OPERATING SYSTEMS LAB

Lab Assignment 2
Group 30

Akash Das 20CS10006
Prakhar Singh 20CS10045
Rohit Kumar Prajapati 20CS30041
Saras Umakant Pantulwar 20CS30046

Heuristic used:

As the malware swamps multiple processes so definitely in some time the total number of childern of this malware (childern counted recursively i.e childern of childern are also counted and so on) will be large in number.
But it can happen that some normal (not a malware process) can also have too many childern.
In addition to above heuristic we also used CPU time of the process, as malware process is active for very small time and then sleeps for relatively long time, so choose the process with less CPU utilization.
In all we want process with many childern and very less CPU utilization. That is the process with maximum value of difference betweeen number of childern and CPU utilization is our suspect.

Following is the method to find the CPU time for a process:

First we determine the total time spent for the process:
total_time = utime + stime

Next we get the total elapsed time in seconds since the process started:
seconds = uptime - (starttime / Hertz)

Finally we calculate the CPU usage percentage:
cpu_usage = 100 * ((total_time / Hertz) / seconds)

NOTE:

1. The sysconf(_SC_CLK_TCK) C function call may also be used to return the hertz value
2. /proc/uptime is used for uptime of the system

Codes to find toatl number of childern (recursively) of a process:

```
char command[50];
  sprintf(command, "pstree -p %d | wc -l", pid);
  int num_children = 0;
  if (system(command) == 0) {
    FILE *fp = popen(command, "r");
    if (fp == NULL) {
      perror("popen failed");
      return -1;
    }
    fscanf(fp, "%d", &num_children);
    pclose(fp);
  }
```

This code first gets the PID of the current process using getpid(), then generates the command pstree -p <pid> | wc -l using sprintf(). The pstree command generates a tree of all processes, with the given pid as the root. The wc -l counts the number of lines in the output, which is equivalent to the number of processes in the tree.

The code then runs the command using system(), and if it returns success, it opens a pipe using popen() to read the output of the command. The output is then read using fscanf(), and the number of children is stored in the num_children variable. Finally, the pipe is closed using pclose(), and the number of children is printed to the console.