Project 1(b).

I use the code and hash map data structure in (a) to list the bug numbers for all the pairs. Then I can figure out which 2 pairs can form 10 locations. As seen below, I list the 2 pairs I found.

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%

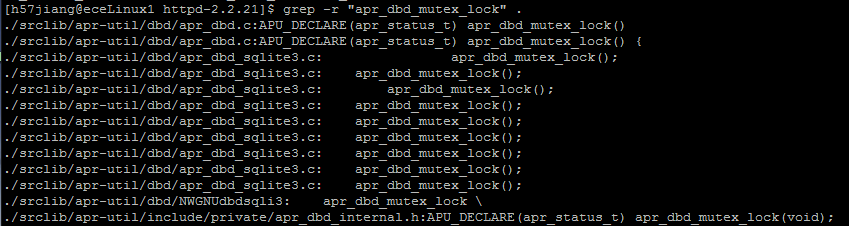
bug: apr\_thread\_mutex\_unlock in apr\_dbd\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

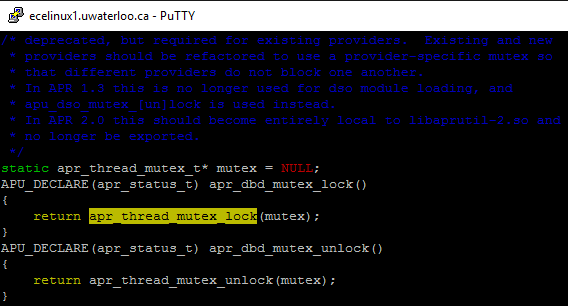
bug: apr\_thread\_mutex\_unlock in apr\_global\_mutex\_trylock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

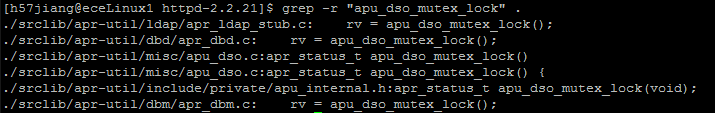
bug: apr\_thread\_mutex\_unlock in apr\_global\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

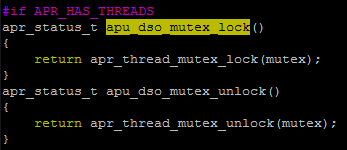
bug: apr\_thread\_mutex\_unlock in apu\_dso\_mutex\_unlock, pair: (apr\_thread\_mutex\_lock, apr\_thread\_mutex\_unlock), support: 43, confidence: 91.49%

Then I 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 only used to define mutex\_lock, 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.









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

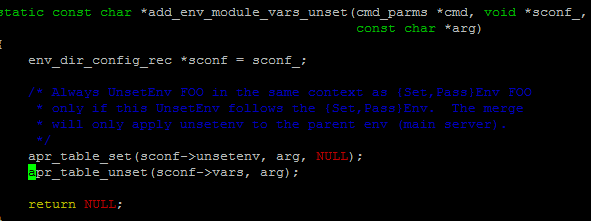
bug: apr\_table\_unset in add\_env\_module\_vars\_unset, pair: (apr\_table\_setn, apr\_table\_unset), support: 8, confidence: 66.67%

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 check the source code of several locations and I find out the false positive. 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 add\_env\_module\_vars\_unset is generally achieve an unset functions, I believe it is right that apr\_table\_set() instead of apr\_table\_setn() is ahead of apr\_table\_unset().



bug: apr\_dir\_close in apr\_match\_glob, pair: (apr\_dir\_close, qsort), support: 4, confidence: 80.00%