ProcVis - 3D Process Visualisation

Suraj Kesavan
Computer Science
University of California, Davis
Davis, USA
Email: spkesavan@ucdavis.edu

***Abstract*—Monitoring computer process is a very important component of security analysis to maintain a secure system being accessed by numerous people at one time. One of the major ability is to exercise authority in the system by killing (or) quiting unauthorised usage of resources by a process and the other is to prevent wasteful usage of memory by certain never ending processes. The metrics about processes also lead to discovery of intricate situations where the system's computing ability is not used to its full potential. The main aim of the paper is to visualize the processes in a UNIX system and also analyze dynamic procedure of the process creation and hierarchy. Unix systems provide a file system that acts as an interface to internal data structures in the kernel called /proc file system to provide metrics on processes. We discuss how we modelled Proc-vis - an interface to represent processes in 3-Dimensional environment using server-client architecture as experiment and discuss the visualisation challenges and overcoming with design model**

***Index Terms*—3D Visualization, process monitoring**

I. Introduction

Processes in UNIX systems are an instance of a computer program that is being executed and they enable the user to execute (or) run several programs concurrently based on the number of threads available to the system. Processes use the system resources based on the program's requirement and greatly affect the performance of the system. The processes which consume a lot of resources will rather slow down the functioning of the system as a whole and affect other processes.

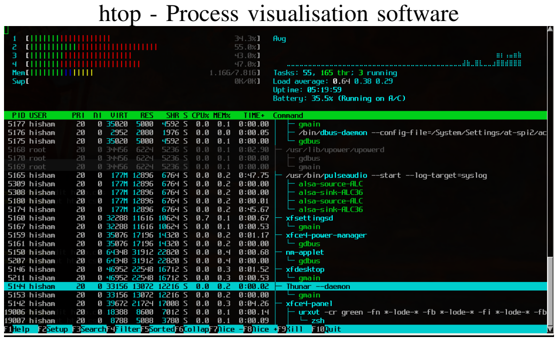
Performance forms the crucial feature of any software being run on a system. In a perfect world, An efficient one would possess short response time, high throughput, low utilisation of computer memory and high availability of computing power. But sad reality disrupts the perfect world, softwares tend to perform poorly and consume a lot of memory whereby affected subsequent processes. For example, A fork bomb is a process that continually replicates itself to deplete all the available system resources and eventually crashing down the system. Apart from computing concern, Some processes try to access other user's data and resources when the system is compromised by an attacker. Intruders of the system tend to break the system and get access to the root user's resource and data.

II. Background

A. Related Work

Currently, the ways to analyze and monitor the processes in UNIX system is using terminal based scripts such as ps[1], top[2], and htop[3]. The scripts list the process in a tabular column fashion and sorts the real time data based on memory. The unix script, ps, doesn't provide real time analysis of the process data as it only displays the current snapshot of the system. Another disadvantage in its usage is the necessity to provide the accurate arguments to display the right information. The command "top", provides a respite from ps by providing real time data with a lot fields eliminating the need to provide arguments. The script, "htop" functions the best among the other available scripts. It follows the same tabular column visualisation but has colors and line graphs to represent the process's metrics.

htop - Process visualisation software



```
  1  [|||||||||||||||||||                    34.3%]   Avg
  2  [||||||||||||||||||||||||||||||||       55.0%]
  3  [||||||||||||||||||||||||               43.0%]
  4  [||||||||||||||||||||||||||             47.0%]
  Mem[||||||||||||||||                  1.16G/7.81G]   Tasks: 55, 165 thr; 3 running
  Swp[                                        0K/0K]   Load average: 0.64 0.38 0.29
                                                       Uptime: 05:19:59
                                                       Battery: 35.5% (Running on A/C)

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 5177 hisham     20   0 35020  5000  4592 S  0.0  0.1  0:00.00 │  ├─ gmain
 5176 hisham     20   0  2952  2080  1976 S  0.0  0.0  0:00.05 │  ├─ /bin/dbus-daemon --config-file=/System/Settings/at-spi2/ac
 5175 hisham     20   0 35020  5000  4592 S  0.0  0.1  0:00.00 │  └─ gdbus
 5168 root       20   0 34456  6224  5236 S  0.0  0.1  0:02.90 ├─ /usr/lib/upower/upowerd
 5170 root       20   0 34456  6224  5236 S  0.0  0.1  0:00.00 │  ├─ gdbus
 5169 root       20   0 34456  6224  5236 S  0.0  0.1  0:00.00 │  └─ gmain
 5165 hisham     20   0  177M 12896  6764 S  0.0  0.2  0:47.75 ├─ /usr/bin/pulseaudio --start --log-target=syslog
 5309 hisham     20   0  177M 12896  6764 S  0.0  0.2  0:00.00 │  ├─ alsa-source-ALC
 5308 hisham     20   0  177M 12896  6764 S  0.0  0.2  0:00.00 │  ├─ alsa-sink-ALC36
 5180 hisham     20   0  177M 12896  6764 S  0.0  0.2  0:00.01 │  ├─ alsa-source-ALC
 5174 hisham     20   0  177M 12896  6764 S  0.0  0.2  0:45.67 │  └─ alsa-sink-ALC36
 5160 hisham     20   0 32288 11616 10624 S  0.7  0.1  0:00.67 ├─ xfsettingsd
 5167 hisham     20   0 32288 11616 10624 S  0.0  0.1  0:00.53 │  └─ gmain
 5159 hisham     20   0 35076 17196 14320 S  0.0  0.2  0:01.17 ├─ xfce4-power-manager
 5161 hisham     20   0 35076 17196 14320 S  0.0  0.2  0:00.00 │  └─ gdbus
 5150 hisham     20   0 64348 31912 22820 S  0.0  0.4  0:00.68 ├─ nm-applet
 5207 hisham     20   0 64348 31912 22820 S  0.0  0.4  0:00.00 │  └─ gdbus
 5146 hisham     20   0 46952 22548 16712 S  0.0  0.3  0:01.52 ├─ xfdesktop
 5211 hisham     20   0 46952 22548 16712 S  0.0  0.3  0:00.53 │  └─ gmain
 5144 hisham     20   0 33156 13072 12216 S  0.0  0.2  0:00.02 ├─ Thunar --daemon
 5153 hisham     20   0 33156 13072 12216 S  0.0  0.2  0:00.00 │  └─ gmain
 5142 hisham     20   0 39672 21724 17008 S  0.0  0.3  0:04.26 ├─ xfce4-panel
19006 hisham     20   0 18388  8600  7012 S  0.0  0.1  0:00.14 │  ├─ urxvt -cr green -fn *-lode-* -fb *-lode-* -fi *-lode-* -fb
19007 hisham     20   0  8788  5088  3780 S  0.0  0.1  0:00.09 │  │  └─ zsh
F1Help  F2Setup F3SearchF4FilterF5SortedF6CollapF7Nice -F8Nice +F9Kill  F10Quit
```

Fig. 1. The interface to view and control process metrices

One of the main disadvantage of these scripts is that provide lines and lines of logging information, which can be intimidating and less conclusive. Also they fail to visualize the interesting situations which arise and it is tough for the user to conclude break of authority or resource depletions. Although the visualization is not very appealing, the real time interface provides the user a variety of actions to be performed on the processes.

Apart from the open source softwares discussed above, Nagios[4] and Centreon[5] have been used as an industry standard to monitor the process related information. The idea of visualisation is inspired from PSDoom[6], a 3D game to provide process information in a game environment. The game involves a currently running processes instantiated as "process monsters" with the program continuously pooling the operating system and creating new processes. The user also gets to interact with the processes and can inflict wounds to kill the corresponding CPU process. The project also exhibited an unique phenomenon, certain processes attacking other processes due to their implementations and exhibits a funny side to the perpetual problem of bad coding practice.

PSDoom

Fig. 2. Doom Game - Visualisation of processes in a 3D environment as game characters

In other operating systems, processes are visualized using softwares such as ”Windows task manager” in Microsoft OS’es and ”Activity monitor” in OSx. These softwares list the processes based on the their maximum CPU usage or memory usage in tabular column. They also have line plots which plot the memory usage of the process with time. The interface also has options to kill the process, stop the process, etc which help the user to organize the processes currently in context. The latest versions of these softwares provide more visualization by introducing more graphs but nevertheless, they fail to convey information about the situations which arise like explained before.

B. /proc Filesystem

Linux Operating systems have a directory /proc, which acts as an interface to internal data structures in the kernel and provide information about the system. The proc filesystem (procfs)[7] is a special filesystem in Unix-like operating systems that presents information about processes and other system information in a hierarchical file-like structure, providing a more convenient and standardized method for dynamically accessing process data held in the kernel than traditional tracing methods or direct access to kernel memory. For example, the process with pid equal to 1 would have a subdirectory named /proc/1/ in the proc file system. The file read operation on /proc/[PID]/status provides all the information about the process with pid equal to PID. Likewise, the files /proc/[PID]/statm reveal the memory usage of the entire system specifying various metrics such as RAM size, VRAM size, number of processors in the system, etc. The Table 1 reveals some of the common files which are accessed to analyse the metric of memory, process, drivers, kernel, etc.

We use /proc file system to get the metrics about every process running in the system. The prototype version of the visualisation uses the data from /proc/[PID]/status, /proc/[PID]/statm, /proc/[PID]/cmdline, /proc/[PID]/cpu and /proc/meminfo alone to obtain the process metrices. Also the prototype version concentrates on visualising only the process memory.

Filename	Purpose
/proc/[PID]/cmdline	Command line arguments.
/proc/[PID]/cpu	Current and last cpu in which it was executed.
/proc/[PID]/cwd	Link to the current working directory.
/proc/[PID]/environ	Values of environment variables.
/proc/[PID]/exe	Link to the executable of the process.
/proc/[PID]/fd	Directory, which contains all file descriptors.
/proc/[PID]/status	Status information of a process.
/proc/[PID]/maps	Current memory mapped regions.
/proc/[PID]/smaps	Memory usage of process.
/proc/[PID]/statm	Memory usage of a process with pid.
/proc/interrupts	Current interrupts in use.
/proc/buddyinfo	A tool to diagnose external fragmentation.
/proc/pagetypeinfo	External fragmentation
/proc/meminfo	Memory utilization across the system.
/proc/ide/	Information about the drivers.
/proc/net	Network usage across the system.

TABLE I
/PROC FILESYSTEM

III. Visualisation

A. System and environment

Ubuntu 16.04 was used to provide the real-time data from the /proc filesystem with server logic coded in NodeJS. An API was built to query the metrics on providing the pid. The client logic was coded in Javascript using Three.js library, written on top of webgl. The server and client communicated using sockets with events responded by a promise (Q library).

B. Idea and methodology

Hive plot

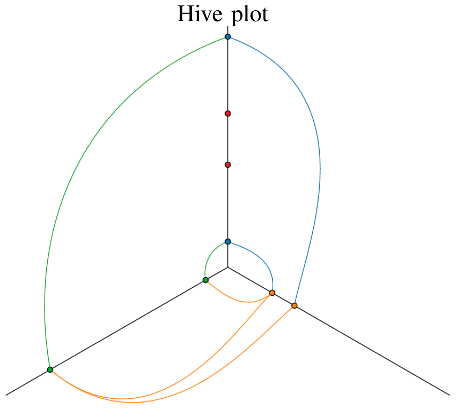


Fig. 3. Hive plot: A prototype version of axises representing users and nodes as processes

The visualization was primarily aimed at providing three abstract level of detailing, First, the metrics derived from /proc filesystem must be represented in a single scope or container similar to the top and htop. Second is that visualisation must answer questions like ”Who owns the process?”,”How much memory does it consume?”,”What is the current state of the process?”, etc. And the third level involves understanding of the inter-process communication between the sub-processes. The third aspect of visualisation is a key and innovative concept which none of the previously existing applications could denote. The basic structure of the prototype is derived from hive plots[8] as shown in the figure 3, The axises represent the users accessing the system and the nodes represent each process. The z-axis line represents the root user who possess all the permissions relating the processes and memory. The curved lines between the users represent the inter process relationship between these nodes or processes. Each axis is scaled by memory exponentially, meaning a node at the end of the axis consumes more memory than the other processes. The user can also click on the node to reveal the metrics relating that process node. In the figure 3, there are 3 users, namely root, user1, user2, are accessing the system with users accessing the root permissions twice and as well communicating between each other. One major drawback of this method is that, if there are more than 3 users, then the whole plot gets cluttered and accessing each node information becomes extremely different.

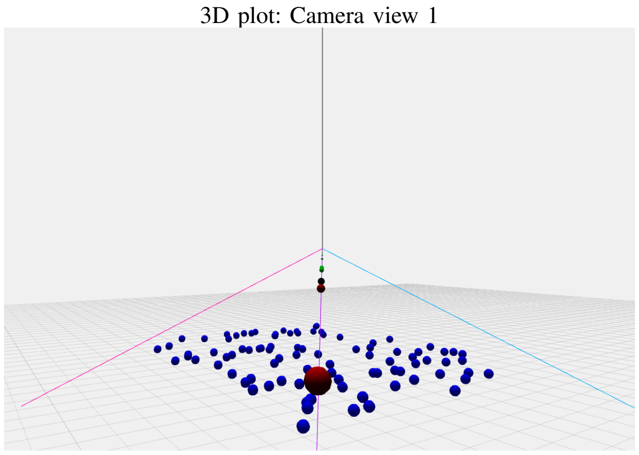


Fig. 4. A 3 dimensional prototype version of axises representing users and nodes as processes

C. *Three dimensional Representation*

The two-dimensional plotting gets cluttered when there are many users accessing the system, to counter this, a 3-Dimensional prototype was built with the same design structure. Figure 4, represents the 3 dimensional plot of the process nodes, with a zoomable and dragable plane. The advantage of 3 Dimensional representation is very evident with the ability to move around the entire environment and accessing each node individually. The nodes when clicked would reveal the metrics as it was in the 2D prototype.

Figure 5 is the view when the camera is moved to a perpendicular plane to figure 4. This figure shows the processes are scaled according to the memory usage. A color hue is used to differentiate between the process based on their current state. Darker nodes represent a sleeping process and a lighter node represents an active process. Green nodes represent the process which consume less than 100MB to execute, whereas red nodes consume more than 100MB to run. Blue nodes on the x-y plane are the processes which aren't using any memory to run. The table 2 shows the various color representation used to differentiate the respective processes

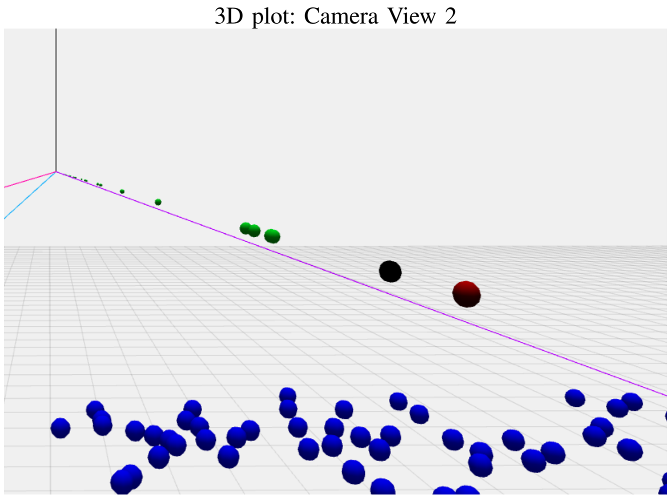


Fig. 5. A view from the perpendicular plane

Node color	Meaning
Green node	Process with memory lesser than 100MB
Red node	Process with memory greater than 100MB.
Blur node	Dormant process.
Hue factor = 1	Running process.
Hue factor = 0.75	sleeping uninterruptible wait.
Hue factor = 0.5	Sleeping process.
Hue factor = 0.25	Zombie process.
Hue factor = 0.6	Stopped process.

TABLE II
COLOR HUE REPRESENTATION

Figure 6 shows how the prototype handles processes which require authorisation. The red line connecting the green node on the user's axis and root axis denotes that the process failed to establish as a root process. If the user was in the whitelist of root, then the line connecting the nodes would be green, showing a success in establishment of authority. The prototype can be extended to represent fork processes as well, where the process node creates itself repeatedly and causes the system crash by showing the loops. These lines also help us to represent the inter-process communication.

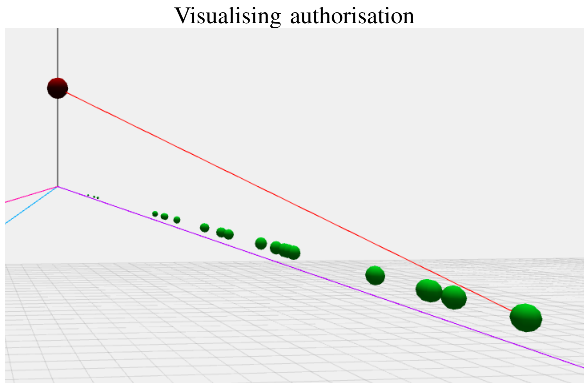


Fig. 6. Red line representing sudo process from user

Number of nodes	Load time [3D model]	Load time [2D model]
10	1.61s	0.56s
100	2.96s	1.32s
500	6.78s	2.44s
1000	20.34s	4.53s

TABLE III
COLOR HUE REPRESENTATION

D. Client - Server Architecture

The nodes are made clickable and onclick, a query is made by the client requesting the metric data of the node using sockets. The server looks up into the specific file in the /proc filesystem and packages the output into JSON.

IV. EVALUATION

The project involved two prototypes, one with 2D visualization and a 3D visualization both having its own merits and demerits. The 2D visualisation would get cluttered with more users and processes making it hard to visualise the process communication. It wasn't possible to represent all the processes as nodes as the axis would get cluttered with nodes making it an inefficient representation in comparison to top or htop. But on a brighter side, 2D visualisation is much faster than 3D visualisation as it avoids the unnecessary GPU computations. While the 3D visualisation eliminates the cluttering and accessibility, field of view affects the rendering, as the camera moves away from the map, the nodes become smaller and smaller leading to an inefficient representation when there are many nodes of higher memory usage.

The method used in 3D model proves to be a better method to denote much dynamic situations but it comes with cost of rendering speeds. A small experiment was performed on to find the effect of number of nodes on the rendering of 3D model. Table 3 shows the comparison of load time to the number of nodes to be rendered in the map. It clearly shows that as the number of nodes increase, the 3D model took more time to create geometries and textures which were rendered by the webgl library three.js. In comparison, 2D model took far less time to project the hive plot written in d3.js. The rendering almost got done in a linear time with the 2D model visualisation.

V. CONCLUSION AND FUTURE WORK

The project established that there is a requirement for a process visualisation software with more functionality and should also convey the happenings inside the machine with clear distinction. Although 3D visualisation with base structure similar to a hive plot solves the problem, there could be better ways of visualising the information. For future work, the project's heavy reliability on three.js (over webgl) can be moved to openGL to increase the speeds. Another advantage with server written in C as well would be helpful to pool data more efficiently. The current pooling rates tend to approximately 20s in case of many nodes. This interval can be reduced by rendering lesser number of nodes. Data sanitation based on the important processes would lead to a much fast rendering. Another functionality is that we can store the process metrics in a data structure by recording over time and relaying the information as a visualisation rather than a real-time visualisation. This method helps us to get rid of the higher load times as we pre-calculate from the metrics.

The visualisation can also be extended to other data from the metrics such as kernel, cpu timings , drivers and the network providing the highest layer of visualisation that could be built on top of the proc file system.

REFERENCES

[1] ps Man page, "https://www.freebsd.org/cgi/man.cgi?query=psmanpath=SuSE+Linux/i386+11.3"

[2] top Man page, "http://www.unixtop.org/man.shtml"

[3] Htop, "http://hisham.hm/htop/"

[4] Nagios, "https://www.nagios.com/products/nagios-core/"

[5] Centreon, "https://www.centreon.com"

[6] PSDoom:, "http://www.cs.unm.edu/ dlchao/flake/doom/after.html"

[7] Krzywinski, Martin and Birol, Inanc and Jones, Steven JM and Marra, Marco A, "Hive plotsrational approach to visualizing networks," -