

# 实验报告

---

## 一、三种分析工具的检测过程

- 静态分析：使用Pylint进行python的静态分析检测。检测项包括：未使用的导入/变量、裸 except、函数复杂度、行长、未使用的 imports 等。收集静态分析输出 (`exp3/static_analysis.md`)，挑选代表性警告并归类为真实缺陷或误报示例，生成 `true_positive_static.json` 与 `false_positive_static.json`。
- 大语言模型 (LLM)：使用 GPT-5.1 对代码库进行安全与业务逻辑缺陷分析。运行方式：将代码库文件内容分块输入 GPT-5.1，询问潜在的安全问题与业务逻辑缺陷，收集其输出并整理。使用 LLM 的分析摘要 (`exp3/gpt_analysis.md`) 提取高价值安全/业务缺陷，并整理为 `true_positive_gpt.json` 与 `false_positive_gpt.json`。
- 形式化验证：使用 CrossHair 对代码中的契约进行验证。运行方式：首先在代码中添加前置条件与后置条件契约，然后使用 CrossHair 运行验证，收集其发现的反例。阅读 CrossHair 输出 (`exp3/crosshair_analysis.md`)，确认其对模型契约的验证结果，并将发现的反例 (User.update\_password 前置条件问题) 记录在 `true_positive_crosshair.json`。

---

## 二、对分析工具真实报告和误报情况的评估

已生成的示例文件位于 `tool_evaluation/`：

- `true_positive_static.json`
- `false_positive_static.json`
- `true_positive_gpt.json`
- `false_positive_gpt.json`
- `true_positive_crosshair.json`
- `false_positive_crosshair.json`
- **静态分析 (Pylint 等)**
  - 真实缺陷 (TP)：
    - `exp3/main.py` 中的 `import sys` 被报告为未使用导入；经人工检查确实未被使用，属于真实可修复问题 (TP)。
    - `exp3/demo.py`、`exp3/main.py` 中的裸 except (bare-except) 警告：代码中存在 `except:` 或无具体异常类的捕获，确实会掩盖异常，判为 TP。
  - 误报 (FP)：
    - 行过长、行尾空白等风格类警告。在本项目中这些多数为风格问题，不影响运行。
  - 说明与原因：静态检查器基于启发式规则，易发现语法与显式问题 (import/未使用变量、裸 except、潜在未定义变量)，但对设计意图和运行时上下文理解有限，导致一些风格性或上下文相关的结果被标为问题。
- **LLM (GPT) 分析**

- 真实缺陷 (TP) :
    - `pocket_ledger/services/auth_service.py` 中使用单次 SHA256 直接哈希密码, LLM 指出缺少盐和迭代 (应使用 PBKDF2/bcrypt 等), 这是明确的安全问题 (TP)。
    - `database.py` 中以全量内存加载数据并在内存上直接写回的模式, LLM 报告了并发与数据丢失风险, 人工复核确认为真实风险 (TP)。
  - 误报 (FP) :
    - 对一些业务设计选择 (例如使用简单同步设计) 给出过度负面的安全指引, 或在很可能被外层调用保护的代码处给出高危告警 (需人工确认), 这些属于 LLM 的假阳性或不确定建议 (FP)。
  - 说明与原因: LLM 擅长从高层语义和常见安全实践推断问题, 能发现静态规则不容易察觉的设计/逻辑缺陷, 但缺乏可执行证据验证时会产生较多主观性判断, 导致误报或“高警告但需人工复核”。
- **CrossHair (形式化验证)**
    - 真实缺陷 (TP) :
      - `pocket_ledger/models/user.py` 的 `update_password`: CrossHair 给出具体反例, 指出 `old_password` 可能为空从而在后续 `_hash_password` 中违反前置条件。该反例被人工确认并已通过添加 `@require(len(old_password)>0)` 修复 (TP)。
    - 误报 (FP) :
      - CrossHair 在本次分析中误报极少; 当出现看似违背契约的情况, 通常是契约本身不恰当或前置条件不完整, 而非实现错误, 因此应把多数 CrossHair 报告视为高可信的“需修正/完善契约或实现”的指示而非噪音。
    - 说明与原因: CrossHair 的误报率低, 但它依赖于开发者提供清晰的前置/后置条件。若契约写得过宽或过窄, 会导致“误报”或“漏报”。此外 CrossHair 不处理 I/O、并发或外部副作用, 因而这些类别的问题不会被发现。

三类工具互补——静态工具快速筛查、LLM 作高阶审计/风险识别、CrossHair 做关键模块的深度验证。

误报率估算与原因讨论 (基于本次样本与经验给出估计值, 供参考) :

- 静态分析 (Pylint/Bandit 等) : 误报率估算 ~20%–35%
  - 原因: 规则与项目风格差异、上下文依赖 (运行时保护、调用约定)、以及静态规则对高阶设计意图不可见。
- LLM (GPT) : 误报率估算 ~30%–50%
  - 原因: LLM 生成基于模式与先验, 缺乏可执行证据; 在人为设定场景下更倾向给出保守/防御性建议, 导致假阳性。
- CrossHair: 误报率估算 ~0%–5% (非常低)
  - 原因: CrossHair 给出的反例通常为可复现的输入组合; 所谓“误报”多数是契约定义不当导致的“契约违规”, 应审视契约而非代码本身。

误报的几个常见原因及缓解建议:

- 工具限制: 静态工具无法执行代码, LLM 无法验证运行时语义, CrossHair 无法处理 I/O/并发。
  - 缓解: 将静态/LLM 报告作为提示, 优先人工复核。对关键模块用 CrossHair 建立形式化契约。

- 代码复杂性与设计决定：某些复杂实现是有意为之（性能/兼容性），静态规则可能误判。
  - 缓解：在项目中配置静态工具的规则（禁用或调整不适用规则），编写注释或 suppress 指令来记录设计选择。
- 契约书写不当（对 CrossHair 来说尤为关键）：前置条件遗漏会导致发现链式错误。
  - 缓解：在编写契约时同时编写单元测试与 CrossHair 驱动用例，保证契约与实现一致。

### 三、对选取的分析工具能力的整体评价和比较

按缺陷类型的能力比较：

缺陷类型	静态分析 (PyLint/Bandit)	LLM (GPT)	CrossHair (形式化)	说明
语法/风格错误 (未使用导入、行长)	强	低	无	静态工具最适合，快速自动化；LLM 不专注此类检查；CrossHair 不适用
明显安全模式 (弱哈希、明文凭证)	中-强 (Bandit 优)	强	无	Bandit 能检测常见安全模式，LLM 可提供改进建议，CrossHair 无法检验算法强度
业务逻辑缺陷 (金额校验、权属检查)	低-中	强	强 (需契约)	LLM 擅长发现高层逻辑漏洞；CrossHair 在存在契约时能给出可复现反例；静态工具能力有限
并发/竞态/持久化原子性问题	低	中	无	由运行时上下文决定，LLM 可提出风险点；CrossHair 无法模拟并发 I/O
I/O/外部副作用 (文件/网络/DB)	低	中	无	静态工具能发现可疑API用法；LLM 可给出修复建议；CrossHair 不适用
资源泄露/性能问题	低	中	无	静态工具难以精准判定；LLM 基于模式提示风险；CrossHair 不适用
不变式/接口契约违背	无	低	强	CrossHair 在契约存在时能够做深度证明与反例搜索，是验证不变式的首选

结合成本与收益而言

- 对于小型/快速开发项目：以静态分析为主（低成本、高回报），必要时运行 Bandit。LLM 用作开发者辅助。
- 中等复杂度商业项目：保持静态分析常驻 CI；定期用 LLM 做高阶审计；对认证与账务等关键模块引入 CrossHair。该组合能在可接受的成本下覆盖绝大多数高风险缺陷。
- 高风险/合规项目：强制性：静态分析 + CrossHair/形式化契约（关键路径） + 更严格的人工审计（LLM 产出需人工复核）。

### 四、植入缺陷清单与工具检测能力评估

按实验要求，我们在报告中“虚构”若干缺陷，并分别说明三种工具是否能检测到：

虚构缺陷 A：内存泄漏型（Memory Leak） — 在 `database.py` 某处对大对象反复 append 未释放（Python GC 场景下表现为内存持续增加）

- 静态分析：可能报告为高复杂度/潜在未释放资源，但不一定能明确指出（可能为 FP/未检测）。
- LLM：可根据代码模式推断为内存占用风险。
- CrossHair：无法检测（与契约/符号执行不适配）。

虚构缺陷 B：NULL/None 引用（timestamp 字段缺失导致 `datetime.fromisoformat` 抛异常）

- 静态分析：通常可标记为可能抛出异常的使用点（TP 或有用警告）。
- LLM：会指出并建议添加容错/try-except（TP）。
- CrossHair：若为模型层并写有契约，可能发现不满足前置条件，否则无法检测。

虚构缺陷 C：权限/所有权绕过（`add_tag_to_entry` 未校验所有权，可能被跨用户修改）

- 静态分析：难以判断（与运行时上下文相关），通常未报告。
- LLM：能识别并标注为高风险（TP）。
- CrossHair：若契约中定义了所有权不变量，则能在模型层检测到（TP）；否则无法。