

Real Time Cyber Analytics Data Collection Framework

Herbert Maosa
Cyber Security Research Centre
London Metropolitan University
London, UK
h.maosa1@londonmet.ac.uk

Prof. Karim Ouazzane
Cyber Security Research Centre
London Metropolitan University
London, UK
k.ouazzane@londonmet.ac.uk

Viktor Sowinski-Mydlarz
Cyber Security Research Centre
London Metropolitan University
London, UK
w.sowinskimydlarz@londonmet.ac.uk

Abstract— For effective security, it is critical that event data is collected in near real time as possible to enable early detection and response to threats. Performing analytics from event logs stored in databases slows down the response time due to the time cost of database insertion and retrieval operations. We present a data collection framework that minimizes the need for long term storage. Events are buffered in memory, up to a configurable threshold, before being streamed in real time using live streaming technologies. The framework deploys virtualized data collecting agents that ingest data from multiple sources including external Threat Intelligence. The framework enables the correlation of events from various sources, improving detection precision. We have tested the framework in a real time, machine-learning based threat detection system. Our results show a time gain of 300 milliseconds in transmission time from event capture to analytics system, compared with storage-based collection frameworks. Threat detection was measured at 95%, which is comparable to the benchmark snort IDS.

Keywords— *Data Collection, Event Correlation, Cyber event analytics, Real time detection, Log analysis*

I. INTRODUCTION

Security analytics systems rely upon data sourced from multiple network infrastructure devices such as Intrusion Prevention and Detection Systems (IDPS), network firewalls and routers, network switches and various application firewalls. Before this data can be analyzed for possible security threats, it needs to be collected. Therefore, data collection is a crucial and critical step in the cyber analytics process. Consequently, data collection might as well be a performance-critical path for analytics systems (Ramah et al., 2006),(Qadeer et al., 2010), especially when the need to consume big data or perform analysis in real time arises.

The figure below presents a typical cyber analytics process.

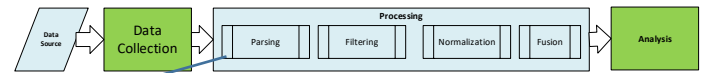


Fig 1. Typical Analytics Process

Different analytics applications will be the consumers of the data collected and processed in the preceding stages. Suppose the analytics applications and consequent processes are time sensitive. In that case, the data collection stage must make the ingested data available as quickly as possible, while at the same time collecting sufficient data on which to form accurate inferences.

While detection capability remains key in any cyber response system, the timeliness of the detection is even more paramount as attacks detected too late would have caused significant damage by the re-action time.

This paper presents a data collection framework that enables the real time detection and response of cyber threats. The near total elimination of local long-term storage of collected data saves significant time cost complexity. The use of state-of-the-art real time streaming technologies ensures that data is available to analytics applications as soon as practically possible, enabling our analytics applications to implement real-time reactions.

The innovation of our solution comes in several ways. Firstly, the system ingests from multiple source types including external cyber threat intelligence. This improves the maturity capability of the overall security operations. Secondly, the architecture improves the storage layer by allowing in-memory analytics, which improves the overall detection and response time. Further, the architecture embraces modern technologies to enable real time streaming and analysis of security events, mitigating technology limitations prevalent in the state-of-the-art solutions.

Our contribution is a flexible, scalable, expandable, and multi-source collection architecture and framework for data collection that enables timely detection of security threats and response.

The rest of this paper is organized as follows:

First, we review some of the recent research in data collection for cyber security, where we critically analyze and highlight the research gaps this paper addresses. Then we present our proposed framework, highlighting the architectural pillars that differentiate our work. We then illustrate an implementation based on our Framework, followed by experimentation and results. We conclude this work and propose some future work.

II. RELATED WORK

Various research on data collection methods and technologies can be found in the research literature.

The collection module proposed in (Razaq et al., 2016) populates security-related data in a local MySQL database, after which a Hadoop snoop job exports the data to an off-shore data store based on Hadoop File System. Analytics applications then run atop the data in the Hadoop system.

The real-time cyber threat detection platform in (Carvalho et al., 2016) collects data from both internal and external sources. After some pre-processing, the data is loaded into multiple databases according to data type (Malware Database, Social Media Database, Email Database, etc.). Big data analytics is then deployed using machine learning algorithms that train and detect threats in real-time data flows. Open Source technologies are used in (R. More et al., 2017) to detect threats in real time. Captured Sensor data is uploaded to Apache Hadoop Clusters before being trained and classified using Apache Mahout.

Deliu et al (Deliu et al., 2017), compared the performance of extracting intelligence data from hacker forums using Convolutional Neural Networks(CNN) vs traditional machine learning algorithms. The emphasis of both these papers is the comparison of different machine learning algorithms in extracting cyber threat intelligence.

The same authors (Deliu et al., 2018) deployed a two stage process to extract Cyber Threat Intelligence (CTI) from

hacker forums using Support Vector Machine algorithm for the classification. A second stage involved utilizing Latent Dirichlet Allocation algorithm to aggregate the data based on topics of discussion.

A server/client based agent model was used to develop a system for collecting open source intelligence from multiple sources in (Kim et al., 2018). The system was used to collect and extract Indicators of Compromise (IOC) from Open-Source Intelligence (OSINT) feeds and correlate them.

The framework proposed in (Jin et al., 2018) implements a collection layer that integrates the data collected from various source types including network, application logs, and external intelligence.

For real time detection, the authors in (Lopez et al., 2018) evaluate the use of Open Source Platform for Network Function Virtualization(OPNFV) for real time threat detection. They implement a capture module that use Bro Probes and employs Spark Streams to stream the captured packet header data to cloud-based Apache Kafka Infrastructure for temporary storage. The batch data is then fed to machine learning based trainers. Classifiers use this historic batch-processed data to detect threats in live streams in an implementation of the Lambda Architecture.

The framework proposed in (Arizona State University et al., 2019) employs custom web crawlers to collect darknet data from webpages and then archives the data for long term storage. In (Koloveas et al., 2021), web crawlers are also used in the data acquisition module to collect data from various Intelligence feeds. The collected data is then stored in an internal NoSQL Database (MongoDB) for later Data Analysis.

The IoT Data Collection Framework presented in (De Vita et al., 2020) uses a micro controller unit capable of controlling several sensors to address heterogeneity of data captured from various sensors at the physical layer. The data is then stored in Influx DB, a non-relational database optimized for storing long time series data.

In (P. More & Mishra, 2020), time complexity is reduced by enhancing the feature reduction algorithm of Principal Component Algorithm (PCA) based analysis.

The real-time Network Intrusion Detection System (NIDS) presented in (Seo & Pak, 2021) relies on optimizing the feature selection of a first level packet based machine learning classifier to achieve faster detection speeds, before engaging a second-level full featured packet session classifier to achieve accurate detection.

The literature reviewed reveals that active research in data collection for cyber security is concentrated on methods and techniques with little work done in addressing the need for the timeous collection and delivery of data for real time analytics. Additionally, the state-of-the-art in data collection frameworks rely on early storage of collected data in some form of database, making real time analytics of cyber events a futile effort.

In this paper, we address the above concerns by presenting a system architecture for data collection that is biased toward real time detection and response to cyber threats. The system allows for both online and offline analytics by using filtered event logs that are buffered in memory using data structures. The in-memory data is streamed to cloud-based Apache Kafka storage clusters at an optimized threshold and interval. This architecture allows for real time detection as well as forensic analysis.

III. PROPOSED DATA COLLECTION FRAMEWORK

The framework is presented in figure 2 below

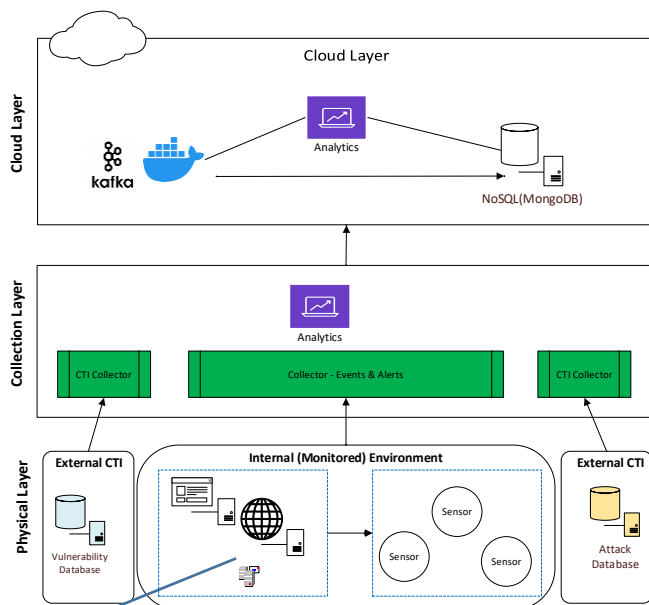


Fig 2. Data Collection and Analytics

Physical layer: This represents the IT infrastructure systems comprising servers, network, and security infrastructure devices. Various sensors are deployed in the internal environment, capturing events of interest. The External Cyber Threat Intelligence (CTI) Databases are located elsewhere on the Internet and act as sources of various Cyber Intelligence feeds of interest. In this research, a virtualized sandbox environment containing real malware samples represents the internal infrastructure, while other virtualized servers host scripts for external data collection.

Collection Layer: This layer implements several custom scripts, applications, adapters, and plugins for collecting data from the various physical layer systems. SNORT in IDS mode is the probe used to collect network activity from real malware samples in the form of packet captures. Due to the heterogeneity of the physical layer systems, the incoming data is in different formats. This layer is therefore also responsible for pre-processing steps such as filtering and normalization. As discussed earlier, one of the key bottlenecks in the reviewed frameworks is the use of local long-term storage for the collected data at this stage. In our framework, we keep this data in memory data structures, eliminating the computational cost of database operations. Streaming modules export data to cloud-based storage and analytics systems when data structures grow to an optimized threshold.

Cloud layer: The cloud layer is home to cloud-based storage and analytics applications. We implement this layer by installing Kafka Brokers and servers in the cloud and using custom applications to stream events. Further, for long-term storage, data is exported from Kafka servers to NoSQL-based database (MongoDB) at configurable thresholds. This allows for the analytics applications to work in both online and offline modes.

A. Data Sources

The system currently processes data from multiple sources as follows:

- **Network Packets :** Live Network activity is captured using the snort IDS tool. The captured traffic is from network activity from live malware that has been executed in windows machines in a controlled

(sandbox) environment. The network traffic is saved at periodic intervals in the standard **.pcap** format. Additionally, forensic network traffic is obtained from public sources. This traffic is also representative of various malware activities as captured in different environments elsewhere around the internet.

- **Event Data** : Snort has been set up in IDS mode to capture suspicious traffic according to various rules. As snort detects possible intrusions, it generates alerts . These alerts are captured in syslog format. Our system ingests these alerts for future analysis and correlation. Additionally, the activities of the malware inside the windows host machines are captured by the Windows Operating System using the Windows Event Management system and stored as EVTX files. Likewise, our system takes these EVTX files and pre-processes them before onward transmission and storage for further analysis and correlation.

B. Data Pre-Processing

Before onward transmission of the data and analysis, certain pre-processing procedures are performed.

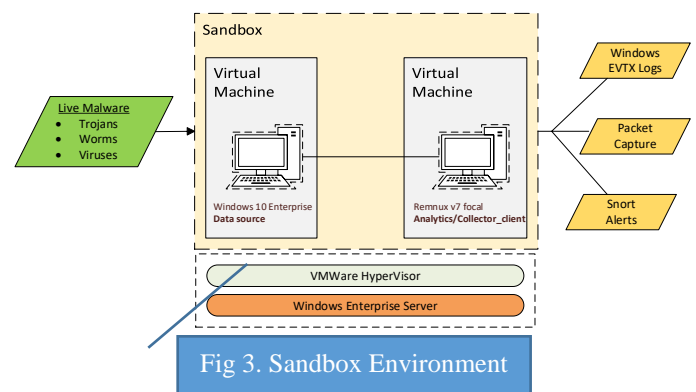
- **Packet Filtering** : The captured **.pcap** network packets comprise the IP header and the payload. We eliminate the payload from further transmission and analysis in our current implementation. This increases the throughput of our system as we only need to transmit the IP header , which is maximum 20 bytes in length.
- **Alert and Log Filtering** : Further, the snort alerts and windows EVTX logs contain more information than is needed for our analytics. For correlations to work, we need to extract the fields of interest upon which we can establish relationships with the other sources. We pass the alerts and evtx logs through a filter, to extract attributes of interest.
- **JSON formatting**: Further, the filtered IP header, alert and evtx logs are digitally serialized in **JSON** format . The JSON structure is stored in memory up

to a configurable buffer limit, after the data is streamed into cloud-based analytics applications.

IV. EXPERIMENTS AND RESULTS

A. Data Generation

The diagram below illustrates the implementation of a Sandbox Environment used for generating and collecting network activity from real malware samples.



The set-up is based on isolating the infected environment using virtualization technology. The various components are described below

- **Host System** : The host system is a dedicated cloud server configured with 4TB of HDD storage, 64GB RAM and 16 CPU cores.
- **Host Operating System**: The Host Operating System is Ubuntu Linux 20.04 Focal, with hardware acceleration enabled for virtualisation support
- **Virtualisation Layer** : Virtualisation is implemented using the Linux native Kernel Virtual Machine Manager (KVM) with Qemu.
- **Virtual Machines** : Four Virtual Machines have been set up in the sandbox. Three are running Microsoft Windows 10 Home, configured with 60GB HDD and 4GB RAM. The fourth virtual machine is a Remnux system(ref), which is based on Ubuntu 18.04 bionic. The Remnux system is a purpose-built Linux system comprising a curated list of applications, tools, and utilities for malware analysis. The screenshot below shows a live capture

of the virtual machines from the Linux Kernel Virtual Machine Manager.

Live malware samples have been downloaded and are introduced into the windows host machines. This malware comprises viruses, worms and trojans. As the malware executes inside the windows hosts, the windows event management system logs open file handles, processes, files, and command execution and records all these events as log files in the Windows EVTX format. Further, the malware attempts network connections as they scan for more targets or contacts their command-and-control Centre. These network activities are captured by the SNORT IDS in the remnux machine. Further, SNORT triggers alert if the network connections match the snort IDS rules.

1) Collection Process

Several python scripts are utilized in the collection process

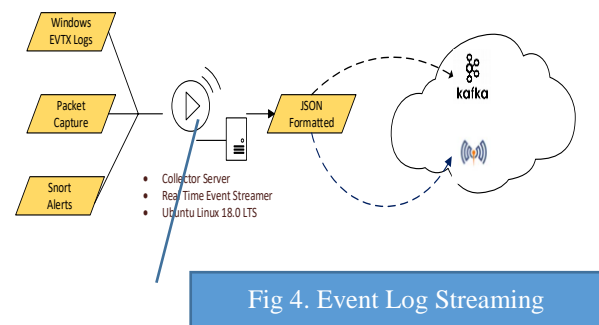
- *Collector server* : This is the server side of a centralised collector script. It runs outside the controlled environment (sandbox) and listens for TCP connections on a configurable port, currently 7508. Upon connection, it receives raw bytes of data from a client collector and saves them into a .pcap file in a central location.
- *Collector client* : This is the client side of the collector script. It runs in a controlled environment. The script scans the storage directory at configurable intervals and transfers all new .pcap files to the central collector above
- *Pcap_filter* : This script takes a pcap file and dissects each packet into individual header fields, discarding the payload and serializing the filtered header into a json structure and file.
- *Json_producer* : This script takes as input Json formatted data structure from either persistent file storage or in-memory and streams the contents to a listening Kafka broker in the cloud using Apache Kafka real time streaming technology. The script implements Kafka Producer client.
- *mqtt_producer* : Like the script above, mqtt_producer.py takes json formatted structure and streams the contents to a listening MQTT broker in

the cloud using eclipse MQTT broker using real time MQTT real time messaging technology

- *cve_parser*: This script parses NVD CVE file, which is in JSON format and filters the CVE attributes of interest, saving the contents into a new JSON formatted file. This filtered JSON CVE file will later be used for software vulnerability detection.

The packet captures, log files and alerts are collected by the client collector script running on the remnux machine and sent to the central collector running on the host system. After parsing and filtering as earlier described, these data are streamed into MQTT and Kafka brokers in the cloud.





The diagram below depicts the onward processing of the data



B. Analytics

To complete the experiment and demonstrate our framework's use case, we implement threat detection through containerization and parameterization of machine learning code. The details of the containerization and parameterization of the machine learning code are a subject for another paper. Here we demonstrate the application of the resultant analytics based on the data streamed by our framework. Fig 5 below illustrates the output from the analysis

▼ Metrics

Name	Value
Number of total ACK 	6645
Number of total PSH ACK 	102
Number of total RST 	0
Number of total regular packets 	14146

▼ Tags







Name	Value	Actions
ACK packet destination address	192.168.1.104	 
ACK packet protocol	DHCP	 
ACK packet source address	192.168.1.1	 
PSH ACK packet destination address	192.168.1.104	 

Fig 5. Analytics

We use Multi-Layer Perceptron Neural Network (MLP NN) and Support Vector Machine (SVM). The NN model is sequential with three layers. The SVM uses radial basis function kernel. The training data for our research comes from Netresec (public packet capture repository - <https://www.netresec.com>). The format of the files is PCAP, and CSV and their size varies from 6MB to 318MB. The PCAP files are used for initial model training. We used 7 CSV files and 10 PCAP files, including Ursnif and Trick Bot infected traffic. We also used DDoS flood data examples (PSH-SYN-FIN, URG-PSH-SYN). Each record has numerical data, describing the packet numbers, the time and length of the packets. Categorical variables are the source and destination IP addresses, protocol used and packet info. The model has trained over 250 epochs (number of passes through the whole dataset) with batch size of 128 training examples in a single batch.

Below we show successful reception of the data streams from our sandboxed data sources.

Using MLFlow, we show below a packet dissection analysis of the received data, showing metrics of interest used for Intrusion detection.

V. SUMMARY

Data collection is a critical part of an analytics system. If the collection framework involves storing data into secondary storage before processing and use, a significant time penalty can be incurred. Timely response is key in alleviating the consequences of a cyber-attack.

In this paper, a data collection framework for real time analytics has been presented. In-memory data structures are used to hold filtered real-time data. Event records are immediately presented into the data structure without the time wasteful need first to accumulate them into log files. This presents an opportunity to treat time-series and otherwise static data in real time. Using Apache Kafka, records in the data structure are streamed to cloud-based analytics systems. Our Containerized machine learning analytics system ingests this data and detects threats in real time. The results show that there is a time gain of 300 milliseconds in transmission time from event capture to analytics system, when compared with storage-based collection frameworks. The threat detection was measured at 95%, which is comparable to the benchmark snort IDS.

VI. FUTURE WORK

The framework currently uses multiple data collectors per stream but in one virtualized system node. This could result in sub-optimal resource utilization if the volume of data to be processed is limited, wasting the entire virtualized resource. On the other hand, if the data velocity and volume from the various sources are big, as would be expected in today's era of big data, the framework might not scale. We therefore look towards containerization of the collectors, with a control module that implements real time performance monitoring of the containers that allows the system to apply container auto-scaling depending on resource utilization. This is the future direction of our research.

REFERENCES

- Arizona State University, Benjamin, V., Valacich, J. S., University of Arizona, Chen, H., & University of Arizona. (2019). DICE-E: A Framework for Conducting Darknet Identification, Collection, Evaluation with Ethics. *MIS Quarterly*, 43(1), 1–22. <https://doi.org/10.25300/MISQ/2019/13808>
- Carvalho, V. S., Polidoro, M. J., & Magalhaes, J. P. (2016). OwlSight: Platform for Real-Time Detection and Visualization of Cyber Threats. *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, 61–66. <https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.73>
- De Vita, F., Bruneo, D., & Das, S. K. (2020). A Novel Data Collection Framework for Telemetry and Anomaly Detection in Industrial IoT Systems. *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 245–251. <https://doi.org/10.1109/IoTDI49375.2020.00032>
- Deliu, I., Leichter, C., & Franke, K. (2017). Extracting cyber threat intelligence from hacker forums: Support vector machines versus convolutional neural networks. *2017 IEEE International Conference on Big Data (Big Data)*, 3648–3656. <https://doi.org/10.1109/BigData.2017.8258359>
- Deliu, I., Leichter, C., & Franke, K. (2018). Collecting Cyber Threat Intelligence from Hacker Forums via a Two-Stage, Hybrid Process using Support Vector Machines and Latent Dirichlet Allocation. *2018 IEEE International Conference on Big Data (Big Data)*, 5008–5013. <https://doi.org/10.1109/BigData.2018.8622469>
- Jin, X., Cui, B., Yang, J., & Cheng, Z. (2018). An Adaptive Analysis Framework for Correlating Cyber-Security-Related Data. *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 915–919. <https://doi.org/10.1109/AINA.2018.00134>
- Kim, N., Lee, S., Cho, H., Kim, B.-I., & Jun, M. (2018). Design of a Cyber Threat Information Collection System for Cyber Attack Correlation. *2018 International Conference on Platform Technology and Service (PlatCon)*, 1–6. <https://doi.org/10.1109/PlatCon.2018.8472775>
- Koloveas, P., Chantzios, T., Alevizopoulou, S., Skiadopoulos, S., & Tryfonopoulos, C. (2021). inTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence. *Electronics*, 10(7), 818. <https://doi.org/10.3390/electronics10070818>
- Lopez, M. A., Gonzalez Pastana Lobato, A., Duarte, O. C. M. B., & Pujolle, G. (2018). An evaluation of a virtual network function for real-time threat detection using stream processing. *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, 1–5. <https://doi.org/10.1109/MOBISECSERV.2018.8311440>

- More, P., & Mishra, P. (2020). Enhanced-PCA based Dimensionality Reduction and Feature Selection for Real-Time Network Threat Detection. *Engineering, Technology & Applied Science Research, 10*(5), 6270–6275. <https://doi.org/10.48084/etasr.3801>
- More, R., Unakal, A., Kulkarni, V., & Goudar, R. H. (2017). Real time threat detection system in cloud using big data analytics. *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 1262–1264. <https://doi.org/10.1109/RTEICT.2017.8256801>
- Qadeer, M. A., Iqbal, A., Zahid, M., & Siddiqui, M. R. (2010). Network Traffic Analysis and Intrusion Detection Using Packet Sniffer. *2010 Second International Conference on Communication Software and Networks*, 313–317. <https://doi.org/10.1109/ICCSN.2010.104>
- Ramah, K. H., Ayari, H., & Kamoun, F. (2006). Traffic Anomaly Detection and Characterization in the Tunisian National University Network. In F. Boavida, T. Plagemann, B. Stiller, C. Westphal, & E. Monteiro (Eds.), *NETWORKING 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems* (Vol. 3976, pp. 136–147). Springer Berlin Heidelberg. https://doi.org/10.1007/11753810_12
- Razaq, A., Tianfield, H., & Barrie, P. (2016). A big data analytics based approach to anomaly detection. *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies - BDCAT '16*, 187–193. <https://doi.org/10.1145/3006299.3006317>
- Seo, W., & Pak, W. (2021). Real-Time Network Intrusion Prevention System Based on Hybrid Machine Learning. *IEEE Access, 9*, 46386–46397. <https://doi.org/10.1109/ACCESS.2021.3066620>