



北京邮电大学
Beijing University of Posts and Telecommunications



Queen Mary
University of London

Undergraduate Project Report 2021/22

Design and Implementation of data monitoring
probes for network traffic analysis in cybersecurity

Name:	Mudong Guo
School:	International School
Class:	2018215117
QMUL Student No.:	190018801
BUPT Student No.:	2018213067
Programme:	Internet of Things Engineering

Date: 28-04-2022

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Table of Contents

Abstract.....	3
Chapter 1: Introduction.....	5
1.1 Motivation.....	5
1.2 Objectives.....	6
1.2.1 Objective 1: Capture and analyse network traffic data flow.....	6
1.2.2 Objective 2: Deploy Kafka subscription system.....	6
1.2.3 Objective 3: Deploy Logstash pipeline.....	7
1.2.4 Objective 4: Deploy Elasticsearch and Kibana.....	7
1.2.5 Extra work.....	7
1.3 Contribution.....	7
1.3.1 For objective 1: Documents review, data capture and data transformation	7
1.3.2 For objective 2: Documents review, deploy and configure Kafka subscription system and create Kafka producer.....	8
1.3.3 For objective 3: Documents review, deploy Logstash and create configuration files for different pipelines.....	8
1.3.4 For objective 4: Documents review, deploy Elasticsearch and create Kibana graphs.....	8
1.3.5 For extra work.....	9
1.4 Technical Context.....	9
Chapter 2: Background.....	10
2.1 Data Capturing Technology.....	10
2.1.1 Data capturing based on Tcpcdump.....	10
2.1.2 Data capturing based on Wireshark.....	11
2.1.3 Data transformation based on Tshark.....	11
2.2 Kafka Subscription System.....	11
2.2.1 Reasons for importing Zookeeper.....	11
2.2.2 Structure of Kafka.....	11
2.2.3 ISR(IN-SYNC Replicas).....	11
2.3 The Combination of Kafka, Prometheus and Grafana.....	12
2.4 ELK Stack.....	12
Chapter 3: Design and Implementation.....	13
3.1 Data Capture.....	13
3.1.1 Data capture for Linux users	13
3.1.2 Data capture for Windows users.....	14
3.1.3 Data transformation in Linux.....	15
3.2 Design and Deployment of Kafka Subscription System.....	16
3.2.1 Deployment of Zookeeper.....	16
3.2.2 Deployment and configuration of Kafka broker cluster.....	17
3.2.3 Creation of Kafka topics.....	18
3.2.4 Kafka producer.....	19
3.3 Prometheus and Grafana.....	21
3.3.1 Reason for importing Prometheus and Grafana.....	21
3.3.2 Deployment of Prometheus.....	21
3.3.3 Deployment of Grafana.....	22
3.4 Logstash.....	25
3.4.1 Creation of Logstash pipelines.....	25

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity	
3.4.2 Logstash pipeline that supports adding comments.....	26
3.4.3 Logstash pipeline that supports time form transformation.....	26
3.4.4 Analysis of Logstash pipelines.....	26
3.5 Elasticsearch and Kibana.....	26
3.5.1 Deployment of Elasticsearch.....	26
3.5.2 Data visualization.....	27
Chapter 4: Results and Discussion.....	28
4.1 Result	29
4.2 Discussion.....	29
4.1.1 Possible way to deploy system in real production environment.....	29
4.2.2 Feedbacks from users.....	29
Chapter 5: Conclusion and Further Work.....	31
5.1 Conclusion.....	31
5.2 Future work.....	31
5.2.1 Cloud deployment.....	31
5.2.2 Support monitor to more kinds of data records.....	32
References.....	33
Acknowledgement.....	35
Appendix.....	36
Specification, part1.....	36
specification, part2.....	37
Early-term Report.....	40
Mid-term Progress Report.....	43
Supervision Log.....	46
Risk and environmental impact assessment.....	48

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Abstract

This project aims to build a system for Linux users which is able to capture, store, transmit, monitor and analyse network traffic data flow by integrating network technical products, mainly including Tcpdump, Kafka and ELK (Logstash, Elasticsearch and Kibana) and other products. In first stage of the system, Tcpdump will be used to capture data packets from local WLAN and store packets to cap format file. Windows users can capture packets by using Wireshark. Taking the advantage of integrated tool provided by Wireshark or Tshark, users can transform the cap format files that store captured data to csv format that is easier for further analysis. In the second stage, Kafka producer analyses csv files that store data packets and transfer them to Kafka broker cluster. Kafka cluster consists of 3 Kafka brokers deployed at same machine and each of them listens to new events through different ports. Kafka brokers receive and store data packets to certain topic according to date. In order to better manage and monitor Kafka subscription system, this project imports Zookeeper, Prometheus and Grafana. In third stage of the system, ELK is the consumer of Kafka system. Logstash pipelines subscribe from Kafka by topic names and process on subscribed data after receiving data packets. Then Logstash pipelines send processed data to Elasticsearch. In last stage, Elasticsearch receives data sent from Logstash. Elasticsearch is a well-developed searching engine which also provides data storage function. With powerful chart display function, Kibana will be used to monitor and visualize data records stored in Elasticsearch for further analysis.

摘要

此项目目标为集成包含 Tcpdump, Kafka 以及 ELK 在内的一系列工具、系统与架构以搭建一个面向 Linux 用户的系统, 使其可以对网络数据流量进行捕获, 存储, 传输, 监控与分析。在第一阶段, Tcpdump 将会从本地无线网络对经过的流量进行抓包操作并将数据包储存在 cap 格式文件中。Windows 用户可以通过 Wireshark 进行抓包操作。通过利用 Wireshark 或 Tshark 提供的内置工具, 将 cap 或 pcap 格式的储存数据包的文件转换为更便于分析的 csv 格式文件。在第二个阶段, 本地 Kafka 生产者将解析储存被捕获数据包的本地文件, 并将其传输到 Kafka 节点集群。Kafka 节点集群包含三个节点, 每个节点通过不同的端口监听事件以便做出应对。Kafka 节点按照日期将数据储存在指定的主题内。为了更好地管理与监控 Kafka 节点集群, 项目引入了 Zookeeper, Prometheus 以及 Grafana。在第三个阶段, ELK 将会充当 Kafka 系统的消费者的角色。Logstash 管道将会向 Kafka 按主题订阅消息, 并且在收到来自 Kafka 节点发送的数据后, 通过调用指定配置文件对数据进行操作, 并将数据传输到 Elasticsearch 中。在最后一个阶段, Elasticsearch 接收来自 Logstash 的数据。Elasticsearch 是一款强大的搜索引擎, 同时也提供了数据存储的服务。Kibana 则会提供强大的图表展示功能, 对 Elasticsearch 内存储的信息进行监控与可视化, 以便更好地进行分析。

Chapter 1: Introduction

This chapter mainly introduces the motivation of this project, the objectives of this project, the author's contribution to this project and technical context in developing this project.

1.1 Motivation

Nowadays, as the amount of data records on the network increasing, it is unavoidable that amount of data that goes through our personal computers increases considerably. At the meanwhile, rapid improvement in computer techniques and astonishing enhancement in computer performance are convincing computer users to be more dependent on computers, since computers are able to do what people cannot do, to compute tasks too difficult for people to work on, and to store massive data records that people cannot remember. One straight result of this world-wide trend is extreme increase in value of network data since those data may contain important information including people's personal identification, home address, contact number, all kinds of password and other private data like secret business contracts in digital version. However, the more valuable the data, the more dangerous situation it can be in since all kinds of network attacks are coming for them. Hackers will try their best to find, access, decrypt and steal valuable data from its original owner, and even with firewall deployed and anti-virus application installed, computer users still face serious security violation since all security systems have vulnerabilities. What worsen the situation is many computer users only rely on default defending service instead of actively track and monitor data traffic flow that goes through their personal computer in daily life, this could bring various damage. In early 2021, reams of Facebook user data were discovered on hacker forums. The contents included full names, phone numbers, emails and location information. A total of 533 million Facebook users were affected. According to report from Facebook, the hackers behind the breach took advantage of a security hole fixed in 2019. But that only highlights the fact that users often find out about these kinds of events long after it's too late.

Facing such a threat, it will be useful to build a system that allows users to track and monitor network traffic data that go through their personal computers. With such a system, personal computer users are able to capture network data flow according to their own time schedule. They are able to track and monitor the captured data packets. They are also able to make different kinds of operation on captured network data according to their own preference, they can write some comments on a group of data packets which are captured among a certain time

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

period. At webpage, different kinds of graph will be provided which can give users clear cognition about the captured data. With such a system, personal users will have proactive knowledge of what have gone through their personal computer everyday instead of letting the potential harmful data unchecked.

1.2 Objectives

From academic standpoint, this project aims to build up a data traffic flow monitoring system used by Linux users. Users are able to use this system to gather, store and monitor all data packets that have gone through their computers among a period.

From technical perspective, the purpose of this project is to build a system that integrates Tcpcmdump, Kafka subscription system and ELK (Logstash, Elasticsearch and Kibana) and other software products. The system is able to use Tcpcmdump to capture network data traffic flow that go through local network; to extract and transform captured data to analysable format at local; to transmit captured data to Kafka broker cluster from local producer; to monitor Kafka broker cluster on both terminal and webpage with the help of Zookeeper, Prometheus and Grafana; to subscribe data records from Kafka subscription system by Logstash; to do some data processing job on subscribed data records within Logstash pipelines; to store subscribed data records to Elasticsearch and to use Kibana to monitor and visualize the data records stored in Elasticsearch.

1.2.1 Objective 1: Capture and analyse network traffic data flow

The system should be able to use Tcpcmdump (in Linux OS) or Wireshark (in Windows OS) to capture data packets from local network. Data packets captured by Tcpcmdump in Linux will be saved to cap format file. Since cap files store information in byte stream format, we need another format of data records that is easier to analyse. For Linux users, the system imports Tshark to transform cap file to csv format file, which is easier to analyse. For Windows users, they can use Wireshark to capture data packets and use integrated tools provided by Wireshark to directly transform the output to csv format file.

1.2.2 Objective 2: Deploy Kafka subscription system

The system should be able to transmit captured data packets to Kafka broker cluster from local Kafka producer. At the same time, the system should be able store captured data packets at Kafka broker cluster. Data packets will be sent to certain topic created by brokers according to date when packets were captured. Kafka subscription system also can publish data to various consumers.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

1.2.3 Objective 3: Deploy Logstash pipelines

The system should be able to use Logstash pipelines to make multiple kinds of operation on data subscribed from Kafka subscription system. The function of Logstash pipelines is to set up connection between Kafka subscription system and multiple kinds of consumers. Consumer is Elasticsearch and Kibana in this project. Every Logstash pipeline consists of an input plugin that takes data records from multiple data sources, a filter plugin that provides data processing function and an output plugin that sends processed data to multiple destinations.

1.2.4 Objective 4: Deploy Elasticsearch and Kibana

The system should be able to use Elasticsearch to receive and store data packets sent by Logstash pipelines. The system also should be able to use Kibana to visualize the general information of data packets that have been stored at Elasticsearch.

1.2.5 Extra work

The system should be able to monitor data stored in Kafka broker cluster in two ways. One way is to monitor Kafka broker cluster by Zookeeper client on command line. Another way is to monitor Kafka broker cluster through Kafka exporter, Prometheus and Grafana, by which we could monitor Kafka on webpage. Different graphs about metrics of Kafka cluster and topics will be provided by Grafana.

1.3 Contribution

I have built up the whole system in my Linux virtual machine. The software products I have used include Tcpdump, Wireshark, Tshark, Finalshell, Zookeeper, Kafka, Prometheus, Grafana and ELK. Based on four objectives and extra work previously mentioned, my contributions are as follows:

1.3.1 For objective 1: Documents review, data capture and data transformation

There are 3 steps required in order to achieve objective 1:

1. I need to read documents provided by Tcpdump officials in order to know how to use Tcpdump to capture data packets in most efficient ways. Since there are nearly 30 different parameters that could be configured during data capturing process, it is a hard task for me to find out the most efficient way of using Tcpdump to capture network data packets, Therefore I think the most difficult part of objective 1 is to master how to use Tcpdump to capture packets.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

2. I import Finalshell in order to transfer files between Windows OS and Linux OS through network based on SSH security protocols.
3. I need to think of ways to transform raw data packets captured by Tcpdump to more readable format. Through investigation I decided to import Tshark to this project. I need to read documents provided by Tshark official to find out how to use Tcpdump to transform raw data captured by Tcpdump.

1.3.2 For objective 2: Documents review, deploy and configure Kafka subscription system and create Kafka producer

There are 3 steps required in order to achieve objective 2:

1. Kafka is a well-known distributed system for data storage, publication and subscription. In order to understand how to build up Kafka subscription system and the meaning of each parameter that appears in configuration file, I need to read the document provided by Kafka official first. Then I need to build up the Kafka subscription system.
2. I need to write a local producer in order to transfer captured data to Kafka broker cluster. I think the design for Kafka producer is the most important process in this task.
3. The creation of topics that store data records will be a daily work. Every day when I have captured new packets from network, I need to create a new Kafka topic named according to date and send data packets to it.

1.3.3 For objective 3: Documents review, deploy Logstash and create configuration files for different pipelines

There are 2 tasks that I mainly work on for objective 3:

1. Logstash is the bridge between multiple kinds of data sources and multiple kinds of destinations. Each Logstash pipeline is a single entity that is able to process on data records uniquely. A typical Logstash pipeline consists of an input plugin, a filter plugin and an output plugin. Different parts of Logstash pipelines are also independent to each other. Therefore, I need to read document provided by official to pick up the input plugin, filter plugin and output plugin that I need.
2. I need to create multiple workable and efficient Logstash pipelines in order to provide various functions to users. I think the most challenging part of this process is the creation of different Logstash pipelines, since I always need to proactively create new Logstash pipelines that can provide new functions to the data processing part.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

1.3.4 For objective 4: Documents review, deploy Elasticsearch and create Kibana graphs

There are 2 task that I need to work on to achieve objective 4

1. Elasticsearch is a strong engine that could take in data records sent from Logstash. Kibana is used to visualize data stored in Elasticsearch. Therefore, I need to read document provided by official to understand how to deploy Elasticsearch and Kibana.
2. I also need to create graphs in Kibana in order to provide efficient presentation of data collected by Elasticsearch. I think the most challenging part in this process is the creation of graphs.

1.3.5 For extra work

Nearly all of the extra work in this project is related to Kafka subscription system. In order to better monitor Kafka subscription system, I import Zookeeper, Prometheus and Grafana. First of all, I need to read documents provided by Apache official in order to deploy Zookeeper and know how to monitor Kafka system through Zookeeper. Then, I need to learn knowledge about Prometheus and Grafana to know how to combine them to gather data from Kafka and present useful information on webpage of Grafana. And the most important and challenging part in this process is to create graphs that presents metrics of Kafka subscription system.

1.4 Technical Context

The project is developed by Java language. All of the process are developing in following environment:

Hardware:

Operating system: Windows 10

CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

RAM: 16G

Software:

VMware Workstation 16 Player

Operating system of middleware: Linux (CentOS 7)

JDK: 1.7.0

Chapter 2: Background

This chapter mainly introduces the background knowledge of main technologies used in this project. Firstly, it introduces data capturing technology used in this project: Tcpdump and Wireshark. Secondly, it introduces the basic structure and mechanism of Kafka subscription system used in this project. Then it introduces additional technology used in this project, including Zookeeper, Prometheus and Grafana and explains the reason for importing them. Finally, it introduces basic structure of ELK stack.

2.1 Data Capturing Technology

Data capture is basically the process that collect or extract structured and unstructured information from document and convert it into data readable to a computer. More broadly, data capturing also means collecting information from paper, network or other sources. It is the technology that every data system has to rely on.

2.1.1 Data capturing based on Tcpdump

Tcpdump is a widely used command-line packet analyser in Linux. It can present and capture all kinds of data packets that are going through a network. A Tcpdump command is as below:

```
tcpdump [ - aAbdDefhHIJKlLnNOpqStuUvxX#] [ - B size ] [ - c count ]  
[ - C file_size ] [ - E algo:secret ] [ - F file ] [ - G seconds ]  
[ - i interface ] [ - j tstamptype ] [ - M secret ] [ -- number ]  
[ - Q | - P in|out|inout ]  
[ - r file ] [ - s snaplen ] [ -- time - stamp - precision precision ]  
[ -- immediate - mode ] [ - T type ] [ -- version ] [ - V file ]  
[ - w file ] [ - W filecount ] [ - y datalinktype ] [ - z postrotate  
- command ]  
[ - Z user ] [ expression ]
```

(1)

By using the command above and configuring different parameters, users are able to capture network data traffic flow according to their own preference. There are 2 parameter that we need to put extra care. The content after parameter ‘-i’ indicates name of network interface, and content after parameter ‘-w’ indicates name of file that will store output. If parameter ‘-w’ is ignored, information of all captured packets will be printed on terminal.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

2.1.2 Data capturing based on Wireshark

Wireshark is an application used to analyse network packets in Windows. Compared to Tcpdump, Wireshark provides brief user interface.

2.1.3 Data transformation based on Tshark

Tshark is like the command-line version of Tshark. Since Tcpdump is not able to transform format of output, the system imports Tshark to transform cap format output to csv format for better analysis.

2.2 Kafka Subscription System

Kafka is a distributed data streaming platform that can publish, subscribe to, store and process data in real time. It is designed to handle data records from multiple sources and send data records to multiple kinds of consumers.

2.2.1 Reasons for importing Zookeeper

Zookeeper is a centralized service that used to maintain configuration information and name information, to provide decentralized synchronization and to provide group service. It is able to provide reliable configuration management service to distributed system like Kafka, in this project a special Zookeeper node will be used to monitor Kafka broker cluster. Each Kafka broker registers at Zookeeper first when it starts. Users can monitor the availability of Kafka brokers and list of created topics by logging in Zookeeper client.

2.2.2 Structure of Kafka

Kafka subscription system consists of producer, broker cluster and consumer. Producer is able to generate data records and send to broker cluster, the transmission between producer and brokers can be in real time or unsynchronized. Broker cluster receives data records sent by Kafka producer and stores it to topics appointed by producer, this process is called publication of data records. Consumer subscribes data records from Kafka broker cluster according to topic name.

2.2.3 ISR (IN-SYNC Replicas)

In Kafka subscription system, one topic can have more than one partition, and each partition can have more than one replication. For each partition of a topic, one replica will be the leader and rest will be the follower. Leader is responsible for reading and writing, and followers will synchronize data according to leader. When a leader shut down, controller will choose another leader from rest of the followers. During that process one rule has to be followed: only

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

follower in ISR list has the right to be chosen as new leader. ISR consists of leader and those followers who keep synchronized with leader. A follower will be kicked out of ISR list if it has fallen too far behind the leader or do not send synchronization request to the leader for a long time. ISR mechanism is considered to be an optimization mechanism based on fully synchronization among leader and followers

2.3 The Combination of Kafka, Prometheus and Grafana

The combination of Prometheus and Grafana is probably the most widely-used monitor tools to complicated system. Prometheus is a time series database that stores data records of the monitored system in different time nodes. Grafana provides graphs of the monitored system that enable users to know the situation of monitored system in real time. Data presented by Grafana graphs is obtained by querying Prometheus database.

In this project, Prometheus monitors the Kafka broker cluster through `kafka_exporter` and track all kinds of message, and Grafana collects data records from Prometheus and presents users with graphs to measure the metrics of Kafka broker cluster. With the help of Prometheus and Grafana, users are able to monitor metrics of data transmission within Kafka subscription system; users are also able to monitor Kafka topics according to their preference.

2.4 ELK Stack

ELK is not the name of an application or a platform, but the abbreviation of three separate software product: Elasticsearch, Logstash and Kibana. They are called ELK since they all come from Elastic.co and always work together in production environment.

The simplest structure of ELK consists of one Logstash instance, one Elasticsearch instance and one Kibana instance. Logstash pipeline obtains data records through input plugin from multiple data sources, push them into filter plugin for further data processing and sends the data records to Elasticsearch through output plugin. Kibana uses graphs to presents data stored in Elasticsearch.

Chapter 3: Design and Implementation

The design and implementation of this project is ordered according to main functions mentioned in the Chapter 1.2, which are “Data capture”, “Implementation of Kafka system”, “Monitor to Kafka system” and “Implementation of Logstash, Elasticsearch and Kibana”. Among them, implementation of Kafka subscription system and implementation of Logstash pipelines are the key functions of the project, which are also the parts that I put most of my effort in. In this chapter, I will explain in details about technologies and products used in this project. The picture below is the architecture of Kafka subscription system.

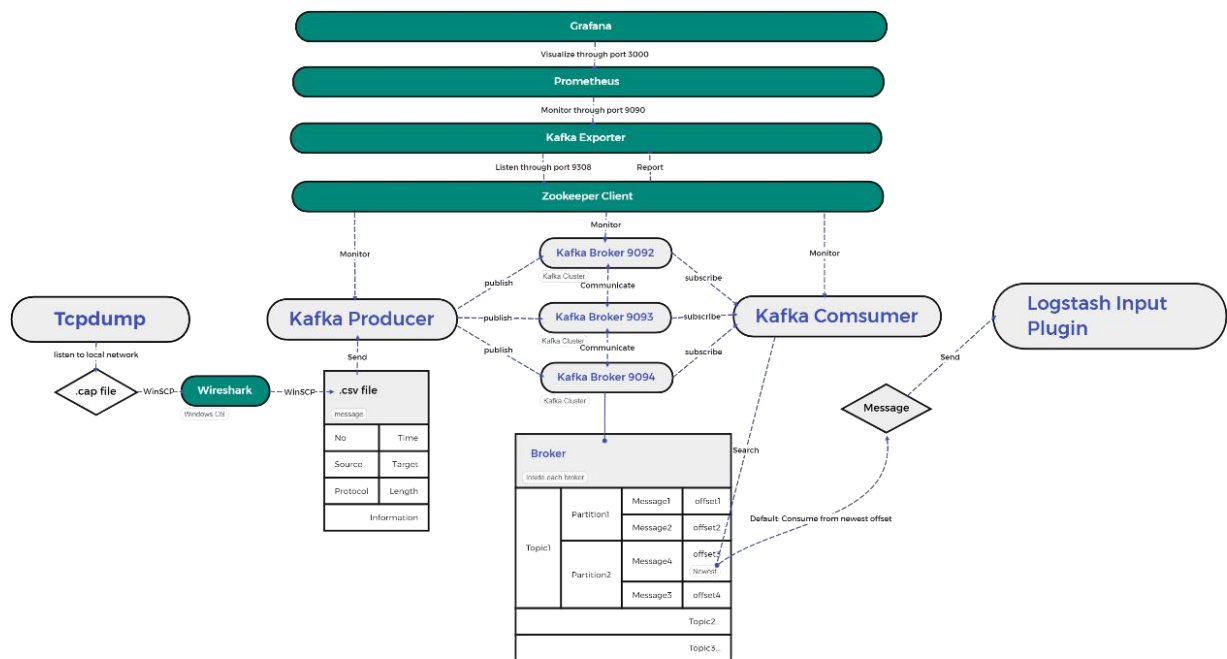


Figure 1 Architecture of Kafka subscription system

3.1 Data Capture

3.1.1 Data capture for Linux users

In most circumstances, users should use the command ‘*tcpdump -i ens33 -nn -s0 -w xxxx.cap*’ to capture packets. The parameter ‘*-i ens33*’ indicates the network interface that Tcpcdump is going to capture packets from network interface ens33; the parameter ‘*-nn*’ means the domain name and port number will not be resolved, it is not only easier for users to check IP address and port number but also more efficient in capturing large amount of data, since domain name resolution slows down capture speed; the parameter ‘*s0*’ indicates Tcpcdump to capture entire message of each packets; the parameter ‘*-w xxxx.cap*’ indicates all

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

packets captured will be stored to a file named xxxx.cap in current path, and without parameter '-w' all packets will be printed in command line. In picture below, data packets captured by Tcpcdump are printed on terminal. The first section is timestamp in hour: minute: second: fraction format. The second section is the source IP address, destination IP address, port number of source and destination and data flow direction. The packets with Flags [S /. /P /F /R] indicate the handshake process of TCP protocol: 'S' represents SYN which indicates connection starts; '.' Means ACK; 'P' represents PSH which indicates push data; 'F' represents FIN which indicates connection finish; 'R' represents RST which indicates connection reset. The next section 'seq num1: num2' represents byte range, the initial sequence number is randomly generated, as you can see in the picture. The sequence number for rest of the bytes in the connection are incremented from ISN. The parameter 'win' indicates number of bytes of buffer space the host has available for receiving data.

```
|root@bogan /| # tcpdump -i ens33 -nn -s0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
02:02:23.423634 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 3743319356:3743319519, ack 848559399, win 250, length 164
02:02:23.423804 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 164:232, ack 1, win 250, length 68
02:02:23.423876 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [.], ack 232, win 4105, length 0
02:02:23.425630 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 232:316, ack 1, win 250, length 84
02:02:23.425759 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 316:352, ack 1, win 250, length 36
02:02:23.425847 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [.], ack 352, win 4104, length 0
02:02:23.426565 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 352:596, ack 1, win 250, length 244
02:02:23.426692 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 596:632, ack 1, win 250, length 36
02:02:23.426775 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [.], ack 632, win 4103, length 0
02:02:23.427716 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 632:892, ack 1, win 250, length 260
02:02:23.427864 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 892:980, ack 1, win 250, length 88
02:02:23.427946 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [.], ack 980, win 4102, length 0
02:02:23.427952 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 980:1052, ack 1, win 250, length 72
02:02:23.428164 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [P.], seq 1:53, ack 1052, win 4101, length 52
02:02:23.428192 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [.], ack 53, win 250, length 0
02:02:23.428650 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [P.], seq 53:105, ack 1052, win 4101, length 52
02:02:23.428666 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [.], ack 105, win 250, length 0
02:02:23.429169 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 1052:1104, ack 105, win 250, length 52
02:02:23.429456 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [P.], seq 105:157, ack 1104, win 4101, length 52
02:02:23.429493 IP 192.168.116.1.49681 > 192.168.116.135.22: Flags [P.], seq 157:209, ack 1104, win 4101, length 52
02:02:23.429526 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [.], ack 209, win 250, length 0
02:02:23.430139 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 1104:1156, ack 209, win 250, length 52
02:02:23.440552 IP 192.168.116.135.22 > 192.168.116.1.49681: Flags [P.], seq 1156:1224, ack 209, win 250, length 68
```

Figure 2 Packets captured by Tcpcdump

Besides, in specified circumstances, users could add expression behind Tcpcdump command to filter the packets. For example, the command 'tcpdump -i ens33 -nn -s0 host local and \((a or b)\)' indicates to print traffic between local and a or traffic between local and b.

3.1.2 Data capture for Windows users

Although the whole system is deployed in Linux operating system, it meaningful to think of ways to capture data packets in Windows operation system since in real production users using Windows OS can also capture data at local and transfer captured data packets to subscription system.

In most circumstances, Windows users are able to use Wireshark to capture data packets from local. Compared to Tcpcdump, Wireshark is more friendly to users since it provides concise user interface. What's more, Wireshark is able to resolve raw packets automatically and print the results on its user interface, which saves time for manual resolving. 7 features will be

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

parsed out by Wireshark, including numerical order, time, source, destination, protocol, length and information. As picture shown below, after finishing data capture users are able to directly export data packets to CSV file, which is very concise to analyse.

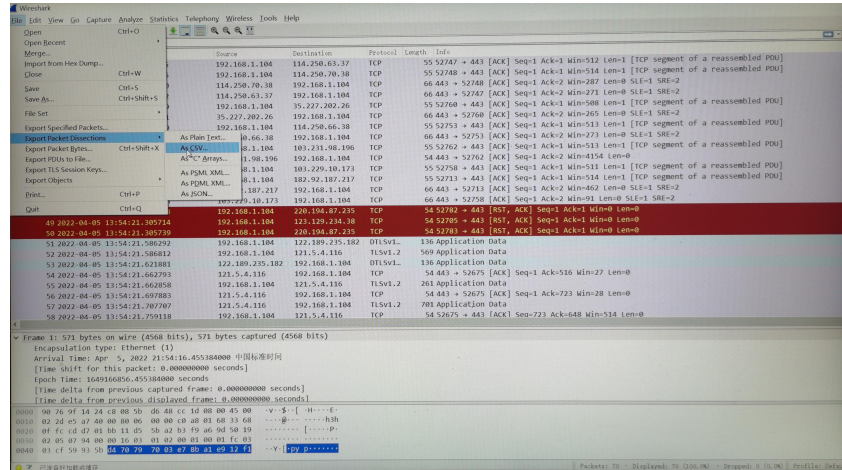


Figure 3 Packets captured by Wireshark and its output

If users want to transfer the output files to Linux server, one way is to use applications like WinSCP or Finalshell. These kinds of applications implement SFTP protocol based on SSH protocol.

3.1.3 Data transformation in Linux

Tshark is always considered to be the command line version of Wireshark, and an obvious difference between Wireshark and Tshark is that Tshark does not provide user interface. However, Tshark can still be useful for Linux users who try to capture and analyse network traffic, because Tshark supports the transformation of packets storage format in command line. After capturing packets by Tcpcap, users can use the command below to transform cap format file to csv format.

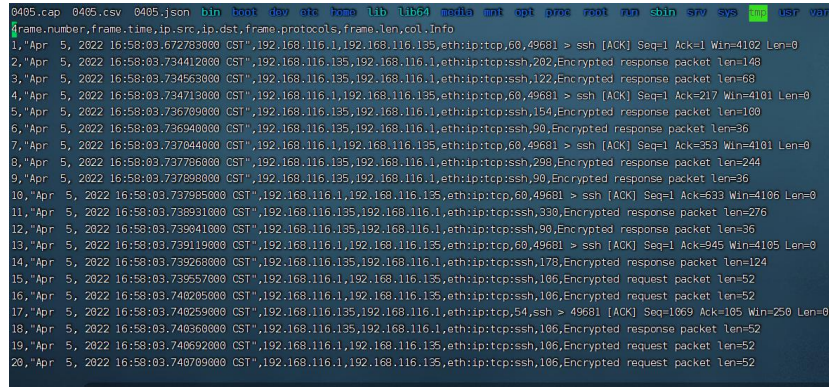
```
tshark -r xxxx.cap -T fields -e frame.number -e frame.time
-e ip.src -e ip.dst -e frame.protocols -e frame.len -e col.Info
-E header = y -E separator = , > xxxx.csv
```

(2)

This command can be partitioned into several sections. The first section ‘-r xxxx.cap’ indicates Tshark will read the file xxxx.cap from current path. The second section is from ‘-T fields’ to ‘-e col.Info’, this section defines output format and choose which part of the packets will be exported to csv file. The parameter ‘-E header = y’ indicates there will be headers at

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

first line of csv file that correspond to each feature. ‘-E separator =,’ means in each line a comma will separate different values. The picture shown below presents data packets stored in csv format file transformed by Tshark.



```
0405.cap 0405.csv 0405.json bin boot dev etc home lib lib64 media opt opt-prec root run sbin srv sys usr var
frame.number,frame.time,ip.src,ip.dst,frame.protocols,frame.len,col.info
1,"Apr 5, 2022 16:58:03.672783000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:60,49681 > ssh [ACK] Seq=1 Ack=1 Win=4102 Len=0
2,"Apr 5, 2022 16:58:03.734412000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,202,Encrypted response packet len=148
3,"Apr 5, 2022 16:58:03.734563000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,122,Encrypted response packet len=68
4,"Apr 5, 2022 16:58:03.734713000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:60,49681 > ssh [ACK] Seq=1 Ack=217 Win=4101 Len=0
5,"Apr 5, 2022 16:58:03.736789000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,154,Encrypted response packet len=108
6,"Apr 5, 2022 16:58:03.736948000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,90,Encrypted response packet len=36
7,"Apr 5, 2022 16:58:03.737044000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:60,49681 > ssh [ACK] Seq=1 Ack=253 Win=4101 Len=0
8,"Apr 5, 2022 16:58:03.737786000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,298,Encrypted response packet len=244
9,"Apr 5, 2022 16:58:03.737998000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,50,Encrypted response packet len=36
10,"Apr 5, 2022 16:58:03.737985000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:60,49681 > ssh [ACK] Seq=1 Ack=633 Win=4106 Len=0
11,"Apr 5, 2022 16:58:03.738931000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,330,Encrypted response packet len=276
12,"Apr 5, 2022 16:58:03.739041000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,90,Encrypted response packet len=36
13,"Apr 5, 2022 16:58:03.739119000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:60,49681 > ssh [ACK] Seq=1 Ack=945 Win=4105 Len=0
14,"Apr 5, 2022 16:58:03.739268000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,178,Encrypted response packet len=124
15,"Apr 5, 2022 16:58:03.739557000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:ssh,106,Encrypted request packet len=52
16,"Apr 5, 2022 16:58:03.740205000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:ssh,106,Encrypted request packet len=52
17,"Apr 5, 2022 16:58:03.740259000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:54,ssh > 49681 [ACK] Seq=1869 Ack=105 Win=258 Len=0
18,"Apr 5, 2022 16:58:03.740368000 CST",192.168.116.135,192.168.116.1,eth:ip:tcp:ssh,106,Encrypted response packet len=52
19,"Apr 5, 2022 16:58:03.740692000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:ssh,106,Encrypted request packet len=52
20,"Apr 5, 2022 16:58:03.740789000 CST",192.168.116.1,192.168.116.135,eth:ip:tcp:ssh,106,Encrypted request packet len=52
```

Figure 4 Csv file transformed by Tshark

3.2 Design and Deployment of Kafka Subscription System

As a distributed system with high throughput capacity and permanent data storage, Kafka subscription system has been world-widely used in real production environment. In this project, Kafka system is deployed in order to store csv files that contains data packets.

3.2.1 Deployment of Zookeeper

It is meaningful to deploy Zookeeper since it is designed as distributed coordination service that provides consistent service to distributed system like Kafka. In this project, only one Zookeeper node is deployed at Linux virtual machine since all three Kafka brokers has been deployed in same virtual machine, which means all Kafka nodes can connect to the same Zookeeper server through same port. The configuration file of Zookeeper is shown below: the variable ‘*tickTime*’ defines minimum time unit of Zookeeper, which is 2000ms in this project. The client port has been set to 2181, which means all Kafka brokers will connect to Zookeeper server through 2181 port. Variable ‘*initLimit*’ and ‘*synLimit*’ has been changed to 8 and 4 from initial value 10 and 5, since 3 brokers and Zookeeper server is deployed at same machine and the message transmission between different part does not require time limit as much as default.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# a placeholder for real paths.
dataDir=/opt/zookeeper-3.7.0/data
# the port on which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to '0' to disable auto purge feature
autopurge.purgeInterval=1
```

Figure 5 Configuration file of Zookeeper

When the system is going to start Kafka, it should always use the command ‘*bin/zkServer.sh start*’ to start Zookeeper server first. Then after all Kafka brokers have started, go back to Zookeeper and use the command ‘*bin/zkCli.sh*’ to start Zookeeper client and monitor Kafka subscription system in command line. As picture shown below, the system is able to monitor status of brokers and Kafka topics that have been created in client. The function of Zookeeper is very useful since it allows users to monitor Kafka subscription system in macro level, which is especially important in bigger Kafka cluster.

```
[zk: localhost:2181(CONNECTED) 0] ls /
[admin, brokers, cluster, config, consumers, controller, controller_epoch, feature, isr_change_notification, latest_producer_id_block, log_dir_event_notification,
zookeeper]
[zk: localhost:2181(CONNECTED) 1] ls /brokers
[ids, seqid, topics]
[zk: localhost:2181(CONNECTED) 2] ls /brokers/ids
[0, 1, 2]
[zk: localhost:2181(CONNECTED) 3] ls /brokers/topics
[Apr01, Apr02, Feb28_1, Jan, Jan16, Jan17, Jan21, Jan25, Jan28, Jan28_1, Mar01, Mar03, __consumer_offsets]
[zk: localhost:2181(CONNECTED) 4] ls /brokers/seqid
[]
[zk: localhost:2181(CONNECTED) 5]
```

Figure 6 Monitor Kafka through Zookeeper client

3.2.2 Deployment and Configuration of Kafka broker cluster

There are mainly two ways to deploy Kafka broker cluster. One way is to deploy each Kafka broker in one single machine, which means different nodes does not share computer resource with each other. In this way, every node listens to new events through port 9092, which is the default port number settled in configuration file. However, this way of deployment is most suited for large amount of data in real production environment, especially for those technical companies which have special requirement for performance of Kafka nodes. However, this project aims to build up a data capture system for personal usage, so it takes another way of deployment: to deploy 3 Kafka brokers in same machine. In this way of deployment, different nodes listen to new events through different ports of the same machine. In this project, three

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

nodes will listen through port 9092, 9093 and 9094. In this project names for configuration files of three different nodes are '*server.properties*', '*server1.properties*' and '*server2.properties*'. Names for different nodes should be divided since each broker is a single entity.

The meaning of some parameter in configuration file should be given extra care: '*broker.id*' indicates id of each broker, which should be set up to a unique integer for each broker. Two parameters '*listeners*' and '*advertised.listeners*' should be separated although they have same format '*IP : port number*', the former one is used when Kafka cluster is set up by administrator and only service on the intranet can be used; the latter one will register on Zookeeper server in order to be found when receiving requests from outside network, it will be set up to same value as '*listeners*' by default; In general, '*listeners*' is responsible for intranet communication and '*advertised.listeners*' is responsible for extranet communication. There will be a section for Zookeeper connection in configuration file as well. In this project, this section is same for all three nodes.

```
##### Server Basics #####
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

##### Socket Server Settings #####
# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#   listeners = listener_name://host_name:port
#   EXAMPLE:
#   listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://192.168.116.135:9092

# Hostname and port the broker will advertise to producers and consumers. If not set,
# it uses the value for "listeners" if configured. Otherwise, it will use the value
# returned from java.net.InetAddress.getCanonicalHostName().
advertised.listeners=PLAINTEXT://192.168.116.135:9092
```

Figure 7 Part of configuration file of Kafka broker

After deployment and configuration, the system uses the command '*bin/kafka-server-start.sh -daemon config/serverx.properties*' to start single Kafka nodes. The parameter '*-daemon*' makes the Kafka node start-up information hidden. After starting all Kafka nodes, users can go to Zookeeper client to check status of Kafka nodes. Another way to check the status of Kafka nodes is to run '*jps*' command on terminal.

3.2.3 Creation of Kafka topics

In this project, topics will be created and named according to the date, for example topic created in April 10th will be named '*Apr10*'. Use the command '*bin/kafka-topics.sh --bootstrap-server IP:9092 IP:9093 IP:9094 --create --topic Apr10 --replication-factor 2 --partitions 1*' to create new topic Apr10. The parameter '*partitions*' refers to minimum storage

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

unit in Kafka system, each partition of a same topic contains part of the data of this topic. Each partition is a single log file, and new data records will be added to the tail of the log file. Each piece of data record in partition will be assigned a unique sequence number named offset. This project sets the value of partitions to 1 in consideration of the order of messages, since although message in one partition is ordered, the order of data records within a topic with more than one partitions cannot be guaranteed. Set '*partitions*' to 1 can promise the whole system is ordered. Default partition strategy is Round-robin.

parameter '*replication-factor*' is used to set up the number of replications for each topic. In this project the it has been set up to 2, which means every piece of data record in this topic will be stored twice in two different brokers. This way of data storage prevents information loss caused by the failure of a single node, since all data records have one replica in one of rest brokers. The mechanism ISR implemented between different replications has been introduced in 2.2.3. The picture below shows how data records will be synchronized within a Kafka cluster when '*partitions*' is set up to 2 and '*replication-factor*' is set up to 2.

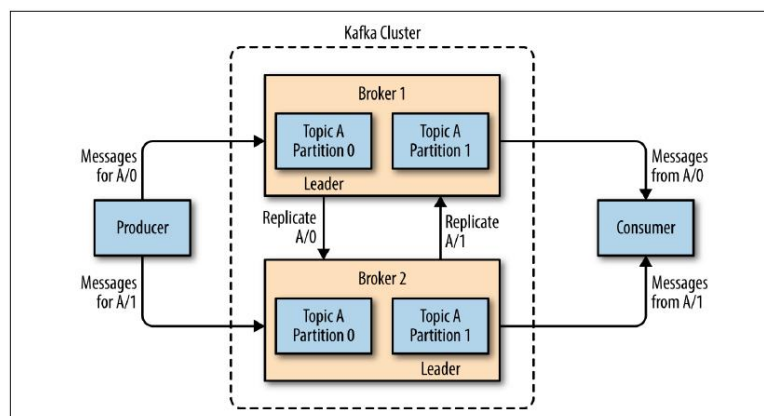


Figure 8 Topics of Kafka subscription system

3.2.4 Kafka producer

In this project, Kafka producer aims to send data records from local to Kafka broker cluster. The first step of constructing Kafka producer is to configure Kafka producer. Each Kafka producer can have its unique features. This is done by creating an object of '*ProducerConfig*' class and assign it to certain producer. There are several properties within this class that we need to give extra care. The first one is '*bootstrap.servers*', which stores all IP addresses of brokers in Kafka cluster. The variable '*batch_size*' controls the maximum size of each batch. Kafka producer compresses many pieces of data records which will be sent to the same partition to a batch. Each Kafka request is to send several batches, and each batch may contain multiple records. sending in batches in this way reduces network requests and helps

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

improve the performance of producer clients and Kafka cluster. The variable *'ack'* controls durability of data records that have been sent. When set up to 0, producer will push all records to buffer area without waiting for response from Kafka server. When set up to 1, producer will wait for acknowledgement from leader of a partition after the leader has written the records into log file. When set up to -1, producer will wait for acknowledgement after all nodes in ISR list have written the records into log file.

After constructing the properties of the producer, the producer then read in data records from csv file saved at local. The producer uses a *'BufferedReader'* object to read in new records since it allows producer to receive records with different length and to avoid unreadable code. Each line of records will be stored in a LinkedList. The reason why producer does not analyse and operate on data during reading process is that data processing can be better done by Logstash pipelines. What system cares most in this stage is the integrity of data records.

Then the producer sends data records to Kafka cluster. *'ProducerRecord'* is a core class in Kafka since each of its entity represents a group of key-value pair that producer is going to send. There are two traditional way to send data records, one way is to directly call method *'send()'* by producer which is also called simple message sending. However, one drawback of this method is that the producer cannot confirm whether the message has been sent successfully or not. Therefore, we use another way to send the message, which is also called synchronized message sending. Producer firstly calls method *'send()'*, and then calls method *'get()'* to wait for response from Kafka cluster (*'producer.send(producerRecord).get()'*). If no error occurs, producer obtains *'RecordMetadata'* which can be used to check data record.

In order to support data sending in real production environment, a producer that can read in new data records continuously is required. In real production environment, data capturing process always continues for a long time, so it will be necessary if the system can capture and send data records to Kafka cluster at the same time. Some variables should be given extra notice during the implementation of this kind of producer. The first variable *'lastFileLen'* records the place where next read should begin. Time interval between two read in process is set up to 5 seconds. The producer can take advantage of *'RandomAccessFile'* class since it implements input and output stream methods of most kinds of file. The method *'seek()'* can specify the position of the cursor, users can use it with variable *'lastFileLen'* to specify where producer start to send. If data records have been sent successfully, the producer will print hint on console.

3.3 Prometheus and Grafana

3.3.1 Reason for importing Prometheus and Grafana

The main reason for deploy Prometheus and Grafana is to monitor data transmission between Kafka producer and Kafka cluster. Although there will be hints printed on console after sending data records to Kafka cluster from producer, the system still needs better way to monitor Kafka system, since amount of data records can be really large, it would be impossible for users to manually check how many pieces of data records have been stored into the log file. What is more, even users can monitor Kafka cluster in Zookeeper client, it is still available only on terminal. System needs a more efficient and intuitive way to monitor Kafka cluster. From that perspective, this project imports the combination of Prometheus and Grafana which is a very common way to monitor distributed system.

3.3.2 Deployment of Prometheus

Prometheus is a time series database which can use HTTP protocol to collect and present time series data. Since Prometheus does not rely on extra memory space, one Prometheus node can support monitor job to whole system. From technical perspective, nodes monitored by Prometheus is called instance, one instance correspond to one thread. The set including several instances that have same objective is called job.

Exporter is a very important component of monitoring system based on Prometheus. In order to monitor another system, a 'Node exporter' is always deployed in order to expose metrics to Prometheus. More broadly speaking, every program that is able to provide sample data to Prometheus can be called an exporter. In this project, Kafka exporter is imported to be an agent between Kafka system and Prometheus and to expose Kafka cluster metrics to Prometheus. Use the command `./kafka_exporter --kafka.server=192.168.116.135:9092` to run Kafka exporter. The picture shown below indicates that Kafka exporter listens to Kafka server and exposes metrics of Kafka cluster through port 9308. In this way, Prometheus is able to obtain metrics of Kafka cluster through listening to port 9308.

```
[root@localhost kafka_exporter-1.2.0.linux-amd64] # ./kafka_exporter --kafka.server=192.168.116.135:9092
INFO[0000] Starting kafka exporter (version=1.2.0, branch=HEAD, revision=830660212e6c109e69dcblcb58f5159fe3b38903) source=kafka_exporter.go:474"
INFO[0000] Build context (go=go1.10.3, user=root@981cdel78ac4, date=20180707-14:34:48) source=kafka_exporter.go:475"
INFO[0000] Done Init Clients source=kafka_exporter.go:213"
INFO[0000] Listening on :9308 source=kafka_exporter.go:499"
```

Figure 9 Kafka exporter running

In order to enable Prometheus to obtain metrics from Kafka exporter, configuration file of

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Prometheus should be configured. The picture shown below is the configuration file of Prometheus. Content inside '*scrape_configs*' will be entity that monitored by Prometheus. The parameter '*target*' inside job '*kafka*' has been set up to '*localhost:9308*', which means Prometheus will obtain data records through port 9308, through which Kafka exporter has exposed metrics of Kafka cluster.

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'kafka'
    static_configs:
      - targets: ['localhost:9308']
```

Figure 10 Configuration of Prometheus

To see details of metrics collected by Prometheus, users can use the URL '*localhost:9090*' to direct to webpage of Prometheus. In webpage of Prometheus, users can obtain metrics of Kafka cluster in text format.

3.3.3 Deployment of Grafana

Grafana is a cross-platform data analysis tool based on Go language. It can request to and visualize collected data from multiple kinds of data source. Default port for visiting Grafana webpage is port 3000. Users are able to visit webpage of Grafana through URL '*IPaddress:3000*'. And default login id and password are both '*admin*'. After logging into the Grafana panel, users should configure Prometheus as the data source first. Then user can use Grafana to make multiple kinds of dashboards to visualize data collected from Kafka cluster.

The graph shown below is alive Kafka broker number. The horizontal axis represents time and vertical axis represents number of available nodes. Once we starting Grafana, it will constantly report on the number of alive Kafka brokers in cluster. The number of alive nodes increases to 3 once the system activates three Kafka brokers. Some people may doubt whether it is be necessary to monitor the node number of a system with only 3 nodes. The answer is positive since this functionality provides the foundation for system scalability. As the more nodes added to the system, for example one hundred nodes, it will be difficult for users to

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

manually count active nodes. Under such situation, a graph that shows number of active nodes at all time will be extremely useful.



Figure 11 Alive Kafka brokers' number

The second graph shown below is about '*go_memstats_alloc_bytes*'. Horizontal axis represents time and unit of vertical axis is bytes. This variable represents for Go memory statistics allocated bytes. It represents the status of memory usage of the machine where Kafka cluster deployed on. The picture shown below represents memory usage when data records is transmitted from producer to Kafka cluster. By observing the graph, the memory is allocated regularly in wavy shape, and there is an obvious upper bound for the amount of allocated memory which is about 3600000 bytes. Therefore, by setting an alert which will send an alarm when the memory allocated is out of the upper bound will be helpful to users to better monitor the metrics of the system.

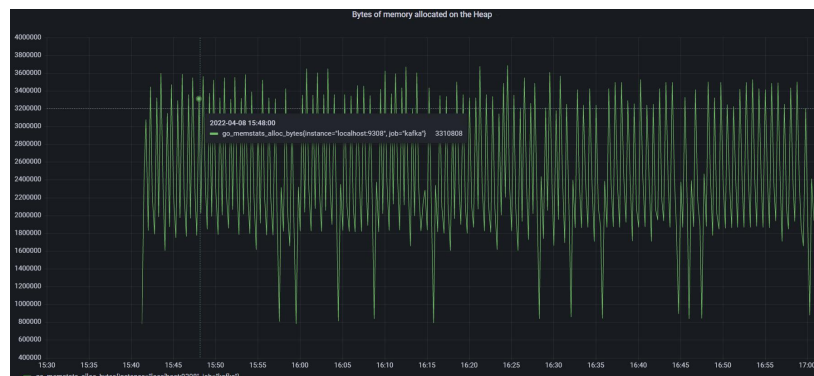


Figure 12 Memory usage

The third graph that Grafana presents is about the current offset of all topics. Horizontal axis represents time and vertical axis represents number. Kafka read in new data records ordinarily and write each of them one by one to partition logs. During that process, the concept of offset is introduced. Each data record in a partition will be assigned a sequence number representing its order, which we call offset. Offset is used to identify each piece of data record uniquely.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

There are two kinds of offsets: current offset and committed offset. The former one is stored in consumers' clients and stands for the place of the message where consumers start their next consuming process. Current offset is only used in '*poll()*' method of consumer, if consumer calls method '*poll()*' and receive 20 messages for example, current offset will be set up to 20. In this way when consumer calls '*poll()*' next time, it will start from message with sequence number 21. The introduction of current offset promises that consumer will receive new messages when it calls '*poll()*'. The latter one is recorded by brokers, which stands for order of messages that have surely been consumed by consumers. It is mainly used during the process of consumer rebalance. In Kafka consumer, each consumer belongs to a certain consumer group, and one consumer group contains one or more consumers. Consumers of the same group subscribe from a topic together, take the example of a situation where a consumer group with 20 consumers instances and subscribe to a topic with 100 partitions, normally Kafka will assign 5 partitions to each consumer, this process is called rebalance. Therefore, when a consumer is assigned a partition that has been consumed partially by another consumer of the same group, this consumer will start to consume new message from committed offset. The picture below shows current offset of all topics. It will be an efficient way for users to monitor the whether messages have been consumed successfully.

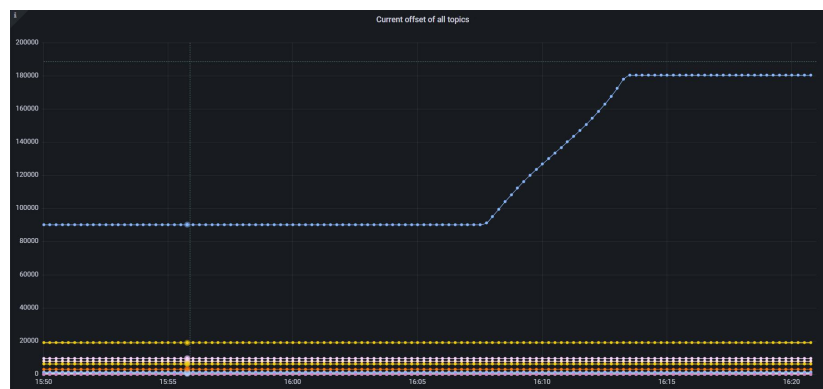


Figure 13 Current offset of all topics

Besides, users are also able to monitor single topic in Grafana. The picture shown below represents monitor to topic Apr08. The blue line represents current offset of the partition '0'. The green line represents current offset of the consumer group, there is an obvious latency between the update of single consumer offset and consumer group offset. The yellow line stands for consumer group lag, its number represents the extent to which consumers currently lag behind producers. The value of consumer group current offset and consumer group lag is always negatively related. This precise monitoring of individual topics will help make the system more scalable, since as more topics created it will be difficult to monitor metrics of all

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

topics in one single graph.

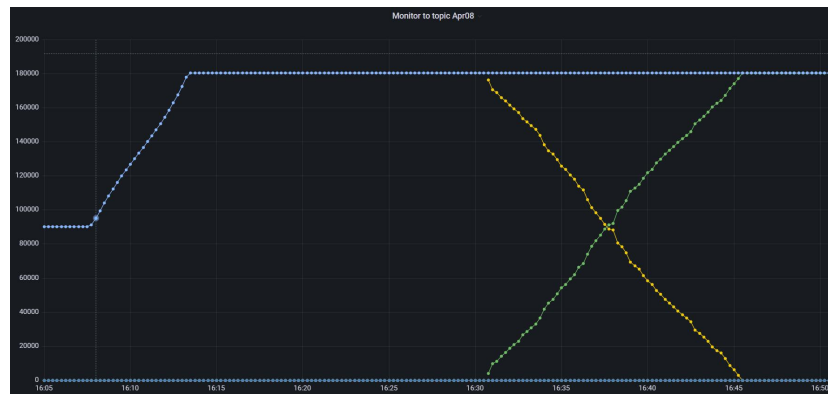


Figure 14 Monitor to topic Apr08

3.4 Logstash

Logstash is a part of ELK stack, it will be used to subscribe from Kafka, to process on data records and to transmit data record to Elasticsearch.

3.4.1 Simplest Logstash pipelines

Each pipeline of Logstash is an independent unique. Therefore, after downloading and deploying Logstash to machine, I need to create different Logstash pipelines that are able to process on data records differently. Every Logstash pipeline consists of three basic parts: input plugin, filter plugin and output plugin. Let us take the simplest Logstash pipeline for example, Input plugin is responsible for receiving data records from multiple data sources. *'bootstrap_servers'* contains IP addresses of all nodes in data source.

```
input {
  kafka {
    bootstrap_servers => "192.168.116.135:9092,192.168.116.135:9093,192.168.116.135:9094"
    group_id => "logstash1"
    client_id => "logstash1"
    auto_offset_reset => "earliest"
    topics => ["Apr08"]
  }
}
```

Figure 15 Input plugin

Filter plugin is responsible for processing data record. The filter of the simplest pipeline is shown below, it will transfer the format of value in column1, column 2 and column 6 of every piece of data record into appropriate format and rename the features, then it will remove unnecessary data.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

```
filter {
  csv {
    convert => {
      "column1" => "integer"
      "column2" => "date_time"
      "column6" => "integer"
    }
    columns => ["No.", "Time", "Source", "Destination", "Protocol", "Length", "Info"]
  }
  mutate {
    remove_field => ["@version", "@timestamp", "host", "type", "message"]
  }
}
```

Figure 16 Filter plugin

The output plugin is responsible for sending data records to Elasticsearch through port 9200.

```
output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "topic-test"
    user => "es"
    password => "guo200002"
  }
  stdout{ }
}
```

Figure 17 Output plugin

In this project, different kinds of Logstash pipelines have been developed. For example, users are able to add some comments on a certain group of data records. Every time when the system needs additional functionalities, we can encapsulate the Logstash pipeline for new functionalities into a new conf format file. Users use the command `'logstash -f ../logstash-sample.conf'` to start a certain Logstash pipeline.

3.4.2 Logstash pipeline that supports adding comments

This Logstash pipeline allows users to add comments on transferred data as their will. The main difference between this pipeline and the one introduced in 3.4.1 is in filter plugin. This kind of Logstash pipelines enable users to add comment on their own preference, which will be useful in specified situation. For example, when one of the users has just searched information from a certain website like CSDN (a computer science technical forum) and has been continuously capturing data packets since he started to use computer, he can add comment 'captured when I search information from CSDN' by using this Logstash pipeline to subscribe from Kafka subscription system. By doing this, he can still recall the circumstances under which the data was captured by looking at comments many days later. This function is achieved by adding the mutate block to filter plugin. Each piece of data packets captured will be added a new feature called '*Comment*'. Input plugin and output plugin are both same with pipeline introduced in 3.4.1.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

```
mutate {  
  add_field => { "Comment" => "This is the default comment." }  
}
```

Figure 18 Pipeline that support adding comments

3.4.3 Logstash pipeline that supports time form transformation

This Logstash pipeline will automatically transform '*Time*' feature to more readable format. Originally, '*Time*' feature will be accurate to microsecond. However, this kind of presentation can cause serious problem when Kibana classifies data records according to feature '*Time*'. Because the value of time is too precise, each piece of data record has a unique time value. Therefore, when users use Kibana to classify data records according to feature '*Time*', each piece of data record is divided into a separate group, which makes it impossible for users to count the amount of data collected over a period of time.

Therefore, for more efficient data classification in Kibana, we create another Logstash pipeline which extracts data from original one and generate more readable '*time*' feature. This pipeline extracts several time features from original data, including year, month, day and hour, this part is shown by figure 19. '*Time*' feature will be analysed and divided inside grok block. The value of four important units of time of each piece of data record will be extracted and assigned, and the remaining part will be discarded. Then it combines extracted feature to generate more readable '*Time*' feature for each piece of data record before sending data records to Elasticsearch. By doing this, the system is able to show when data was captured more concisely.

```
grok {  
  match => {"Time" => "(?<YEAR>\d*)-(?<MON>\d*)-(?<DAY>\d*)\s*(?<HOUR>\d*);\d*;\d*\d*"}  
}
```

Figure 19 Part of pipeline that supports time form transformation

3.4.4 Analysis of Logstash pipelines

An obvious drawback during the configuration of Logstash pipelines is that users have to modify the configuration file when subscribing from new topics, and users have to create new Logstash pipelines each time when they want to expand the functionalities of Logstash. This is because each Logstash pipeline is considered to be an independent entity that processes on data according to pre-configured configuration file. However, this feature also increases the scalability of Logstash pipelines as connection between data sources and destination. Because different functions are assigned to different pipelines. The failure of a single pipeline can never affect availability of other parts. This feature definitely makes Logstash pipelines more

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

maintainable.

3.5 Elasticsearch and Kibana

3.5.1 Deployment of Elasticsearch

Elasticsearch is a strong searching engine that is able to both search and analyse data records. After deploying Elasticsearch in my own machine, I need to create another user in Linux in order to start Elasticsearch. In output part of figure 20, the name of *'user'* is *'es'* instead of *'root'*.

In output part of figure 20, the content after *'index'* represents an index in index database. Data records subscribed from Kafka cluster will be stored in different index in Elasticsearch, the creation of index is just like the creation of database when users using MySQL or Oracle.

3.5.2 Data visualization

In this project, the system uses Kibana to visualize data collected by Elasticsearch. The default visiting port to Kibana is port 5601, users are able to login to webpage of Kibana with URL *'localhost:5601'*. Kibana provides graphs that present basic information of all data stored in Elasticsearch. The picture shown below is data records classified according to destination.

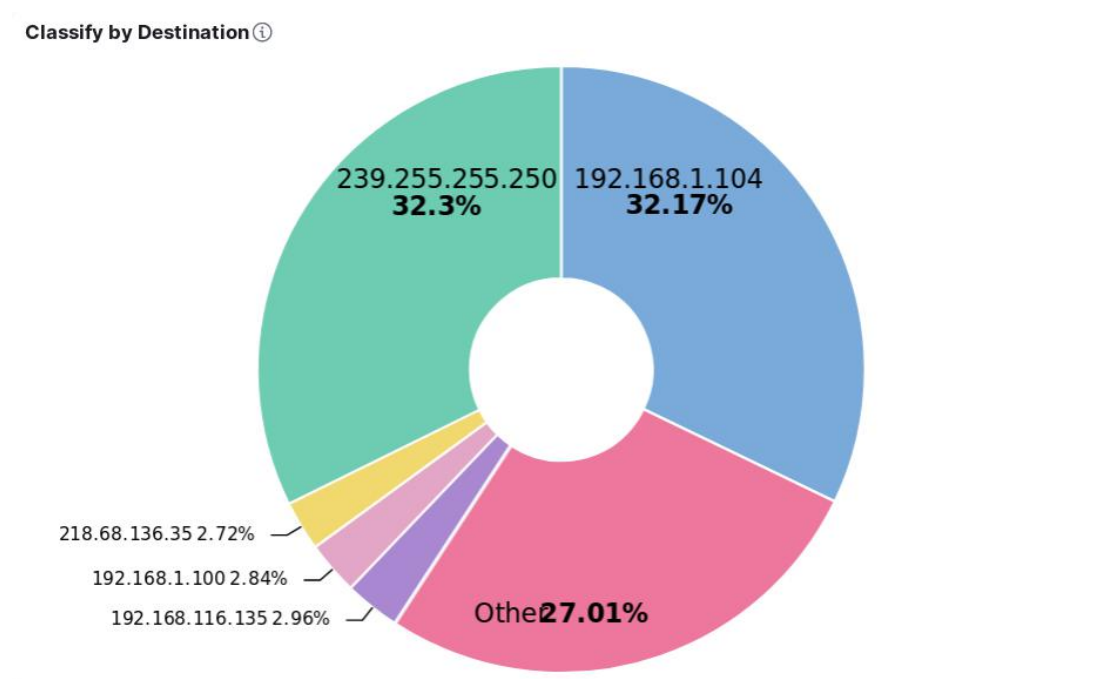


Figure 20 Kibana panel

Chapter 4: Results and Discussion

4.1 Result

In this chapter, the implementation of the system will be introduced. According to 5 objectives designed at the initial stage, all functions of the system have been successfully realized, and the whole system has been deployed successfully. The connection between different parts of the system has been tested many times which is proved to be qualified. According to design and test, both users in Linux and Windows are able to use this system to capture network data. The operation of the whole system is smooth, bringing users very good experience.

There is still one obvious defect in the performance of the system when all functions are fully realized: the occupation of computer resource. Because the system integrates many different software products, it can take up a significant amount of system memory. If users decide to deploy part of the system in cloud end, it also can cause extra spending.

Some changes have been made on the structure of final report compared with the one written in mid-term report, because more content have been added to the project since mid-term check.

4.2 Discussion

4.2.1 Possible way to deploy system in real production environment

In real production environment, different parts of system are always separated and deployed in different cloud end. Take the deployment of Kafka cluster for example, in real production environment different Kafka nodes should be deployed in different machines, and one Zookeeper node will be deployed with each Kafka node as well. Different Zookeeper nodes will form a Zookeeper cluster at first, and each Kafka brokers will register on local Zookeeper server and form Kafka cluster based on Zookeeper cluster. Communication between different nodes is done through opening access to certain port. One advantage of deploying Kafka cluster in real production environment is that computer resource will no longer be shared between nodes, which means that failure of single node is less likely to influence the performance of whole cluster. What is more, the latency of Kafka system can be measured if

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

nodes are deployed separately. Through measuring and analysing latency, developers are able to modify the configuration and make comparison to find out way of configuration that makes latency lowest.

4.2.2 Feedbacks from users

In the process of system development, there are two ways to establish an effective user feedback mechanism. The purpose is to provide adjustment opinions for the iterative development of system for the lack of products and the potential needs and characteristics of users.

1. Active search user feedback: System developers can use the search engine or microblog search function to actively search users' feedback about the system, reply in time and try to solve problems encountered by users.
2. Put the whole system in open-source website like Github and invite more users to join the project. By doing this, users are able to make their own comments on the whole system. what is more, talent users can also write their own code and add new functions to the system based on original project. This is a very good way to develop non-profit program.

Chapter 5: Conclusion and further Work

5.1 Conclusion

This project successfully developed a network data traffic flow monitor system with several main components connected. Each of the main modules of the system has realized the expected functions, including data capture part, Kafka subscription system, Logstash pipelines and data visualization part. Besides, extra function has been added in order to better monitor Kafka subscription system.

Among them, the construction of Kafka subscription system of the system is a highlight. The structure of Kafka subscription system in this project is very concise with all basic functions realized. Besides, the combination of Prometheus and Grafana extremely enhance user experience since it allows users to better monitor metrics of Kafka subscription system. The producer part of Kafka subscription is really easy for user to construct which increases scalability of the system since users are able to transfer captured data to Kafka by producer from remote.

In addition, the Logstash pipelines of the system have been constructed successfully. Not only it is able to subscribe data records from Kafka subscription system and transfer data records to Elasticsearch, but also it allows users to make multiple kinds of operation before data records transferred to Elasticsearch.

Functions of all parts of the system have been developed according to original plan and have been tested many times, and new functions have been continuously added during the whole developing process. Data transmission between different parts of the system has been tested many times and proved to be successfully. All tests are done in the virtual machine.

5.2 Future work

5.2.1 Cloud deployment

In real production environment, different components of a data monitoring system are always deployed separately for better performance. Another advantage of separating different part is that administrator of the system is able to measure latency of data transmission between different part in order to judge the performance. With all parts deployed at same machine, computing resource will be shared between different parts and the latency of data transmission is nearly equal to zero, which does not allow developers to analyse the performance of the system. Therefore, in the future a very important task is to deploy Kafka

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

producer (data capture part), Kafka broker cluster and Kafka consumer (ELK) in different cloud machine.

5.2.2 Support monitor to more kinds of data records

By now, the system can only provide data monitoring service to data records captured from local network with default features extracted by Tcpdump. In the future, the system should be able to monitor data records from more kinds of data sources, HTML webpage for example.

References

1. Xuesong, Tian. (2019). *Elastic Stack Application Book*. Beijing: China Machine Press. (ISBN 9787111634447)
2. Zhonghua, Zhu. (2019). *In-depth understanding to Kafka: Core design and practice principles*. Beijing: Electronic Industry Press. (ISBN 9787121359026)
3. Marios, I. (2020). *Tcpdump filters*. Retrieved Jan 22, 2022, from Tcpdump & Libpcap official Web site: <https://www.tcpdump.org/>
4. Julia, E. (2021). *Let's learn tcpdump*. Retrieved Jan 23, 2022, from Tcpdump & Libpcap official Web site: <https://www.tcpdump.org/>
5. Daniel, M. (2021). *A tcpdump tutorial with examples*. Retrieved Jan 23, 2022 from Tcpdump & Libpcap official Web site: <https://www.tcpdump.org/>
6. Ban, D. (2020). *Kafka: Comprehension about ISR Mechanism*. Retrieved Feb 10, 2022 from CSDN Web site: https://blog.csdn.net/daima_caigou/article/details/109390705?spm=1001.2014.3001.5506
7. Xin, B. (2020). *Learn Kafka through this article*. Retrieved Feb 11, 2022 from CSDN Web site: <https://blog.csdn.net/cao1315020626/article/details/112590786?spm=1101.2014.3001.5506>
8. Jian, D. (2020). *Kafka cluster monitor through Prometheus & Grafana*. Retrieved Feb 22, 2022 from CSDN Web site: https://blog.csdn.net/qq_34864753/article/details/103953385?spm=1001.2014.3001.5506
9. Ge, W. (2020). *Deployment of Prometheus & Grafana*. Retrieved Feb 27, 2022 from CSDN Web site: <https://blog.csdn.net/finghting321/article/details/107832698?spm=1001.2014.3001.5506>
10. Kai, G. (2018). *Big data search and log mining and visualization scheme ELK Stack: Elasticsearch, Logstash and Kibana*. Beijing: Tsinghua University Press. (ISBN 9787302433286)
11. Yao, A. (2018). *Introduction, deployment and usage of Kibana*. Retrieved Mar 11, 2022 from CSDN Web site: https://blog.csdn.net/qq_18769269/article/details/80843810?spm=1001.2014.3001.5506

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

12. Shi, J. (2021). *Network analysis tool: Usage of Wireshark*. Retrieved Mar 17 from CSDN

Web site:

<https://blog.csdn.net/zzwwhhpp/article/details/113077747?spm=1001.2014.3001.5506>

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Acknowledgement

I would like to thank my supervisor for her supervision and support throughout this past two semesters. My supervisor continuously offers me useful instruction about the development of the whole system, which benefits me a lot. I also like to thank one of my classmates, who had discussed the structure of this project with me before I started to develop the system, some of his ideas gives me new perspective to design the whole system. Finally, I would like to thank my family for their support to me throughout the past four years.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Appendix

Specification, part1

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 1 – Supervisor

论文题目 Project Title	Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity		
题目分类 Scope	Software Development	Research	Software
主要内容 Project description	<p>This project is focused on the design and implementation of an architecture oriented to monitor network traffic data by means of distributed light probes. The light probes must be able to:</p> <ul style="list-style-type: none"> • Capture network traffic data, mainly using tools such as tcpdump/tshark. • Transmit logs to central storage unit (assuming it is cloud hosted). Main technologies: ELK, logstash, Kafka. • Be remotely controlled by a remote controller software (to be developed also as a task of the project). <p>The data transmitted to the central shall be correctly received (by tuning ELK server parameters) and visualized (Kibana dashboards).</p> <p>A preliminary PoC architecture implementing capturer software, logstash and ELK can be implemented in order to explore preliminary results (basic implementation is already done by Ceit), but this project also requires to explore implementations and benefits of more complex architectures involving suitable software, such as Kafka.</p>		
关键词 Keywords	network traffic, big data, cybersecurity, elk, data analysis		
主要任务 Main tasks	1 Design and Implementation of preliminary PoC. Implementation of: simple capturer software, logstash and ELK interaction. Design&Implementation report and corresponding software as outcome.		
	2 Analysis of state of the art of enhanced architecture involving software such as Kafka. Design&Implementation report as outcome.		
	3 Implementation of a PoC of the enhanced architecture with Kafka. Implementation software as outcome.		
	4 Implementation of controller software for commanding light probes monitoring actions. Implementation software as outcome		
主要成果 Measurable outcomes	1 Task 1: Design&Implementation report and corresponding software as outcome.		
	2 Task 2: Design&Implementation report as outcome. Implementation software as outcome.		
	3 Task 3: Implementation software as outcome.		

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Specification, part 2

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 2 - Student

学院 School	International School	专业 Programme	Internet of Things Engineering		
姓 Family name	Guo	名 First Name	Mudong		
BUPT 学号 BUPT number	2018213067	QM 学号 QM number	190018801	班级 Class	2018215117
论文题目 Project Title	Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity				
论文概述 Project outline Write about 500-800 words Please refer to Project Student Handbook section 3.2	<p>The system in my project is about the capture, analysis, transmission and receive about network traffic flow. A developed software is required which the system will be able to use to grab network traffic flow locally with Tcpdump or Tshark. The use of Tcpdump or Tshark is integrated with Logstash, and by using the software, the system should also be able to make operation on captured traffic data with the help of Logstash. To increase the efficiency, Kafka will be used between local data logs and Logstash; besides, the system should be able to communicate with the server end through internet and transmit the logs of traffic data logs to server end; at the server end, the logs should be transmitted successfully and the system should be able to achieve the storage and visualization of transmitted logs with Elasticsearch and Kibana. The user could use dashboard of Kibana for further analysis.</p> <p>In the following, the main technological process will be described.</p> <ol style="list-style-type: none"> 1. The system will collect the network data by using Tcpdump/ Tshark. When network traffic data captured successfully and stored to local; After the storage, the system will be able to use the software to make operation on collected data with the help of Logstash. 2. To increase efficiency, Kafka is introduced between local collected logs and Logstash. During this process, local collected logs will be published to Kafka, Logstash will play the role of consumer which is able to subscribe specific logs from Kafka. 3. At the server end, Elasticsearch and Kibana will be deployed. Elasticsearch is an open-source distributed searching engine which support fuzzy inquiry, we could use it to achieve the obtain of specific logs. Kibana is a free user interface which allows the visualization of Elasticsearch and enables users to navigate through Elastic Stack. The data that transmitted to central will be correctly received by tuning ELK server parameters and visualized by Kibana dashboard. 4. A software is required to enable the control of the configuration of the capturer at the end-device. This design shall involve an API design(no graphical GUI required) to control the capturer end-device. <p>To sum up, the techniques that will be employed in this project include Wireshark, Kafka and ELK (ElasticSearch, Kibana and Logstash).</p> <p>During my program, Java and Python will be mainly used as the programming language since its simplicity and availability in object-oriented programming which will be helpful when constructing structured logs object. At the same time, Java is well compatible with Logstash and Kafka. In aspects of the</p>				

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

	database, beside the cloud database provided by Kafka and ElasticSearch, a local database is required to save raw logs collected by Wireshark. No special software packages or hardware is required as things stand.
道德规范 Ethics	Please confirm that you have discussed ethical issues with your Supervisor using the ethics checklist (Project Handbook Appendix 1). [YES]
	Summary of ethical issues: (put N/A if not applicable) N/A
中期目标 Mid-term target. It must be tangible outcomes, E.g. software, hardware or simulation. It will be assessed at the mid-term oral.	<ol style="list-style-type: none"> 1. Task1.2 Successful storage of data captured by using Tcpdump/ Tshark 2. Task2.3 Workable manipulation of logs by using Logstash. 3. Task3.3 Successful publication of local stored logs to Kafka and successful subscription of logs from Logstash. 4. Task4.2 Transmission of manipulated logs to cloud end. 5. Task4.3 Receive logs at server end, use Kibana to obtain visualized analysis of data. 6. System that implements functions mentioned above.

Work Plan (Gantt Chart)

Fill in the sub-tasks and insert a letter X in the cells to show the extent of each task

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

	Nov 1-15	Nov 16-30	Dec 1-15	Dec 16-31	Jan 1-15	Jan 16-31	Feb 1-15	Feb 16-28	Mar 1-15	Mar 16-31	Apr 1-15	Apr 16-30
Task 1 [Capture and analysis of network traffic dataflow using Tcpdump/ Tshark]												
1.1 Learn basic skills required to use Tcpdump/ Tshark.	X	X										
1.2 Successful storage of data captured by using Tcpdump/ Tshark.		X	X									
1.3 Analysis of traffic data captured from Internet.		X	X									
1.4 Development of data capture part of the system		X	X									
Task 2 [Learn ELK then operate and transmit local data using Logstash]												
2.1 Learn basic knowledge and skills required to build ELK.			X	X								
2.2 Use Logstash of ELK to receive, operate and transmit logs.			X	X	X							
2.3 Workable manipulation of logs with the help of Logstash			X	X	X							
2.4 Development of the data capture part which is integrated with Logstash of the system.			X	X	X							
Task 3 [Use Kafka for publication/ subscription of logs between local and Logstash]												
3.1 Learn basic knowledge and skills of usage of Kafka.						X	X					
3.2 Deploy Kafka between local stored data and Logstash.						X	X					
3.3 Successful publication of local stored logs to Kafka and successful subscription of logs from Logstash.						X	X	X				
3.4 Development of the part of the system that work with Kafka between local logs and Logstash.						X	X	X				
Task 4 [Use ElasticSearch and Kibana at server end for receive and visualization of logs]												
4.1 Learn basic knowledge of ELK and deploy Kibana and ElasticSearch.								X	X			
4.2 Transmission of manipulated logs to cloud end.								X	X	X		
4.3 Receive logs at server end, use Kibana to obtain visualized analysis of data.								X	X	X		
4.4 Design and implement a software that enable the control of the configuration of the capturer at the end-device. Test the system with different configuration until the system will work in most circumstance.								X	X	X	X	X

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Early-term Report

北京邮电大学 本科毕业设计（论文）初期进度报告

Project Early-term Progress Report

学院 School	International School	专业 Programme	Internet of Things Engineering		
姓 Family name	Guo	名 First Name	Mudong		
BUPT 学号 BUPT number	2018213067	QM 学号 QM number	190018801	班级 Class	2018215117
论文题目 Project Title	Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity				

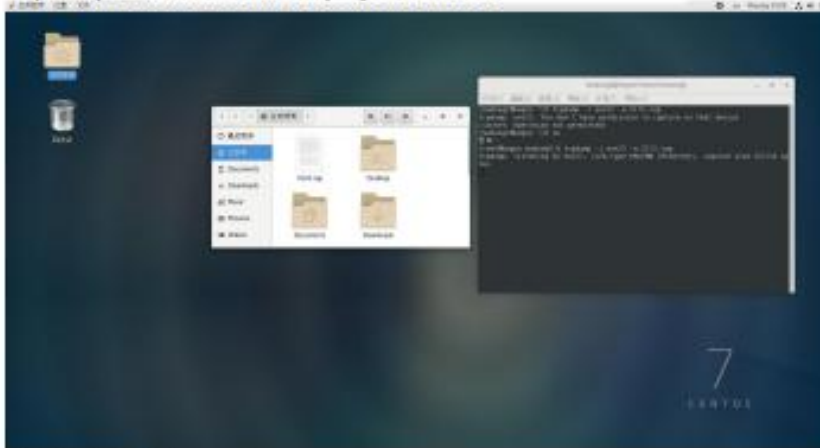
已完成工作 Finished work:

1. Literature Review

- Document of Tcpdump read by December
- Document of Kafka read in January

2. Current Progress

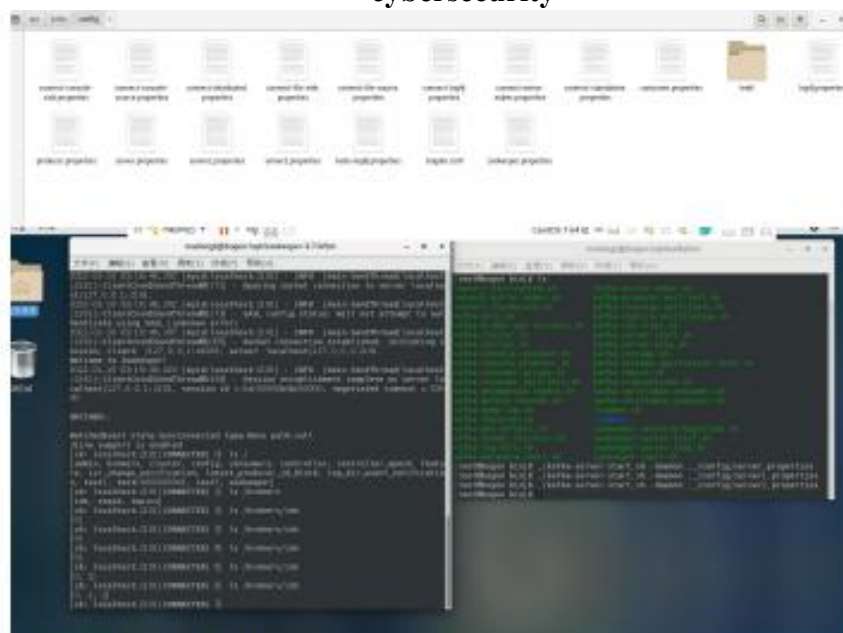
- Tcpdump downloaded and deployed to VMM



- Network traffic captured successfully by using Tcpdump and saved as .cap file for further analysis
- .cap files that store network traffic data have been analysed and transferred to .csv file by Wireshark for further operation.

- Basic knowledge of deployment of Kafka has been learned.
- Kafka cluster has been successfully deployed in VMM with the support of Zookeeper. 3 brokers will be used in Kafka cluster for storage of data.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity



3. Problems and proposed solution

a. How to analyse .cap files captured by Tcpdump?

Solution: All .cap file captured by Tcpdump will be transferred to Windows OS and analysed with the help of Wireshark for better visualization. All .cap file will be transferred to .csv file with integrated tools in Wireshark for better performance in further data processing. All .csv files will be transferred back to Linux OS with the help of WinSCP.

b. What kind of network data would be captured by Tcpdump?

Solution: All network data would be listened, captured and saved to .cap file by using Tcpdump. All network traffic data would then be transferred to the subscription system implemented under Kafka frame for further analysis. Users would subscribe for specific data on their own demand. Subscribed network traffic data would be transferred to Logstash through a message queue.

c. Which version of Kafka would be deployed?

Solution: the 3.0.0 version of Kafka released in Sep, 2021 would be deployed. The reason is that in this version the deployment of Kafka is independent from Zookeeper. In new version, complexity of the system would be decreased and efficiency of the system would be increased.

d. How to determine the configuration of Kafka cluster?

So far, the Kafka cluster has 3 brokers. The settings of topics still requires further discussion.

是否符合进度? On schedule as per GANTT chart?

[YES/NO]

YES

下一步 Next steps:

1. Deploy Kafka in VMM with all basic functions nicely designed and implemented. This step would be finished in January. The configuration of producer, consumer and brokers still requires further discussion.
2. Design an API between .csv file that stores network traffic data and Kafka. This step would be

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

finished in January. The form of transformed network traffic data still requires further discussion. All transformed data would be seen as content that published to Kafka cluster by administrator.

3. Study of basic knowledge of Logstash, try to deploy Logstash to VMM and design API between Kafka and Logstash as soon as possible.
4. Design of API between Kafka consumer and Logstash. All subscribed content would be consumed by consumer in Kafka cluster and transferred to Logstash for further operation. Those content that has passed through Logstash would be then transferred to ES for higher priority and more clear analysis.

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Mid-term Progress Report

北京邮电大学 本科毕业设计（论文）中期进度报告

Project Mid-term Progress Report

学院 School	International School	专业 Programme	Internet of Things Engineering		
姓 Family name	Guo	名 First Name	Mudong		
BUPT 学号 BUPT number	2018213067	QM 学号 QM number	190018801	班级 Class	2018215117
论文题目 Project Title	Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity				
是否完成任务书中所定的中期目标? Targets met (as set in the Specification)? [YES/NO] YES					
已完成工作 Finished work: <ol style="list-style-type: none"> All points included in Task 1 of specification have been finished. The detailed progress of Task1 include: <ol style="list-style-type: none"> Tcpdump has been deployed in CentOS virtual machine and basic skills that needed to use Tcpdump have been mastered. Data log packets have been successfully captured by Tcpdump. All data packets captured at one time will be stored in one single cap format file. Every packet has 7 features including package identifier, time, source, destination, protocol, length and information. To achieve simple and efficient analysis of the captured data, WinSCP is deployed in local machine to provide transmission of captured data between virtual machine and local computer. We use WinSCP to transfer cap format file from virtual machine to local machine. Wireshark has been deployed in local machine which could provide clearer presentation of captured data stored in cap file and transfer cap file to CSV format. Data stored in CSV format file is easier to analyse. All points included in Task 3 of specification have been finished. Some additional work has been added in order to obtain better monitoring to Kafka subscription system. The detailed progress of Task 3 include: <ol style="list-style-type: none"> Kafka system has been deployed in virtual machine and basic skills that needed to build and improve Kafka system have been mastered. A cluster that includes 3 brokers have been deployed inside the Kafka system which receive data from producer, store logs and put forward logs to consumer. Different brokers listen to new events through different ports in one single virtual machine. Topics are created according to date of captured data. For example, data packet that captured on February 28th would be published to a topic named Feb28. The creation of topics is done manually on terminal. Producer of Kafka system has been successfully developed. Producer receives CSV file that locally stored in virtual machine according to topic and publish it to brokers. During the transmission, producer would not manipulate the content of transferred data. Consumer of Kafka system is Logstash which will be described later. <p>The detailed progress of additional work include:</p> <ol style="list-style-type: none"> Successful deployment of zookeeper. Zookeeper is able to monitor Kafka subscription system in terminal through zookeeper client. Successful deployment of Prometheus. Prometheus could be used to monitor Kafka subscription system with the help of kafka_exporter which, is an open-source tool that could be downloaded on website and used to export Kafka system to other monitoring tools. With the help of kafka exporter, Prometheus is able to present metrics of Kafka 					

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

<p>subscription system on web page through port 9308.</p> <p>3) Successful deployment Grafana. Grafana could be used to visualize the data metric that presented by Prometheus, it provided strong and clear visualization which supports better analysis of Kafka system.</p> <p>3. All points included in Task 2 of specification have been finished. The detailed progress of Task 2 include:</p> <p>1) Successful deployment of Logstash, all basic skills related to Logstash that needed in this project is mastered.</p> <p>2) Logstash tune has been successfully configured which plays the role of Consumer of Kafka subscription system. Basic Logstash tune will subscribe from Kafka subscription system according to topic name, transcribe the data to readable format and transfer the data to ElasticSearch for further analysis.</p> <p>4. Most points included in Task 4 of specification have been finished. The detailed progress of Task 4 include:</p> <p>1) Successful deployment of ElasticSearch in local virtual machine.</p> <p>2) Data operated by Logstash has been successfully received by ElasticSearch.</p> <p>3) Data received by ElasticSearch has been successfully shown by Kibana. Kibana could provide strong and clear visualization of data stored in ElasticSearch, all kinds of graphic presenting tools are included in Kibana.</p> <p>4) Logstash, ElasticSearch and Kibana have been deployed in cloud virtual machine.</p>
<p>尚需完成的任务 Work to do:</p> <p>1. Configuring more Logstash tunes which could provide more options in operating captured data forwarded by Kafka.</p> <p>2. Adding more panels in Kibana which could provide more options in visualization of data stored in ElasticSearch.</p> <p>3. Trying to separate Kafka producer (Tcpdump) and Kafka brokers. This could better emulate system that working in real production environment.</p> <p>4. Test a real time scenario to measure the behaviour of the system designed. For that purpose, 3 VM, or preferably AWS instances should be deployed.</p> <p>5. Measure latency and throughput of data transmission between different parts of the system.</p> <p>6. Trying to use environment variable to replace topic name in input plugin of Logstash. This could improve performance of the system.</p> <p>7. Properly document the project in the created Github repository, including next sections: name, contributors, description, architecture, requirements, getting started, how to use, troubleshooting and to do.</p> <p>8. Depending on the results, in terms of real-time deployment and latency measurements to be performed, the publication of a conference paper could be considered.</p>
<p>存在问题 Problems:</p> <p>1. All parts are deployed in a single virtual machine which could result in insufficient resource. Besides, since all part of the system are deployed in one machine, it is difficult for administrator to measure latency between different part since all latency are nearly equal to zero. This is not good since it does not similar to real production environment where most data monitoring system work in.</p> <p>2. More panels should be created to provide abundant visualization of data stored in ES.</p> <p>3. No automatic way to configure input-plugin of Logstash, which means that user has to configure Logstash tunes manually every time.</p>
<p>拟采取的办法 Solutions:</p> <p>1. Rent a cloud server or use the student account of the AWS service which I could deploy ELK part of the system in cloud server to separate different parts of the system. Besides, this could allow the system emulating real working environment where latency between different part exists. Otherwise, the latency between different parts of the system will be nearly to zero if all</p>

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

<p>parts were deployed in single machine.</p> <ol style="list-style-type: none"> Continue to learn skills required to creating Kibana panels, so that more effective panels could be created to monitor data stored in ES. Think of ways to configure one environment variable that represents all unique input plugin in Logstash. This could efficiently improve user experience since user does not have to adding new input plugins when new topics were created.
<p>论文结构 Structure of the final report:</p> <p>Abstract</p> <ol style="list-style-type: none"> Introduction <ol style="list-style-type: none"> Problem statement Define problem solved by the project Data source used Description of legal data used in creating this project and summaries of the data sets Background <ol style="list-style-type: none"> Analysis of data monitoring under cyber security Summary of the basic research and system in the area of data monitoring Data Indexing techniques Summary of data indexing techniques used in this project Data monitoring techniques Summary of data monitoring techniques used in this project System design <ol style="list-style-type: none"> Analysis of structure Describe the basic working mode of the system and describe the connection between different parts of the system Analysis of performance Analyse the performance of the system Simulation of data monitoring system System functions <ol style="list-style-type: none"> Description of system functions Describe basic functions of the system Data visualization realization Describe data visualization/monitoring in all level of the system Conclusions <ol style="list-style-type: none"> Describe major result from the project summarising sections Future work Short section describing other things that I might do <p>Bibliography</p> <p>Structured list of all the many papers I used.</p> <p>The format should be as below:</p> <p>[1] R.G. Clegg, J. Smith, Nature Scientific Reports "Richard's Great Paper" pages 200-210(2021)</p> <p>[2] R.G. Clegg, J. Smith, Nature Scientific Reports "Richard's Other Paper" pages 200-210(2021)</p> <p>...</p>

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Supervision Log

北京邮电大学 本科毕业设计（论文）教师指导记录表

Project Supervision Log

学院 School	International School	专业 Programme	Internet of Things Engineering		
姓 Family name	Guo	名 First Name	Mudong		
BUPT 学号 BUPT number	2018213067	QM 学号 QM number	190018801	班级 Class	2018215117
论文题目 Project Title	Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity				
<p>Please record supervision log using the format below:</p> <p>Date: dd-mm-yyyy</p> <p>Supervision type: face-to-face meeting/online meeting/email/other (please specify)</p> <p>Summary:</p>					
<p>[These are examples, please DELETE them and add yours. DELETE this line.]</p> <p>Date: 13-10-2021 Supervision type: email Summary: Inform the start of kick-off meeting</p> <p>Date: 15-11-2021 Supervision type: First project progress face-to-face meeting Summary: Discussed the specification of the project</p> <p>Date: 17-11-2021 Supervision type: email Summary: Instruction about the details written in specification</p> <p>Date: 13-12-2021 Supervision type: Second project progress face-to-face meeting Summary: Discussed the progress that has been achieved between first group meeting and second group meeting</p> <p>Date: 7-1-2022 Supervision type: email Summary: Discussed the details that has been written in early term report</p> <p>Date: 23-1-2022 Supervision type: online meeting Summary: Discussed progress achieved between January 17 – January 21</p> <p>Date: 28-1-2022 Supervision type: online meeting Summary: Discussed the progress that has been achieved in January</p> <p>Date: 7-2-2022 Supervision type: online meeting Summary: Discussed progress achieved between January 31 – February 4</p>					

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Date: 14-2-2022

Supervision type: online meeting

Summary: Discussed progress achieved between February 4 – February 11

Date: 21-2-2022

Supervision type: online meeting

Summary: Discussed progress achieved between February 14 – February 18

Date: 27-2-2022

Supervision type: online meeting

Summary: Discussed progress achieved between February 21 – February 25

Date: 27-2-2022

Supervision type: email

Summary: Discussed the details that has been written in midterm report

Date: 28-2-2022

Supervision type: online-meeting

Summary: Discussed the progress that has been achieved in February

Design and Implementation of data monitoring probes for network traffic analysis in cybersecurity

Risk and environmental impact assessment

Description of Risk	Description of Impact	Likelihood Rating	Impact Rating	Preventative Actions
Privacy disclosure of local network during data capture	During data capture process, users may capture packets that goes through public WLAN.	1	2	Suggest users to capture data packets from their private network.
User's computer is out of memory	Since this project indicates users to deploy Tcpcap, Kafka and ELK in their computer, computer's memory may be not sufficient to run whole system.	2	3	Suggest users to deploy ELK in cloud machine, since Elasticsearch and Kibana have minimum requirement for memory.
Data leak in Kafka subscription system	Un authorized users may successfully subscribed data from Kafka subscription system.	1	4	Suggest users to keep information related to Kafka secret.
Hardware damage	Project loss.	1	5	Backup the whole system in cloud like Github.