Enhanced filtering of data using data-driven analysis

Jean-Christian Kouamé
Michel Dagenais

Progress Report Meeting
May 2015

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Presentation Plan

- Introduction
- Update from December
- Other features
- Advantages
- Performance
- Interesting use cases
- Filter events view
- Further work
- Demo

Introduction

- Reminder
  - Previous works (Florian Wininger)
    - Custom XML analysis
  - Goal of the project
    - Filter data
    - Detect complex defaults
    - Follow mecanism
    - Generate high-level events

Update from December

- XML structure
  - 3 main entities
    - **Finite state machine (FSM)**
      - Describe the scenarios
    - **Transitions**
      - Conditions that trigger the state transitions
      - Conditions based on events or on the time
    - **Actions**
      - Action to execute
      - Supported actions :
        - State changes
        - Generate synthetic events
        - Start a new FSM

Update from December

- **XML structure**

```
<filterHandler filterName="sched_switch">
    <initialFsm id="sched_switch" />
    <transitionInput id="sched_switch">
        <event eventName="sched_switch"/>
    </transitionInput>
    <action id="update Current_thread">
        <stateChange>
            <stateAttribute type="location" value="CurrentCPU" />
            <stateAttribute type="constant" value="Current_thread" />
            <stateValue type="eventField" value="next_tid" />
        </stateChange>
    </action>
    <fsm id="sched_switch" multiple="false">
        <precondition input="sched_switch"/>
        <stateTable>
            <stateDefinition name="sched_switch">
                <transition input="sched_switch" next="sched_switch" action="update Current_thread" />
                <transition input="#other" next="sched_switch" />
            </stateDefinition>
        </stateTable>
        <initialState id="sched_switch"/>
    </fsm>
</filterHandler>
```

Transition

Action

FSM

Others features

- Precondition
- Preaction
- Circular FSM
- FSM Singleton
- Activate/deactivate debug mode

Advantages

- Flexibility
- Isolate regions
- Realize abstraction
- Possibility of use of a graphic interface in order to design the filters (Simon's research)
- Reduce the difficulty of the analysis

Interesting use cases

- Chroot jail escape
  - Select a directory as a root folder for a file system file
  - Possibility to escape the jail with a root privilege
  - Chroot(), fchdir()
  - More details at https://filippo.io/escaping-a-chroot-jail-slash-1/

Interesting use cases

- Chroot jail escape

```
<fsm id="chroot">
	<stateTable>
		<stateDefinition name="wait_syscall_entry_chroot">
			<transition input="syscall_entry_chroot" next="syscall_entry_chroot" action="sys_chroot_founded" saveSpecialFields="true"/>
			<transition input="#other" next="wait_syscall_entry_chroot" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_chroot" >
			<transition input="syscall_exit_chroot:thread_thread" next="syscall_exit_chroot" saveSpecialFields="true" clearSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_chroot" />
		</stateDefinition>
		<stateDefinition name="syscall_exit_chroot">
			<transition input="syscall_entry_chdir:thread_thread" next="syscall_entry_chdir" saveSpecialFields="true"/>
			<transition input="syscall_entry_open:thread_thread" next="syscall_entry_open" saveSpecialFields="true"/>
			<transition input="syscall_entry_exit:thread_thread" next="syscall_entry_exit" saveSpecialFields="true"/>
			<transition input="#other" next="syscall_exit_chroot"/>
		</stateDefinition>
		<stateDefinition name="syscall_entry_chdir" >
			<transition input="syscall_exit_chdir:thread_thread" next="exit_fsm" saveSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_chdir" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_exit" >
			<transition input="syscall_exit_exit:thread_thread" next="exit_fsm" saveSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_exit" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_open" >
			<transition input="syscall_exit_open:thread_thread" next="chrootAttack" action="warning_chroot_jail_escape" saveSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_open" />
		</stateDefinition>
		<stateDefinition name="exit_fsm">
			<transition input="#other" next="exit_fsm"/>
		</stateDefinition>
		<stateDefinition name="chrootAttack">
			<transition input="#other" next="chrootAttack"/>
		</stateDefinition>
	</stateTable>
	<initialState id="wait_syscall_entry_chroot"/>
	<enState id="chrootAttack" />
	<abandonState id="exit_fsm" />
</fsm>
```

Interesting use cases

- File access
  - Find all file access
  - From their opening to their closing

Interesting use cases

- File access

```
<fsm id="file_access">
	<stateTable>
		<stateDefinition name="wait_syscall_entry_open">
			<transition input="syscall_entry_open" next="syscall_entry_open" action="sys_x_founded" saveSpecialFields="true"/>
			<transition input="#other" next="wait_syscall_entry_open" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_open" >
			<transition input="syscall_exit_open:thread_thread" next="wait_syscall_entry_close" saveSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_open" />
		</stateDefinition>
		<stateDefinition name="wait_syscall_entry_close">
			<transition input="syscall_entry_close:thread_thread" next="syscall_entry_close" saveSpecialFields="true"/>
			<transition input="#other" next="wait_syscall_entry_close" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_close" >
			<transition input="syscall_exit_close:thread_thread" next="syscall_exit_x" saveSpecialFields="true" clearSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_close" />
		</stateDefinition>
		<stateDefinition name="syscall_exit_x">
			<transition input="#other" next="syscall_exit_x"/>
		</stateDefinition>
	</stateTable>
	<initialState id="wait_syscall_entry_open"/>
	<endState id="syscall_exit_x" />
</fsm>
```

Interesting use cases

- Average of data read
  - Possibility to compute statistic
  - Usage of a timer FSM
  - Possibility to use XML view to show the data

Interesting use cases

- Average of data read

```
<fsm id="syscall">
	<precondition input="tid_18985" /> -->
	<preaction action="restoreDataCount" /> -->
	<stateTable>
		<stateDefinition name="wait_syscall_entry_x">
			<transition input="syscall_entry_x" next="syscall_entry_x" action="sys_x_founded" saveSpecialFields="true"/>
			<transition input="#other" next="wait_syscall_entry_x" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_x" >
			<transition input="syscall_exit_x:thread_thread" next="syscall_exit_x" action="exit_syscall_founded" saveSpecialFields="true" clearSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_x" />
		</stateDefinition>
		<stateDefinition name="syscall_exit_x">
			<transition input="#other" next="syscall_exit_x"/>
		</stateDefinition>
	</stateTable>
	<initialState id="wait_syscall_entry_x"/>
	<endState id="syscall_exit_x" />
</fsm>
SCHED_SWITCH -->
<fsm id="sched_switch" multiple="false">
	<precondition input="sched_switch"/>
	<stateTable>
		<stateDefinition name="sched_switch">
			<transition input="sched_switch" next="sched_switch" action="update Current_thread" />
			<transition input="#other" next="sched_switch" />
		</stateDefinition>
		<stateDefinition name="fake_end">
			<transition input="#other" next="fake_end"/>
		</stateDefinition>
	</stateTable>
	<initialState id="sched_switch"/>
	<endState id="fake_end" />
</fsm>
<fsm id="timer" multiple="false" >
	<stateTable>
		<stateDefinition name="restart_timer" automatic="true">
			<transition input="dummy" next="check" action="restoreDataCount" />
		</stateDefinition>
		<stateDefinition name="check">
			<transition input="init" next="reach_20_ms" action="init_timer" /> -->
			<transition input="reach_20_ms" next="restart_timer"/>
			<transition input="#other" next="check"/>
		</stateDefinition>
	</stateTable>
	<initialState id="restart_timer"/>
</fsm>
```

Interesting use cases

- Follow a process creation tree
  - Intruders could access to a server and obtain root privilege and get a shell.
  - Follow all process that descending from a shell process created by Apache

Interesting use cases

- Follow process creation tree

```
<fsm id="fork_sh" multiple="false">
    <stateTable>
        <stateDefinition name="wait_sched_process_fork">
            <transition input="sched_process_fork" next="process_forked" action="sched_process_fork1" />
            <transition input="#other" next="wait_sched_process_fork" />
        </stateDefinition>
        <stateDefinition name="process_forked" automatic="true">
            <transition input="process_is_sh" next="wait_sched_process_fork" action="process_is_sh" />
            <transition input="#other" next="wait_sched_process_fork" />
        </stateDefinition>
    </stateTable>
    <initialState id="wait_sched_process_fork"/>
</fsm>
<fsm id="follow_process">
    <stateTable>
        <stateDefinition name="wait_sched_process_fork">
          <transition  input="sched_process_fork:root_is_sh" next="wait_process_exit" action="save_data" />
          <transition input="#other" next="wait_sched_process_fork" />
        </stateDefinition>
        <stateDefinition name="wait_process_exit">
            <transition input="sched_process_exit:same_tid" next="process_exited" action="gen_process_event" />
            <transition input="#other" next="wait_process_exit" />
        </stateDefinition>
        <stateDefinition name="process_exited">
            <transition input="#other" next="wait_root_sh" />
        </stateDefinition>
    </stateTable>
    <initialState id="wait_sched_process_fork"/>
    <endState id="process_exited" />
</fsm>
```

Interesting use cases

- XML latency analysis
  - Get the latency data of all system calls
  - Generate a synthetic event for each system call
  - Usage of an abstraction filter of system calls
  - Work also for the IRQs

Interesting use cases

- XML latency analysis

```
<fsm id="syscall">
	<precondition input="tid_18985" />
	<stateTable>
		<stateDefinition name="wait_syscall_entry_x">
			<transition input="syscall_entry_x" next="syscall_entry_x" action="sys_x_founded" saveSpecialFields="true"/>
			<transition input="#other" next="wait_syscall_entry_x" />
		</stateDefinition>
		<stateDefinition name="syscall_entry_x" >
			<transition input="syscall_exit_x:thread_thread" next="syscall_exit_x" action="exit_syscall_founded" saveSpecialFields="true" clearSpecialFields="true"/>
			<transition input="#other" next="syscall_entry_x" />
		</stateDefinition>
		<stateDefinition name="syscall_exit_x">
			<transition input="#other" next="syscall_exit_x"/>
		</stateDefinition>
	</stateTable>
	<initialState id="wait_syscall_entry_x"/>
	<endState id="syscall_exit_x" />
</fsm>
SCHED_SWITCH -->
<fsm id="sched_switch" multiple="false">
	<precondition input="sched_switch"/>
	<stateTable>
		<stateDefinition name="sched_switch">
			<transition input="sched_switch" next="sched_switch" action="update Current_thread" />
			<transition input="#other" next="sched_switch" />
		</stateDefinition>
		<stateDefinition name="fake_end">
			<transition input="#other" next="fake_end"/>
		</stateDefinition>
	</stateTable>
	<initialState id="sched_switch"/>
	<endState id="fake_end" />
</fsm>
```

Performance

- Criterias that affect the performance :
  - Complexity of the filter
  - Number of current scenarios
- Optimization :
  - Indexing
  - Refactoring
  - precondition

Performance (evaluation)

- Read a trace
  - Comparison between XML and Java

- Comparison of the time of reading a trace

Performance(evaluation)

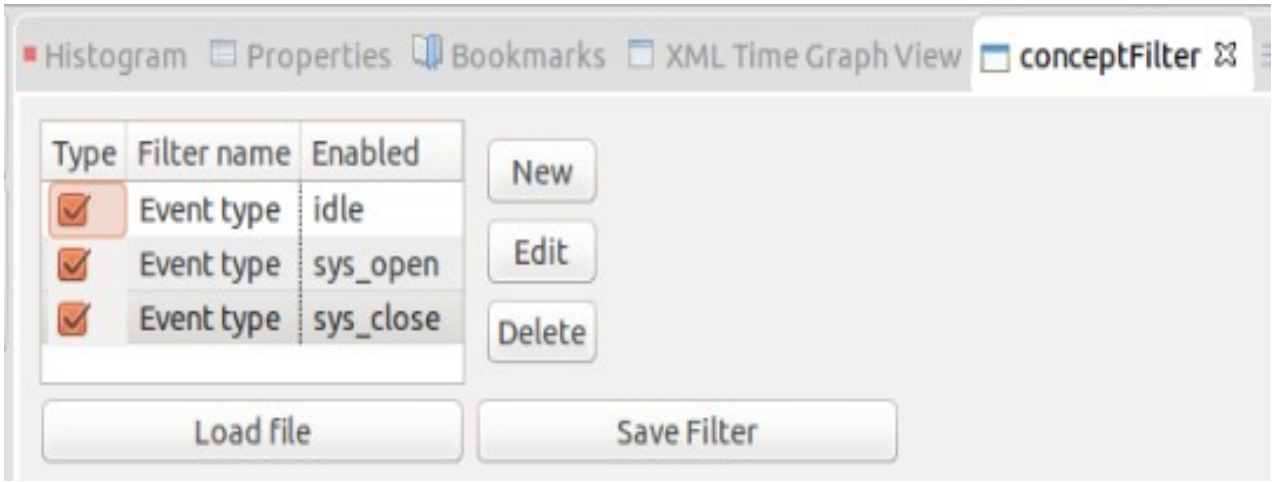
- Java hard-coded filter vs XML approach
  - find all file access
  - 500MB trace file
  - 19M of events
  - 328510 relevants events
  - 1494 file access found
  - 5 coexisting FSMs

Analysis	Java	XML	% of slowdown
Time (s)	31.1155	32.1194	3.23

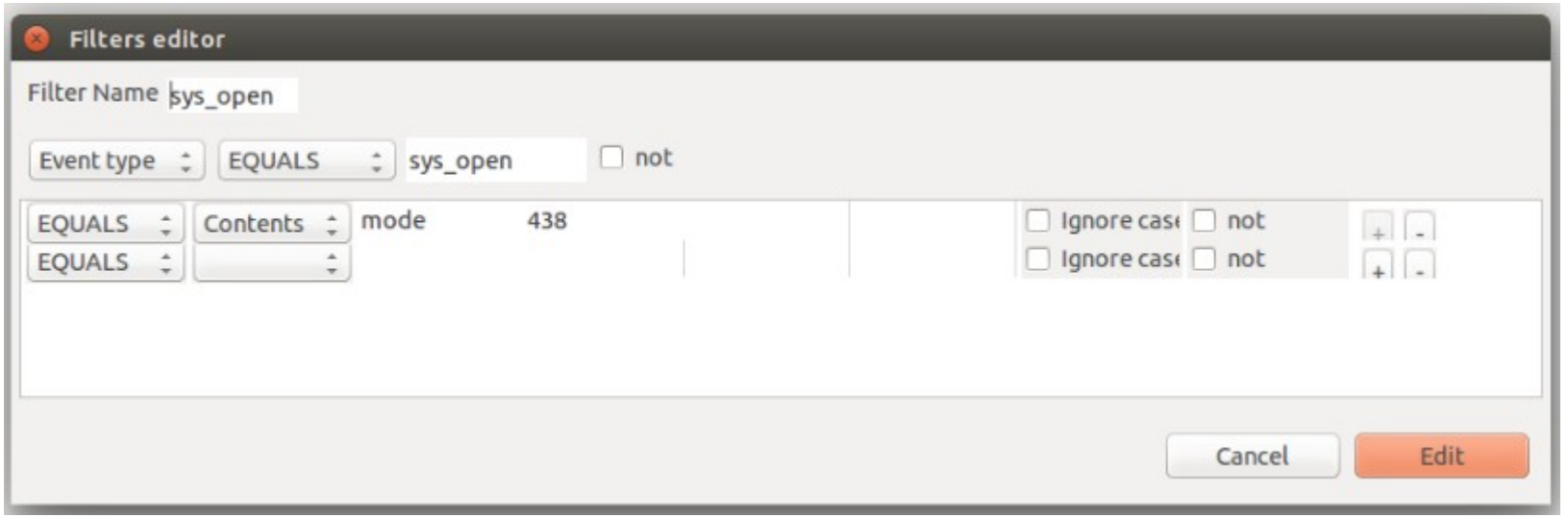
Filter events view

- A new filter tool to filter all the views
  - Based on the existing Filter View
  - A user friendly view
  - Possibility to manage the filters
    - Activate / Deactivate
  - Filter applied on multiple views
  - Updates the views instantly

Filter events view



Filter events view



Further work

- Performance :
  - Optimize multiple scenario analysis
  - Comparison with prior work
- More use cases
  - Different fields
- The need of a “filter manager tool”
  - (Jonathan's project)
- Filter events view
  - Make more views react to the filters

DEMO

- D E M O