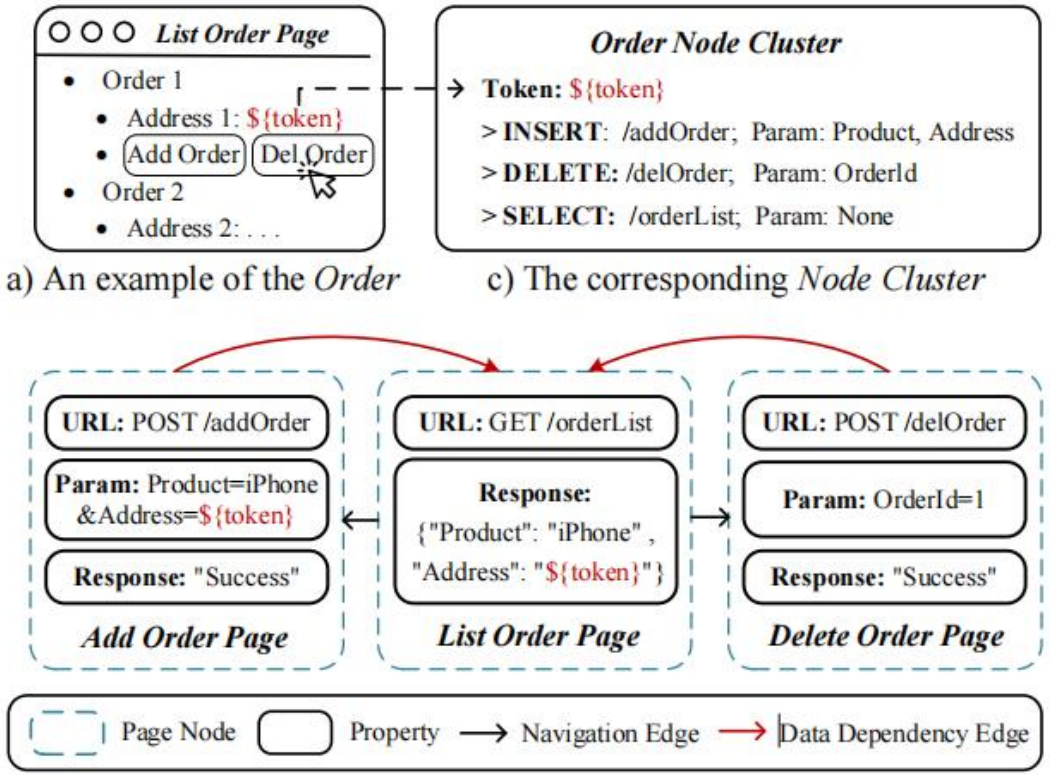
算法原理 基本框架

- 核心思想：修改页本身不一定告诉你改没改成功，但相关状态页会泄露结果
- Step1：构建IDDG
  - Inter-page Data Dependency Graph，页面间数据依赖图
  - 表示Web应用中修改页面与状态页面之间数据依赖关系的一种有向图结构
- Step2：BAC漏洞检测
  - 双预言机策略
  - 判断MBAC/RBAC漏洞



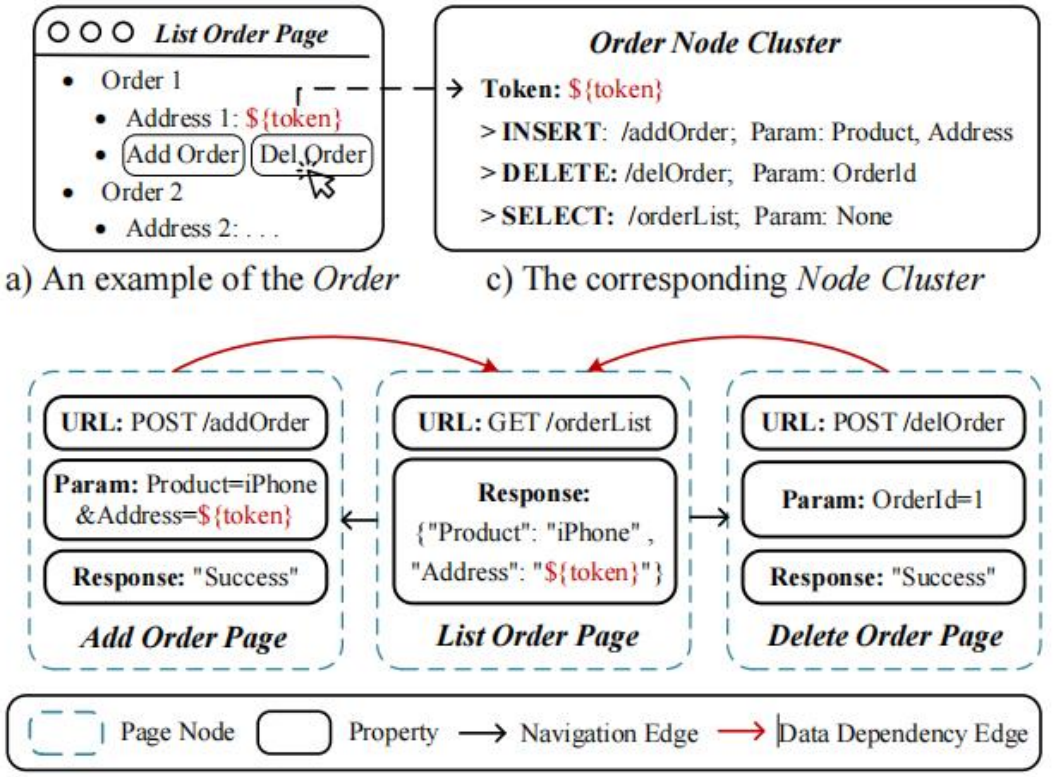
算法原理 IDDG

- 节点：
  - 修改页面节点：产生数据变更的页面，如提交订单、修改个人信息、删除记录等
  - 状态页面节点：能够显示数据当前状态的页面，如订单列表、个人信息展示页、数据详情页等
- 边
  - 数据依赖边（IDDE）：表示一个修改页面的操作结果会在某个状态页面上体现出来
    - INSERT型、UPDATE型、DELETE型



算法原理 IDDG流程

- 1、导航图构建
  - 爬虫遍历Web应用，记录页面节点与跳转关系
  - 仅使用受害用户凭证，捕获所有可达页面及其导航边
- 2、修改请求拦截与Token注入
  - 拦截修改页面的请求，将非结构化参数值替换为伪随机唯一Token
  - Token形式：6位小写字母随机串（熵足够高），用于后续追踪

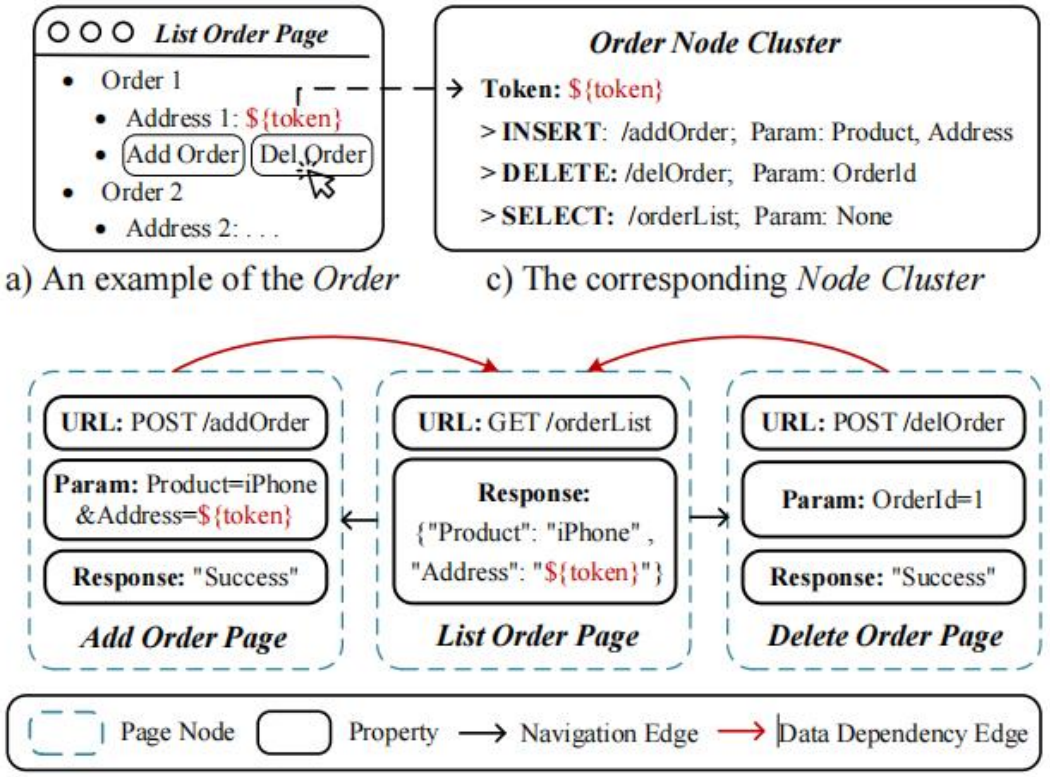


算法原理 IDDG流程

- 3、分层导航图遍历
  - 优先访问更可能是状态页的页面，查找Token出现/消失的位置
  - 评分函数S(m, r)综合URL相似度、参数相关性、导航距离

$$S(m,r) = \frac{w_1}{1+\mathrm{Sim}(m,r)} + w_2 \cdot \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(p_i \in r) + \frac{w_3}{\mathrm{Dist}(m,r)}$$

- 4、数据依赖边建立
  - 根据Token在状态页面中的行为确定IDDE类型
  - INSERT（出现）、UPDATE（新旧替换）、DELETE（消失）



算法原理 漏洞检测

- MBAC检测
  - 攻击者重放受害者的修改请求
  - 沿IDDE找到对应状态页
  - 在受害者界面根据token变化判断是否越权成功
  - 三种判定规则：
    - INSERT：新token出现→可疑漏洞
    - UPDATE：新token出现且旧token消失→可疑漏洞
    - DELETE：旧token消失→可疑漏洞
- RBAC检测
  - RBAC还是沿用响应相似度比较
  - 攻击者重放读取请求，看和受害者响应是否足够相似

实验流程

- 4个RQ
  - RQ1：BACScan是否有效
  - RQ2：相比现有方法是否更强
  - RQ3：效率如何
  - RQ4：IDDG构建是否可靠
- 实验对象
  - 20个真实开源Web应用
  - 14个testing set
  - 6个ground-truth set，含44个已知BAC漏洞
- 实现与环境
  - Python+Playwright
  - 4116行代码
  - Ubuntu22.04，64核CPU，256GB内存

Testing Set	# CVEs / Vulns[1]
Mall-swarm	0 + 1 / 1 + 1
Newbee_mall	2 + 2 / 2 + 3
Invoiceninja	1 + 0 / 2 + 0
PrestaShop	0 + 0 / 0 + 0
XMall	5 + 4 / 6 + 4
SpringBlade	0 + 0 / 0 + 0
Ruoyi	0 + 0 / 0 + 0
Joomla	0 + 0 / 0 + 0
Ampache	1 + 0 / 2 + 0
OpenEMR	4 + 2 / 5 + 6
Supermarket	5 + 2 / 6 + 3
PhpBB	0 + 0 / 0 + 0
Apache_inlong	2 + 0 / 4 + 2
Webid	2 + 2 / 5 + 2
Total	22 + 13 / 34 + 22
Ground Truth Set	**# Known Vulns[2]**
Memos-0.9.0	4 + 5
WordPress_SPM-4.57	6 + 1
Snipe_it-5.0.3	0 + 3
Lunary-1.2.7	6 + 6
MyBloggie-2.1.4	2 + 0
Collabtive-2.1	6 + 5
Total	24 + 20

实验结果 RQ1 BACScan是否有效

- 总实验结果
  - 54个0-day
  - 35个已知漏洞
  - 开发者确认39个漏洞，收到35份CVE确认函，其中包含22个MBAC漏洞和13个RBAC漏洞
- Ground-truth上的结果
  - MBAC：Precision 100%，Recall 83.33%，误报（0次）和漏报（4次）
  - RBAC：Precision 83.33%，Recall 75.00%
- Testing set上的结果
  - 33个MBAC 0-day，且都被人工验证可利用
  - 21个RBAC 0-day，准确率80.77%

实验结果 RQ2 相比现有方法是否更强

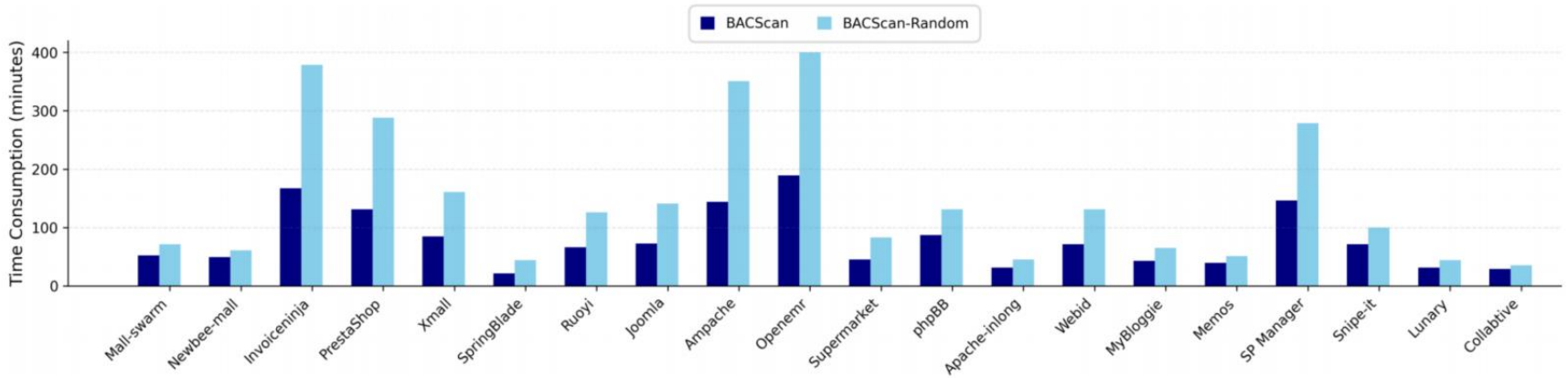
- 基线方法
  - EvoCrawl（2023）
  - BurpSuite+Autorize
- 对比结论
  - 在MBAC检测上，BACScan明显领先
- MBAC：
  - EvoCrawl：Precision 51.52%，Recall 29.82%
  - BurpSuite：Precision 62.00%，Recall 54.39%
  - BACScan：Precision 100.00%，Recall 92.98%
- RBAC：
  - BACScan也最好，但优势没MBAC那么夸张

Baselines	MBAC Vulnerability				
	TP	FP	FN	Prec (%)	Recall (%)
EvoCrawl	17	16	40	51.52%	29.82%
BurpSuite	31	19	26	62.00%	54.39%
BACScan	53	0	4	100.00%	92.98%

Baselines	RBAC Vulnerability				
	TP	FP	FN	Prec (%)	Recall (%)
EvoCrawl	23	9	18	71.88%	56.10%
BurpSuite	35	15	6	70.00%	85.37%
BACScan	36	8	5	81.82%	87.80%

实验结果 RQ3 效率

- 平均耗时
  - BACScan：1.1小时/应用
  - BACScan-Random：2.3小时/应用
- 结论
  - 层次化遍历策略带来109.10%的性能改善
  - 在大应用上优势更明显
  - 随机策略在OpenEMR上甚至会超时



实验结果 RQ4 IDDG结构可信度率

- IDDG是反馈驱动预言机的基础
- 关键数字
  - 共构建1135个IDDE
  - 抽样10%（113个）人工检查
  - 仅4个错误，错误率3.54%
- 进一步分析
  - 331 INSERT
  - 452 UPDATE
  - 352 DELETE
  - DELETE型IDDE虽只占31.40%，但DELETE型MBAC占总检测漏洞的48.89%

特点总结与未来展望

特点总结与未来展望

- 特点总结

算法	YuraScanner	BACScan
优势	1.把Web扫描从无目标页面遍历推进到面向任务的工作流执行 2.更擅长发现深层攻击面	1.解决传统方法难以检测的MBAC 2.一种可信的证据验证机制，基于反馈驱动的oracle 3.成本低、效率快、实证结果较强
劣势	1.浅层攻击面覆盖度不足 2.执行稳定性、状态依赖和LLM成本	1.依赖随机遍历爬虫覆盖率，需要人工补充路径 2.仅能检测直接依赖关系

- 未来展望
  - 从页面遍历走向业务流程理解
  - 从单步漏洞检测走向多步骤攻击链挖掘
  - 从规则驱动走向大模型驱动的智能决策

参考资源

[1]. Stafeev A, Recktenwald T, De Stefano G, Khodayari S, Pellegrino G. YuraScanner: Leveraging LLMs for Task-driven Web App Scanning. 32nd Annual Network and Distributed System Security Symposium (NDSS 2025) [C]. San Diego, CA, USA: The Internet Society, 2025: 1-16.

[2]. Liu F, Zhang Y, Li E, Meng W, Shi Y, Wang Q, Wang C, Lin Z, Yang M. BACScan: Automatic Black-Box Detection of Broken-Access-Control Vulnerabilities in Web Applications. Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25) [C]. New York, NY, USA: ACM, 2025: 1320-1333.

[3]. Eriksson B, Pellegrino G, Sabelfeld A. Black Widow: Blackbox Data-driven Web Scanning. 2021 IEEE Symposium on Security and Privacy (SP) [C]. Los Alamitos, CA, USA: IEEE Computer Society, 2021: 1125-1142.

道德经

道可道，非常道。名可名，非常名。无名天地之始。有名万物之母。故常无欲以观其妙。常有欲以观其徼。此两者同出而异名，同谓之玄。玄之又玄，众妙之门。

谢谢！