**ECE 653 Project Report**

**Group 63**

Student Name: Hai Jiang Student No.: 20597659 Email: [h57jiang@uwaterloo.ca](mailto:h57jiang@uwaterloo.ca)

**Part (1) Building an Automated Bug Detection Tool**

1. **Inferring Likely Invariants for Bug Detection**

I use the code …

1. **Finding and Explaining False Positives**

Reasons for the occurrence of false positives could be:

1. Although two methods have to be paired to realize a target function, one of them may be included in a specific method function that only gives a broader definition associated with this method. In such a case, this specific method function does not need to include the other method, since it is not for the target purpose.
2. Although two methods are paired to realize a target function in some circumstances, one of them can be paired with another method to achieve similar functions. In such a case, a false positive is generated if the new pair is regarded as a bug.

I use the code and hash map data structure in (a) to count and list the bug numbers for all the pairs. Then I figure out which 2 pairs can form 10 locations. As seen below, I list the 2 pairs I found and trace back to the source code in /opt/testing/apache\_src to check if the bug is a false positive.

1. Pair (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock) as shown below, which has totally 6 bug locations.

bug: apr\_thread\_mutex\_lock in apr\_dbd\_mutex\_lock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 95.56%

bug: apr\_thread\_mutex\_lock in apu\_dso\_mutex\_lock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 95.56%

bug: apr\_thread\_mutex\_unlock in apr\_dbd\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

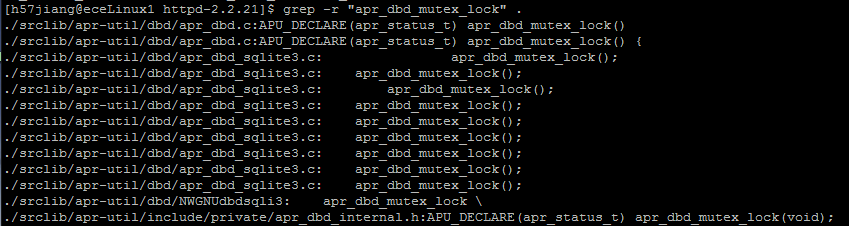
bug: apr\_thread\_mutex\_unlock in apr\_global\_mutex\_trylock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

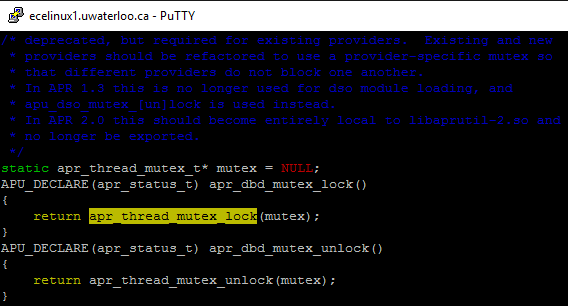
bug: apr\_thread\_mutex\_unlock in apr\_global\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

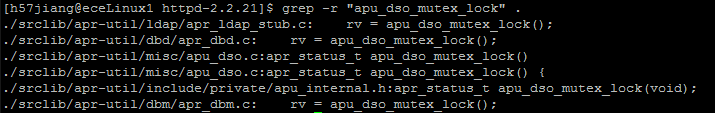
bug: apr\_thread\_mutex\_unlock in apu\_dso\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

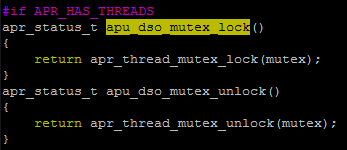
Then I use the linux instruction “grep –r” to find out which file contains the corresponding method. As a result, I check the source code of several locations and I find out some false positives.

As shown in the picture below, I read the source code of apr\_dbd\_mutex\_lock(). Apparently, this method apr\_dbd\_mutex\_lock() is used to define mutex\_lock, by using the apr\_thread\_mutex\_lock. There is no need for apr\_dbd\_mutex\_lock() to include an unlock function. In practical use, apr\_dbd\_mutex\_lock() is called by the other functions to perform a lock operation, and apr\_dbd\_mutex\_unlock() is called to perform an unlock operation. In such case, apr\_dbd\_mutex\_lock() does not need an unlock, and similarly apr\_dbd\_mutex\_unlock() does not need a lock. As such, the associated bugs shown in the report from part (a) is false positives.









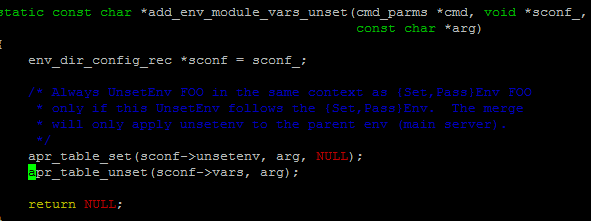
1. Pair (apr\_table\_setn, apr\_table\_unset) as shown below, which has totally 4 bug locations.

bug: apr\_table\_unset in add\_env\_module\_vars\_unset, pair: (apr\_table\_setn, apr\_table\_unset), support: 8, confidence: 66.67%

bug: apr\_table\_unset in ap\_read\_request, pair: (apr\_table\_setn, apr\_table\_unset), support: 8, confidence: 66.67%

bug: apr\_table\_unset in basic\_http\_header, pair: (apr\_table\_setn, apr\_table\_unset), support: 8, confidence: 66.67%

bug: apr\_table\_unset in strip\_headers\_request\_body, pair: (apr\_table\_setn, apr\_table\_unset), support: 8, confidence: 66.67%



Then I use the same method as mentioned above to check the pair (apr\_table\_setn, apr\_table\_unset). As shown in the picture below, I read the source code of add\_env\_module\_vars\_unset. In this function, apr\_table\_unset() follows apr\_table\_set() instead of apr\_table\_setn(). As apr\_table\_set() has already performed a set function, I believe that it is reasonable to have apr\_table\_unset() without a apr\_table\_setn() here.

1. **Inter-Procedural Analysis**

I use the code …

**Part (2) Using a Static Bug Detection Tool**

1. **Resolving Bugs in Apache Commons**

I use the code …

1. **Analyzing Your Own Code**

I use the code …