SOFTWARE SECURITY PROJECT – PROJECT PART 2

Team members: Aakash Jayavel, Diya Anna Biju and Sneha Ramkumar

Files:

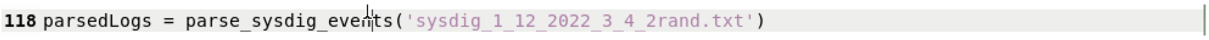
1. **backtracking-points-of-interest.py** – This file contains the sysdig logs parser, graph creation function and backtracking points of interest function.
2. **sysdig_1_12_2022_3_4_2rand.txt** – This file contains all the sysdig logs for a timeframe of interest. This file has the logs for the execution of our point of interest. Which is executing execution.sh which writes to shfile.sh
3. **execution.sh** – This file writes content to shfile.sh and executes shfile.sh
4. **shfile.sh** – This file will echo all its contents.

How we generated the log file and captured out point of interest:

1. We use the command given below to get the sysdig logs
   sudo sysdig -p "%evt.num %evt.rawtime.s.%evt.rawtime.ns %evt.cpu %proc.name (%proc.pid) %evt.dir %evt.type cwd=%proc.cwd %evt.args latency=%evt.latency exepath=%proc.exepath proc_pid =%proc.pid file_id=%fd.num fd_name=%fd.name fd_filename=%fd.filename" "proc.name!=tmux and (evt.type=read or evt.type=readv or evt.type=write or evt.type=writev or evt.type=accept or evt.type=execve or evt.type=clone or evt.type=pipe or evt.type=rename or evt.type=sendmsg or evt.type=recvmsg)" and proc.name!=sysdig > sysdig_28_11_2022_3_4_2.txt
2. In another terminal execute execution.sh using sh execution.sh
3. The event will be captured in the sysdig logs post which we can stop the sysdig command.

Instruction to execute the backtracking algorithm to find points of interest:

1. In backtracking-points-of-interest.py file please add the path to **sysdig_1_12_2022_3_4_2rand.txt** in line 118 where we call the function **parse_sysdig_events.** For Example,

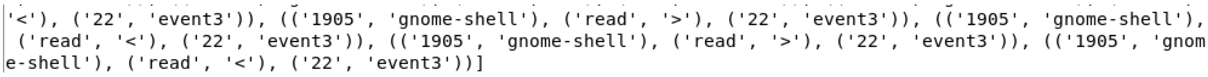


```
118 parsedLogs = parse_sysdig_events('sysdig_1_12_2022_3_4_2rand.txt')
```

2. The whole logs file will be created as a graph and stored in the file logsTestGraph.svg after the file backtracking-points-of-interest.py is run. The svg file can be opened in any browser. The graph is created by the create_graphs_from_tuples function.
3. The backtracking algorithm backtracks our point of interest and outputs the backtracked graph to the BackTrackGraph.svg file. The output graph is generated by the backtrackPointOfInterest function to which we pass out points of interest 17840 sh -> 1 shfile.sh and the graph of the logs.

Results and Analysis:

1. From the logs **sysdig_1_12_2022_3_4_2rand.txt** file we can generate the tuples.



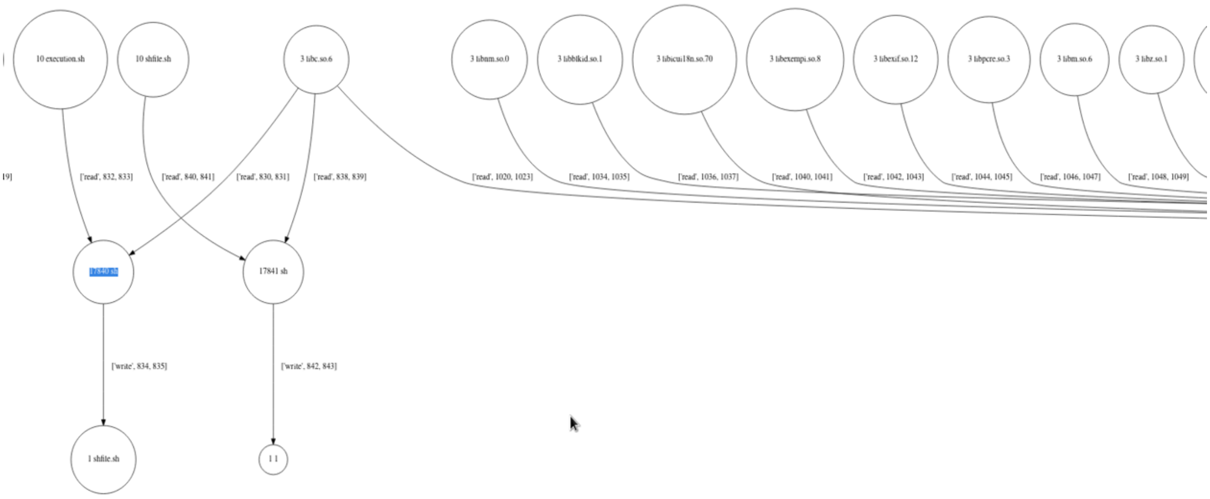
```
'<'), ('22', 'event3')), (('1905', 'gnome-shell'), ('read', '>'), ('22', 'event3')), (('1905', 'gnome-shell'),
 ('read', '<'), ('22', 'event3')), (('1905', 'gnome-shell'), ('read', '>'), ('22', 'event3')), (('1905', 'gnom
e-shell'), ('read', '<'), ('22', 'event3'))]
```

   Sample incoming tuple:
   (('1905', 'gnome-shell'), ('read', '>'), ('22', 'event3'))
   1905 – Process ID
   gnome-shell – Process Name
   read – operation
   > - incoming event
   22 – Object identifier
   event3 – Object name

   Sample outgoing tuple for the same event:
   (('1905', 'gnome-shell'), ('read', '<'), ('22', 'event3'))
   < - outgoing event

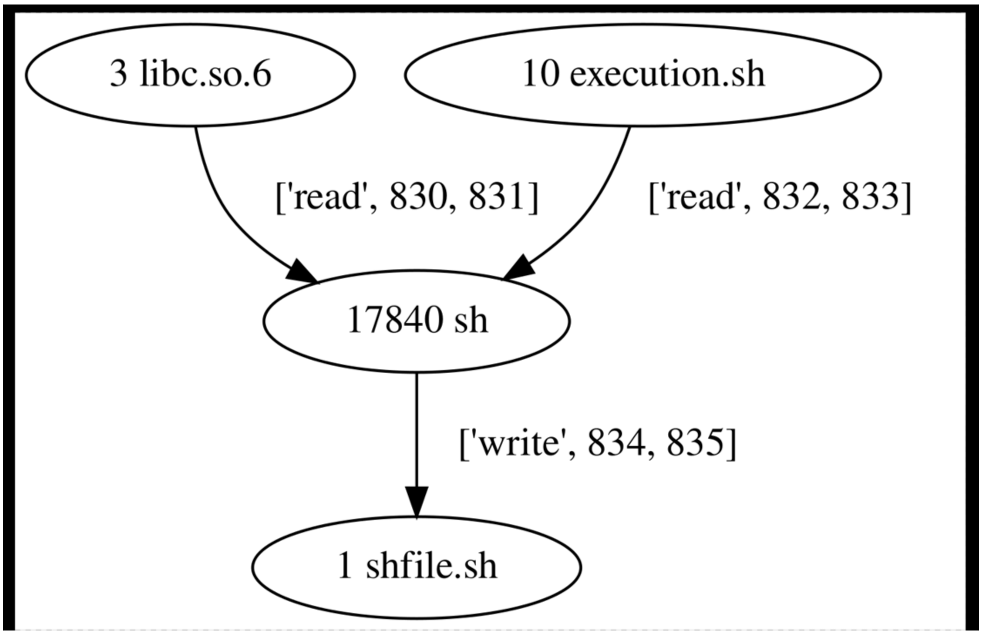
   The incoming and outgoing event is used to calculate the latency.

2. A part of the graphs containing part of our point of interest is shown below. This graph is present in logsTestGraph.svg



   The node 10 execution.sh -> 17840 sh -> 1 shfile.sh is where we have run the execution.sh

3. Finally, after backtracking our point of interest, we get the output graph containing only the required nodes.



3 libc.so.6
10 execution.sh
['read', 830, 831]
['read', 832, 833]
17840 sh
['write', 834, 835]
1 shfile.sh