开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler

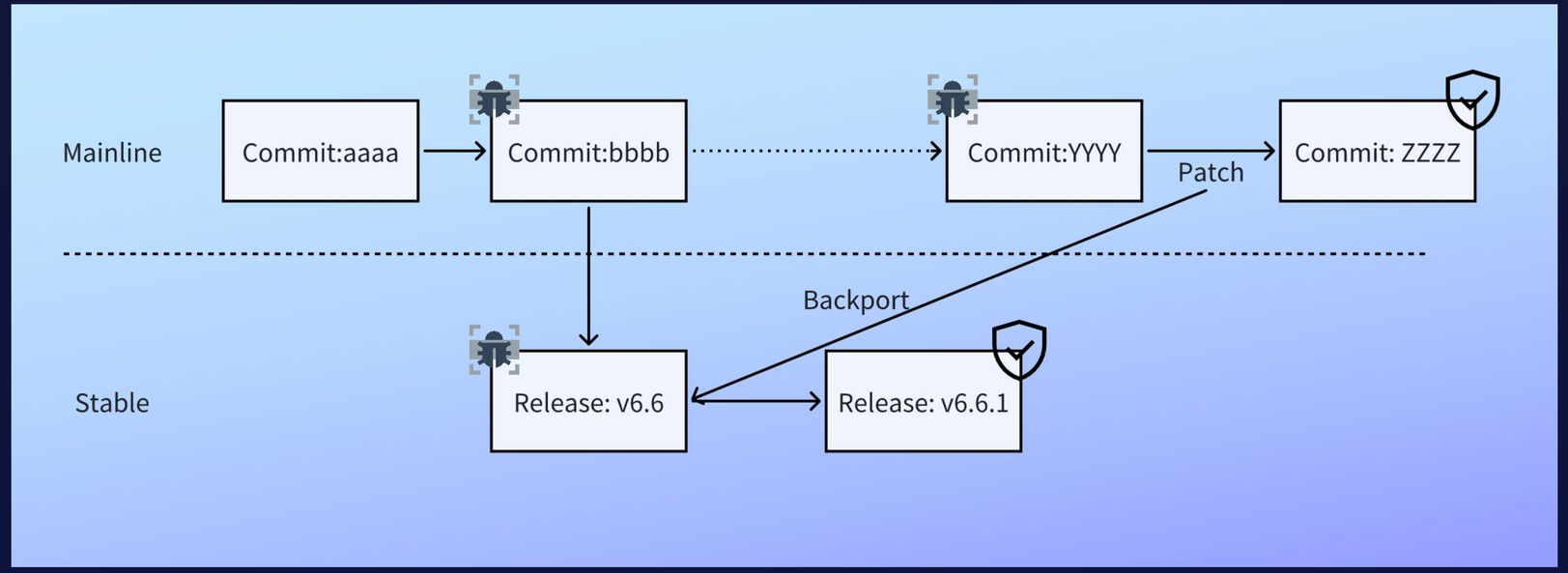
PORTGPT：基于 LLM 的自动化后向移植研究

李朝阳

华中科技大学开放原子俱乐部

研究动机

- 大型开源项目长期维护多个分支（mainline、stable、LTS）。
- 安全漏洞修复通常先在主线完成，随后需要**“向后移植”（Backport）**到旧版本。
- 现有维护模式依赖专家**手动维护**，耗时。
- 自动化研究依赖语法/语义规则，**难以处理复杂结构变化**。
- 该研究已被 IEEE Symposium on Security and Privacy **2026 (S&P) 接收**。



Patch Backporting 示意图

问题定义

补丁后向迁移（Backporting）定义：

将主线版本中的补丁 P_n 迁移到旧版本 P_o，保持漏洞修复与原本功能的正确性。

核心挑战：

1. **定位（Localization）**：确定旧版本中对应的修改位置；
2. **变换（Transformation）**：调整补丁以适配旧版本上下文；
3. 二者都受到版本差异、符号变更、结构调整等因素影响。

传统方法的局限

文本匹配

patch utils

依赖上下文完全匹配

语法匹配

Fixmorph

基于 AST 实现

难以处理语义上的变化

语义模板

TSBPORT

PDG 匹配语义

扩展性有限
仅实现部分定位

基于*LLM*的方法

PPathF
Mystique

微调

仅针对转化阶段

启发性示例：CVE-2022-32250

- **漏洞类型：Use-After-Free。**
- **人类开发者通常：**
  1. 查找函数定义与上下文；
  2. 分析 Git 历史追踪函数迁移；
  3. 手动调整变量名、修复编译错误；
  4. 最后运行测试验证。
- 启发：Backporting 需要动态检索、追踪历史与基于反馈的修正。

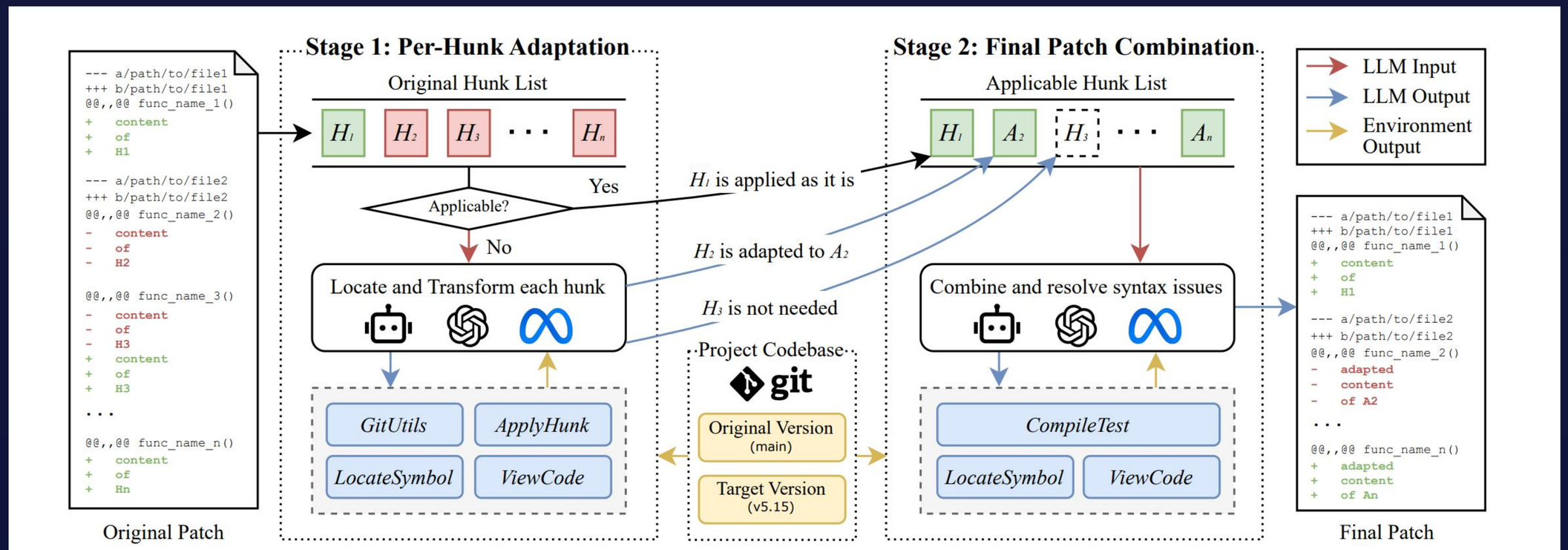
```
--- a/net/netfilter/nf_tables_api.c
+++ b/net/netfilter/nf_tables_api.c
@@ -2873,27 +2873,31 @@ *nft_expr_init(
  err = nf_tables_expr_parse(ctx, nla, ...);
  if (err < 0) goto err1;

+ err = -EOPNOTSUPP;
+ if (!(expr_info.ops...flags & NFT_EXPR_STATE))
+    goto err_expr_stateful;

  err = -ENOMEM;
  expr = kzalloc(expr_info.ops->size, ...);
@@ -5413,9 +5417,6 @@ *nft_set_elem_expr_alloc(
    return expr;

  err = -EOPNOTSUPP;
- if(!(expr->ops->type->flags & NFT_EXPR_STATE))
-   goto err_set_elem_expr;
-
  if (expr->ops->type->flags & NFT_EXPR_GC) {
    if (set->flags & NFT_SET_TIMEOUT)
      goto err_set_elem_expr;
```

Listing 8: Original Patch for CVE-2022-32250

关键观察

- 无法一次性提供全部信息，需按需查询；
- 信息应简洁、可由模型自行调用；
- 需要试错式（trial-and-error）过程；（验证与反馈）
- 因此——应采用 Agent 式设计，而非单轮提示。
- 关键思想：将 LLM 设计为具备“工具调用”和“自我反馈”的智能体。

系统架构概览



- 阶段一：每个补丁块（hunk）的定位与转换；
- 阶段二：补丁组合、编译与自我修正；

工具设计

- **模拟人工迁移行为；**
- **为模型提供“工具”与“推理链”：**
  - 代码访问： ViewCode；
  - 符号定位： LocateSymbol；
  - 历史追踪： GitUtils；
  - 补丁应用与修复： ApplyHunk；
  - 编译验证： CompileTest。
- **构成端到端自动化迁移流程。**

ApplyHunk 机制流程

```
--- a/foo.c
+++ b/foo.c
@@ -11,7 +11,9 @@
 }}

 int check (char *string) {{
+    if (string == NULL) {{
+        return 0;
+    }}
-    return !strcmp(string, "hello");
+    return !strcmp(string, "hello world");
 }}
 int main() {{
```

处理补丁格式并反馈失败原因

	补丁格式错误	上下文错误	文件不存在
错误原因	行号，开头	误用、幻觉	错误文件名、路径
解决方法	自动修正	编辑距离 反馈相似代码	符号定位 相似文件名

GitUtils 设计

- 利用 Git 辅助大模型分析代码演进历史。
- 包含两个协同工作的组件：
  - History 组件：追踪局部演变
    - 功能：展示代码片段从“分叉点”（fork point）到新版本的所有相关提交历史。
    - 设计：仅显示影响当前代码块（hunk）内代码行的提交。
  - Trace 组件：识别全局变更
    - 功能：专门用于识别并追踪关键的全局性代码变更，特别是代码块的重定位（移动）。
    - 设计：利用最小编辑距离算法，在 commit 的代码修改中检查是否存在相似的代码在不同文件中增删。
- 两者结合提供完整修改视角，帮助处理因代码重构或迁移导致的复杂情况。

实验分析

- 实验设置
  - GPT-4o，1815 纯 Linux 数据集，146（C/C++/Go）混合数据集。
- 综合成功率
  - 89.15%（提升 18%），60.87%（提升 26%）
- 模型通用性
  - 52% - 60%，仅 Llama 3.3 由于工具调用成功率较低。
- Real-World Applicability
  - Mainline to LTS: 9 个补丁被 Linux 6.1-stable 接收
  - LTS to downstream: 完成了 Ubuntu 中 10/16 的补丁对。

结论与展望

- PORTGPT 模拟人类专家迁移行为，实现端到端自动化；
- 结合多工具与验证链，有效提高智能化迁移效果；
- 实际补丁已被社区接受，具实用价值，对 AI 与安全的探索；
- 未来工作：
  - 上下文限制 vs 补丁分块；
  - 语言/项目定制 vs 可扩展性；
  - 高可靠性验证链

关于我们

- 华中科技大学开放原子俱乐部
  - 操作系统内核贡献团队 —— 慕冬亮老师指导
  - 内核贡献团队建设 —— 新人友好型
    - 明确贡献流程
    - 内部审核小组
    - Patch Copilot 机制
    - 辅助工具（机器人）
  - Linux 内核开源工坊
  - 团队成果
    - Google Open Source Peer Bonus
    - 向 Linux 社区提交 180+ 补丁，其中 90% 为安全补丁
    - 向 OpenEuler 社区 Kernel 提交近百个补丁
  - hust-os-kernel-patches@googlegroups.com