Visualizing the Run Time Execution of Command Patterns

Zainab Saeed Wattoo
Department of Computer Science
University of British Columbia
zswattoo@cs.ubc.ca

1 Introduction

Hein Lab (Chemistry Lab) at University of British Columbia (UBC) is carrying out different chemical experiments using robot arms and other modules for assistance to automate these experiments where manual labor is involved. The Hein Lab automated solubility experiment setup is shown in Figure 1 where a lab computer running python scripts that sends commands to the Robot Arm and other modules such as MT Quantos (Solid Dosing), Tecan Cavro (Liquid Dosing) and IKA Magnetic Stirrer (Stirring/ Heating) connected to it to carry out different experiments.

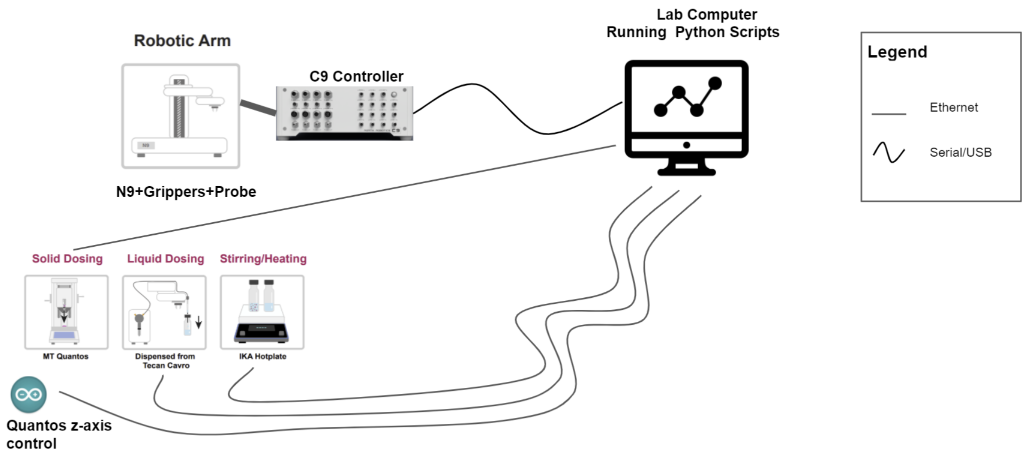


Figure 1: Hein lab setup for Automated Solubility experiment. A lab computer running python scripts is sending commands to the robot arm and its assisting modules

However, this setup could be vulnerable to various attacks from an intruder or a hacker who can intercept the communication that happens between the lab computer running python scripts and sending commands to the Robot Arm and other modules. This communication is unprotected as the hacker who has gained control over the lab computer can manipulate the python scripts and send malicious commands that change the behavior of the robot arm or the modules leaving them in an unwanted state or perform dangerous experiments. In order to protect the Hein Lab setup, we proposed a middlebox that sits between the lab computer and the robot arm along with the modules.

This middlebox shown in Figure 2 is a computer connected to the lab computer via Ethernet; further, this middlebox is connected to the robot arm along with the modules. It will have an Anomaly based Intrusion Detection System (IDS) that analyzes each command that is being sent to the the robot arm and the modules and will accept or reject it on the basis of its impact on the overall setup. It will ensure that the robot arm along with its assisting modules performs safe experiments correctly and accurately.

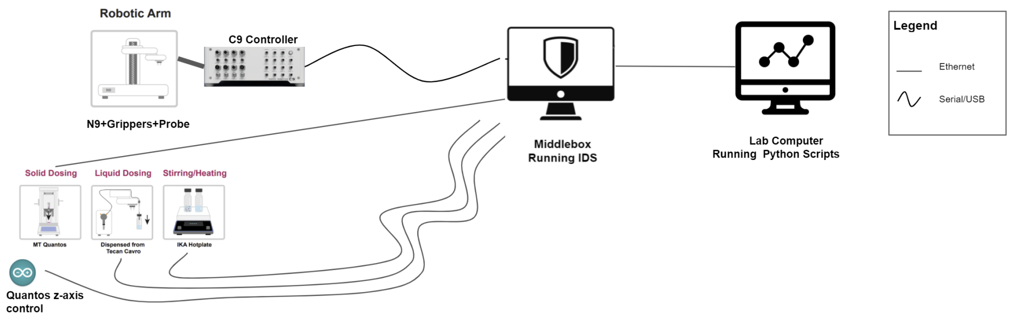


Figure 2: Middlebox running Anomaly Based Intrusion Detection System is connected to the robot arm and its assisting modules. Further, it is connected to the lab computer via Ethernet

As the initial step in building the Anomaly Based IDS, we are collecting traces of the commands sent via the lab computer to carry out the experiments on the middlebox. These traces are preprocessed to divide the commands into sub-signatures based on data mining techniques in order to observe the commands that are often sent together in a sequence. In this research, we want to visualize the run time execution of the command patterns for a particular experiment sent via the lab computer where these sub-signatures are going to be encoded via separate colors. The main goal of visualizing these command patterns along with its sub-signatures is to produce different signatures of different experiments along with validating the sub-signatures produced by the data mining techniques.

Overall, this course research project ties with the project that I along with my team at Systopia Lab in UBC are working on since April 2021 to secure the Hein Lab.

2 Related Work

We review the work where visualizations have helped to build an Intrusion Detection System and Intrusion Detection System based Tools. Further, we explore the work that has been done to visualize different patterns in different areas of research.

2.1 Intrusion Detection Systems and Tools

Several works [3; 4] have been done to create Intrusion Detection System with Visualizing Capabilities. Luo et al. [4] proposed a four-angle-star based visualized feature generation approach (FASVFG) to evaluate the distance between samples in a 5-class classification problem that is used for building an IDS. The four-angle-star approach is used to classify an unknown point to a class where the four vertexes represent four anomaly classes and from which new features are generated. It is a feature reduction approach that is used before the data is fed into the classification model. Moreover, Karami [3] worked on a novel anomaly-based IDS where visualization capabilities uses modified Self-Organizing Map (SOM) in the presence of benign outliers. Self-Organizing Map - an unsupervised learning algorithm is used to convert high dimensional data to low-dimensional data that is similar to a 2D discrete map with nodes and connections. It detects the attack and anomalies correctly and also provides useful insights to the end users.

LogicMonitor [1] is an Intrusion Detection Based tool that uses advanced machine learning algorithms to visualize the expected data patterns for data points, so one could see the data that falls outside of these patterns. The LogicMonitor provides dashboards and widgets to display the data the way users wants in order to catch the issues before hand in order to avoid further more severe events.

Their approaches [3; 4; 1] visualized and analyzed the network traffic data and helped with feature reduction of the dataset or building an IDS using visualization capabilities. However, our data consists of the command traces being used for carrying out the chemical experiments. The main goal is to visualize these command patterns at run time execution which is the stage before building an IDS.

2.2 Visualizing Patterns

Singh [5] analyze and visualize system calls that capture useful information of system call logs to detect malicious activity from benign activity. They find patterns by visualizing their data via scatter plot and line chart. Further, Cadez et al.[2] propose a new methodology for observing the patterns of navigating through Web Sites. They divide the users with similar navigation paths into one cluster. They create a tool called WebCANVAS (Web Clustering Analysis and VisuAlization of Sequences) for visualizing clusters of users with same behavioral pattern.

Both works [5; 2] are used to visualize the patterns in their own domain such as observing system calls or user behavior on Web Site. However, our work is concerned with visualizing the commands patterns where the commands are sent to the robot arm during a chemical experiment and further dividing them into sub sequences to observe them at a lower granularity.

3 Data and Task Abstraction

In the Hein Lab, the traces are collected by our middlebox for every experiment that runs on the lab computer. Once the traces are collected, they are processed and stored in the form of a json file, csv file and further stored in MongoDB. For our research project, we will be provided with the csv files which is in table format to visualize the data. The data description is shown in Table 1. The current dataset contains normal and anomalous traces and the size of this dataset is 3.82 MB with 25 csv files. In the current dataset, there are 6 fields and 30126 commands in total that have been executed. In the Hein lab setup, there are five number of modules including the robot arms.

Data Type	Table
Data Size	3.82 MB
Total Number of CSV files	25
Total Number of Fields	6
Total Number of Commands	30126

Table 1: Dataset Description

The traces contains the timestamp, commands and arguments sent to the robot arms and the assisting modules and contains the responses and exceptions that are received while the experiment is running. Table 2 shows the data abstraction of the tracing dataset containing the description, data type and the cardinality. There are 6 fields: Timestamp, Module, Command Name, Arguments, Responses, Exceptions from which we will use the first four fields (Timestamp, Module, Command Name, Arguments) for our visualization. The ordered Timestamp contains the timestamp when the command was executed, the categorical Module contains the name of the module, the categorical Command Name contains the name of the command that was executed, the categorical Arguments, categorical Responses and categorical Exceptions field contains the arguments, responses, exceptions of the command that was executed.

We will derive the sub-signatures that is going to be part of the visualization from the csv data provided to us using data-mining techniques. The sub-signatures are selected on the basis of the most frequent commands that appear together in all of the dataset. Currently, this work is still in progress

Field Name	Description	Data Type	Cardinality
Timestamp	Time of the command	Ordered	12th October 2021 - 22nd October 2021
Module	Name of the Module	Categorical	5
Command Name	Name of the command	Categorical	41
Arguments	Arguments executed with the command	Categorical	5745
Responses	Responses received once the command executed	Categorical	0
Exceptions	Exceptions got once the command executed	Categorical	0

Table 2: Dataset Abstraction

and we will receive the sub-signatures soon to further work on visualizing them. Therefore, we can not report the exact cardinality of sub-signatures at this point in time.

The high-level task abstraction is to use visualization to consume and produce information for the dataset provided. We produce further information from the data provided which is the sub-signatures of the command traces. For our task, we want to visualize the pattern of the commands that were sent over a period of time for a particular experiment, along with its arguments for this project. At this stage, we are not looking at the responses and exceptions thrown by the commands. Moreover, along with the signature of the commands, we want to visualize the sub-signatures of a particular experiment derived from the data mining techniques. This will help us achieve our main goal of giving us an insight about how the commands change over time and give an idea about the most frequent command sub-signatures that appear together.

4 Solution

4.1 Proposed Solution

For our solution, we plan to create a step graph to represent the commands along with the time. For the first part, the commands will be along the y-axis and the timestamp will be along the x-axis. The line will be used as a mark to visualize the problem. Further, the sub-signatures will be encoded via different colors to distinguish between them. For every experiment, we will have a different step graph. The figure 3 shows the step graph that we plan to achieve to visualize the signatures along with the sub-signatures for every experiment. Further, we plan to have a second part to the graph that will be used to visualize the arguments that the user wants for a specific command. On this graph, the arguments along with its values will be shown as a popup. The figure 5 shows the popup showing the arguments for a specific command at a given timestamp.

4.2 Proposed Implementation

We are planning to use Python with Plotly and Flask for our implementation. Python is a programming language that has several packages for visualizing different graphs. Plotly and Flask are python packages that are used to visualization and creating web framework respectively. Flask is going to be used to create a drop down menu to select an experiment and the Plotly will be used to create the step graph which will take information given in the drop down menu. Further, Plotly will show the sub-signatures encoding using different colors and also a pop up when the user selects a command at a particular time.

As the initial step, currently we have created a drop down menu that has experiments listed shown in 4. We are creating a Web application to display our visualization. Currently, we are facing problems in rendering the graphs that we are working on.

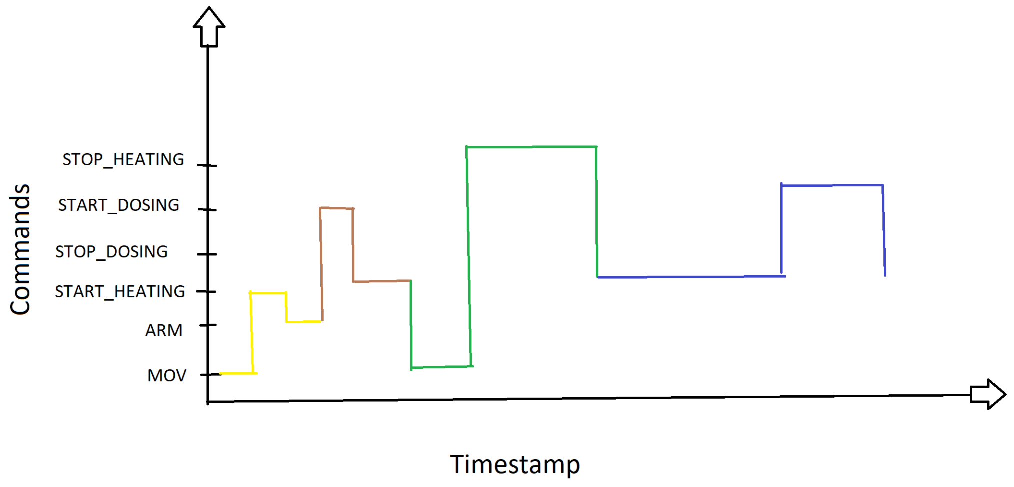


Figure 3: The step graph shows the solution that we plan to achieve to visualize the command patterns at run time execution along with the sub-signatures.

127.0.0.1:5000
Visualizing the Run Time Execution of Command Patterns
Automated Solubility
Joystick
Crystal Solubility Profiling
Automated Solubility with UR3
Automated Solubility

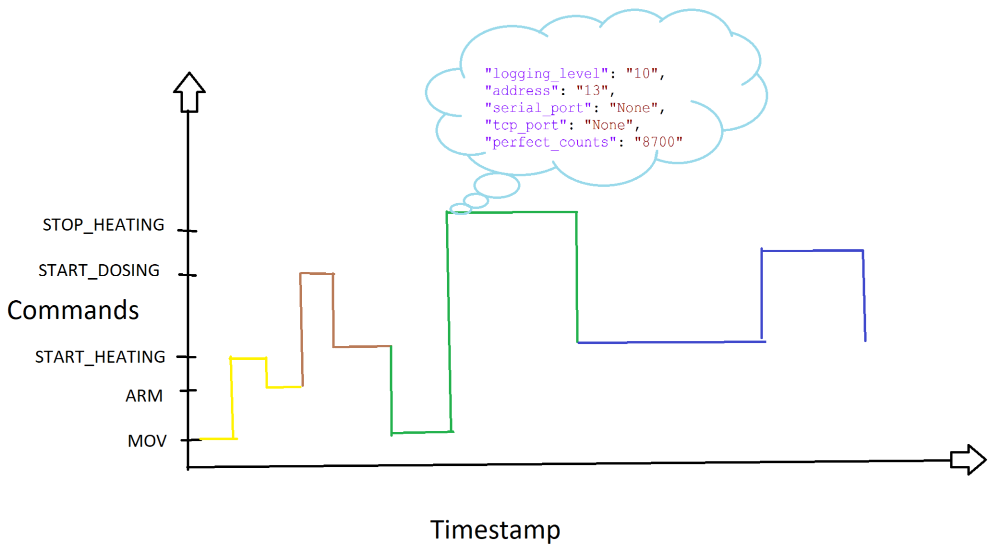
Figure 4: This figure shows the work that has been done so far.

4.3 Usage Scenario

First, the user clicks on the filter to select the experiment they want to visualize. Once the filter is clicked the graph is displayed that gives an overall view of the commands that are executed over time. Moreover, it also shows the sub-signature encoded with different colors. Second, the user can click the command and the time for which they want to visualize the arguments for. Once the line is clicked, a pop up appears showing the arguments at that time of that particular command.

5 Future Work

For future work, we hope to expand our solution further to accommodate the arguments and exceptions as well. Further, if time permits we plan on using the anomalous data points to be represented on our design solution to get an idea of the anomaly data points.



```
"logging_level": "10",
"address": "13",
"serial_port": "None",
"tcp_port": "None",
"perfect_counts": "8700"
```

Figure 5: The popup on top of the graph shows the arguments for a specific command at a given timestamp for a specific experiment.

6 Milestones

The following is a phase-based schedule for carrying out our research problem. The dates are a rough estimate, that we hope to achieve as we explore the problem and may change accordingly. However, we will try to be on track with our deadlines for each of the proposed dates.

Phase 1: Related Work section and Defining the Task Abstractions More Concretely *Number of Hours : 10 (November 16th 2021)*

- read related papers to the research problem
- finalize the dataset to be used
- finalize the task abstractions concretely after understanding the dataset further.

Phase 2: Implementation *Number of Hours : 30 (30th November 2021)*

- defining the solution with visual encoding and idioms
- implementing the solution

Phase 3: Evaluations and Completing the Writeup *Number of Hours : 5 + 8 (15th December 2021)*

- evaluating the solution
- ensuring the command patterns are accurately visualized
- comparing the solution with the sub-signatures
- completing the writeup

References

[1] Anomaly Detection Visualization. `https://www.logicmonitor.com/support/forecasting/anomaly-detection/anomaly-detection-visualization`.

[2] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of Navigation Patterns on a Web site Using Model-Based Clustering. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–284, 2000.

[3] A. Karami. An Anomaly-Based Intrusion Detection System in Presence of Benign Outliers with Visualization Capabilities. *Expert Systems with Applications*, 108:36–60, 2018.

[4] B. Luo and J. Xia. A Novel Intrusion Detection System Based on Feature Generation with Visualization Strategy. *Expert Systems with Applications*, 41(9):4139–4147, 2014.

[5] A. Singh. System Call Analysis and Visualization. 2019.