

Data Visualization System

Introduction

The purpose of the Data Visualization System is to a tool that provides the user with an integrated view of the data generated by [the Evaluator Centric and Extensible Logger daemon \(ECELd\)](#) and the ability to tag and modify the data associated with a capture taken by ECELd.

Problem statement:

Data is a critical asset in cybersecurity; it helps researchers and practitioners develop novel technologies based on real past incidents. However, this asset is lacking, and even more, tools that specifically focus on the analysis of data are far and few between. The Evaluator Centric and Extensible Logger daemon (ECELd) is a tool that attempts to reduce this gap. The tool allows users to collect data for the purpose of analysis. Collected data include network data, keystrokes, system calls, and screenshots, among others. Additionally, the ECELd includes a wireshark component which allows users to annotate the network data using the popular packet sniffer application. Both the ECELd tool and the ECELd-wireshark component are integrated into the ECELd-netsys.

Team Members:

- Briana Sanchez
- Stephanie Medina
- Luisana Clarke
- Rocio Cardona
- Bianca Alvarez
- Dima Abdel Jaber

Client:

- Dr. Jaime Acosta

Guidance Team:

- Dr. Salamah Salamah

Description

The system consists of two primary components: the packet view shows ECELd-Wireshark and the timeline view shows the parsed ECELd-collected data using a horizontal layout. These two windows combined will allow for the user to have a better visualization of what is being analyzed. Many features are integrated in this system in order to bring better visualization and each feature will be specific to the type of data that is going to be displayed. In this system, each specific set of data will be considered a dataline. For example, a mouseclicks dataline will contain

different information than what a keypresses dataline contains, which will result in specific features for each. Moreover, with different datasets, there must be a way to be able to distinguish between each of them, so each dataline will have an assignable color in order to make it easier to tell them apart. This color assignment shall be displayed in both the timeline view and the packet view. Finally, the most important feature of this system will be synchronization. The synchronization feature shall be enabled and disabled depending on the user's preference. The main purpose for this feature is to allow the user to use both timeline view and packet view at the same time and be able to see the data that is correlated to where the user is currently analyzing.

Installing the Data Visualization System (DVS)

The Data Visualization System's source code can be found in the following link: <https://github.com/smedina7/DVS>. The system works in both Linux and Windows. Follow the instructions below for installation.

Windows Installation

The first step of installation is to clone the repository provided in the link above and navigate to the project folder.

Virtual Environment

Install eceld-wireshark using the installer provided. Type the following in the command prompt.

```
> nsis\Wireshark-win64-3.2.5.exe
```

You will get a prompt to install Wireshark. Install using default settings provided.

- Install Python's Virtual Environment Builder:
- Create and activate virtual environment:
- Install required dependencies:

```
> pip install virtualenv  
> virtualenv dvs-venv  
> dvs-venv\Scripts\activate
```

```
(dvs-venv)> pip install -r requirements.txt
```

Lastly, run main.py to start the DVS GUI.

```
(dvs-venv)> python main.py
```

Linux Installation

Requirements

In order for DVS to run properly, Eceld-Wireshark must be installed.

The following are ways to install it:

- Install Eceld-Netsys: Refer to the Git-Hub page

<https://github.com/ARL-UTEP-OC/eceld-netsys.git>

- Install Eceld-Wireshark: Refer to the Git-Hub page

<https://github.com/ARL-UTEP-OC/eceld-wireshark>

- Install with the DVS Installer

DVS installation steps

The first step of installation is to clone the repository provided in the link above and navigate to the project folder.

```
kali@kali:~/DVS$ sudo ./installDeb.sh
```

You will be prompted if you would like to install Eceld-Wireshark:

```
kali@kali:~/DVS$ sudo ./installDeb.sh Running apt-get updateHit:1
https://packages.microsoft.com/repos/vscode stable InReleaseHit:2
http://kali.download/kali kali-rolling InReleaseReading package
lists... DoneDVS depends on : eceld-wireshark would you like to
install it [Y/n]
```

Input "Y" to install Eceld-Wireshark, if already installed input "n" to skip.

Activate Environment:

```
kali@kali:~/DVS$ source venv/bin/activate
```

Run DVS:

```
(venv) kali@kali:~/DVS$ sudo python3 main.py --no-sandbox
```

Using the Data Visualization System (DVS)

The Data Visualization System has two main components: Packet View and Timeline View. This section will provide a tutorial on basic usage of each component. Furthermore, users will be able to use the DVS as a standalone program or integrated as part of ECEld.

Standalone Version

In the standalone version, users will be able to choose from what folder they'll be analyzing their data. In the event that the user already has the data that they need, this gives the flexibility to just do the data visualization and analyzation without having to open ECEld-NetSys.

Iteration 1 Components:

For the first iteration of the DVS system, the team focused on getting the Timeline View set up. This means that the packet view will not appear yet for this iteration. The reason for that is because the packet view is already implemented, and we will just need to combine the two windows once the timeline view is ready.

Iteration 2 Components:

For the second iteration of the DVS system, the team focused on getting the Packet View set up. The team updated front end and backend for this iteration.

Iteration 3 Components:

For the third iteration of the DVS system, the team focused on Synchronization features, color assignment, Suricata alerts, and more frontend features. The team also tested the system using large data and updated the installer.

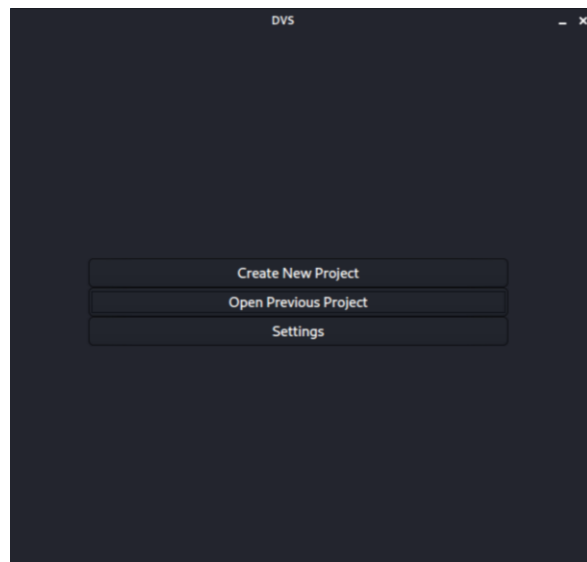
Iteration 4 Components:

For the fourth iteration of the DVS system, the team focused on Synchronization features, color assignment, Suricata alerts, and more frontend features. The team also tested the system using large data and updated the installer. This iteration was expanding on features implemented in iteration 3.

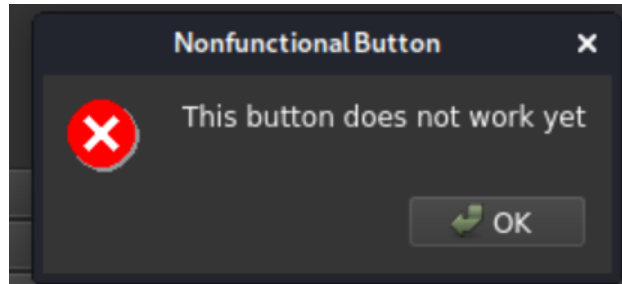
The following sections will review what are the current available features.

Start Window

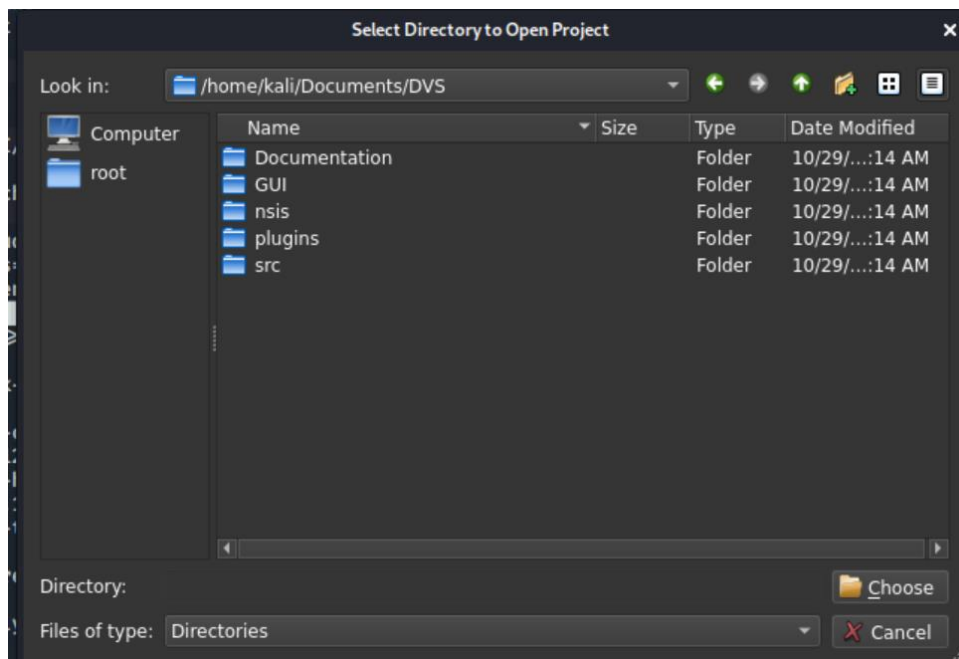
Once the DVS is started, the following window will appear:



Only 2 out of the 3 buttons will take you to the timeline view. The settings button will only display a message indicating that it hasn't been implemented yet. The following is the message box displayed:



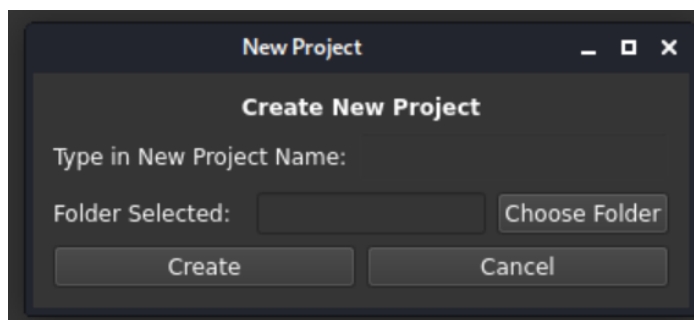
The "Open Previous Project" button shows a file browser to allow you to select a previous project. The following will appear when you select "Open Previous Project":



Once you select a project, the Timeline and the Packet view will appear.

The third option "Create New Project" will prompt you to name the project and select the folder where the data is stored. The create option will create a copy of the folder selected, which will be

stored in DVS/Project Data.

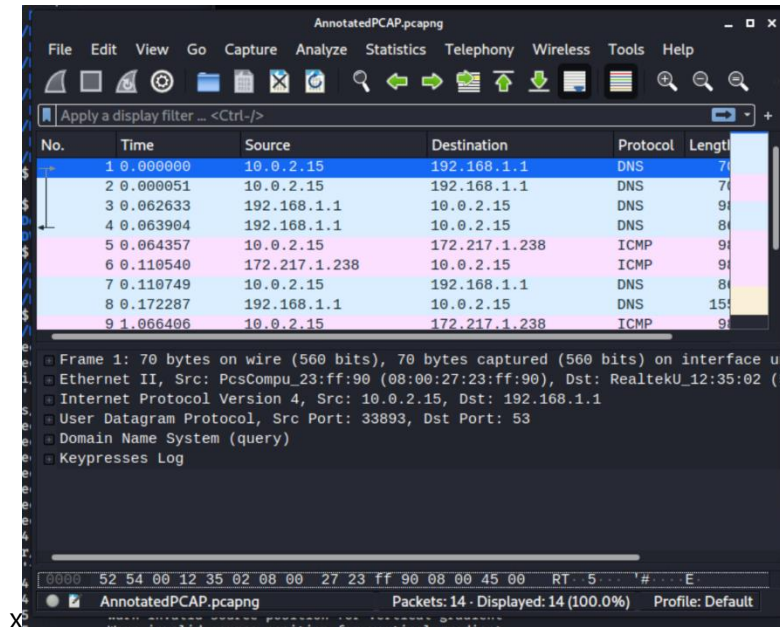


Once the user selects a project, two views will be shown, the Timeline View and the Packet View.

Packet View

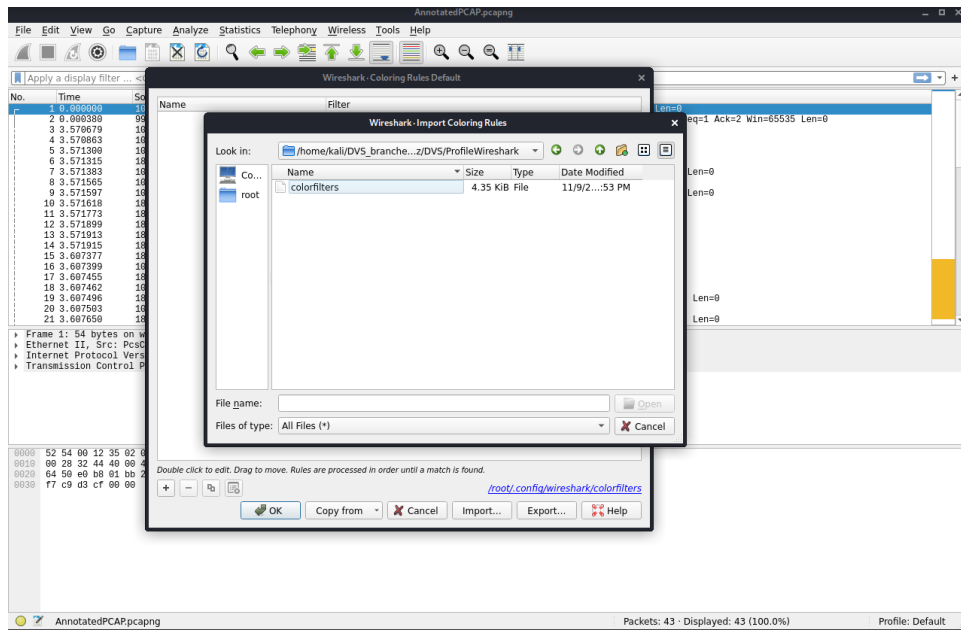
The packet view consists of the ECEld version of Wireshark that is available at

<https://github.com/ARL-UTEP-OC/eceld-wireshark>.

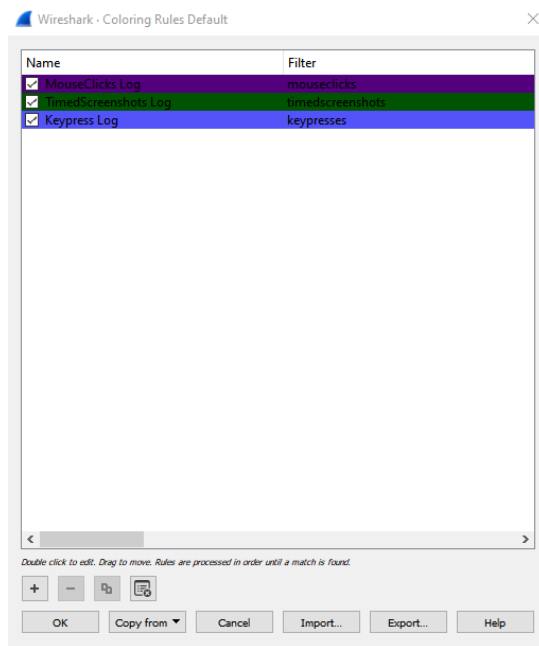


Wireshark Coloring Rules

Once each dataline is selected, the system creates a file which is the coloring rules according to the color selected for each dataline. The user can import this file into Wireshark:

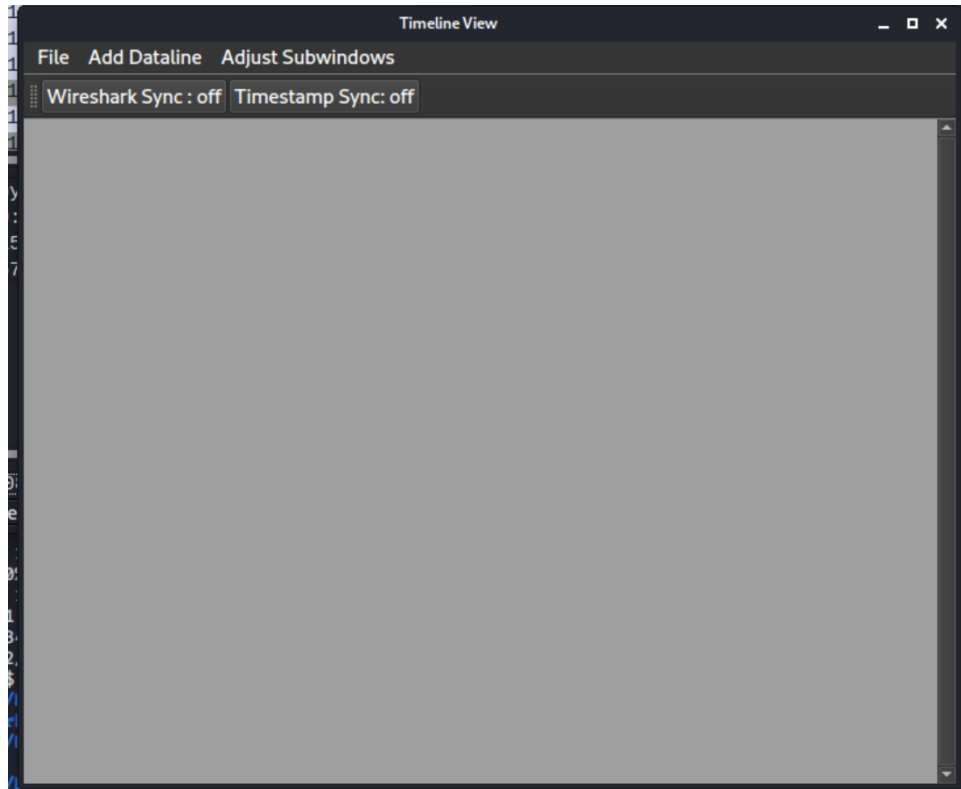


After selecting the colorfilter file, the rules will be added:



Timeline View

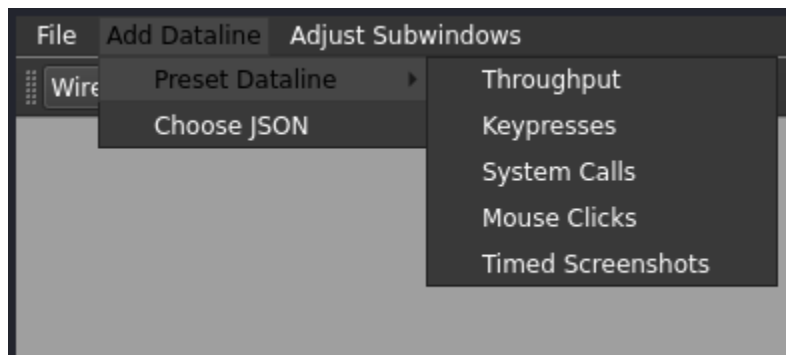
The Timeline View looks like the following:



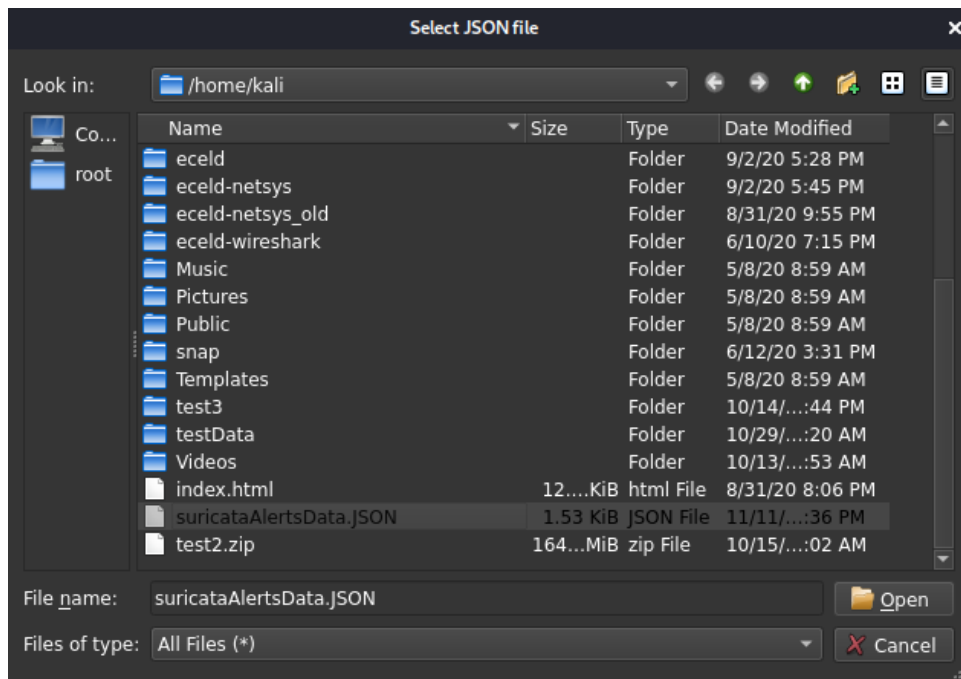
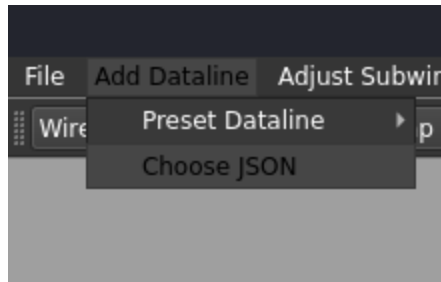
Adding Datalines

The “Add Dataline” menu option has two options: Preset Dataline and Choose JSON .

When selecting the Preset Dataline option, a list of the predefined datalines will become visible. These datalines will use the data from the project created in the start window.



When selecting the Choose JSON option, the user can select a JSON file to open it on the timeline view. According to the JSON data, the correct render will be used to display the data.

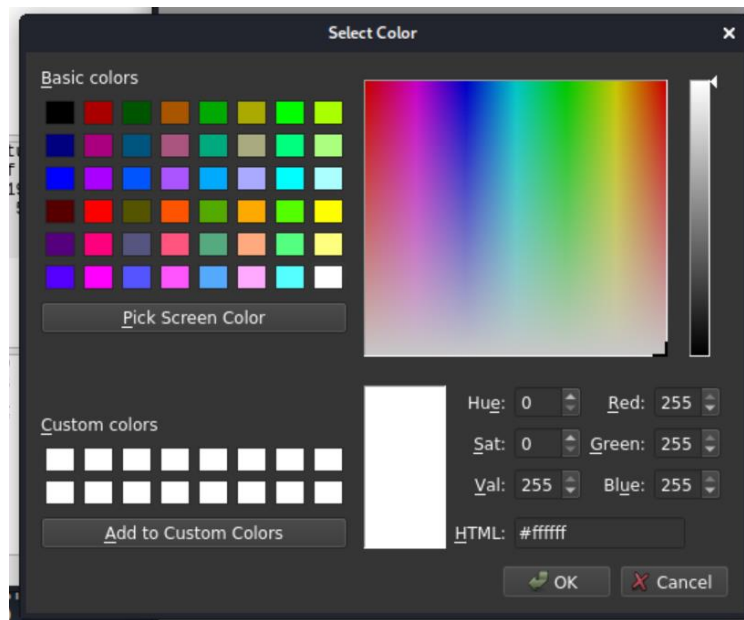


As an example, the user can select a new JSON file like the Suricata Alerts Data that will be rendered as a text table display.

Qt Suricata					
	suricata_id	suricata_rule_id	content	className	start
1	0	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:51
2	1	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:51
3	2	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:51
4	3	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:51
5	4	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:53
6	5	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:54
7	6	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:54
8	7	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:55
9	8	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:55
10	9	[1:2:0]	Test dns_query option	suricataAlert	2020-10-07T02:02:55

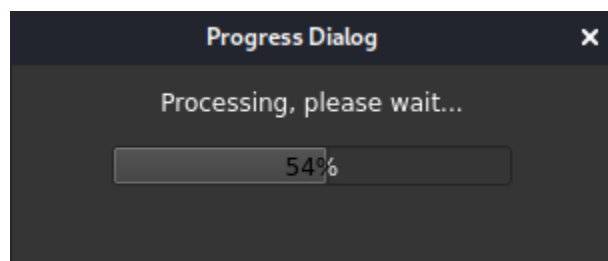
Color Selection

Once each dataline is selected, the user will be asked to select a color to assign to the dataline. The color selector looks like the following:

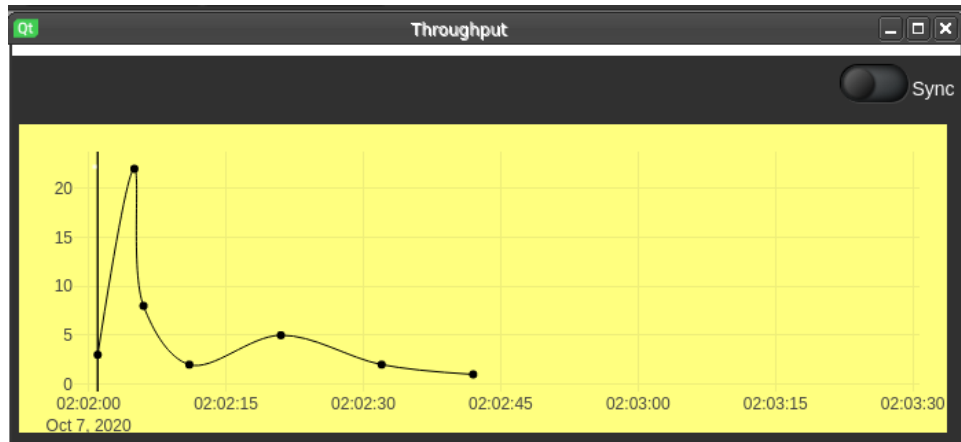


Throughput

The first dataline option is throughput. When the user selects this option, a progress dialog will be visible indicating that the dataline is loading.



The following dataline will be shown in the timeline view.



This dataline is a dash graph render displaying the tshark network throughput data in time intervals.

Keypresses

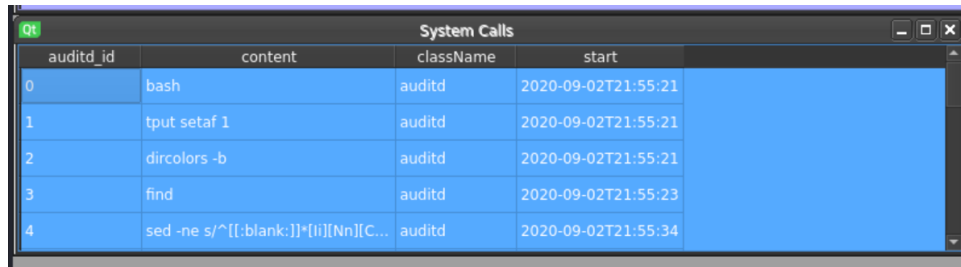
The second dataline option is keypresses. When the user selects this option, the following dataline will be shown in the timeline view:

Keypresses			
keypresses_id	content	className	start
0	find[Return]	Keypresses	2020-09-02T21:55:23
1	ht[BackSpace][BackSpace]ping...	Keypresses	2020-09-02T21:55:36
2	[Control_L]	Keypresses	2020-09-02T21:55:38

This dataline is a text render in a table display containing the parsed data from the keypresses JSON file. This table includes the id of the keypress (keypresses_id), the content, className, and the timestamp.

System Calls

The third dataline option is the system calls. When the user selects this option, the following dataline will be shown in the timeline view:

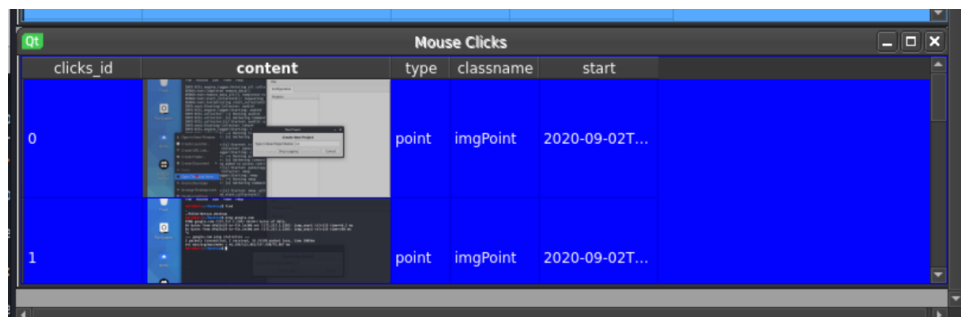


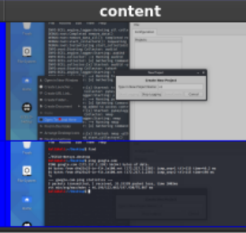
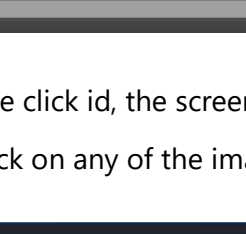
auditd_id	content	className	start
0	bash	auditd	2020-09-02T21:55:21
1	tput setaf 1	auditd	2020-09-02T21:55:21
2	dircolors -b	auditd	2020-09-02T21:55:21
3	find	auditd	2020-09-02T21:55:23
4	sed -ne s/^([[:blank:]]*)[li][Nn][C...	auditd	2020-09-02T21:55:34

This dataline is a text render in a table display containing the parsed data from the auditdData JSON file. This table includes the auditd id, the system call, className and timestamp in which the system call was used.

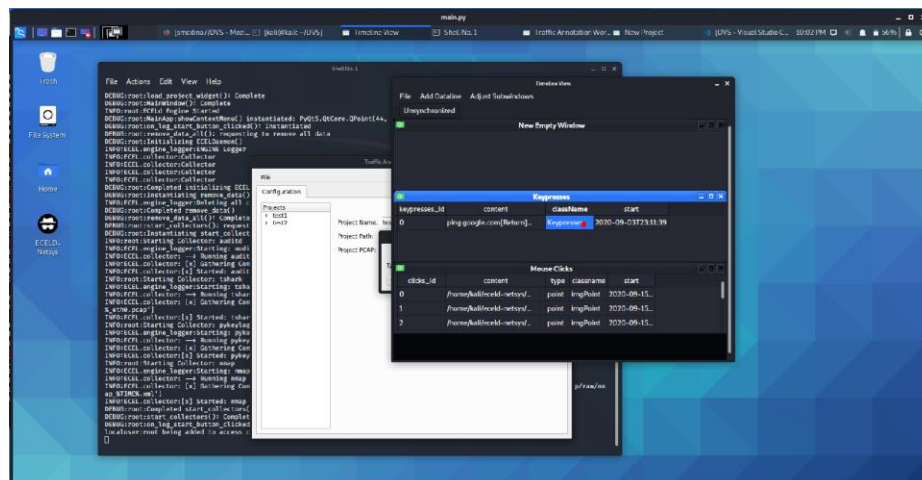
Mouse Clicks

The fourth dataline option is the mouseclicks. When the user selects this option, the following dataline will be shown in the main window:



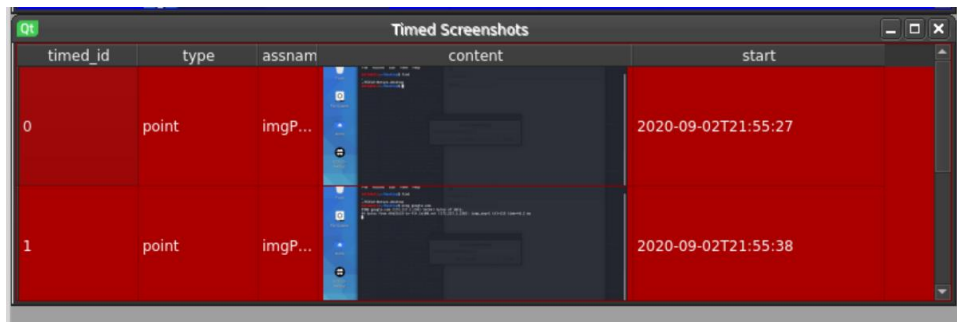
clicks_id	content	type	classname	start
0		point	imgPoint	2020-09-02T...
1		point	imgPoint	2020-09-02T...


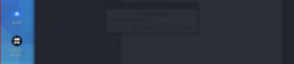
This table includes the click id, the screenshot, the type, className and the timestamp of the click. The user can click on any of the images to view it in a new window.



Timed Screenshots

The last option is the Timed screenshots. When the user selects this option, the following dataline will be shown in the main window:



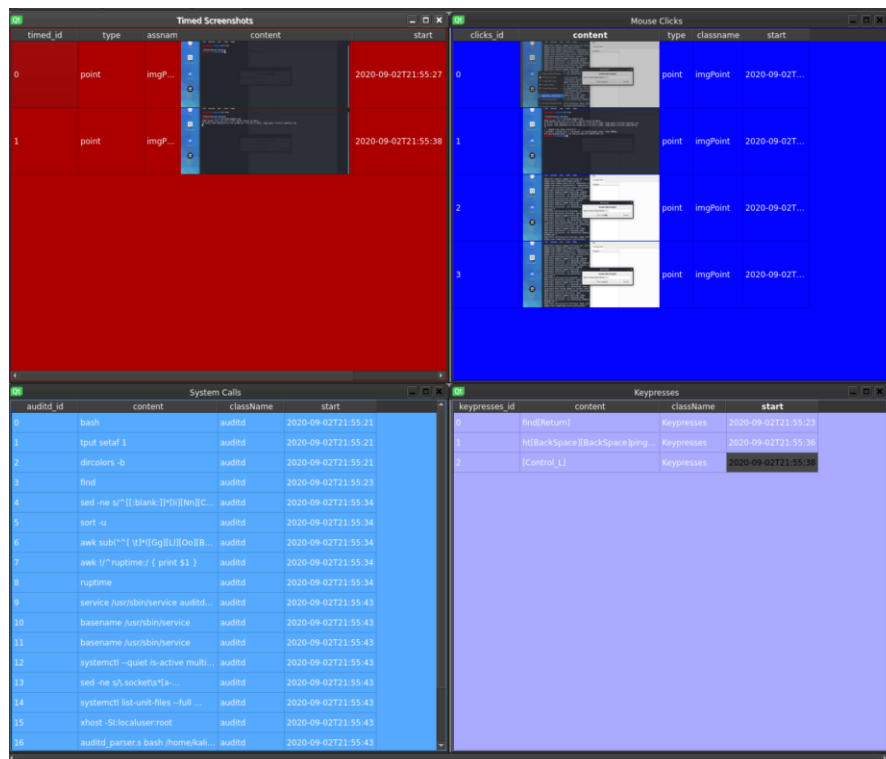
timed_id	type	assnam	content	start
0	point	imgP...		2020-09-02T21:55:27
1	point	imgP...		2020-09-02T21:55:38

This table includes all the screenshots that were timed, the `timed_id`, the `type`, `className` and the timestamp of the screenshot. The user can click on any of the images to view it in a new window.

The user is able to open, close and minimize each dataline window individually as well as displaying a combination of the datalines available at the same time.

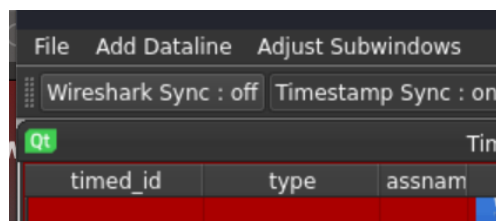
Adjust Subwindows

This option allows the user to readjust the windows in a tile formatting. The following is a sample of what happens when there are 4 windows open and this option is selected:



Buttons to enable/disable synchronization

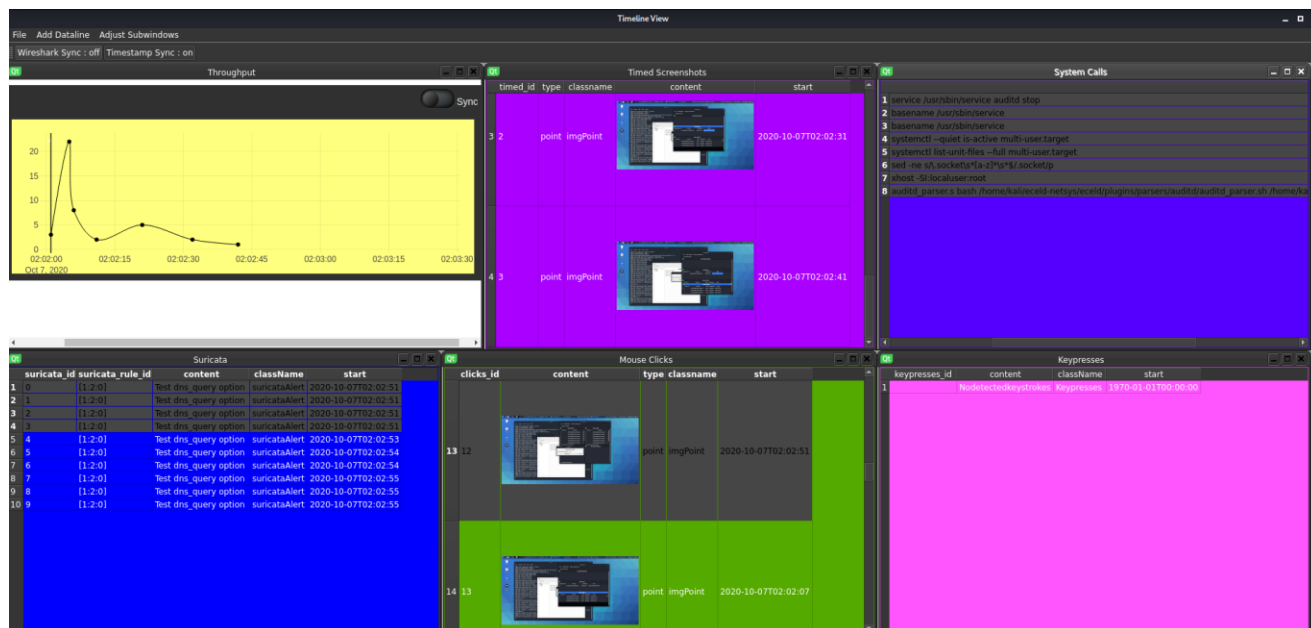
This feature will allow the user to synchronize views. The following are the sync buttons:



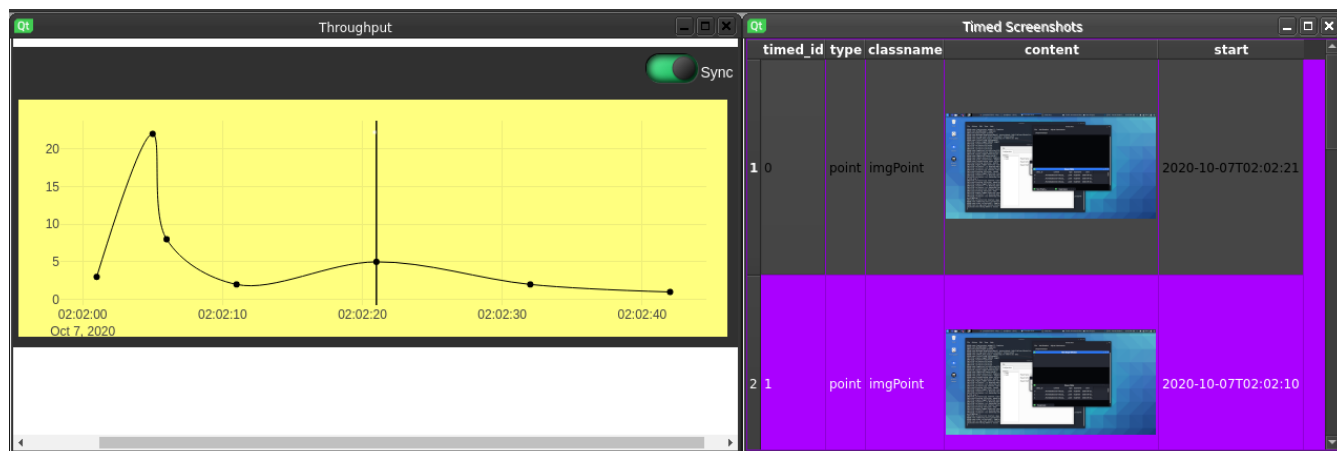
Timestamp Sync

When the user clicks on “Timestamp Sync: off” it will be updated to “Timestamp Sync: on.” All the datelines in the timeline view will be synchronized according to the timestamp, highlighting the rows in the tables were the system finds a matching timestamp.

The user can select any row of any dateline to sync the datelines according to the timestamp selected. The following shows the sync behavior:



The Throughput has a sync toggle button, that when activated will send a signal to the system, this will sync the throughput when the user selects a timestamp from the other datalines.



Appendix

Keywords

DVS - Data Visualization System

ECEld – Evaluator-Centric and Extensible Logger

