Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

大模型指导的协议模糊测试

硕士研究生 徐菊彬

2025年05月18日

问题回溯

- 问题回溯
  - 准备不充分，过多陈述思考内容
  - 两篇算法衔接生硬，创新点引入不自然
- 相关内容
  - 2024.09.03 张浩然《大模型赋能的模糊测试用例生成技术》
  - 2024.12.23 徐菊彬《网络未知协议逆向技术》

内容提要

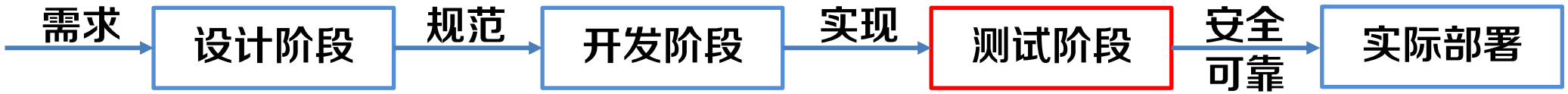
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - CHATAFL
  - ChatHTTPFuzz
- 特点总结与工作展望
- 参考文献

预期收获

- 预期收获
  - 1. 了解大模型指导的协议模糊测试的基本概念和研究方向
  - 2. 理解两种大模型指导的协议模糊测试的基本原理
  - 3. 了解现有方法的缺陷以及未来发展方向

内涵解析与研究目标

- 内涵解析
  - 网络协议：计算机网络中进行数据交换的规则，保证节点之间的相互通信



  - 模糊测试：生成大量随机突变测试用例，旨在触发软件程序中的异常运行时行为
    - 协议模糊测试：通信复杂性高、测试环境相对受限
  - 大语言模型：具有大规模参数，使用自监督学习或半监督学习对大量未标记文本进行训练，具备复杂的语言理解能力
- 研究目标
  - 利用LLM自动化学习和理解协议规范，指导协议模糊测试策略
  - 提高协议模糊测试效率和有效性

研究背景与意义

- 研究背景
  - 网络协议是互联网通信的基础，协议实现的复杂性使得开发过程中易引入漏洞，导致严重的安全问题
    - Heartbleed漏洞存在于OpenSSL的TLS协议实现中，在实现TLS的心跳扩展时没有对输入进行适当验证（缺少边界检查），导致缓冲区过读，影响了全球超过17%的服务器
  - 传统模糊测试在协议场景中具有局限性，需针对协议规范设置结构化测试用例，实现状态空间的有效探索
- 研究意义
  - 降低协议模糊测试对专业人员的依赖度与人力成本，提高测试效率
  - 通过LLM解析协议规范，自动化生成高质量测试用例，提高测试覆盖率

研究历史与现状

2023

Hu等人认为传统基于变异和基于生成的模糊测试方法受限于结构化输入数据，提出利用生成式大语言模型生成合规输入改进灰盒模糊测试CHATFUZZ，根据初始种子利用ChatGPT生成变体，提高输入质量

2024

Ma等人提出首个针对Matter设备的模糊测试工具mGPTFuzz，利用自然语言处理能力理解和生成测试数据，实现了复杂协议测试用例的自动生成，减少了协议结构的人工干预，根据测试结果的反馈调整输入生成策略，提高了测试效率和准确性

2024

Meng等人提出大模型指导的模糊测试方法CHATAFL，将大语言模型与经典协议模糊测试器AFLNet相结合，利用LLMs学习协议规范文档RFC，指导生成合规且多样化的输入数据，同时设置覆盖高原检测机制，提高了覆盖率和漏洞发现效率

2024

Wang等人使用大型语言模型来提取IoT设备的协议信息并进行设备相应推理，实现LLMIF模糊测试算法。识别和理解网络协议中的关键字段，生成有针对性的测试用例输入，并为针对Zigbee协议设计了高效的模糊测试模型

2024

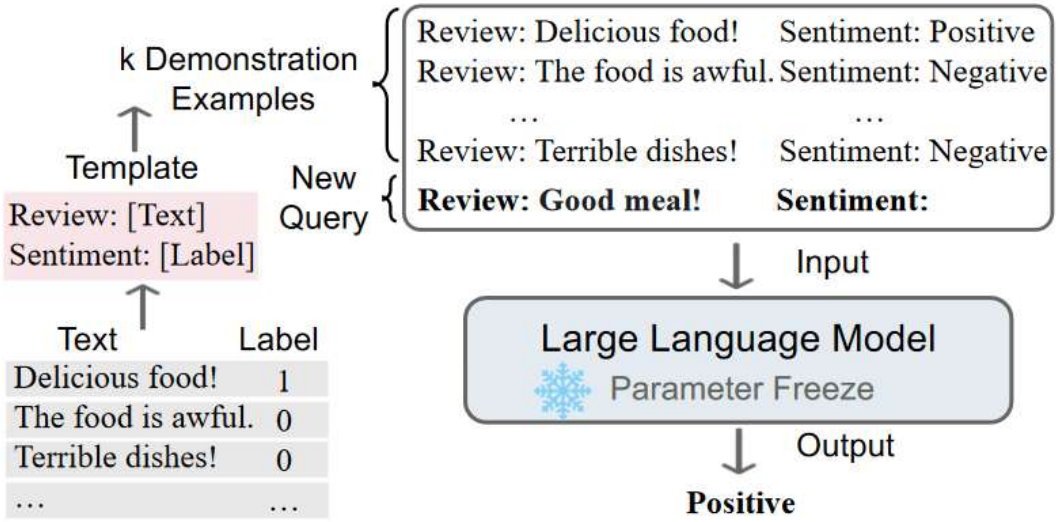
Cheng等人提出了一个具有消息语法理解的协议模糊方法MSFuzz，从协议实现源代码中提取关键代码片段，利用LLM的代码理解能力提取消息语法并构建消息语法树。然后利用这些语法树来扩充种子库和指导种子变异，从而提高测试用例的有效性，进而提高模糊测试的效率

2025

Yang等人提出针对测试物联网设备中的HTTP协议而定制的模糊测试框架ChatHTTPFuzz，利用LLMs对数据字段进行注释、构造结构化种子模板、确定最佳变异位置的能力。然后分析服务代码逻辑，生成相应的数据包，从而提高种子输入的质量和数量，同时扩展模板和字段值的变异空间

- 语境学习（In-Context Learning，ICL）
  - 不需要额外参数训练，输入中提供自然语言形式的任务实例（语境演示）
  - 通过离散的文本提示（输入或输出对）引导模型理解任务的逻辑
  - 适用于零样本（仅指令）或少样本场景

- 有效性说明
  - 在贝叶斯框架中，ICL 被解释为隐式贝叶斯推理，其中模型通过识别示例之间的共享潜在概念来执行 ICL，利用注意力机制聚合多示例信息，实现“隐式概念归纳”



算法原理 CHATAFL

Large Language Model guided Protocol Fuzzing

算法原理 TIPO

T	目标	大模型指导下生成合规测试用例，触发协议服务端bug
I	输入	待测程序：6种基于文本的网络协议实现（RTSP、FTP、SIP、SMTP、DAAP协议服务端） 初始种子：从ProFuzzBench捕获的测试服务器与客户端的网络流量 自定义模糊测试时间
P	处理	1. 语法引导变异 2. 种子语料增强 3.覆盖高原突破
O	输出	1. 覆盖结果：状态覆盖、转移覆盖、代码覆盖 2. 漏洞报告

P	问题	缺乏机器可读的协议规范，导致无法有效生成合法消息或探索深层状态
C	条件	协议具有公开的自然语言规范（如 RFC 文档），且已被 LLM 训练数据覆盖
D	难点	LLM提示词工程，生成符合机器可读格式的语法和状态信息
L	水平	NDSS 2024 CCF A

算法原理 创新说明

- 传统基于突变的协议模糊测试
  - 有效性严重受限于初始种子质量，预先记录的消息序列将难以涵盖协议规范中所讨论的协议状态和输入结构的巨大多样性
  - 变异未知消息结构难度大，在没有关于消息结构的机器可读信息的情况下，模糊器无法对种子消息进行结构上的有效更改
  - 未知状态空间探索困难，如果没有关于状态空间的机器可读信息，模糊器就不能识别当前状态，也不能被引导去探索以前看不见的状态
- CHATAFL
  - 利用LLM学习公开RFC文档内容，实现自动化解析协议规范
  - 提取协议的机器可读语法和状态转移逻辑
  - 根据当前通信历史（消息序列与服务器响应）实时推断协议状态，并生成可触发新状态的消息

算法原理 CHATAFL

- 基于AFLNET使用3个LLM驱动组件进行增强
  - 语法引导变异
    - LLM输出所需格式的协议文法
    - 实现语法指导的突变
  - 种子语料增强
    - 扩展初始种子语料库
  - 覆盖高原突破
    - 动态状态推断与引导

Algorithm 1: LLM-guided Protocol Fuzzing

Input: P_0: protocol implementation
Input: p: protocol name
Input: C: initial seed corpus
Input: T: total fuzzing time
Output: C: final seed queue
Output: $C_\times$: crashing seeds

```
1  P_f ← INSTRUMENT (P_0)
2  Grammar G ← CHATGRAMMAR (p)
3  C ← C ∪ ENRICHCORPUS (C, p)
4  PlateauLen ← 0
5  StateMachine S ← ∅
6  repeat
7      State s ← CHOOSESTATE (S)
8      Messages M, response R ← CHOOSESEQUENCE (C, s)
9      ⟨M_1, M_2, M_3⟩ ← M    (i.e., split M in subsequences,
         s.t. M_1 is the message sequence to drive P_f to arrive
         at state s, and message M_2 is selected to be mutated).
10     for i from 1 to ASSIGNENERGY (M) do
11         if PlateauLen < MaxPlateau then
12             if UNIFORMRANDOM () < ϵ then
13                 M_2' ← GRAMMARMUTATE (M_2, G)
14                 M' ← ⟨M_1, M_2', M_3⟩
15             else
16                 M' ← ⟨M_1, RANDMUTATE (M_2), M_3⟩
17             end
18         else
19             M_2' ← CHATNEXTMESSAGE (M_1, R)
20             M' ← ⟨M_1, M_2', M_3⟩
21             PlateauLen ← 0
22         end
23         R' ← SENDTOSERVER (P_f, M')
24         if ISCRASHES (M', P_f) then
25             C_✗ ← C_✗ ∪ {M'}
26             PlateauLen ← 0
27         else if ISINTERESTING (M', P_f, S) then
28             C ← C ∪ {(M', R')}
29             S ← UPDATESTATEMACHINE (S, R')
30             PlateauLen ← 0
31         else
32             PlateauLen ← PlateauLen + 1
33         end
34     end
35 until timeout T reached or abort-signal
```

协议语法提取

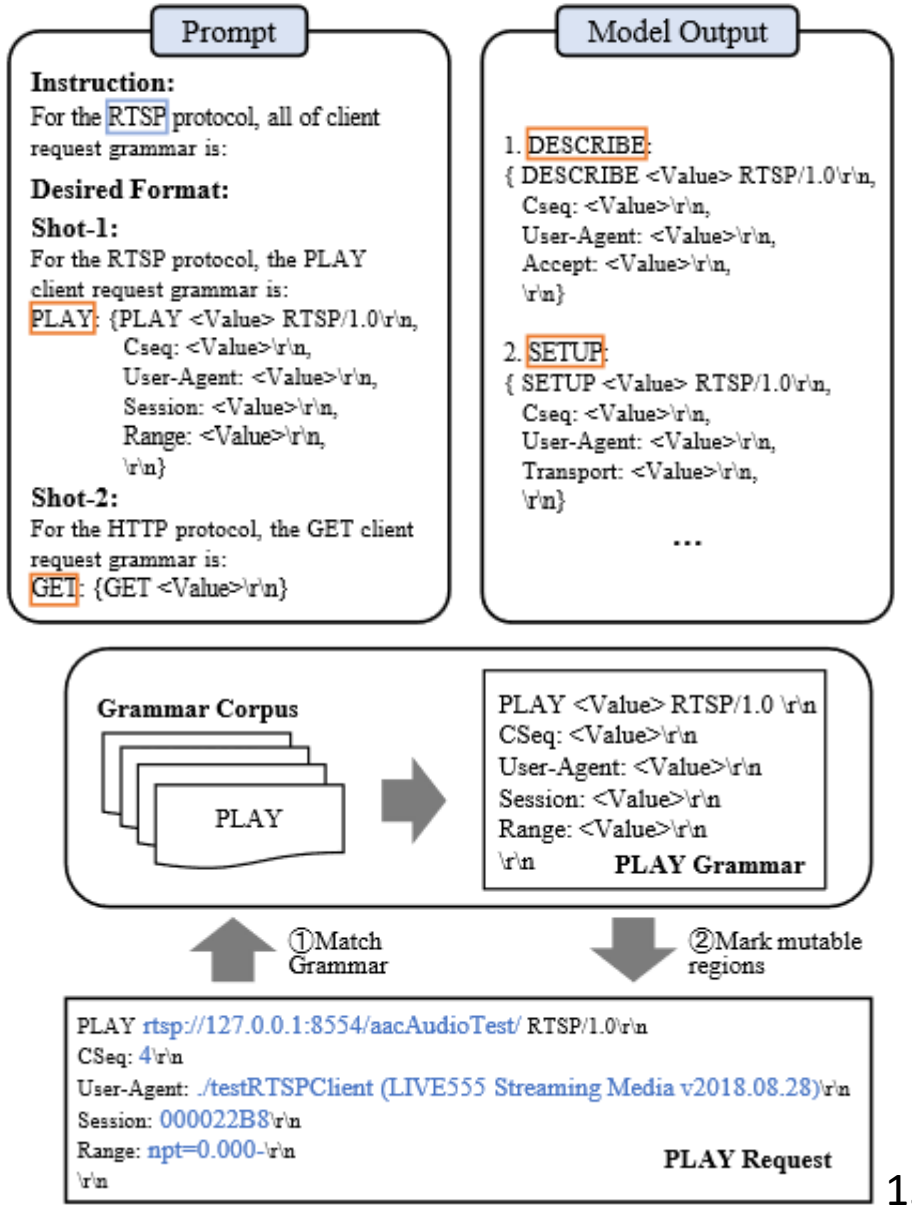
种子语料增强

语法引导变异

覆盖高原突破

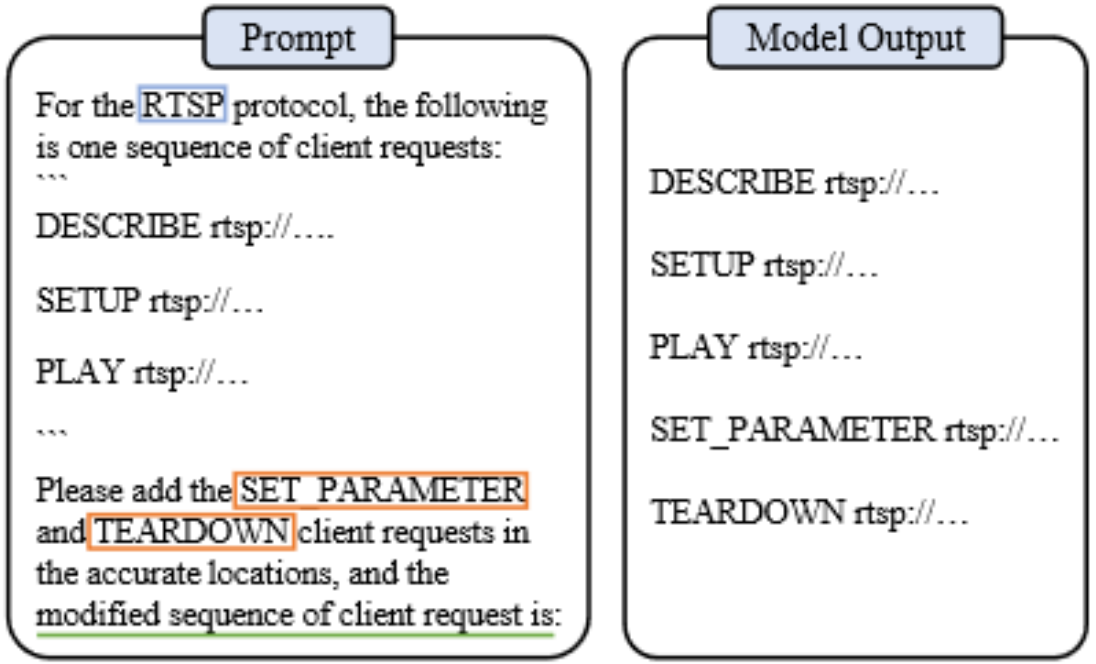
算法原理 CHATAFL

- 语法引导变异
  - 解决消息结构未知问题，生成符合协议语法的有效测试用例
- 原理：
  - LLM 语法提取
    - 通过少样本语境学习让 LLM 生成协议消息的机器可读语法
  - 结构化变异
    - 消息划分为固定字段（如 PLAY）和可变字段根据语法将（如<Value>）
    - 仅对可变字段进行变异（如替换 URL、修改 CSeq 值），确保消息格式合法



算法原理 CHATAFL

- 种子语料增强
  - 解决初始种子多样性不足问题，补充缺失的消息类型和状态转移路径
- 原理：
  - 缺失类型检测
    - 对比 LLM 生成的全量消息类型与初始种子中的类型，确定需补充的类型
  - LLM生成新消息序列
    - 提示 LLM 在现有种子中插入缺失类型的消息，并确保符合状态转移逻辑
    - 上下文注入：将服务器响应中的动态信息（如 Session ID）传入提示，解决 LLM 生成消息时的上下文缺失问题



算法原理 CHATAFL

- 覆盖高原突破
  - 覆盖高原：模糊器无法探索新的覆盖范围
    - 多次变异后无缺陷触发、状态率覆盖提高
  - 解决状态空间探索停滞问题，引导模糊器突破覆盖高原模式表示
- 原理：
  - 覆盖高原检测
    - 通过计数器记录连续变异无效次数，超出阈值触发LLM干预
  - LLM状态转移引导
    - 向LLM提供当前通信历史，请求生成可触发新状态的消息

```
11  if PlateauLen < MaxPlateau then
12      if UniformRandom () < ϵ then
13          M2′ ← GrammarMutate (M2, G)
14          M′ ← ⟨M1, M2′, M3⟩
15      else
16          M′ ← ⟨M1, RandMutate (M2), M3⟩
17      end
18  else
19      M2′ ← ChatNextMessage (M1, R)
20      M′ ← ⟨M1, M2′, M3⟩
21      PlateauLen ← 0
22  end
```

Prompt Template

In the [protocol-name] protocol, the communication history between the [protocol-name] client and the [protocol-name] server is as follows:
Communication history:
```
[Put the communication history here]
```

The next client request that can induce the server's state transition to other states is:

Desired format of one real client request:
```
[Put one real message example from the initial seed corpus here]
```

实验设计　数据资源

- 数据资源

Subject	Protocol	#LOC	#Stars	Version
Live555	RTSP	57k	631	31284aa
ProFTPD	FTP	242k	445	61e621e
PrueFTPD	FTP	29k	572	10122d9
Kamailio	SIP	939k	1915	a220901
Exim	SMTP	118k	662	d6a5a05
Forked-daapd	DAAP	79k	1718	2ca10d9

- 对比方法
  - AFLNET（2020）
    - 领域内经典基于突变的协议模糊测试方法
  - NSFuzz（2023）
    - 通过静态分析识别状态变量，并将状态变量值作为模糊器反馈，以最大化状态空间的覆盖

- 评价指标
  - 代码覆盖率
  - 状态空间覆盖率
    - 不同状态的数量（状态覆盖率）
    - 状态之间转换次数（转换覆盖率）
  - Vargha-Delaney效应量度量$\widehat{A_{12}}$
    - 非参数效应量度量方法，用于量化两个独立组之间的差异程度
    - 量化 CHATAFL 与基线工具的性能差异程度

实验结果　对比实验

- **RQ1：状态空间覆盖率**
  - 参数解释
    - **Total：24h内运行过程中3种工具覆盖的状态总数**
    - **Speed-up：实现与基线相同覆盖率用时提升倍数**
    - $\widehat{A_{12}}$：**量化CHATAFL与基线在状态转移覆盖率上的性能**

24h内运行过程中3种工具覆盖的状态总数

Subject	CHATAFL	AFLNET	Improv	NSFUZZ	Improv	Total
Live555	14.20	10.00	41.75%	11.70	21.16%	15
ProFTPD	28.70	22.60	26.84%	24.30	17.81%	30
PureFTPD	27.90	25.50	9.37%	24.00	16.20%	30
Kamailio	17.00	14.00	21.43%	15.10	12.50%	23
Exim	19.50	14.10	38.19%	14.40	35.42%	23
forked-daapd	12.10	8.70	39.74%	8.00	51.39%	13
AVG	-	-	29.55%	-	25.75%	-

24h内运行10次，平均状态转换次数、平均状态发现个数

CHATAFL实现更全面有效的状态空间探索

Subject	CHATAFL	Transition comparison with AFLNET				Transition comparison with NSFUZZ			
		AFLNET	Improv	Speed-up	$\hat{A}_{12}$	NSFUZZ	Improv	Speed-up	$\hat{A}_{12}$
Live555	160.00	83.80	90.98%	228.62×	1.00	90.20	77.38%	63.09×	1.00
ProFTPD	246.70	172.60	42.91%	7.12×	1.00	181.20	36.11%	4.97×	1.00
PureFTPD	281.80	216.90	29.91%	5.61×	1.00	206.10	36.72%	7.94×	1.00
Kamailio	130.00	99.90	30.14%	5.53×	1.00	105.30	23.42%	4.58×	1.00
Exim	108.40	62.70	72.98%	40.27×	1.00	69.50	55.97%	13.25×	1.00
forked-daapd	25.40	21.40	18.65%	1.58×	1.00	20.10	26.52%	1.79×	0.86
AVG	-	-	47.60%	48.12×	-	-	42.69%	15.94×	-

实验结果　对比实验

- **RQ2：代码覆盖率**
  - 使用分支覆盖率进行比较，CHATAFL均优于对比方法

Subject	ChatAFL	Branch comparison with AFLNet				Branch comparison with NSFuzz			
		AFLNet	Improv	Speed-up	$\hat{A}_{12}$	NSFuzz	Improv	Speed-up	$\hat{A}_{12}$
Live555	2,928.40	2,860.20	2.38%	9.61×	1.00	2,807.60	4.30%	21.60×	1.00
ProFTPD	5,143.30	4,763.00	7.99%	4.04×	1.00	4,421.80	16.32%	21.96×	1.00
PureFTPD	1,134.30	1,056.30	7.39%	1.60×	0.91	1,041.10	8.96%	1.60×	1.00
Kamailio	10,064.00	9,404.10	7.02%	12.69×	1.00	9,758.70	3.13%	2.95×	1.00
Exim	3,789.40	3,647.60	3.89%	4.27×	1.00	3,564.30	6.32%	11.33×	0.77
forked-daapd	2,364.80	2,227.10	6.18%	4.63×	1.00	2,331.30	1.43%	1.66×	0.70
AVG	-	-	5.81%	6.14×	-	-	6.74%	10.18×	-

- **RQ3：消融实验**
  - 逐一启用语法引导变异S_A、种子语料增强S_B、覆盖高原突破S_C

Subject	CL0	Enable strategy S_A (CL1)			Enable strategies S_A and S_B (CL2)			Enable all strategies (CL3)		
		Improv	Speed-up	$\hat{A}_{12}$	Improv	Speed-up	$\hat{A}_{12}$	Improv	Speed-up	$\hat{A}_{12}$
Live555	2,860.20	0.28%	1.60×	0.89	1.49% (1.21*pp*)	8.45×	1.00	2.38% (0.89*pp*)	9.61×	1.00
ProFTPD	4,763.00	3.63%	2.45×	0.60	5.27% (1.64*pp*)	3.69×	0.63	7.99% (2.72*pp*)	4.04×	1.00
PureFTPD	1,056.30	6.67%	1.34×	0.61	6.70% (0.03*pp*)	1.36×	0.86	7.39% (0.69*pp*)	1.60×	0.91
Kamailio	9,404.10	0.60%	1.75×	0.96	2.24% (1.64*pp*)	8.92×	1.00	7.02% (4.78*pp*)	12.69×	1.00
Exim	3,647.60	2.36%	2.48×	0.52	2.54% (0.18*pp*)	2.36×	0.58	3.89% (1.35*pp*)	4.27×	1.00
forked-daapd	2,227.10	4.67%	2.48×	0.68	4.93% (0.26*pp*)	2.98×	1.00	6.18% (1.25*pp*)	4.63×	1.00
AVG	-	3.04%	2.02×	-	3.86% (0.82*pp*)	4.63×	-	5.81% (1.95*pp*)	6.14×	-

提高测试用例有效性
拓展输入空间
缩短模糊测试耗时

实验结果 实测性能验证

- **RQ4：Bug发现数量**
  - 使用最新版本的待测协议服务端进行测试
    - **CHATAFL检测到9个明显的未知缺陷**
    - **相同时间内，AFLNET检测到5、6、9共3个缺陷**
    - **相同时间内，NSFuzz检测到5、6、7、9共4个缺陷**

ID	Subject	Version	Bug Description	Potential Security Issue	Status
1	Live555	2023.05.10	Heap use after free in handling PLAY client requests	Remote code execution	CVE-requested, fixed
2	Live555	2023.05.10	Heap use after free in handling SETUP client requests	Remote code execution	CVE-requested, fixed
3	Live555	2023.05.10	Use after return in handling DESCRIBE client requests	Remote code execution	CVE-requested
4	Live555	2023.05.10	Use after return in handling SETUP client requests	Remote code execution	CVE-requested
5	Live555	2023.05.10	Heap buffer overflow in handling stream	Remote code execution	CVE-requested
6	Live555	2023.05.10	Memory leaks after allocating memory for stream parameters	Memory leakage	Reported
7	Live555	2023.05.10	Heap use after free in calling RTPInterface::sendDataOverTCP	Remote code execution	CVE-requested
8	ProFTPD	61e621e	Heap buffer overflow while parsing FTP commands	Remote code execution	CVE-requested, fixed
9	Kamailio	a220901	Memory leaks after allocating memory in parsing config files	Memory leakage	Reported

算法总结

- 算法贡献
  - 引入大语言模型突破协议模糊测试瓶颈
    - 语义级规范提取：利用 LLM 自动解析协议的自然语言规范（如 RFC），生成机器可读的消息语法和状态转移逻辑
    - 动态状态引导：LLM 根据通信历史实时推断协议状态，生成可触发新状态的消息
  - 构建LLM引导的混合模糊测试框架
  - 显著提升覆盖能力与漏洞发现效率
- 算法不足
  - 数据依赖局限，依赖详细的协议规范文档学习协议语法
  - 考虑LLM交互成本
  - 协议类型覆盖范围不足，缺少对二进制协议类型研究

算法原理 ChatHTTPFuzz

ChatHTTPFuzz: large language model-assisted IoT HTTP fuzzing

算法原理 TIPO

T	目标	生成合规http协议测试用例，测试物联网设备http协议bug
I	输入	初始种子：实时捕获的Iot设备http协议流量 待测Iot设备：Cisco、TP-Link等品牌的路由器、VPN等17种设备
P	处理	1.协议解析与种子生成 2.种子模块扩展 3.调度优化
O	输出	待测Iot设备漏洞报告

P	问题	随机变异导致无效数据包多，依赖人工模板，调度策略单一
C	条件	需少量标注示例引导 LLM，设备通信流量和后端代码可获取
D	难点	大模型引导的协议精度解析、种子模板调度
L	水平	International Journal of Machine Learning and Cybernetics 2025 SCI Q2

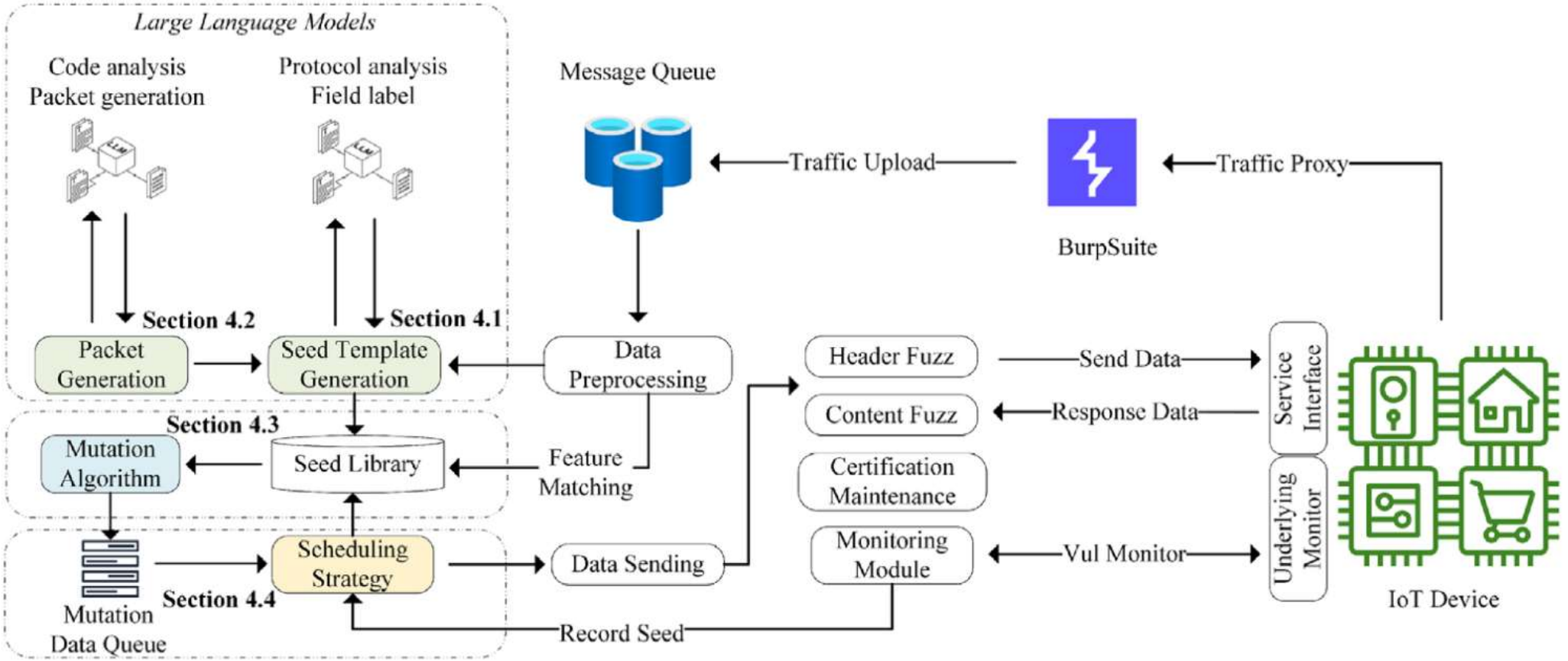
算法原理 创新说明

- 传统协议模糊测试方法
  - 测试用例质量问题
    - 协议解析依赖人工，变异策略随机，生成大量无效数据包
  - 深层状态覆盖问题
    - 初始种子依赖流量捕获，无法覆盖隐蔽逻辑
  - 调度策略随机问题
    - 传统调度算法缺乏对种子性能评估，导致低调用高潜力模板被遗漏且低效模板重复使用，模糊测试效率低下

- ChatHTTPFuzz
  - LLM驱动协议智能解析与种子生成
    - 利用 GPT-4o 解析 HTTP 数据包，生成种子模板、标注可变字段
    - 静态分析提取路由逻辑，LLM生成新数据包
  - 双因子增益的智能调度算法
    - 变异潜在因子，评估模板的变异空间（可变字段值集合大小）
    - 探索平衡因子，根据调用历史平衡探索与利用，避免遗漏低调用模板

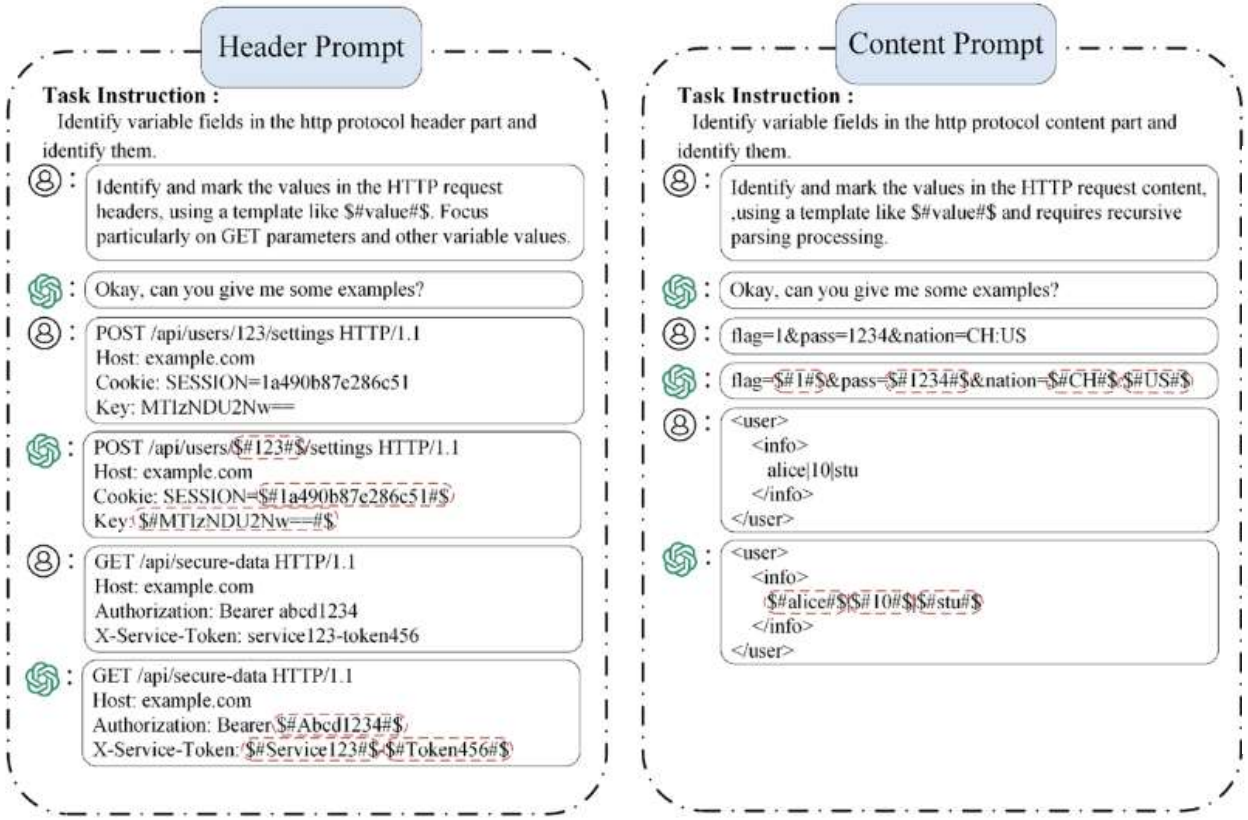
算法原理 ChatHTTPFuzz

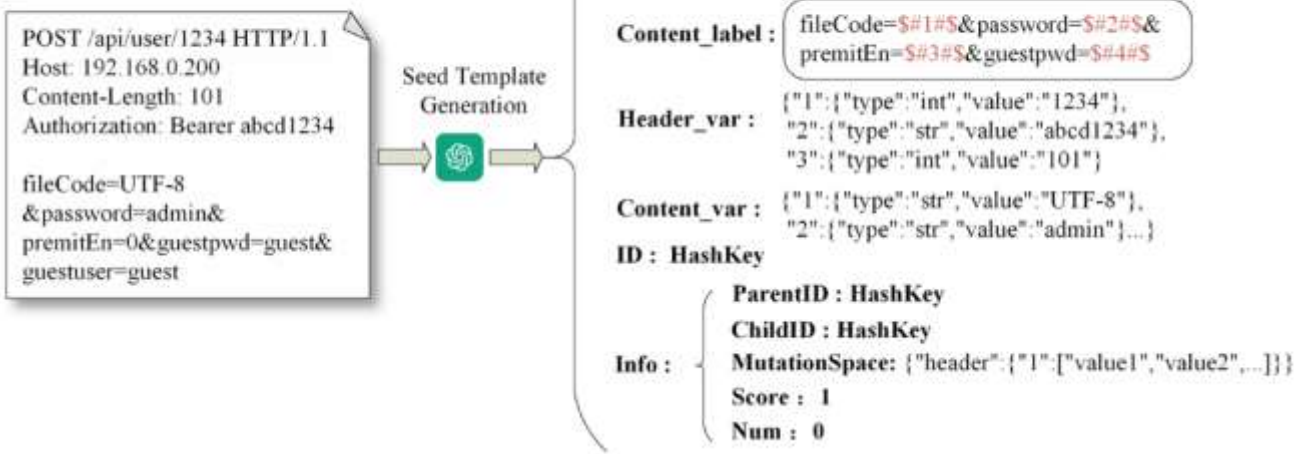
- 核心算法原理
  - LLM 引导的 HTTP 协议变量标注与种子模板扩展
  - 双因子增益种子模板调度算法



算法原理 ChatHTTPFuzz

- LLM引导的HTTP协议变量标注
  - 少样本学习生成提高准确率
  - 对请求头和请求体标注可变字段





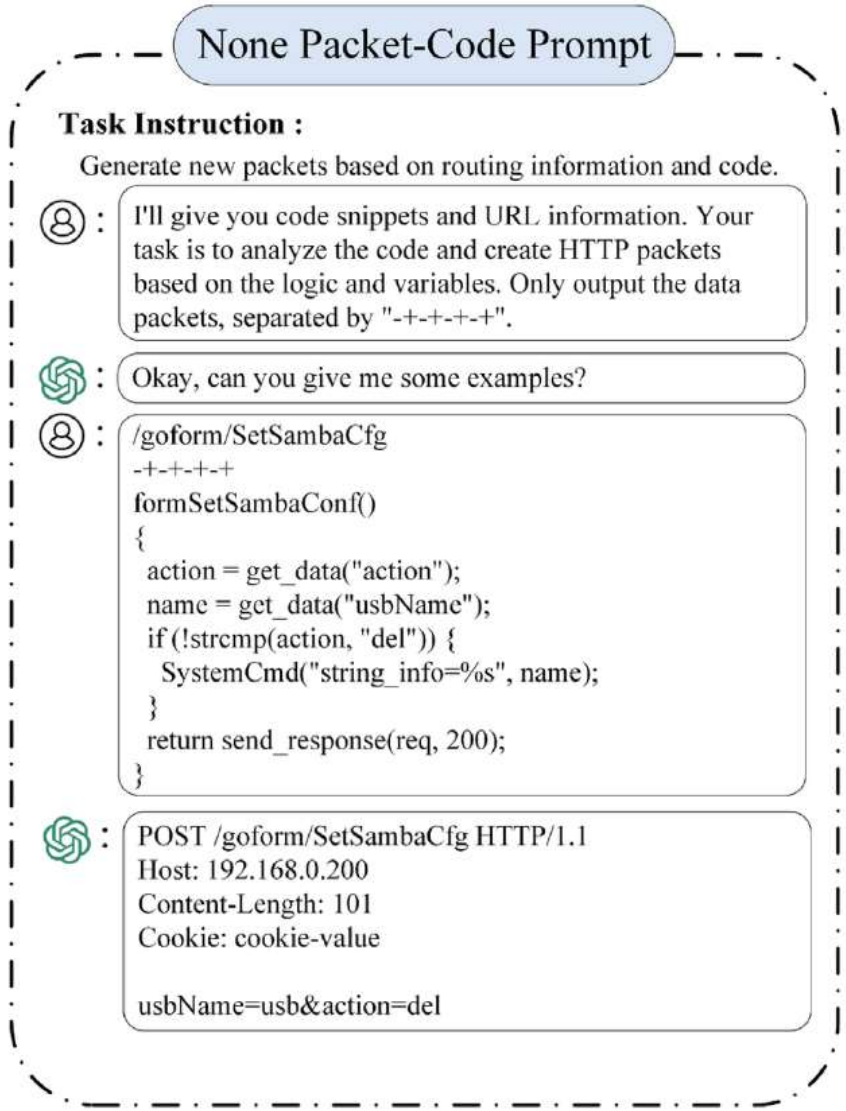
- LLM生成HTTP协议种子模板
  - 可变字段位置
  - 可变字段参数集（类型、数值）
  - 可变字段的变异空间
  - 调用信息与变异得分

算法原理 ChatHTTPFuzz

- LLM引导的种子模板扩展
  - 路由信息与后端代码静态分析
    - 解决隐蔽逻辑问题
    - 无法通过前端直接访问的特定后端服务
  - 利用静态分析结果指导LLM扩展种子模板
    - 变异空间扩展
      - 根据后端代码逻辑扩展可变异值列表

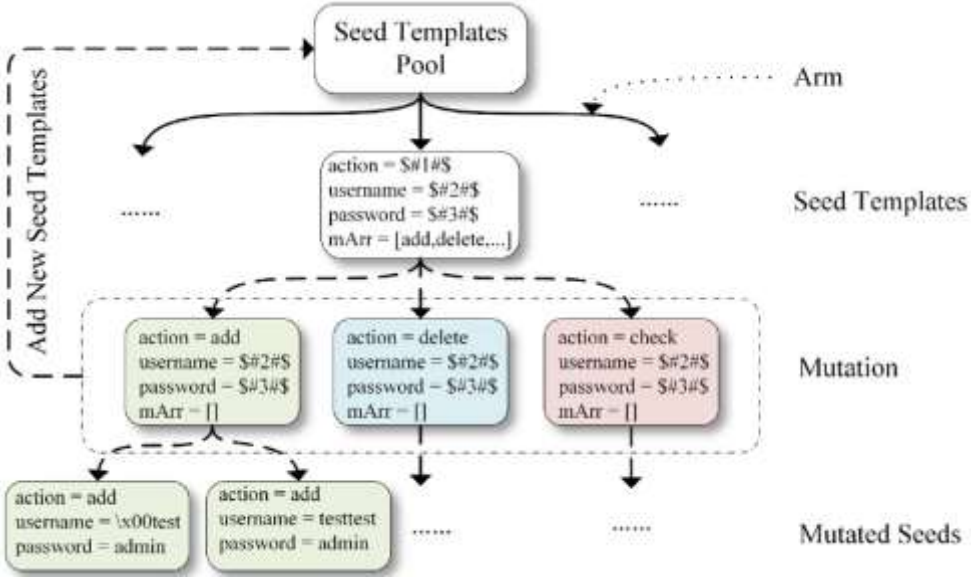
```
void formSetWanConfig() {
    char* action = get_data("action");
    if (strcmp(action, "update")) {
        SystemCmd("config --update %s", action);
    }
}
```

    - 新种子模板生成
      - 生成与代码逻辑一致的数据包



算法原理 ChatHTTPFuzz

- 双因子增益种子模板调度算法
  - 触发有效状态的种子模板
    - 提高测试效率与漏洞检测率
  - 低频调用的种子模板
    - 保持输入空间多样性与全面性
  - 丰富变异空间的种子模板
    - 扩大测试的覆盖范围和深度



- 实施策略
  - Thompson采样
    - 基于历史性能更新贝叶斯后验概率
  - 探索平衡因子
    - 为调用次数较少的模板分配了更高的选择概率

$$E_{factor_i} = \frac{\log(total_{trials}/N + 1)}{\log(\alpha_i + \beta_i + 1)}$$

  - 变异潜在因子
    - 衡量模板的潜在变异空间

$$M_{factor_i} = \frac{\log(v_i + 1)}{\log(\sum_{j=1}^{N} v_j + 1)}$$

  - 综合评分

$$Score_i = \theta_i \cdot (1 + M_{factor_i}) \cdot (1 + E_{factor_i})$$

算法原理 ChatHTTPFuzz

- ChatHTTPFuzz多层次变异策略
  - 基于类型的变异方法
    - 数值类型，突变为边界值或极端值，进行极端条件验证
    - 特殊编码字段，先解码再变异再编码，保持数据完整性和可用性
  - 上下文感知的智能变异方法
    - LLM分析与HTTP数据包相关联的后端代码，提取可变字段的实际用法与相应数值
    - 合并后端代码实际使用的值，模拟在实际中可能的异常或攻击场景
  - 其他的变异方法
    - 特殊字符替换，空字符\0、换行\n、回车\r、缩进\t
    - 预定义变异字典，包含已知的漏洞利用载荷（指令注入、缓冲区溢出、整数溢出）
    - 利用Radasma通用模糊器进行变异

实验设计　数据资源

- 待测数据（设备信息）

Vendor	Device	Version	Type
Cisco	RV110W	V1.2.2.5-8	VPN
D-Link	DIR-816	V1.10	Router
	DNS-320	V2.05b08	NAS
	DNS-320L	V1.09b06	NAS
	DNS-327L	V1.10	NAS
Tenda	AC15	V15.03.05.19	Router
	AC10	V16.03.10.13	Router
	AX1806	V1.0.0.1	Router
TP-Link	WR841N	V3.16.9	Router
	WR940N	V3.20.1	Router
Linksys	WRT54G	V4.21.5	Router
NetGear	WNAP320	V2.0.3	AP
	R9000	V1.0.4.26	Router
	R8500	V1.0.2.160	Router
VIVOTEK	CC8160	0113b	Camera
ToToLink	NR1800X	V9.1.0u.6279	Router
Zyxel	NAS326	V5.21(AAZF.18)	NAS

- 对比方法
  - BooFuzz（2024）
    - 基于模板的结构化模糊测试工具，需人工定义变异字段
  - Snipuzz（2021）
    - 分析测试响应来优化消息突变，黑盒IoT模糊测试工具
  - Mutiny（2017）
    - Cisco 开发的网络协议模糊测试工具，基于 Radamsa 随机变异，支持流量重放
- 评价指标
  - 漏洞发现数量

实验结果 漏洞发现

- 漏洞发现数量
  - ChatHTTPFuzz共识别出116种漏洞，包括70种未披露漏洞，CVE认证23种

Vendor	Device	CVE ID	CVSS v3
Linksys	WRT54G	CVE-2024-41281	8.8 / High
D-Link	DNS-320L	CVE-2024-7828	9.8 / Critical
		CVE-2024-7829	9.8 / Critical
		CVE-2024-7830	9.8 / Critical
		CVE-2024-7831	9.8 / Critical
		CVE-2024-7922	9.8 / Critical
		CVE-2024-8127	9.8 / Critical
		CVE-2024-8128	9.8 / Critical
		CVE-2024-8129	9.8 / Critical
		CVE-2024-8130	9.8 / Critical
		CVE-2024-8131	9.8 / Critical
		CVE-2024-8132	9.8 / Critical
		CVE-2024-8133	9.8 / Critical
		CVE-2024-8134	9.8 / Critical
		CVE-2024-8210	9.8 / Critical
		CVE-2024-8211	9.8 / Critical
		CVE-2024-8212	9.8 / Critical
		CVE-2024-8213	9.8 / Critical
		CVE-2024-8214	9.8 / Critical
		CVE-2024-7849	8.8 / High
		CVE-2024-7832	8.8 / High
		CVE-2024-7715	6.3 / Medium

Vendor	Device	Version	Vul	Undisc. Vul	CVE
Cisco	RV110W	V1.2.2.5	13	11	–
TP-Link	WR841N	V3.16.9	2	–	–
	WR940N	V3.20.1	1	–	–
D-Link	DIR-816	V1.10	12	9	21
	DNS-320	V2.05b08	31	30	
	DNS-320L	V1.09b06	–	–	
	DNS-327L	V1.10	–	–	
Tenda	AC15	V15.03.05.19_multi	17	2	–
	AC10	V16.03.10.13	4	–	–
	AX1806	V1.0.0.1	8	1	–
Linksys	WRT54G	v4.21.5	8	7	2
NetGear	R9000	V1.0.4.26	1	–	–
	R8500	V1.0.2.160	5	1	–
VIVOTEK	CC8160	0113b	1	–	–
ToToLink	NR1800X	V9.1.0u.6279_B20210910	2	–	–
Zyxel	NAS326	V5.21(AAZF.18)C0	11	9	–
Total			116	70	23

实验结果　对比实验

- 漏洞发现数量对比
  - ChatHTTPFuzz优势
    - 多类型漏洞检测：将可利用漏洞载荷作为变异字典
    - 上下文漏洞探索：分析后端代码实现更深层次的漏洞触发
    - 特殊变异机制漏洞：用不同编码格式封装，需正确解包-变异-封装

CVE ID	Type	Device	ChatHTTPFuzz	BooFuzz	Snipuzz	Mutiny
CVE-2020-8423	Buffer Overflow	TP-Link	✓	✓	×	✓
CVE-2020-3331	Buffer Overflow	Cisco RV110W	✓	×	×	✓
CVE-2024-7439	Buffer Overflow	Vivotek CC8160	✓	×	×	×
CVE-2024-41281	Buffer Overflow	Linksys WRT54G	✓	✓	×	×
CVE-2024-2855	Buffer Overflow	Tenda AC15	✓	×	×	×
CVE-2024-2902	Buffer Overflow	Tenda AC15	✓	✓	×	✓
CVE-2019-20760	Command Injection	NetGear R9000	✓	×	×	×
CVE-2024-7922	Command Injection	DLink-DNS 320L	✓	×	×	×
CVE-2024-8130	Command Injection	DLink-DNS 320L	✓	×	×	×
CVE-2024-7715	Command Injection	DLink-DNS 320L	✓	×	×	×
CVE-2022-41518	Command Injection	ToTolink	✓	×	×	×

算法总结

- 算法贡献
  - 使用LLM对HTTP协议进行可变字段标注，生成种子模板
  - 使用LLM分析后端代码与路由信息，扩展种子模板，模拟真实场景
    - 提取可变字段实际可用值与类型，扩展变异空间
    - 生成新种子模板
  - 变异策略中预定义可利用漏洞载荷，对设备进行针对性测试
  - 双因子调度策略，平衡“探索与利用”
- 算法不足
  - LLM模型可用性与成本
  - 代码分析深度限制
    - 闭源固件逆向依赖：静态分析提取路由和后端代码信息
    - 动态逻辑覆盖不足：动态生成路由、加密通信协议

特点总结与未来展望

特点总结与未来展望

- 特点总结
  - CHATAFL & ChatHTTPFuzz
    - 测试用例合规
      - 利用LLM根据RFC文档或实际流量内容生成合规种子模板
    - 扩展输入空间
      - 利用LLM根据RFC文档或静态分析获得的后端代码扩展输入空间
    - 模糊测试效率
      - 覆盖高原突破、双因子增益种子模板调度算法
- 未来展望
  - 未知协议模糊测试，知识库依赖协议逆向工程
  - 完全自动化实现
  - 内存开销优化

参考文献

[1] Zhang X, Zhang C, Li X, et al. A survey of protocol fuzzing[J]. ACM Computing Surveys, 2024, 57(2): 1-36.

[2] Meng R, Mirchev M, Böhme M, et al. Large language model guided protocol fuzzing. Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS) [C]. Los Angeles, CA, Internet Society (ISOC): 2024.

[3] Yang Z, Peng H, Jiang Y, et al. ChatHTTPFuzz: large language model-assisted IoT HTTP fuzzing[J]. International Journal of Machine Learning and Cybernetics, 2025: 1-22.

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！