

CSE-3025 Large scale data processing

Analysis of Traffic and Log Data Captured During a Cyber Defense Exercise

SUBMITTED BY :-

|  |  |
| --- | --- |
| **Name** | **Reg No.** |
| **Devshekhar Pattnaik** | **19BCE1292** |
| **Shubh Pachauri** | **19BCE1754** |

Submitted to:-

Dr. Ramesh Ragala

SCHOOL OF COMPUTER SCIENCE and ENGINEERING

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “Analysis of Traffic and Log Data Captured During a Cyber Defense Exercise” is a bonafide work of **Shubh Pachauri (19BCE1754) , Devshekhar Pattnaik (19BCE1292)** who carried out the J-component under my supervision and guidance.

**Dr. Ramesh Ragala**

Associate Professor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (SCSE)

VIT UNIVERSITY, CHENNAI CAMPUS

CHENNAI-6001273

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Ramesh Ragala,** Associate Professor, SCSE, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work. We are extremely grateful to the Dean of the SENSE, VIT Chennai, for extending the facilities of the School towards our project and for the unstinting support. We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**ABSTRACT**

Cybersecurity research relies on relevant datasets providing researchers a snapshot of network traffic generated by current users and modern applications and services. We have taken one such and tried to analyze the data. The data provides network traffic flows and event logs (Linux and Windows) from a two-day cyber defense exercise involving attackers, defenders, and fictitious users operating in a virtual exercise network. All the users in that exercise were divided into two teams. We have tried to analyze the dataset and classify the IP addresses of the computers based on which team they belong to.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter no** | **Topics** | **Page no** |
| **1** | **Introduction** | 6 |
| **2** | **Dataset** | 7 |
| **3** | **Methodology** | 8 |
| **4** | **Code** | 12 |
| **5** | **Results** | 14 |
| **6** | **Conclusion** | 16 |

**1.INTRODUCTION**

Cybersecurity research relies on relevant datasets providing researchers a snapshot of network traffic generated by current users and modern applications and services. We have taken one such and tried to analyze the data. The data provides network traffic flows and event logs (Linux and Windows) from a two-day cyber defense exercise involving attackers, defenders, and fictitious users operating in a virtual exercise network. All the users in that exercise were divided into two teams. We have tried to analyze the dataset and classify the IP addresses of the computers based on which team they belong to.

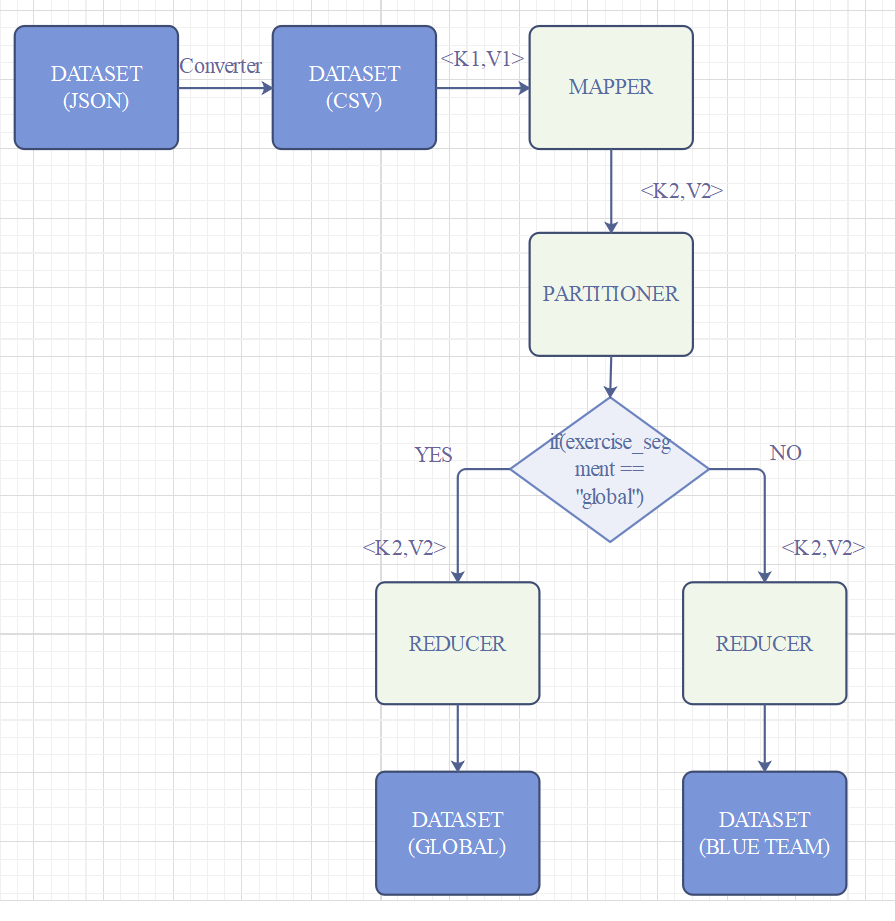
**2.DATA SETS**

The dataset includes traffic flows and event logs from Linux and Windows machines captured and collected during a run of a Red Team/Blue Team cyber defense exercise held on March 19–20, 2019. The exercise network was deployed in the KYPO Cyber Range Platform ] and it was designed to be a full digital twin of a fictitious organization with all exercise hosts and network devices running common operating systems and applications, which can be found in modern organizations. The result of this are network and log data that are equal to data seen in the real world, without the need for anonymization, ready to be processed and analyzed. The captured and collected data were normalized into three distinct event types and they are stored as structured JSON.

We have used “cz.muni.csirt.SyslogEntry.tgz” – an archive of Linux Syslog entries with the payload of corresponding text-based log messages. There are 6083,409 events of this type. The collected logs include, for instance, information from these daemons and applications: cron, smbd, sshd, usermod, useradd, and firewalld.

<https://zenodo.org/record/3746129#.YL4pDfkzZPZ>

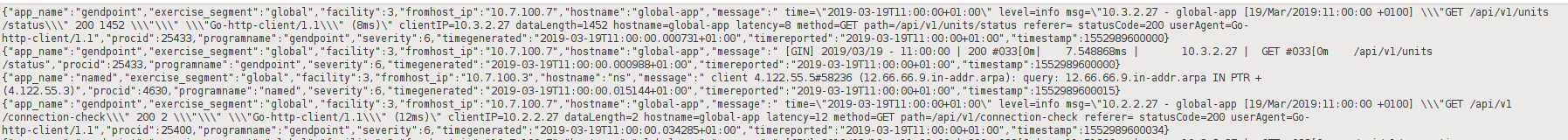
**3.METHODOLOGY**



**DATA CONVERSION**

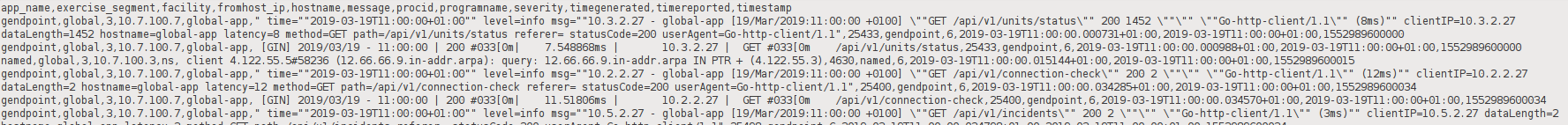
First, we have converted the data in JSON format to data in comma separated CSV format. Then we have implemented a map-reduce program to read the data, process it and partition the data based the team they belong to.

1. Dataset before conversion (JSON)



{"app\_name":"gendpoint","exercise\_segment":"global","facility":3,"fromhost\_ip":"10.7.100.7","hostname":"global-app","message":" [GIN] 2019/03/19 - 11:00:00 | 200 #033[0m| 4.768903ms | 10.6.2.27 | GET #033[0m /api/v1/connection-check","procid":25534,"programname":"gendpoint","severity":6,"timegenerated":"2019-03-19T11:00:00.682432+01:00","timereported":"2019-03-19T11:00:00+01:00","timestamp":1552989600682}

1. Dataset after conversion (CSV)



python,blue-team-6,3,10.7.106.27,ocs," 10.0.3.32 - - [19/Mar/2019 11:00:00] ""GET / HTTP/1.1"" 200 -",823,python,6,2019-03-19T11:00:00.081017+01:00,2019-03-19T11:00:00+01:00,1552989600081

**MAPPER**

Mapper in Hadoop takes each record generated by the RecordReader as input. Then processes each record and generates key-value pairs. This key-value pair is completely different from the input pair. Mapper in Hadoop only understands key-value pairs of data. So, data should be converted into key-value pair before passing to the mapper. The input key-value pair for mapper is <K1, V1>.

K1 = LongWritable key

V1 = Each line of the CSV file

The output key-value pair for mapper is <K2, V2>.

K2 = “timestamp” and “fromhost\_ip”. It is of type Text.

V2 = “exercise\_segment”. It is also of type Text

**PARTITIONER**

We have used custom partitioner to split the data. The partitioner in MapReduce controls the partitioning of the key of the intermediate mapper output. A total number of partitions depends on the number of reduce task. According to the key-value, each mapper output is partitioned and records having the same key value go into the same partition and then each partition is sent to a reducer. Partition class determines which partition a given key-value will go. The “exercise\_segment” has two types of values “global” and “blue-team”. If “exercise\_segment” is global then we put in one partition else in other partition. The input key-value pair for partitioner is <K2, V2>.

K2 = “timestamp” and “fromhost\_ip”. It is of type Text.

V2 = “exercise\_segment”. It is also of type Text

**REDUCER**

Partitioner in Hadoop MapReduce redirects the mapper output to the reducer by determining which reducer is responsible for the particular key. The Reducer process the output of the mapper. After processing the data, it produces a new set of output. At last, HDFS stores this output data. Reducers run in parallel since they are independent of one another. The number of reducers to be used is decided by the number of partitions we want to create. In our case, we have two reducers. The input key-value pair for partitioner is <K2, V2>.

K2 = “timestamp” and “fromhost\_ip”. It is of type Text.

V2 = “exercise\_segment”. It is also of type Text

**4. CODE**

**package** Dev\_hadoop;

**import** java.io.IOException;

**import** java.util.\*;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.conf.\*;

**import** org.apache.hadoop.io.\*;

**import** org.apache.hadoop.mapreduce.\*;

**import** org.apache.hadoop.mapreduce.Reducer.Context;

**import** org.apache.hadoop.mapreduce.lib.input.\*;

**import** org.apache.hadoop.mapreduce.lib.output.\*;

**public** **class** Project

{

**public** **static** **class** Map **extends** Mapper<LongWritable, Text, Text, Text>

{

**public** **void** map(LongWritable key, Text value, Context context) **throws** IOException, InterruptedException

{

String[] line = value.toString().split(",");

String exercise\_segment = line[1];

String timestamp\_ip = line[11] + "\t" + line[3];

context.write(**new** Text(timestamp\_ip),**new** Text(exercise\_segment));

}

}

**public** **static** **class** dpart **extends** Partitioner<Text,Text>

{

**public** **int** getPartition(Text key,Text value,**int** nr)

{

String a=value.toString();

**if**(a.equalsIgnoreCase("global"))

**return** 0;

**else**

**return** 1;

}

}

**public** **static** **class** Reduce **extends** Reducer<Text, Text, Text, Text>

{

**public** **void** reduce(Text key, Text value, Context context) **throws** IOException, InterruptedException

{

context.write(key, value);

}

}

**public** **static** **void** main(String[] args) **throws** Exception

{

Configuration conf = **new** Configuration();

Job job = **new** Job(conf, "Project");

job.setJarByClass(Project.**class**);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(Text.**class**);

job.setMapperClass(Map.**class**);

job.setPartitionerClass(dpart.**class**);

job.setNumReduceTasks(2);

job.setReducerClass(Reduce.**class**);

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

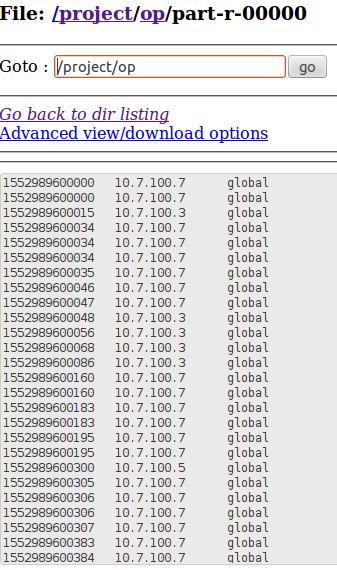
job.waitForCompletion(**true**);

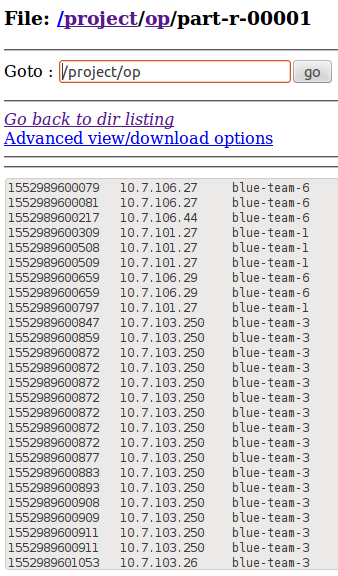
}

}

**5. RESULTS**

In the output the first column is the time stamp, the second column is the IP address and the third column is “exercise\_segment” or which team the belong to.





**6. Conclusion**

Thus, we were able to get inference about the “Traffic and Log Data Captured During a Cyber Defence Exercise” using a map reduce program implemented in Hadoop and HDFS written in java.

We have found and separated the IP addresses of the systems used for hacking along with the time stamp of the attack and which team they belong.