MATATABI: Multi-layer Threat Analysis Platform with Hadoop

Hajime Tazaki
The University of Tokyo, Japan

Kazuya Okada
Nara Institute of Science and Technology, Japan

Yuji Sekiya
The University of Tokyo, Japan

Youki Kadobayashi
Nara Institute of Science and Technology, Japan

***Abstract*—Threat detection and analysis are indispensable processes in today's cyberspace, but current state of the art threat detection is still limited to specific aspects of modern malicious activities due to the lack of information to analyze. By measuring and collecting various types of data, from traffic information to human behavior, at different vantage points for a long duration, the viewpoint seems to be helpful to deeply inspect threats, but faces scalability issues as the amount of collected data grows, since more computational resources are required for the analysis. In this paper, we report our experience from operating the Hadoop platform, called MATATABI, for threat detections, and present the micro-benchmarks with four different backends of data processing in typical use cases such as log data and packet trace analysis. The benchmarks demonstrate the advantages of distributed computation in terms of performance. Our extensive use cases of analysis modules showcase the potential benefit of deploying our threat analysis platform.**

***Index Terms*—Cybersecurity, Multi-layer threat analysis, Hadoop**

I. INTRODUCTION

In parallel to the growth of the Internets functionality as a distributed system, the number of critical threats is also increasing, rendering pro-active defensive approaches troublesome. Various categories of malicious activities have been seen in the wild, and multiple countermeasures have been proposed and deployed. Unfortunately, no defense mechanism has been able to completely hinder attacks and bring an end to this perpetual arms race.

A significant obstacle against deploying a countermeasure for such threats, is the lack of knowledge of what is happening in the system: a single point of observation at the firewall has no knowledge about the other egress nodes of an enterprise network, or scanning applications deployed at various endpoints lack global information which would provide information whether their probing is assisting a Distributed Denial of Service (DDoS) attack. The limited number of observation points and type of information collected needs improvement.

The recently established NECOMA project[1] aims at improving the situation of current cyber-security, by introducing new insight regarding countermeasure. It assumes the missing pieces for creating robust countermeasure are the lack of 1) information or types of datasets for threat analysis and 2) locations that observers should look at. In the other words, *the more data is available about an attack, the more data can be analyzed and thus, the higher the probability that the most effective countermeasures are taken*.

However, such a huge collected dataset easily faces a scalability problem in terms of not only the storage of the collected data, but also computation resource of the analysis itself because it also requires a fair amount of computational resource to investigate the datasets across the multiple sensors at different layers and various locations.

This paper reports our experience on the development of a big-data platform, called MATATABI, in order to fulfill the requirements for cyber-threat analysis (detailed in § II). We combined possible techniques and ideas available to satisfy the requirements, by incorporating and tweaking existing open source software. Our development follows several basic studies ([13][12][14]) in the past, but aimed at creating more complete system that allows us to detect complex security threats involving multiple data sources and locations of attackers.

Our contributions of this paper include:

- We designed and implemented the data collection and analysis platform, MATATABI, to handle multi-terabytes measurement data for the security threat analysis.
- We studied performance of our system with data querying benchmarks and gave the best practice for the implementations of threat analysis modules.

II. REQUIREMENTS

Considering the amount of collected data for the threat analysis, the design of data collection and analysis platform must satisfy the following properties;

R-1) the input and output performance of data must be scalable in terms of size of analyzed data (**scalability**).

R-2) the computation resource must be scaled out with the number of nodes: hopefully real-time performance for the detection of threats on the fly (**real-time analysis**).

R-3) the platform must be adaptable with the multiple kinds of data types (e.g., formatted text, binary data, etc) and layers (e.g., network traffic, user behaviors, etc) without introducing new ways to analyze them (**uniform programmability**).

R-1 and R-2 are obvious requirements, but R-3 is also important since one cannot predict threats, but is required to detect them once they happen.

We will describe the system and how it fulfills the aforementioned requirements in the following section.

[1] http://www.necoma-project.eu/

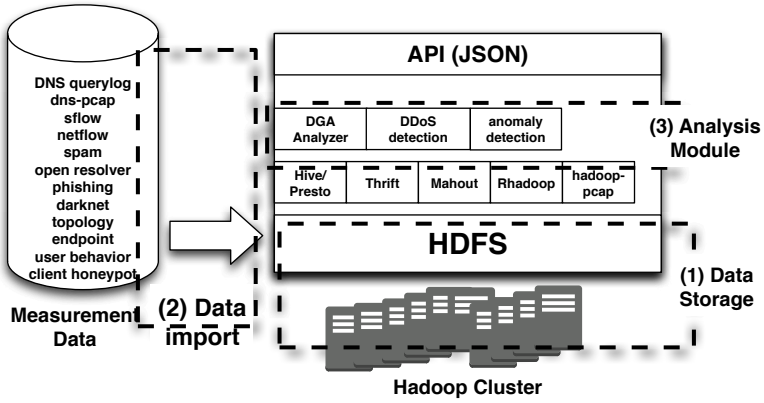

Fig. 1. Overview of MATATABI. Based on the Hadoop platform, we integrated the data storage with import modules, analysis scripts, and an application programing interface in a single platform.

III. DESIGN OF MATATABI PLATFORM

This section presents the design of our platform, so called MATATABI that serves data collection and analysis for the threat detection in order to fulfill the requirements described in § II. Figure I depicts the overview of our implemented platform, which consists of three key components; 1) data storage in a distributed environment, 2) data import modules, 3) analysis modules, and 4) Application Programming Interface (API) with the help of Apache Hadoop [1] software[2].

A. Data Storage and Base Software

The data storage component relies on the Hadoop Distributed File System (HDFS) to locate and access data in a distributed environment so that applications are agnostic to access the data where they are running on. We employed totally nine cluster nodes in total (Table I) distributed across several Japanese universities.

On top of served distributed file system, various data access utilities such as Hive [2] (SQL liked interface), Presto-db [5] (distributed query engine), and language bindings (Thrift [3], Rhadoop [6]) are employed in order to create analysis modules (§ III-C). These varieties of utilities are helpful not only for its friendliness to Hadoop environment, but also we can reuse existing analysis implementations used at stand-alone environments without reimplementing a lot. Furthermore, SQL like interface provided by Hive or Presto-db is useful to analyze multiple layers of data sources with simple query statements.

B. Data Import Module

The data import module basically works for copying various kinds of collected data into HDFS so that the analysis module implemented by the Map-Reduce framework can access directly. It will benefit *locality* during data reading process, as it is arranged by Hadoop automatically.

[2]At the moment, we used Apache Hadoop 2.2.0 version.

TABLE I
EQUIPMENTS OF HADOOP CLUSTER.

	CPU	RAM	Storage
master	2.5GHz (8 cores)	24GB	1.9TB
hadoop1	2.2GHz (16 cores)	38GB	52GB
hadoop2	0.8GHz (8 cores)	68GB	77GB
hadoop4	0.8GHz (8 cores)	68GB	77GB
hadoop6	0.8GHz (8 cores)	32GB	253GB
hadoop7	2.2GHz (16 cores)	50GB	1.9TB
slave02	2.0GHz (24 cores)	64GB	6.6TB
slave03	2.0GHz (24 cores)	64GB	6.6TB
slave04	2.0GHz (24 cores)	64GB	6.6TB

Table II describes the list of data import modules which we have used at the present moment. Some of them are converted to Hive-oriented table, others are stored as-is (binary data).

For the data access via Hive, Hive Serializer/Deserializer (SerDe) is used to read and write HDFS data with a custom format. It allows us to reduce the cost of implementing a data import module. We slightly modified RIPE pcap SerDe [4] for the data stored in `pcap` format.

Figure 2 is an example of a Hive database schema, which represents a custom format definition of `pcap` file containing DNS packets. With `PcapDeserializer` of the RIPE module, `pcap` files can be queried with an SQL-like language.

C. Analysis Module

The analysis module works on top of data store which provides high computation resource with flexible data access interface. Unlike ordinary applications for threat analysis running on a standalone machine, the module will benefit from the distributed computations by Map-Reduce or distributed query engine of Presto-db.

An example of the process of our implemented analysis module, and the corresponding queries to the Hive/Presto-db can be seen in figure 3: 1) to look for events in collected datasets which contain suspicious indications of threat, 2) find the behavior of the indications into another dataset to identify

TABLE II
DATA CONVERSION INTO HDFS.

	format	parser	data size (per day)	remark
DNS pcap	as-is	PcapDeserializer (hadoop-pcap [4])	5GB	date/node partitioned
Netflow	csv	CSV	1.2GB	`nfdump`, lzo compress, date/node partitioned
sFlow	csv	CSV	4.1GB	`sflowtool`, lzo compress, date/node partitioned
DNS querylog	ssv (bind9)	SSV	1.5G	date/node partitioned
SPAM email	SSV		4.5MB	date/MUA partitioned

```
CREATE EXTERNAL TABLE IF NOT EXISTS dns_pcaps (ts bigint,
                            protocol string,
                            src string,
                            src_port int,
                            dst string,
                            dst_port int,
                            len int,
                            ttl int,
                            dns_queryid int,
                            dns_flags string,
                            dns_opcode string,
                            dns_rcode string,
                            dns_question string,
                            dns_answer array<string>,
                            dns_authority array<string>,
                            dns_additional array<string>)
PARTITIONED BY (dt string, server string)
ROW FORMAT SERDE
   'net.ripe.hadoop.pcap.serde.PcapDeserializer'
STORED AS INPUTFORMAT
   'net.ripe.hadoop.pcap.io.PcapInputFormat'
OUTPUTFORMAT
           'org.apache.hadoop.hive.ql.io.\
                       HiveIgnoreKeyTextOutputFormat'
LOCATION 'hdfs:///dns-pcaps/';
```

Fig. 2. Example Hive table scheme for `pcap` data.

```
1) select * from dns_pcaps where regexp_like \
   (dns_question, '[a-z0-9]{32,48}');
2) select * from netflow where srcip='192.168.10.1';
3) select time,client_fqdn from suspicious_flow ;
```

Fig. 3. Steps for an analysis module to seek suspicious flows through multiple data sources.

correlations among multiple data sources (where 192.168.10.1 is the IP address that first query detects it as suspicious host), and 3) extract some attributes from the indications and store into a blacklist.

D. MATATAPI: API for MATATABI

The analytical results obtained from our platform are valuable not only for our own purpose, but also the others who try to detect threats from their analysis. Multi-dimensional analysis using different datasets at the different physical or logical space will help early threat detection: if indication of threats were detected in advance and propagated these information to others, one can countermeasure against such threats.

```
def Query (time_min, time_max):
   headers = {
   'context-type': 'text/plain',
   'x-presto-user': 'presto',
   'x-presto-catalog': 'hive',
   'x-presto-schema': 'default'
   }
   url='http://master:8080/v1/statement'

   data = "select dga_domain from zeus_dga_result
           where dt >= '%s' AND dt <= '%s'"
           % (time_min, time_max)
   request = urllib2.Request (url, data, headers)
   return json.loads (urllib2.urlopen(request).read())
```

Fig. 4. A sample python program of MATATAPI bridge program.

For that purpose, we designed MATATAPI, an application programing interface (API) for MATATABI, in order to provide an interface for accessing analytical results. Our design is a simple wrapper for the existing Presto REST API, which is available by Presto-db and generates JavaScript Object Notation (JSON) objects for a certain request through the API. All we need to provide an API is 1) to create a Hive (or Presto-db) table, and 2) to write a bridge program from client requests to Presto REST API.

Figure 4 represents an example of the bridge program written in `python`, where it bridges requests received via http transport and runs as a CGI program on a web server.

IV. MICRO-BENCHMARKS

In order to assess the performance of the data processing of use cases that are typical to threat analysis, we conducted micro-benchmarks in this section. The objectives of this benchmark are:

- to observe the scalability for the amount of data (data size), and
- to present a best practice for implementing the analysis module.

A. Regular Expression Match on Plain Text Files

The first benchmark shows the response time for data query when the amount of data scanned for the queries increases. Since the data collected for the analysis increases day by day and is easy to fill up the storage, it is important to understand how much data we can process in order not to degrade the performance of analysis.

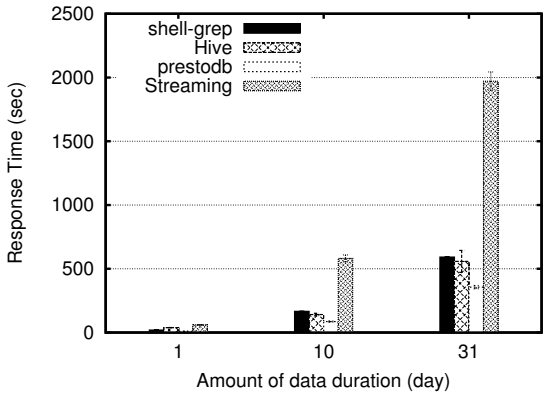

Fig. 5. Performance parsing on plain text data (DNS query log).

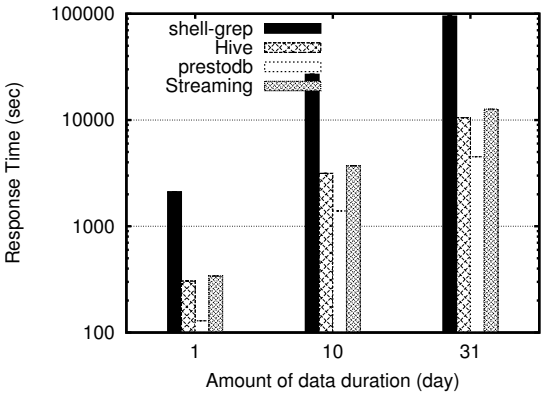

Fig. 7. Performance parsing on binary data (`pcap` file).

```
'[a-z0-9]{32,48}.(ru|com|biz|info|org|net)'
```

Fig. 6. Regular expression of a DGAed domain name.

To measure the performance, we setup scripts that conduct a simple query to pick events from the data sources. The scripts we used are 1) `grep` command-based shell script without Hadoop (`shell-grep`), 2) Hive query language, 3) Presto-db SQL, and 4) python script of Hadoop streaming. All the scripts parse and look for strings using a regular expression shown in figure 6, which represents the Domain Generation Algorithm (DGA) for ZeuS Bot [17]), and then print the results to the console screen.

As a target data for this benchmark, we used a) formatted text based log files which contain `bind9`[3] querylog, and b) `pcap` files which contain DNS traffic.

We ran the scripts on our Hadoop cluster described in Table I (except `1) shell-grep` script, which uses a single node, master, with local storage) and measured the execution time of each script.

Figure 5 represents the result of response time of each script as a function of data size (i.e., we changed the number of date to be parsed) with the 95% confidence interval computed for 3 replications of script execution.

The performance of single node data queries present a slower response time, while distributed computation by Hive and Presto-db gives faster response (40% faster in the best case), when the size of data parsed increased. Note that although our hadoop streaming script dispatched query jobs into distributed node, we did not see much performance gain. This may be due to an implementation matter of the streaming script that we used, but it is possible to achieve such a performance with a simple mapper/reducer implementation for the hadoop streaming.

B. Regular Expression Match on pcap Files

Figure 7 represents another benchmark using different datasets, `pcap` files contain DNS traffic, in the same environment. In this benchmark we have completed only one iteration of the benchmark due to time constraints.

Unlike the performance on queries to plain text files shown in figure 5, the `shell-grep` performance is worse: the response time was 21 times slower in the worst case than the one of Presto-db. This is possible considering heavy tasks in each filtering process: extracting gzip compressed data of original `pcap` files, decoding packet data from `pcap` format by `tshark` command, and regular expression matching by `grep` command. On the other hand, other scripts using hadoop infrastructure employs RIPE `hadoop-pcap` library which effectively dispatches reading files and string matching operation into distributed nodes, resulted in a high performance gain compared to the `shell-grep` implementation. Note that the RIPE `hadoop-pcap` library also stores the raw `pcap` data into HDFS, and then decompresses and decodes the file when data query is issued so, the both response times of `shell-grep` and others involve same procedure.

Presto-db achieves almost the best result among the four different scripts, with a low implementation cost for the data parsing script. If an analysis is based on simply looking up events on a Hive table, Presto-db is the best possible choice for a data store.

C. Processing Multiple Datasets

The last benchmark is a performance measurement on querying multiple datasets at the same time to analyze common interests between different datasets. This is an interesting feature of multi-layer threat analysis: if a dataset contains interests of threat, the other dataset may give the behavior of malicious activities more deeply.

In this benchmark, we used a query across the two Hive tables, Netflow records (i.e. netflow) and suspicious ZeuS DGA domain name list (i.e. zeus_dga_result), as depicted in figure 8. The query tries to find traffic flows that communicates

[3]https://www.isc.org/downloads/bind/

with the Command and Control (C&C) server detected by a DNS traffic scan using JOIN operation of Hive and Presto-db. The netflow table for one-month traffic has about 757,144,720 records while the zeus_dga_result table has 2,171 records. Since the size of the netflow table is relatively big and a query takes a long time for the JOIN operations, we carefully looked at three different file formats available for the Hive table, which are `TextFile`, `SequenceFile`, and `RCFile` (Record Columnar File) [9], and observed the variance of response time.

```
SELECT netflow.* FROM netflow
   JOIN zeus_dga_result ON (zeus_dga_result.c2c_sv =
   netflow.sa AND zeus_dga_result.dt=netflow.dt);
```

Fig. 8. Steps for an analysis module to seek suspicious flows through multiple data sources.

Figure 9 represents the execution time of the query by Hive and Presto-db as a function of the amount of data processed (i.e., days), along with the standard deviation computed for five replications. We measured three different types of file structure (i.e., `TextFile`, `SequenceFile`, `RCFile`) stored in HDFS.

When we used `TextFile` for the data structure, we did not have a benefit in performance since the structure is not able to correctly split the job processing across multiple nodes, resulting in a small number nodes executing the query. On the other hand, `SequenceFile` and `RCFile` are well designed for splitting jobs under MapReduce environment or Presto-db distributed SQL engine, and the performance improves.

Note that while Presto-db outperforms Hive with regular expression matching as shown in previous sections, the result is almost opposite with JOIN operation: Presto-db only outperforms one-day data with `SequenceFile` and `RCFile` structure.

V. Use Cases

In this section, we present several use cases of MATATABI as a threat analysis platform with huge amount of data.

A. Implemented Analysis Modules

ZeuS DGA detector
The first case is the detection of compromised hosts by the ZeuS botnet in an enterprise network by scanning DNS queries with a particular pattern of domain names as used in § IV-A. This module detects compromised hosts of ZeuS bot in a managed network, where a host queries suspicious domain names, based on the Domain Generation Algorithm (DGA), is considered a potential compromised host. In the case of proxied query via a DNS forwarder, we looked at traffic information filtered by the IP address of DNS answer records to identify the client IP address.

NTP amplifier detector
This module searches for Network Time Protocol (NTP) servers sending traffic with a particular packet size corresponding to a well-known NTP-amplification attack [18]. It reports the IP addresses of NTP amplifiers, and the IP address and Autonomous System (AS) number of targeted victims.

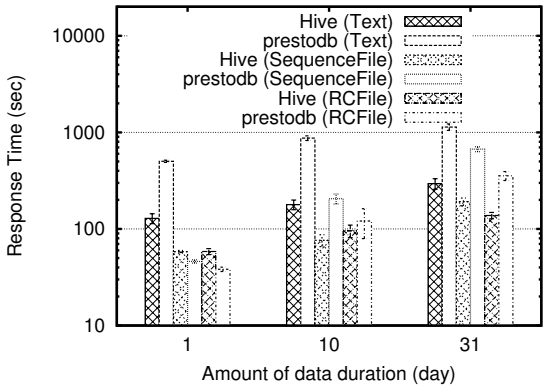

Fig. 9. Performance of SQL JOIN queries on various internal formats of Hive.

An additional module for the detector extracts NTP flows at the backbone sampling traffic (i.e., sFlow records) and lists the top ten NTP flows within a given time period.

Anomalous heavy-hitter detector
By using simple statistical tests, this modules detects IP addresses sending or receiving an abnormally high number of packets or bytes, for example, caused by DoS attacks.

Phishing likelihood calculator
This module is an implementation of a previously proposed system [15], which provides a binary detection whether a given URL points to a phishing site or not. The module consists of dataset preparation by crawling contents on pre-known phishing sites provided by PhishTank[4], analyzed by machine learning method with the help of Mahout. The dataset is updated every day since phishing sites changes frequently.

DNS amplification detector
The module tries to detect anomalous DNS traffic, causing amplification attack which fills the link capacity and makes denial of services. It looks at two different datasets, backbone sFlow traffic records and a list of open DNS resolver servers [19], and ranks top 10 speakers of DNS flow which communicates with open resolvers in sFlow datasets.

UDP fragmentation
As realizing an additional way for cache poisoning attack on DNS server based on IP fragmented packets [10], we started to observe how much traffic employed fragmented packets in the backbone traffic. The script simply extracts a record from sFlow dataset and implements a counter-based detection approach.

DNS anomaly detection
This module tries to detect anomalies of DNS response packets by adapting a machine learning method. Various statistical features such as IP addresses, the country code of DNS server,

[4]http://www.phishtank.com/

TABLE III
ANALYSIS MODULES ON MATATABI.

Name	datasets	frequency	LoC (#lines)	remark
ZeuS DGA detector	DNS pcap, netflow	daily	25	hadoop-pcap
UDP fragmentation detector	sflow	daily	48	
Phishing likelihood calculator [15]	Phishing URLs, Phishing content	1-shot	–	Mahout (RandomForest)
NTP amplifier detector	netflow, sflow	daily	143	pyhive, Maxmind GeoIP
	sflow	daily	24	
DNS amplifier detector	sflow, open resolver [19]	daily	37	
Anomalous heavy-hitter detector	netflow, sflow	daily	106	pyhive
DNS anomaly detection	DNS pcap, whois, malicious/legitimate domain list	daily	57	hadoop-pcap, Mahout (RandomForest)
SSL scan detector	sflow	1-shot	36	
DNS failure graph analysis [11]	DNS pcap	daily	159	pyhive

Malware Domain List[5], legitimate domain list, and the AS number of the DNS server are used for the analysis.

SSL scan detector

This module extracts SSL/TLS scans sFlow traffic data, which frequently happened right after the discovery of Heartbleed bug in OpenSSL library. The module simply counts packets destined to a specific port number, and containing the TCP SYN flag.

DNS failure graph analysis

This module tries to find suspicious non-existing domain names and IP addresses that might belong to botnets. The analysis is an implementation of an existing method, DNS failure graphs [11], based on a clustering technique.

Visualization

Figure 10 shows an example of visualization, representing a ranking of frequent asked domain names that matched with the regular expression of the ZeuS DGA, based on the result which the module generates. This visualization is implemented by using d3js[6] with the data available via MATATAPI.

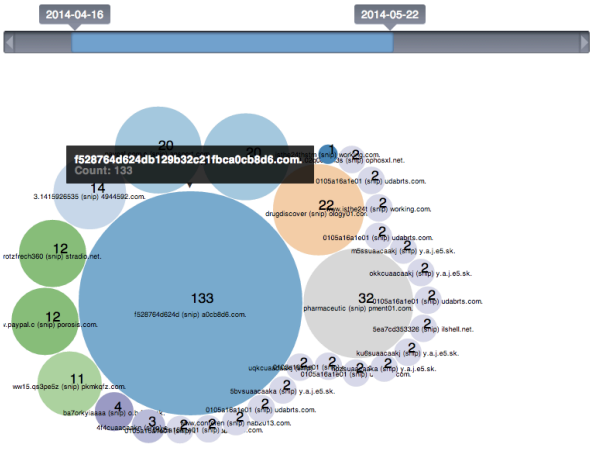

Fig. 10. Visualization of the number of DGAed queries asked.

B. Summary

Table III summarizes all the implemented analysis modules that we have come up with so far (almost for one year). Thanks to the pre-processed data by import module of each dataset and uniform programmability of MATATABI, multiple experiments have been conducted. Furthermore, the script are small and easy to implement, with most of the ranging from from 20 to 160 Lines of Code (LoC).

[5]http://www.malwaredomainlist.com/
[6]http://d3js.org/

VI. DISCUSSIONS

Early warning on a threat: As the number of cyber-attacks grows, detection mechanisms and countermeasures against threats targeting at enterprise are becoming indispensable. The convention on cybercrime [16], which the Japanese government has signed, states that the maximum duration for preserving computer data shall be 90 days. Once a company encounters a cyber-attack, it is required to analyze in detail the information collected during the attack, contained in the data logs, deploy countermeasures for both the source and targets of the attack, and identify the range of influence. As indicated with our benchmark in figure 5, MATATABI with DNS querylog is able to process data from one month (31 days) within 500 seconds, and might be possible to process within 20 minutes if the stored data spans a duration of 90 days. The hadoop-based infrastructure allows us to increase the number of recorded information processed and gives a potential to analyze multiple data sources for a long duration, which include the target incident to be detected.

Best practice for implementing analysis module: currently Presto-db presents the best performance on simple data querying as shown in § IV, but the software is still young and has limitations on creating table onto the original database, among others. Therefore we still use Hive for such operations in the data import and analysis modules. Furthermore, the performance result of SQL JOIN operation between Hive and Presto-db suggests that Hive with `RCFile` achieves good response time in our benchmarks, even for large amount of parsed data. Calculations after the queries need different processing on data and can use available utilities such as Rhadoop, which is provided by the streaming feature of Hadoop.

Users can benefit from each method available for Hadoop depending on what the analysis modules require.

The adaptability of open source tools: The open source tools that we incorporated into our system, introduced ex-

tended functionality during initial deployment, without requiring much effort to be invested during the software development. Indeed, it requires to modify such software not only fixing bugs in the original one, but also optimizations for our purposes to handle the huge size of data[7].

Using a combination of Apache Hadoop, Hive, and Facebook Presto-db makes a faster deployment of our MATATABI system, but introduces difficulties in terms of operating the system. The issues we have faced so far are: 1) appropriate resource dispatch between concurrent threat detections, 2) inability to estimate each job's duration, which may keep occupying processors' resource, 3) precise access control on each job, among others. These are not originally addressed by a particular software, and require careful system operation in the end.

VII. Related Work

P^3 [13] studied Hadoop platform to analyze a large amount of traffic data, with improving RIPE `hadoop-pcap` by reducing computational overhead. DDoS-Hadoop [12] extends P^3 to introduce a counter-based DDoS anomaly detection method on Hadoop. Both very early studies are our basis and gave the potential benefits of multi-terabytes traffic analysis with `pcap` and netflow data.

Li et al. [14] proposed a system to classify the host roles (i.e., clients or servers, etc) by using sFlow traffic with machine learning supported Hadoop environment. Their objective is to provide an analysis platform to detect hosts role from measurement data in timely manner with an online system, while P^3 was focused on an offline analysis. Our proposed MATATABI is also based on an offline analysis, however it is also possible to be an online system providing faster performance to inspect packets and flows on the fly if further optimization to the data access performance would be archived.

Hashdoop [8] introduced a way to speed-up the detection of network anomalies by distributing heavy computations among Hadoop cluster nodes. It shows 15 times speedup, at maximum, compared to standalone version of detectors. Also, accuracy evaluation with MAWILab [7] report as ground truth data highlights better detection with Hashdoop. The key benefit is to use hash function during job splitting to preserve spatial and temporal structure of traffic dataset for the anomaly detection. We plan to integrate their effort into our system in near future.

VIII. Conclusions and Future Work

We have reported MATATABI, a data collection and threat analysis platform that uses the Hadoop environment. The system has been designed to meet a set of requirements for security threat analysis, and achieves scalability with uniform programmable analysis module in a timely manner. Then we have presented benchmarks on querying stored data and shown speedups of up to 21 times (compared to a single machine), when the backend of MATATABI uses Presto-db as a data store. We have also showcased the use cases with our designed analysis modules to detect cyber threats with small amount of effort required for implementation.

Our system is running daily to analyze and detect security incidents from collected data, but there is still room for improvement on the system. At first, most of analysis with MATATABI is running every 24 hours as batch processes, but shorter period and hopefully real time analysis would be desired for certain threats. That would pose another challenge for the system's design such as faster data import rather than importing files collected by measurement sensors. Another direction for a broader system design is to integrate the threat information sharing system with the provisioning mechanism for defense, making information pipeline for more resilient mechanism on cyber-threats.

[7]All of our modified and implemented modules are available at https://github.com/necoma.

Acknowledgments

This research has been supported by the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 608533 (NECOMA). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the Ministry of Internal Affairs and Communications, Japan, or of the European Commission. This work was also supported by JSPS KAKENHI Grant Number 26330101. Thanks to all the volunteers, Romain Fontugne, Daisuke Miyamoto, Wataru Tsuda, for the development of analysis modules on MATATABI.

References

[1] Apache Hadoop. http://hadoop.apache.org/. (Accessed 30th January 2014).

[2] Apache Hive data warehouse. http://hive.apache.org/. (Accessed 30th January 2014).

[3] Apache Thrift. http://thrift.apache.org/. (Accessed 30th January 2014).

[4] Large-scale PCAP Data Analysis Using Apache Hadoop. https://labs.ripe.net/Members/wnagele/large-scale-pcap-data-analysis-using-apache-hadoop. (Accessed 30th January 2014).

[5] Presto: Distributed SQL Query Engine for Big Data. http://prestodb.io/. (Accessed 30th January 2014).

[6] RHadoop. https://github.com/RevolutionAnalytics/RHadoop/wiki. (Accessed 30th January 2014).

[7] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab : Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *CoNEXT '10*, pages 8:1–8:12, Philadelphia, USA, 2010.

[8] Romain Fontugne, Johan Mazel, and Kensuke Fukuda. Hashdoop : A mapreduce framework for network anomaly detection. Toronto, Canada, 2014.

[9] Yongqiang He, Rubao Lee, Yin Huai, Zheng Shao, N. Jain, Xiaodong Zhang, and Zhiwei Xu. Rcfile: A fast and space-efficient data placement structure in mapreduce-based warehouse systems. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1199–1208, April 2011.

[10] A. Herzberg and H. Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 224–232, Oct 2013.

[11] Nan Jiang, Jin Cao, Yu Jin, Li Li, and Zhi-Li Zhang. Identifying suspicious activities through dns failure graph analysis. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 144–153, Oct 2010.

[12] Yeonhee Lee and Youngseok Lee. Detecting ddos attacks with hadoop. In *Proceedings of The ACM CoNEXT Student Workshop*, CoNEXT '11 Student, pages 7:1–7:2, New York, NY, USA, 2011. ACM.

[13] Yeonhee Lee and Youngseok Lee. Toward scalable internet traffic measurement and analysis with hadoop. *SIGCOMM Comput. Commun. Rev.*, 43(1):5–13, January 2012.

[14] Bingdong Li, Mehmet Hadi Gunes, George Bebis, and Jeff Springer. A supervised machine learning approach to classify host roles on line using sflow. In *Proceedings of the First Edition Workshop on High Performance and Programmable Networking*, HPPN '13, pages 53–60, New York, NY, USA, 2013. ACM.

[15] Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi. An evaluation of machine learning-based methods for detection of phishing sites. In *Advances in Neuro-Information Processing*, volume 5506 of *Lecture Notes in Computer Science*, pages 539–546. Springer Berlin Heidelberg, 2009.

[16] Council of Europe. Convention on Cybercrime, November 2001.

[17] CERT Polska. ZeuS P2P+DGA variant mapping out and understanding the threat. http://www.cert.pl/news/4711/langswitch_lang/en. (Accessed 30th January 2014).

[18] Christian Rossow. Amplification hell: Revisiting network protocols for ddos abuse. In *NDSS Symposium 2014*, pages 224–232, February 2014.

[19] Yuuki Takano, Ruo Ando, Takeshi Takahashi, Tomoya Inoue, and Satoshi Uda. A Measurement Study of Open Resolvers and DNS Server Version. In *Internet Conference 2013*. IEICE, 2013.