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.

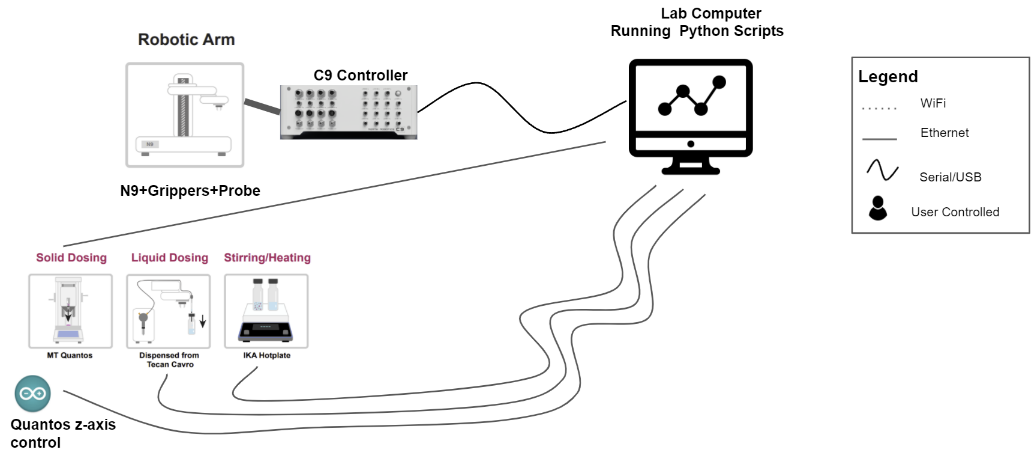


Figure 1: Hein lab setup for Automated Solubility experiment. A lab computer running python scripts is sending commands to the robot arm and its assisting modules

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.

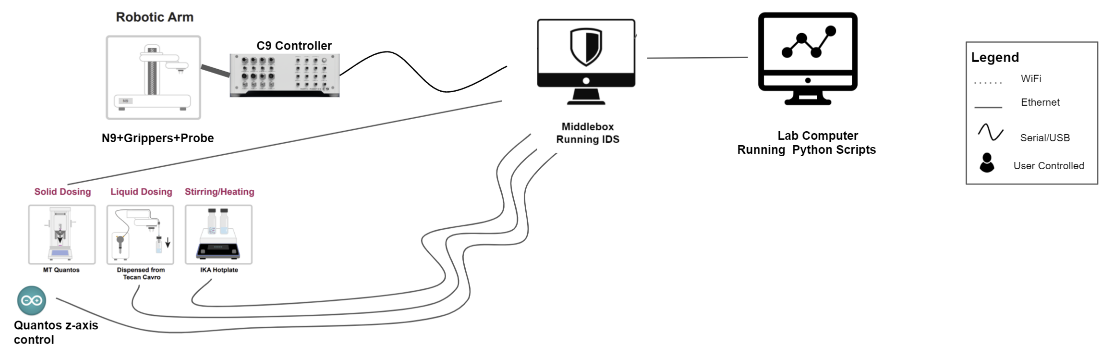


Figure 2: Middlebox running Anomaly Based Intrusion Detection System is connected to the robot arm and its assisting modules. Further, it is connected to the lab computer via Ethernet

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. Further, we explored the commercial tools that have helped in building Anomaly Based IDS.

2.1 Intrusion Detection System

Several work [6; 7] has been done to create Intrusion Detection System with Visualizing Capabilities. In this regard, Luo et al. 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 [7]. It was used as a feature reduction approach before feeding it into a classification model. Their system achieved reasonable performance both in re-substitution test and validation test along with presenting meaningful visualization inference with high-level significance [7].

Moreover, Karami [6] worked on a novel anomaly-based intrusion dection system by visualization capabilities using modified Self-Organizing Map (SOM) in the presence of benign outliers. It not only detects the attack and anomalies correctly but also provides useful insights and visualization to the end users [6]. In addition, their visualization capabilities considered the limitations in human cognitive ability when dealing with IDS that has large volumes of information that is not possible to fit all the requirements into one screen [6].

Their approaches [6; 7] 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 and the main goal is to visualize these command patterns at run time execution.

2.2 Intrusion Detection Based Tools

LogicMonitor [3] 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. It helps the users to catch the issues before hand before escalating more severe events [3].

Amazon SageMaker [2], Athena [1], and Pandas [5] together are used by Machine Learning (ML) and data scientist for anomaly detection from their data by visualizing and analyzing them. Athena [1] is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL [4]. Further, Pandas [5] is an open-source, high-performance, easy-to-use library that provides for data structures and data analysis library like matplotlib for Python programming language [4]. Lastly, Amazon SageMaker [2] is a fully managed service that provides you with the ability to build, train, and deploy ML models quickly [4].

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 dumped into 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 only normal data points and the size of this dataset is 223 MB with 10 csv files. We are still in the process of gathering more data as the experiments in the Chemistry lab are being carried out. Further, we are also planning on gathering anomalous data points by carrying out the experiments or procedures that deviate from the expected behavior but still do not have drastic or dangerous impact. In the current dataset, there are 8 fields and 16818 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	223 MB
Total Number of CSV files	10
Total Number of Fields	8
Total Number of Commands	16818

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 8 fields: Experiment Name, Timestamp, Module, Command Name, Arguments, Responses, Exceptions and Anomaly (Yes/No). The categorical Experiment Name contains the name of the experiment such as Automated Solubility or Crystal Solubility Profiling, the ordered Timestamp contains the timestamp when the command was executed, 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. Lastly, the categorical anomaly (Yes/ No) contains whether the command was anomalous or not.

The high-level task abstraction is to use visualization to consume and produce information for the dataset provided. 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

Field Name	Description	Data Type	Cardinality
Experiment Name	Name of the experiment	Categorical	1
Timestamp	Time of the command	Ordered	12th October 2021 - 15th October 2021
Module	Name of the Module	Categorical	5
Command Name	Name of the command	Categorical	29
Arguments	Arguments executed with the command	Categorical	105
Responses	Responses received once the Command executed	Categorical	5
Exceptions	Exceptions got once the command executed	Categorical	0
Anomaly (Yes/No)	The command is anomalous or not	Categorical	2

Table 2: Dataset Abstraction

insight about how the commands change over time and the validity of the sub-signatures produced by the data mining techniques.

4 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 glyph. The figure 4 shows the glyph that pops up showing the arguments for a specific command at a given timestamp.

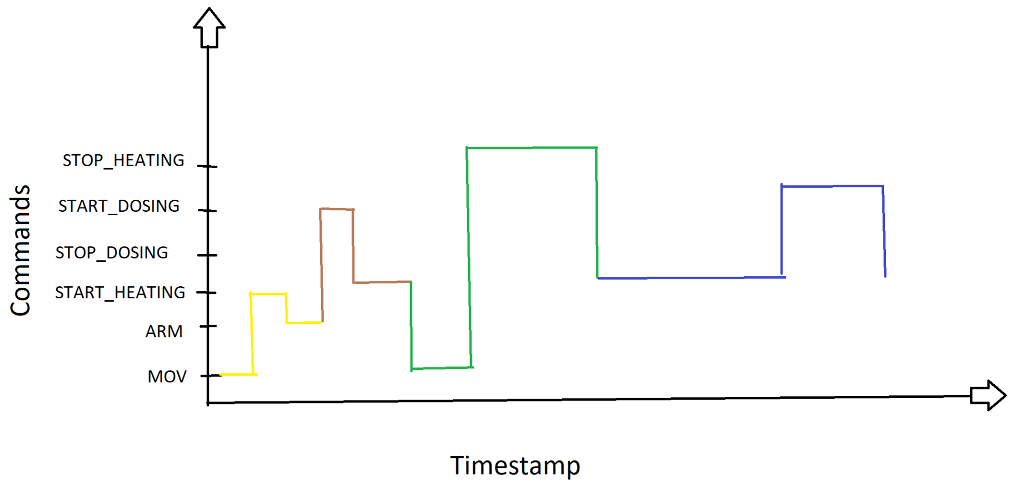
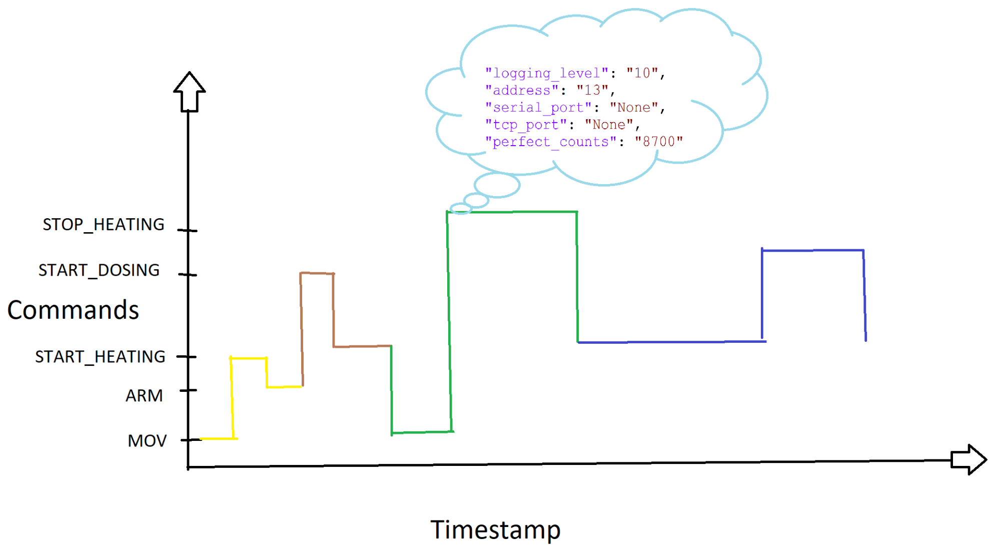


Figure 3: The step graph shows the solution that we plan to achieve to visualize the command patterns at run time execution along with the sub-signatures.

We are planning to use Tableau or Power BI tool for visualizing our solution. We are planning to have a filter from where the user can select the experiment they want to visualize. This will give them an overall view of the commands being executed over the time. Further, the user can select the command and the time for which they want to visualize the arguments for. This will result in a glyph that will give the user an idea of the arguments that were sent for the specific command at a given timestamp for a specific experiment.



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

Figure 4: The glyph on top of the graph shows the arguments for a specific command at a given timestamp for a specific experiment.

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.

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 *(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 *(30th November 2021)*

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

Phase 3: Evaluations and Completing the Writeup *(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] Amazon athena. `https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc`.

[2] Amazon sagemaker. `https://aws.amazon.com/sagemaker/`.

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

[4] Data visualization and anomaly detection using amazon athena and pandas from amazon sagemaker. `https://aws.amazon.com/blogs/machine-learning/data-visualization-and-anomaly-detection-using-amazon-athena-and-pandas-from-amazon-sagemake`

[5] Pandas. `https://pandas.pydata.org/`.

[6] A. Karami. An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities. *Expert Systems with Applications*, 108:36–60, 2018.

[7] 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.