additionalTaintStep(过程间调用)

在smtm中讲到了一些additionalTaintStep的使用，主要常见的为以下2种：

语句的

````

/*示例代码:
 * try {
 *   call(tainted);
 * } catch(Exception e) {
 *   ... e.getMessage()
 * }
 * ```
 */

class ExceptionTaintStep extends TaintTracking::AdditionalTaintStep {
  override predicate step(DataFlow::Node n1, DataFlow::Node n2) {
    exists(Call call, TryStmt try, CatchClause catch, MethodAccess getMessageCall |
      // the call is within the `try` block, which has a corresponding `catch` clause
      call.getEnclosingStmt().getEnclosingStmt*() = try.getBlock() and
      try.getACatchClause() = catch and
      // the `catch` clause is likely to catch an exception thrown by the call
      (
        catch.getACaughtType().getASupertype*() = call.getCallee().getAThrownExceptionType() or
        catch.getACaughtType().getASupertype*() instanceof TypeRu
````

```
ntimeException
      ) and
      // the exception message is read by `getMessageCall` within the `catch` block
      catch.getVariable().getAnAccess() = getMessageCall.getQualifier() and
      getMessageCall.getMethod().getName().regexpMatch("get(Localized)?Message|toString") and
      // taint flows from any argument of the call to a place where the exception message is accessed
      n1.asExpr() = call.getAnArgument() and
      n2.asExpr() = getMessageCall
    )
  }
}
```

方法调用

```
/** Track taint from `stream` to `stream.collect()`. */
class StreamCollectTaintStep extends TaintTracking::AdditionalTaintStep {
  override predicate step(DataFlow::Node n1, DataFlow::Node n2) {
    exists(StreamCollectCall call |
      n1.asExpr() = call.getQualifier() and
      n2.asExpr() = call
    )
  }
}
```