软件工程作业二

题目：给定一个C语言编译器（包含源代码、可执行系统，没有其他文档），试讨论如何使用程序分析技术和软件测试技术，确保该编译器不存在内存泄露、缓冲区溢出等内存安全漏洞？

① **可达性分析**：某个代码无法从入口可达

- return后的代码
- 检测空指针解引用缺陷

② **数据流分析**：是一种用于收集在程序中各个点变量的可能值集合信息的技术。数据流分析通常是基于控制流程图进行的。

- 基于控制流图
- 定义数据流公式
- 应用数据流公式直到所有节点的输出不再变化（即到达不动点）

通过数据流分析可以确定变量在某一点可能的取值并判断是否存在漏洞。除零漏洞代码示例，可通过数据流分析检测。

③ **指针分析**：目的是为了确定在程序的某一点指针可能指向的变量或者内存区域。

在静态分析中使用指针分析可以检测与指针相关的缺陷如空指针引用、use-after-free、double free、内存泄漏等，常见的指针分析算法为 Steensgaard 算法和 Andersen 算法。

④ **别名分析**：用于确定是否可以以多种方式访问某个存储位置。 如果两个指针指向同一位置，则称它们为别名。别名产生的原因包括指针的使用 、函数调用、数组下标和结构体的使用等。

通过别名分析可以提高静态分析的精度，帮助进行函数间分析。

分析算法有 Andersen-style pointer analysis，Steensgaard-style pointer analysis……

⑤ **污点分析**：分为静态污点分析和动态污点分析 ，静态污点分析是指在不运行且不修改代码的前提下，通过分析程序变量间的数据依赖关系来检测数据能否从污点源（source）传播到污点汇聚点（sink）。

污点分析通常用来检测由外部输入引起的程序缺陷，如缓冲区溢出、命令注入等，通过识别污染数据影响的代码范围，可以提高静态分析的效率。

⑥ **融合模糊测试、符号执行检测缓冲区溢出（静态分析+模糊测试）：**

模糊测试：

- 通过程序崩溃检测缓冲区溢出漏洞
    - 通过变异输入恰好导致程序崩溃非常困难
- 状态空间爆炸，无法充分覆盖代码空间→无法充分检测缓冲区溢出漏洞

- 变异输入难以通过窄约束

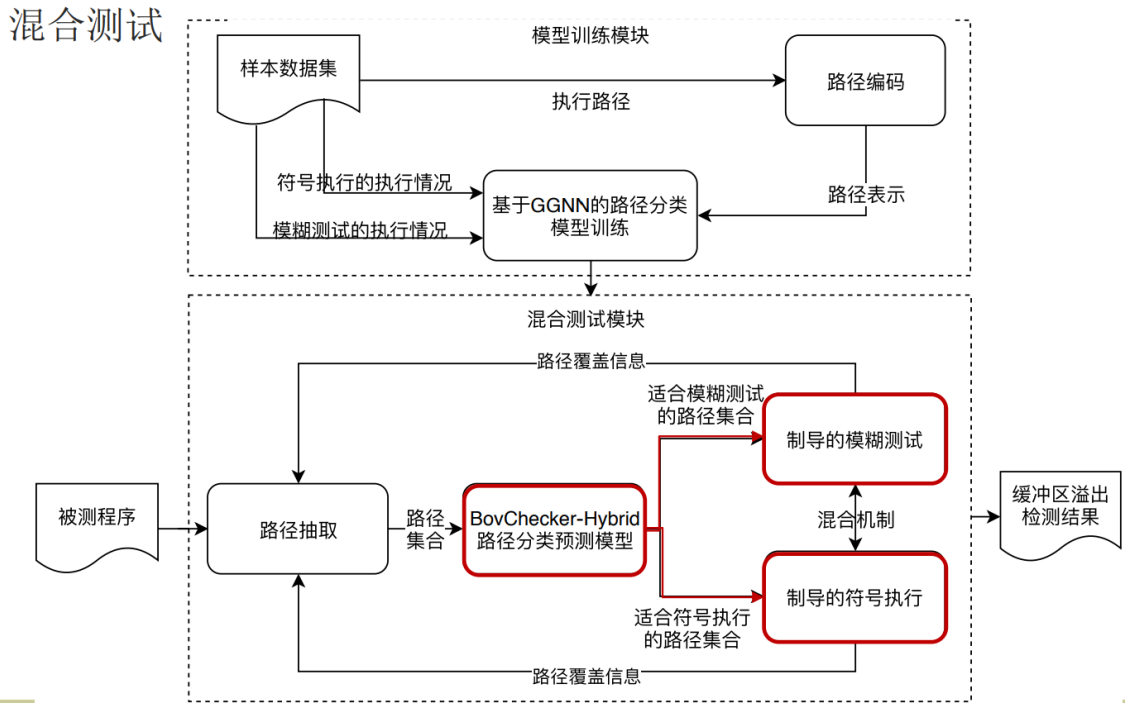
符号执行：

- 约束求解容易处理模糊测试难以通过的窄约束
  - 约束求解耗时
- 状态空间爆炸，无法充分覆盖代码空间→无法充分检测缓冲区溢出漏洞
  - 分支、循环加剧约束复杂度

融合模糊测试、符号执行检测缓冲区溢出：

- 同时执行更高效
- 制导执行缓冲区操作语句的可达路径
- 制导由限探索适合各自的路径
  - 不清楚什么路径适合模糊测试/符号执行
  - 深度学习，基于控制流图表示程序路径
  - 门控图神经网络（GGNN）基于路径图表示学习
- 混合机制，交换路径，提升覆盖

混合测试：



缓冲区溢出漏洞检测：

- 符号执行
  - 路径约束^缓冲区一处约束，约束求解
- 模糊测试
  - 通过程序崩溃检测漏洞
  - 代码插桩，使缓冲区溢出发生时导致程序崩溃

缓冲区溢出报告：

- 符号执行
  - 路径约束^缓冲区溢出约束，可满足
- 模糊测试
  - 检测到程序崩溃，并且崩溃原因被标记为缓冲区溢出

方法：用可达性分析，数据流分析，指针分析和别名分析判断是否有未归还的内存空间和指向控件更改的指针，进而判断是否存在内存泄露；再用污点分析分析程序变量间的数据依赖关系来检测数据能否从污点源传播到污点汇聚点，进而判断置否存在缓冲区溢出；通过最后通过模糊测试和符号执行相结合的方式，用程序崩溃和代码插桩检测出缓冲区溢出。