Michael Salerno

**Pin - A Dynamic Binary Instrumentation Tool**

Another technique to perform instrumentation was using pin. Pin seemed like a good idea since it would not actually run the instrumented program, but emulate it. Pin also allowed us to modify it in real time. First, to get things rolling, we wrote a simple module to count the number of instructions a program executed. We did this by using the INS\_InsertCall object and incrementing a static variable. Next we wrote a module to hook malloc and free. We hooked these by checking for a routine named “malloc” every time an image was loaded. If we found a valid routine when an image was loaded we replaced it with our own rewritten version of malloc. This new version of malloc keeps track of the address space that were malloced and how big it was. Lastly, we detected if there was any reads or writes being done to the heap memory. If there was then we checked the address of what was being accessed and compared it with the list of addresses our new malloc function made. If the address was not in range of anything we malloced, we print out to tell the user that something went.