CSE 545 Software Security
Lab 3: System Auditing

Xusheng Xiao

Associate Professor
School of Computing and Augmented Intelligence
Arizona State University

Installation

- Github wiki: https://github.com/draios/sysdig/wiki
- Under the section Setup, you can find the instruction for installation.
  - https://github.com/draios/sysdig/wiki/How-to-Install-Sysdig-for-Linux
- OS requirement: only Linux will work!
  - Other option: use Virtual Box to launch a Linux VM
- Recommended installation:

```
curl -s https://s3.amazonaws.com/download.draios.com/stable/install-sysdig | sudo bash
```

Virtual Box

- If you are using MacOS or Windows, then you may download the provided VM file:
  - Link:

- Tutorial for using Virtual Box:
  - Download the tutorial in Canvas

Sysdig Output

```
$ sysdig
34378 12:02:36.269753803 2 echo (7896) > close fd=3(/usr/lib/locale/locale-archive)
34379 12:02:36.269754164 2 echo (7896) < close res=0
34380 12:02:36.269781699 2 echo (7896) > fstat fd=1(/dev/pts/3)
34381 12:02:36.269783882 2 echo (7896) < fstat res=0
34382 12:02:36.269784970 2 echo (7896) > mmap
34383 12:02:36.269786575 2 echo (7896) < mmap
34384 12:02:36.269827674 2 echo (7896) > write fd=1(/dev/pts/3) size=12
34385 12:02:36.269839477 2 echo (7896) < write res=12 data=hello world.
34386 12:02:36.269843986 2 echo (7896) > close fd=1(/dev/pts/3)
34387 12:02:36.269844466 2 echo (7896) < close res=0
34388 12:02:36.269844816 2 echo (7896) > munmap
34389 12:02:36.269850803 2 echo (7896) < munmap
34390 12:02:36.269851915 2 echo (7896) > close fd=2(/dev/pts/3)
34391 12:02:36.269852314 2 echo (7896) < close res=0
```

Sysdig Field

- Sysdig output is organized using classes and fields
  - Class:
    - fd: file descriptor
    - proc: process
    - thread: thread
    - evt: system call event
    - Others: user, group, syslog, …
  - Field:
    - fd.num: the unique number identifying the file descriptor.
    - proc.pid: the id of the process generating the event.
    - evt.type :the name of the event (e.g. 'open’).

- By default, sysdig prints the information for each event on a single line, with the following format:

```
*%evt.num %evt.time %evt.cpu %proc.name (%thread.tid) %evt.dir %evt.type %evt.args
```

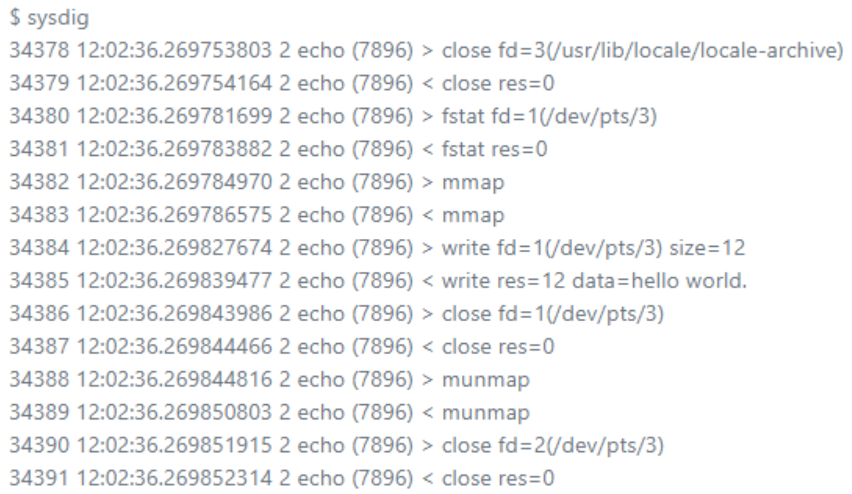
Default Fields

*%evt.num %evt.time %evt.cpu %proc.name (%thread.tid) %evt.dir %evt.type %evt.args

- evt.num is the incremental event number
- evt.time is the event timestamp
- evt.cpu is the CPU number where the event was captured
- proc.name is the name of the process that generated the event
- thread.tid is the TID that generated the event, which corresponds to the PID for single thread processes
- evt.dir is the event direction, > for enter events and < for exit events
- evt.type is the name of the event, e.g. 'open' or 'read'
- evt.args is the list of event arguments. In case of system calls, these tend to correspond to the system call arguments, but that's not always the case: some system call arguments are excluded for simplicity or performance reasons.

Additional Information

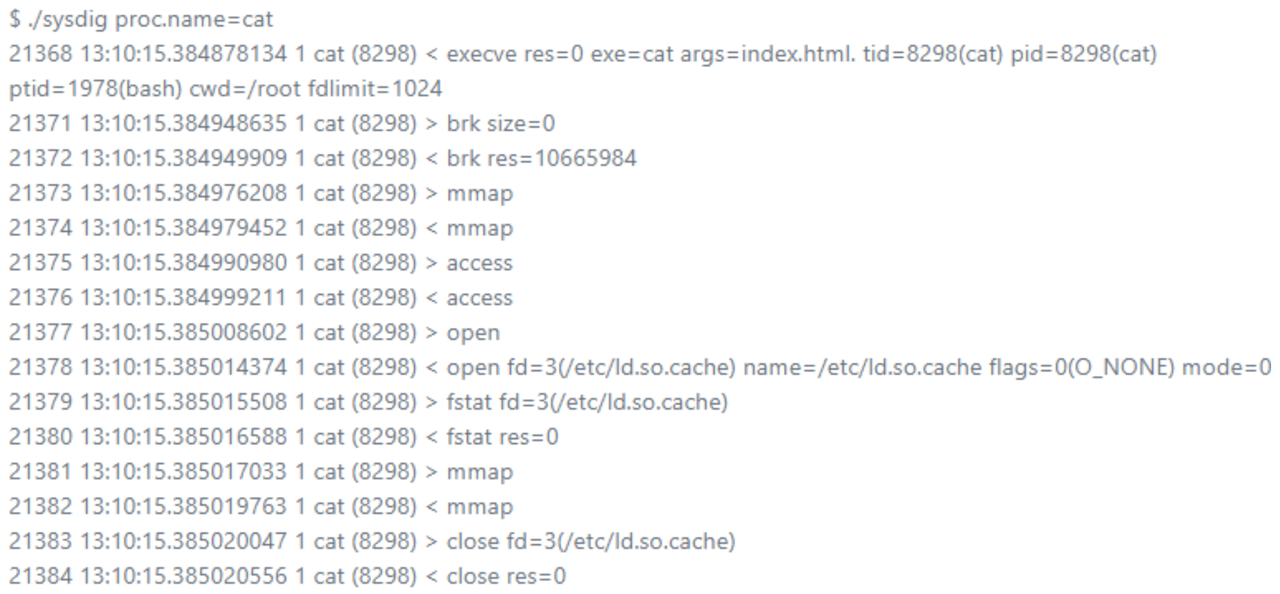
- For most system calls, sysdig shows two separate entries: an enter one (marked with a '>') and an exit one (marked with a '<'). This makes it easier to follow the output in multi-process environments.
- File descriptors are resolved. This means that, whenever possible, the FD number is followed by a human-readable representation of the FD itself: the tuple for network connections, the name for files, and so on. The exact format used to render an FD is the following: num(<type>resolved_string) where:
- num is the FD number
- resolved_string is the resolved representation of the FD, e.g. 127.0.0.1:40370->127.0.0.1:80 for a TCP socket
- type is a single-letter-encoding of the fd type, and can be one of the following:
  - f for files
  - 4 for IPv4 sockets
  - 6 for IPv6 sockets
  - u for unix sockets
  - s for signal FDs
  - e for event FDs
  - i for inotify FDs
  - t for timer FDs



```
$ sysdig
34378 12:02:36.269753803 2 echo (7896) > close fd=3(/usr/lib/locale/locale-archive)
34379 12:02:36.269754164 2 echo (7896) < close res=0
34380 12:02:36.269781699 2 echo (7896) > fstat fd=1(/dev/pts/3)
34381 12:02:36.269783882 2 echo (7896) < fstat res=0
34382 12:02:36.269784970 2 echo (7896) > mmap
34383 12:02:36.269786575 2 echo (7896) < mmap
34384 12:02:36.269827674 2 echo (7896) > write fd=1(/dev/pts/3) size=12
34385 12:02:36.269839477 2 echo (7896) < write res=12 data=hello world.
34386 12:02:36.269843986 2 echo (7896) > close fd=1(/dev/pts/3)
34387 12:02:36.269844466 2 echo (7896) < close res=0
34388 12:02:36.269844816 2 echo (7896) > munmap
34389 12:02:36.269850803 2 echo (7896) < munmap
34390 12:02:36.269851915 2 echo (7896) > close fd=2(/dev/pts/3)
34391 12:02:36.269852314 2 echo (7896) < close res=0
```

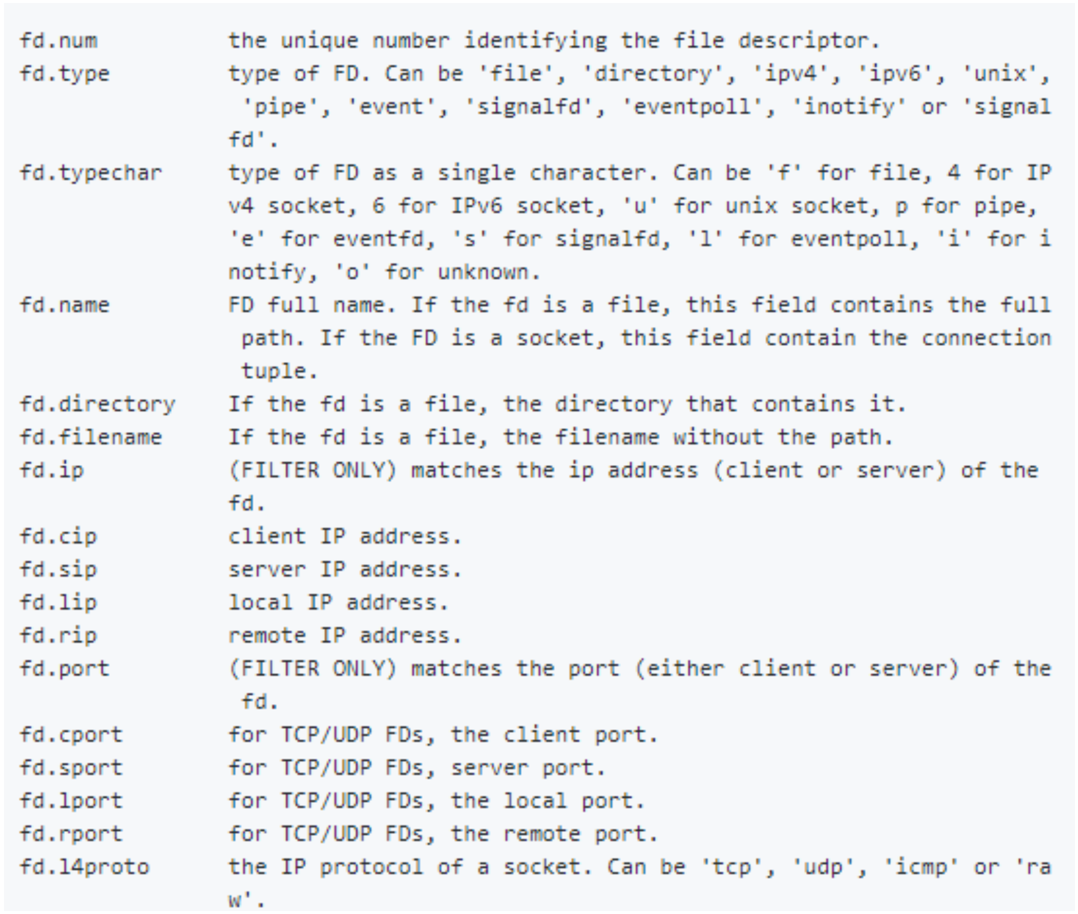
Field Filter

- Sysdig's filtering system is powerful and versatile, and is designed to look for needles in a haystack. Filters are specified at the end of the command line, like in tcpdump, and can be applied to both a live capture or a capture file.
- Example:



```
$ ./sysdig proc.name=cat
21368 13:10:15.384878134 1 cat (8298) < execve res=0 exe=cat args=index.html. tid=8298(cat) pid=8298(cat)
ptid=1978(bash) cwd=/root fdlimit=1024
21371 13:10:15.384948635 1 cat (8298) > brk size=0
21372 13:10:15.384949909 1 cat (8298) < brk res=10665984
21373 13:10:15.384976208 1 cat (8298) > mmap
21374 13:10:15.384979452 1 cat (8298) < mmap
21375 13:10:15.384990980 1 cat (8298) > access
21376 13:10:15.384999211 1 cat (8298) < access
21377 13:10:15.385008602 1 cat (8298) > open
21378 13:10:15.385014374 1 cat (8298) < open fd=3(/etc/ld.so.cache) name=/etc/ld.so.cache flags=0(O_NONE) mode=0
21379 13:10:15.385015508 1 cat (8298) > fstat fd=3(/etc/ld.so.cache)
21380 13:10:15.385016588 1 cat (8298) < fstat res=0
21381 13:10:15.385017033 1 cat (8298) > mmap
21382 13:10:15.385019763 1 cat (8298) < mmap
21383 13:10:15.385020047 1 cat (8298) > close fd=3(/etc/ld.so.cache)
21384 13:10:15.385020556 1 cat (8298) < close res=0
```

Part of the Fields

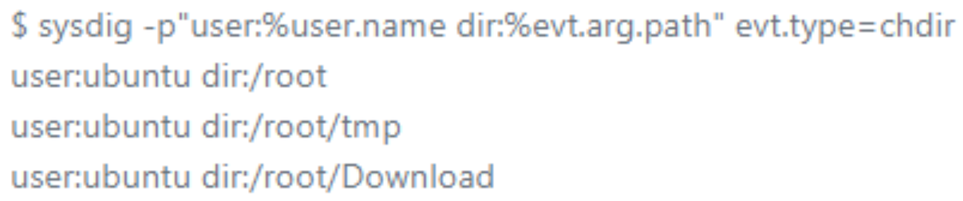


```
fd.num          the unique number identifying the file descriptor.
fd.type         type of FD. Can be 'file', 'directory', 'ipv4', 'ipv6', 'unix',
                 'pipe', 'event', 'signalfd', 'eventpoll', 'inotify' or 'signal
                fd'.
fd.typechar     type of FD as a single character. Can be 'f' for file, 4 for IP
                v4 socket, 6 for IPv6 socket, 'u' for unix socket, p for pipe,
                'e' for eventfd, 's' for signalfd, 'l' for eventpoll, 'i' for i
                notify, 'o' for unknown.
fd.name         FD full name. If the fd is a file, this field contains the full
                 path. If the FD is a socket, this field contain the connection
                 tuple.
fd.directory    If the fd is a file, the directory that contains it.
fd.filename     If the fd is a file, the filename without the path.
fd.ip           (FILTER ONLY) matches the ip address (client or server) of the
                fd.
fd.cip          client IP address.
fd.sip          server IP address.
fd.lip          local IP address.
fd.rip          remote IP address.
fd.port         (FILTER ONLY) matches the port (either client or server) of the
                 fd.
fd.cport        for TCP/UDP FDs, the client port.
fd.sport        for TCP/UDP FDs, server port.
fd.lport        for TCP/UDP FDs, the local port.
fd.rport        for TCP/UDP FDs, the remote port.
fd.l4proto      the IP protocol of a socket. Can be 'tcp', 'udp', 'icmp' or 'ra
                w'.
```

Fulllist is at https://github.com/draios/sysdig/wiki/Sysdig-User-Guide

Formatting Output

- Output customization happens with the –p command line flag, and works somewhat similarly to the C printf syntax. Here's an example:



```
$ sysdig -p"user:%user.name dir:%evt.arg.path" evt.type=chdir
user:ubuntu dir:/root
user:ubuntu dir:/root/tmp
user:ubuntu dir:/root/Download
```

- Notes about the –p formatting syntax:
  - Fields must be prepended with a %
  - You can add arbitrary text in the string, exactly as you would do in the C printf.
  - By default, a line is printed only if all the fields specified by –p are present in the event. You can, however, prepend the string with a * to make it print no matter what. In that case, the missing fields will be rendered as <NA>.

Specific Format for the Provided Parser

- 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.name!=tmux and (evt.type=read or evt.type=readv or evt.type=write or evt.type=writev or evt.type=fcntl 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 > filename.txt (Your log file)
- Red section: defines the format processed by the parser, if you want to change this section, you also need change the given code, otherwise it is very possible that the parser will not work.
- Green section: I provide some filter here. You could change the proc.name based on your own need, but I don't suggest you change evt.type, because only these events can be processed by the parser. Some system events will dramatically increase the size of your log file. You could add more event types if you think they are necessary, but you also need carefully revise the code.
- The blue section: is the path of your log file. If you don't use the absolute path, you can find your log file in current working directory.

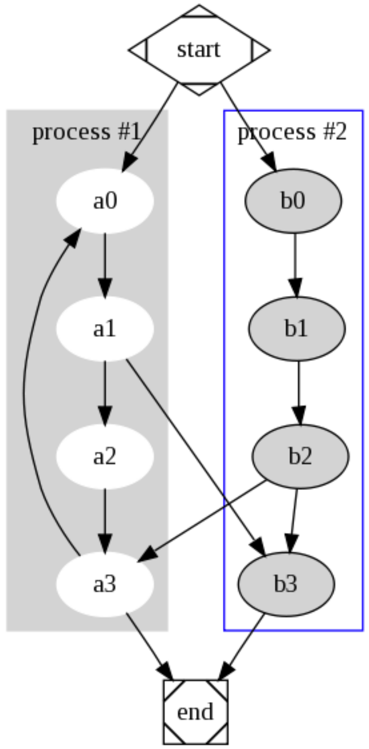
Note: Limit the size of your log file(<=100MB). The huge graph is hard to process and visualize on your personal laptop.

Graph Generation

- Once you run the code, the output should be a dot file.
- Install Graphviz:
  - See next two slides
- You can convert this dot file to svg figure, and view it in your web browser.
  - Command: dot -Tsvg filename(change it based on your own file).dot > filename(change it based on your own file).svg

Graphviz

- Graphviz is open source graph visualization software
- Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks
- Graphviz uses DOT, a graph description language, to generate graphs.



https://graphs.grevian.org/example

Install Graphviz

- Website:
  - Windows (download and execute .msi file): https://graphviz.gitlab.io/_pages/Download/Download_windows.html
  - Mac:
    - Install brew: https://brew.sh/
    - Open terminal and type: brew install graphviz
  - Linux:
    - sudo apt install graphviz

Troubleshooting

- Check the output of sysdig
  - The problem is usually that sysdig gives some output with unexpected format
  - Solution: You could filter it by proc.name

- Check your sysdig command is correct or not. If you change the command, don’t forget to change code.