# Audit Log Generation through Library Interposition

**Mustafa Paksoy, Dr. Benjamin Kuperman (Advisor)**
**Swarthmore, PA**
mpaksoy1@swarthmore.edu

**Abstract**
*Computer security monitoring systems operate by scanning recorded audit data for statistical anomalies and known attack signatures. The accuracy and effectiveness of these systems are limited by the quality of the data that they operate on—the garbage in, garbage out principle. Library interposition is a flexible and robust audit log generation method that operates by intercepting calls to dynamically loaded system libraries. We designed and implemented a thoroughly documented, highly modular system with independent interposing libraries that collect data for attack, intrusion and misuse detection. We would like to follow up on this work by comparing our system with common alternatives as to generate evidence of its advantages and shortcomings.*

**Summary**
Computer Security Monitoring (CSM) systems operate by examining audit data trails on systems for known attack scenarios, sequences of potentially harmful events or statistical anomalies. Majority of security monitoring systems design their analysis scheme around limitations of audit data provided by default on the operating system.[1] Given the very low ratio of attacks (*signal*) to normal use (*noise*) and ambiguous data, CSMs succumb to increased false positives when they try to reduce false negatives. In other words, they are either insecure or they produce too many false alarms. To mitigate this problem, our application provides audit data sources that are discriminatingly sensitive to events associated with a given security-monitoring task.

On modern operating systems, calls to shared system libraries are linked on run time. Thus, any library interposed between system libraries and the calling application can intercept such calls to dynamically loaded libraries (file extension .dll on Windows, and .so on Linux). Our audit log generator has three libraries that can be interposed independent of each other. These are tailored for attack, intrusion and misuse detection respectively. In doing so, they intercept calls to a unique set of functions, log information related to their purposes, then pass the function call on to the next library in sequence.

Information collected from intercepted calls is composed into tokens that represent basic elements of data, such as strings, integers, and file status data. Tokens are combined into log entries that contain information relating to individual events. The entries are created and augmented in memory. When all tokens relating to an entry are in place, the entry is committed and written to a log file. Each interposing library has a unique log file associated with it. A separate application in our system can read these binary logs and translate contents into easily parsable or human readable forms.

Having completed implementing the application, we will design a testing framework to quantitatively compare our system to its alternatives in areas such as: performance impact, volume of log files generated, adaptability to different security monitoring demands. In addition, a set of known attacks will be compiled and tested under our application, demonstrating its capability in generating data suitable for detection.

**References**
[1] Kuperman, Benjamin A. *et al*, *Terminology to Categorize Security Monitoring Systems*, Purdue University, Indiana, 2004