**CS 111 Design Project Lab 1**

**Design Document**

Spencer Tung 004355860

Li Tseng 304272081

**Design Overview**

> Security Objective

What we want to do in this design project is to prevent a process overload attack when using our shell in lab 1ab. When there is an overload attack, out shell may be overloaded with too many bad processes so that our shell can not have processes to be used by other users. Our implementation will try to mitigate this kind of attack and not let them happen if one process tries to fork too many processes.

> Overall Design

First, we aim to limit the maximum number of processes that our shell can fork at any given time. If a process hits the limit, we will kill it if it continues to attempt to fork.

Second, we want to ensure fairness in other programs by checking the number of children that a parent has forked using *pstree*. When the number of processes exceed a certain limit, we will start increasing its nice factor to slowdown its forking.

**Plan for Implementation**

> Monitor the total number of processes

In the first part of our approach, we want to monitor the total number of the processes on the shell, and kill it if it exceeds the given maximum number of processes.

We did this implementation in the function watchdog in execute-command.c. To get the process forked from timetrash, we scan the processes in folder /proc and pass the information in the file to get the gid and pid from it. Then, we can distinguish the processes with same gid as timetrash. As the processes with same gid as timetrash increase, we update the number of process accordingly. When it exceeds the given maximum number of processes, we will kill all the processes with same gid. So that timetrash can not fork anymore processes, or it may occupy too much resources on the shell, which may let the shell crash.

> Environment

At first, we run our program on UCLA CS lion server. However, as we tried to test our program and let it fork as much as it can, it will let the server out of usage. After that, we change our environment to the cs111 distribution on virtual machine. Even if we crash the shell, we just need to reboot it then everything is good again. We won’t affect other users like what we did on the server.

**Summary of Results**

> Robustness Analysis

First, by limiting the maximum number of processes on the shell, we can prevent our shell from being overloaded since it will stop forking when it already has a certain number of processes.

Second, by increasing the nice factor when a process trying to fork too many child processes, we can let it fork slower. Hence, it won’t occupy the shared resources on the shell, which may let other users cannot use the resources.