Collaborative Detection of Cyber Security Threats in Big Data

Jiange Zhang, Yuanbo Guo, and Yue Chen
State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou Information Science and Technology Institute, China

Abstract: *In the era of big data, it is a problem to be solved for promoting the healthy development of the Internet and the Internet+, protecting the information security of individuals, institutions and countries. Hence, this paper constructs a collaborative detection system of cyber security threats in big data. Firstly, it describes the log collection model of Flume, the data cache of Kafka, and the data process of Esper; then it designs one-to-many log collection, consistent data cache, Complex Event Processing (CEP) data process using event query and event pattern matching; finally, it tests on the datasets and analyzes the results from six aspects. The results demonstrate that the system has good reliability, high efficiency and accurate detection results; moreover, the system has the advantages of low cost and flexible operation.*

Keywords: *Big data, cyber security, threat, collaborative detection.*

Received July 20, 2016; accepted February 15, 2017

1. Introduction

In recent years, as the development of the Internet, especially the rapid development of Internet+ such as electronic commerce, Internet Finance (ITFIN), Internet industry, wireless networks [20], etc., more and more attackers appear, thus imposing more threats on the cyber security. It shows in the report about the Internet Development Security in the Third Quarter of 2015, published by Akamai, that the index of cyber security threat has continued to rise, and various data leakage and cyber attacks have continued to emerge, particularly Distributed Denial of Service (DDoS) attacks increased by 180%, thus creating a historical record again! The report also shows the scale of DDoS attacks have continued to increase, and may paralyze the core router of network operators easily. Moreover, cyber attacks will cause serious and significant consequences such as loss of income, damage to reputation and information systems, and stealing of private data or customer sensitive data [8, 13].

Thus it can be seen that the rapid changes in the cyber security threats have gone beyond the current security protection in the industry, and the traditional protection measures, such as intrusion detection system, no longer work satisfactorily. Especially in the era of big data, there will be massive attack data generated at every moment, while the traditional detection systems when in dealing with these attacks seem to be overwhelmed. However, the big data technology is widely applied to other industries, and it is just begun to use in the security industry. Therefore, it will be a development trend to detect the cyber security threats by virtue of the big data technology.

For instance, Zuech *et al*. [22] introduced the specific issues of big heterogeneous data fusion, heterogeneous intrusion detection architectures, and Security Information and Event Management (SIEM) systems from many different heterogeneous sources.

At present, there are five typical models using the big data technology to process the data: a centralized online real-time analysis complex event processing model which is called Esper [2, 12] proposed by Luna [17]; a distributive and batch processing model which is called Hadoop-proposed by Cutting [9]; a distributive and real-time analysis model which is called Agilis [1, 7] proposed by Aniello *et al*. [1]; a distributive fault-tolerant online real-time analysis model which is called Storm [16] proposed by Aniello *et al.* [1]; a distributive multi-iteration batch processing model which is called Spark proposed by Armbrust *et al.* [6]. The emergence of these big data processing models provides a good technical support for the detection of cyber security threat. However, these models are only data processing models and cannot be used as a complete set of independent security threat detection system. Therefore, it is very necessary to construct a security threat detection system by utilizing the big data processing model.

Any attack on the Internet has a certain law, and it needs to use a certain technology to discover. Staheli *et al*. [21] proposed a collaborative investigation system-Cyber Analyst Real-Time Integrated Notebook Application (CARINA), using collaborative data analysis and discovery techniques to provide better decisions for network analysis. It believes in Axiom 6 in [19] that "A relationship always exists between the

Attacker and their Victim(s) even it is distant, fleeting, or indirect." Hence, a rule for detecting the attack is fabricated according to the relationship, and the relationship between the attacker and the victim is found out according to the attack rules. Moreover, Esper engine is a data processing model for processing the data according to the attack rules. Therefore, this paper constructs a collaborative detection system of cyber security threats in big data by using the Esper engine as the data processing model, the Flume as the reliable highly available massive log collection, aggregation and transmission and the Kafka as the data cache.

2. Related Works

2.1. Flume

Flume is a distributive, reliable and highly available system, provided by Apache [3], for efficiently collecting, aggregating, and transmitting massive log data. Its operation core is Agent, Agent is adopted as the minimum independent operation unit, and Event is used as a basic unit of data processing. Each Agent consists of three core components: Source, Channel and Sink.

As a data source, Source receives an event from the external resource, converts it to an Event of a specific format, and pushes the Event into a single or multiple Channels. It provides many methods to collect data from data sources, such as console, RPC, text, tail, syslog, exec, etc.

As a transmission channel, channel stores the event pushed by source until event has been handled by sink. It provides many channel types, such as memory channel, file channel, Java Data Base Connectivity (JDBC) channel, recoverable channel, etc.

As a data receiver, sink acquires the event from the channel, performs the data persistence for the event (for instance, the event is stored into file systems, databases, or submitted to the remote server) or pushes the event into another source. It also provides many data receiving modes, such as console, Remote Procedure Call (RPC), text, HDFS, syslog Transmission Control Protocol (TCP), etc.

Through the three core components, flume makes event flow to channel from source and then flow to sink. It is able to create multiple Agents to work together for customers, and provides three models: one-to-one, one-to-many, many-to-one. Moreover, for the same Agent, source and sink are asynchronous.

2.2. Kafka

Kafka is a high throughput and distributed publish-subscribe messaging system. It has the following characteristics: [4]

1. Fastness: a single Kafka node can handle hundreds of mega bytes per second from hundreds of thousands of customers.
2. Scalability: Kafka allows a single cluster to be used as the center data backbone of large institutions, and the expansion can be performed flexibly and transparently without the downtime. The data stream is divided and dispersed into the cluster, which makes the data flow larger than the capacity of any one machine and achieves the cluster coordination.
3. Persistence: in order to prevent the data loss, messages are stored in a disk and replicated in the cluster, and each node can handle one million Mega bytes while the performance is not affected.
4. Distribution: Kafka is a modern center cluster that provides strong robustness and fault tolerance.

The Kafka cluster consists of three components i.e., Producer, Consumer and Broker. Producer is a messager publisher, and it generates and pushes data to a Broker; Consumer is a message subscriber, and it pulls and then processes the data from a Broke; Broker is a message storage array, and the performance is enhanced because of not maintaining the data consumption state. In addition, the three components carry out the coordination of requests and forwarding through the Zookeeper [5, 14].

Zookeeper is a distributed coordination service of applications, which is based on the Fast Paxos [15]. It generates a leader through the election, and only the leader can submit the proposer.

2.3. Esper

Esper is a flow data processing method of Complex Event Processing (CEP), it has the following characteristics: [12, 17]

1. It uses memory database and processes a complex event through the query of the Event Processing Language (EPL). Compared with the traditional relational database, it has a better processing and query performance and is more suitable for processing the CEP.
2. Two mechanisms are provided to handle the event. The event mode matching based on the expression is implemented by a state machine. The event processing method is to match the event expected to exist, the absent event or the combination of events; and the query of the event flow is implemented through the EPL statement. The event processing method provides functions such as filter, slip window, aggregation, joint and analysis. The EPL adopts the view to put the constructed data into one event flow and to drive the flow of the data; and the data is processed during the data flow process to obtain a final needed result.
3. It provides many data processing methods: Plain Old Java Objects (POJOs), MAP, SOCKET, XML, etc.

4. It handles 500 thousand events per second, and the input flow rate reaches 70Mb/s.
5. When facing new threats, it can dynamically adjust the detection logic by integrating/deleting SQL query statements.
6. The overhead is low.

The pattern of Esper is a rule that is composed of atomic events and operators. The atomic events are customized events, and the pattern operators have 4 types [11]:

1. Repetition operators: every, every-distinct, [num] and until.
2. Logical operators: and, or, not.
3. Order operator: -> (followed by)
4. Event lifecycle operators: timer:within, timer:withinmax, while-expression.

The pattern operators are similar to other operators and also have the priority. According to the priority of the pattern operators [11], the pattern expression "a->b" refers to that if occurrence of an event b on the right side follows the occurrence of an event a on the left side, it enters an Esper engine. The pattern expression "every a->b" means that once an event B enters, all events A before the event B are matched, and if it succeeds in matching, it enters the Esper engine.

The atomic events are combined into different expressions by using the pattern operators so as to form different patterns, and the correlation between the events is explored according to the patterns.

3. Design of Collaborative Detection of Threat

In the collaborative threat detection architecture, the data processing is mainly divided into five stages: data source, data collection, data cache, data processing, and data storage. The data source is various network data packets transmitted on the network; the data collection is to simply handle and filter the network packets, and is used as the Producer of Kafka to generate data; the data cache is to coordinate the Broker by virtue of Zookeeper, so that the data collection speed is consistent with the data processing speed, and it provides the reliability of the data; the data processing is used as the Consumer of Kafka to consume the data through the EPL query and pattern of Esper Engine, and it triggers the listening for the event matching the pattern; the data storage is to persistently store the data in the stage of data collection and data processing, and to provide the data for off-line analysis and presentation. The integral model of the collaborative threat detection architecture is illustrated in Figure 1.

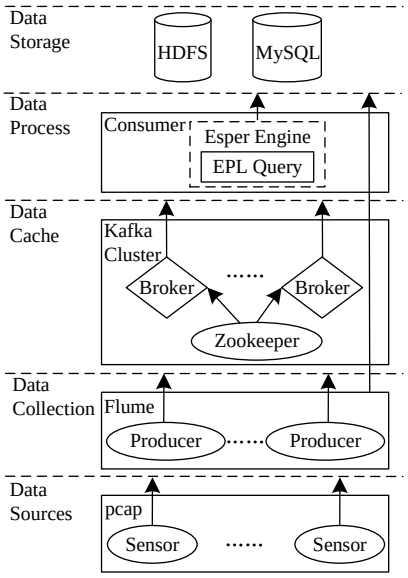

Figure 1. Collaborative threat detection architecture.

3.1. Design of Flume

In the collaborative threat detection architecture, the Flume adopts the one-to-many model to collect logs. The Source is SourceNet, which is a network packet. Each packet is used as one Event to be transmitted in two channels-- ChannelDFS and ChannelKafka. Naturally, there are two Sinks-- SinkDFS and SinkKafka. SinkDFS acquires data from ChannelDFS and stores it in HDFS to be used as a data source for off-line analysis. SinkKafka acquires the data from ChannelKafka and processes it in the Consumer component of Kafka to be used as the data source for real-time on-line analysis. The construction of log collection by using Flume is illustrated in Figure 2.

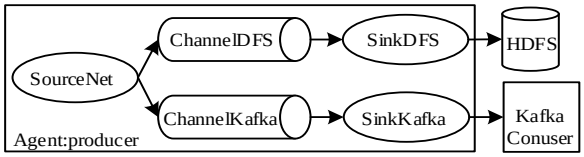

Figure 2. Construction of log collection.

3.2. Design of Kafka

In order to make the speed of data collection consistent with the speed of data processing, a message middleware Kafka is added as a buffer. Kafka manages Producer, Consumer and Broker through Zookeeper, and a plurality of Brokers collaborate to ensure the production and consumption of the data. Moreover, Zookeeper provides a consistent service to the distributed applications. One service or a pluralitiy of services can be deployed. The more services are deployed, the higher the reliability is. Without loss of the generality, this architecture is deployed with a Zookeeper service. The construction of data buffer using Kafka is shown in Figure 3.

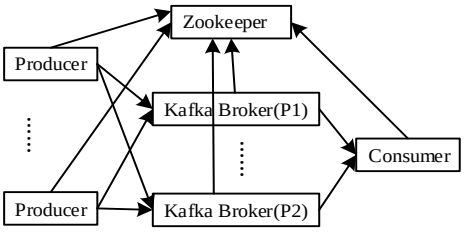

Figure 3. Construction of data buffer.

3.3. Design of Esper

The flow of processing and analyzing the event by using esper is illustrated in Figure 4 First, an Event class and a updatelistener interface are created. According to this class, a configuration instance is created. According to this instance, an EPServiceProvider object is obtained by means of EPServiceProviderManager. According to this object, the event is sent to a listening interface by using the sendEvent of getEPRuntime, meanwhile, by using the EPL query statement, a listening statement instance EPStatement is created and is added to the listening interface by virtue of the addListener.

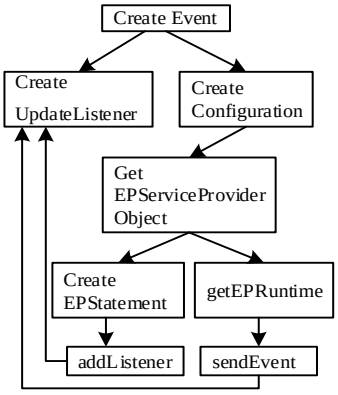

Figure 4. Flow of data process.

The main method for dealing with the event by using the Esper Engine is the event stream query and the event pattern matching.

The pattern matching supports the processing of the complex event with relevance. In the collaborative threat detection collaborative threat detection architecture, the atomic event of the pattern is defined as the pcap packet which is processed by Flume and Kafka, and expressed by a 12-tuple: packet (time, MAC source, MAC destination, IP source, IP destination, port source, port destination, protocol type, SYN, ACK, RST, traffic).

According to the three-handshake process of TCP, the event for the Client to request a data packet is formally defined as a, the event for the Server to confirm the data packet requested by the Client is formally defined as b, the event for the Client to confirm the data packet to the server is formally defined as d, and the event for resetting the data packet is formally defined as c. At the same time, the successful three-handshake is defined as the full-connection, and if a third handshake is not to confirm the data packet but to reset the data packet or has no suck information, it is defined as the incomplete connection or the half-open connection.

The pattern expression of the complete TCP three-handshake (full-connection) is: every a->(b->d)

If the survival time of event b and d is defined as 10 seconds, the pattern expression of the full-connection is: every a->(b->d) where timer: within(10 sec)
The meaning of the pattern expression is:

1. The pattern never stops looking for the event a.
2. When it reaches a1, the pattern starts a new sub-expression, and a1 is kept in the memory, and any event b is searched; and meanwhile, it still always looks for more events a.
3. When it reaches a2, the pattern starts a new sub-expression, and a2 is kept in the memory, and any event b is searched; and meanwhile, it still always looks for more events a.

After the arrival of a2, there are 3 active sub-expressions:

- The first sub-expression matching a1 is looking for any event b in the memory.
- The second sub-expression matching a2 is looking for any event b in the memory.
- The third sub-expression is looking for the next event a.

4. The survival time of event b and d is 10 seconds, so:

- If there is no event b and d after a1 in 10 seconds, the first sub-expression disappears.
- If there is no event b and d after a2 in 10 seconds, the second sub-expression disappears.
- The third sub-expression remains to look for the event a.

Supposing the order of the event is a1, a2, a3, b3, d3, the corresponding time-window is demonstrated in Figure 5.

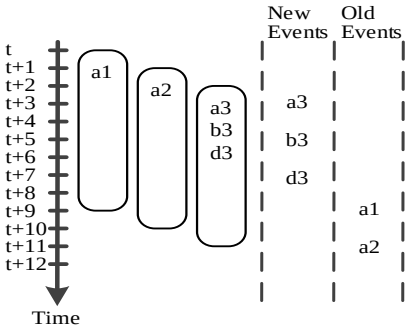

Figure 5. Time-window of events.

After 10 seconds, a1 disappears because the event satisfying the pattern is not found and the service life of the time-window is ended. Similarly, a2 also disappears. However, a3 triggers the Esper engine

because the events matching the pattern are found in the time-window.

The incomplete TCP half-open three-handshake is defined as that the event c occurs instead of the event d after the event a and event b, therefore, the pattern expression of TCP half-open connection is: every a->(b->(c and not d))

If the survival time of events b, c and d is 10 seconds, the pattern expression of TCP half-open connection is: every a->(b->(c and not d)) where timer: within(10 sec)

The goal of the collaborative threat detection architecture is to detect full and half-open connection. If the half-open connection or the failure connection is detected, the corresponding IP will be considered as a scanner, and eventually a series of suspicious IP addresses is output.

4. Experiment and Analysis on Results

The collaborative threat detection architecture utilizes seven hosts to build a test environment: the Kafka middleware is deployed on three hosts; the Zookeeper is deployed on one host and used for coordinating the Kafka clusters; the Flume is deployed on one host to be used as the Producer to acquire source data and to release same into the Kafka cluster; the Consumer is deployed on one host to process and analyze the real-time data; and the HDFS is deployed on one host to perform the persistent storage.

4.1. Datasets

Experimental datasets are downloaded from the MIT DARPA intrusion detection project [10], which are currently relatively authoritative attack testing datasets, and provide two attack scenarios-- DDoS 1.0 and DDoS 2.0.2. Compared to DDoS 1.0, the attack of the DDoS 2.0.2 is more subtle and sophisticated, and the two attack scenarios contain the network data packet with an attack phase marker of Demilitarized Zone (DMZ) and the internal network of an evaluation network acquired in a period of time.

The size, the packet number and the duration of the network data packets of DMZ and the internal network of the two attack scenarios are shown in Table 1.

Table 1. Network packets.

	DDOS1_dmz	DDOS1_inside	DDOS2_dmz	DDOS2_inside
Size(MB)	85.3	116.4	50.4	63.0
Number of packets	394089	649787	236753	347987
Length of the trace(s)	11760	11653	6150	6168

4.2. Results Analysis

The results obtained by processing the network data packets of the two attack scenarios according to the collaborative threat detection architecture are analyzed mainly from six aspects: reliability, processing efficiency, memory performance, overall performance, rules performance, and attack detection results.

1. Analysis on the reliability: Producer, Consumer and Broker are distributed in a number of computers, if the Broker of one computer is killed to fail the node, the data will not be lost, and Consumer can normally receive the data sent by Producer. Even if the main Broker is killed to cause the failure of the lead Broker, the data is still not lost, and Consumer still can normally receive data sent by Producer. Therefore, whether the Broker fails or the lead Broker fails, as long as there is a node which does not fail, Consumer will be able to normally receive the data sent by Producer. Therefore, whether the failure of the node or the failure of the main node exists, the Consumer can normally receive the data sent by the Producer, thereby guaranteeing the reliable transmission of the data.
2. Analysis on the processing efficiency: For the attack scenario of DDoS 1.0, by using the architecture to process the network packets, the processing time, packet rate, data rate, and the average value are illustrated in Tables 2 and 3.

Table 2. Processing of data packets for DDOS1_DMZ.

ID	time(s)	packet-rate(10^5p/s)	data-rate(Mb/s)
1	3	13.14	211.9
2	4	9.85	158.93
3	5	7.88	127.14
4	6	6.57	105.93
AVG	4.5	9.063	146.211

Table 2 is the result that network packets are parsed and processed for ten times in DMZ of DDOS 1.0 attack scenarios by using this architecture. The processing time in three seconds appears once, in four seconds appears four times, in five seconds appears four times, and in 6 seconds appears once. The results show that the average processing time of the network packets is 4.5 seconds, the average packet rate is 9.063 E5p/s, and the average data rate is 146.211Mb/s.

Table 3. Processing of data packets for DDOS1_inside.

ID	time(s)	packet-rate(10^5p/s)	data-rate(Mb/s)
1	5	13	171.29
2	6	10.83	142.74
3	7	9.28	122.35
AVG	6.2	10.644	140.294

Table 3 is the result that network packets are parsed and processed for ten times of the internal network of DDOS 1.0 attack scenarios by using this architecture. The processing time in five seconds appears twice, in six seconds appears four times, and in seven seconds appears four times. The results demonstrate that the average processing time of the network packets is 6.2 seconds, the average packet rate is 10.644 E5p/s, and

the average data rate is 140.294Mb/s.

For the attack scenarios of DDOS 2.0.2, by using the architecture to process the network packets, the processing time, packet rate, data rate, and the average value are illustrated in Tables 4 and 5.

Table 4. Processing of data packets for DDOS2_DMZ.

ID	time(s)	packet-rate(10^5p/s)	data-rate(Mb/s)
1	3	7.89	124.62
2	4	5.92	93.4
3	5	4.74	74.77
AVG	3.7	6.59	104.025

Table 4 is the result that network packets are parsed and processed for ten times in DMZ of DDOS 2.0.2 attack scenarios by using this architecture. The processing time in three seconds appears four times, in four seconds appears five times, and in five seconds appears once. The results demonstrate that the average processing time of the network packets is 3.7 seconds, the average packet rate is 6.59 E5p/s, and the average data rate is 104.025Mb/s.

Table 5. Processing of data packets for DDOS2_inside.

ID	time(s)	packet-rate(10^5p/s)	data-rate(Mb/s)
1	3	11.6	153.92
2	4	8.7	115.44
3	5	6.96	92.35
AVG	4	8.932	118.518

Table 5 is the result that network packets are parsed and processed for ten times in the internal network of DDoS 2.0.2 attack scenarios by using this architecture. The processing time in three seconds appears twice, in four seconds appears six times, and in five seconds appears twice. The results demonstrate that the average processing time of the network packets is 4 seconds, the average packet rate is 8.932 E5p/s, and the average data rate is 118.518Mb/s.

3. Analysis on the memory performance: by using the architecture to process the network data packets of DMZ and the internal network of two attack scenarios, the maximum memory utilization rate and average memory utilization rate are demonstrated in Table 6.

Table 6. Memory utilization rate.

ID	DDOS1_dmz	DDOS1_inside	DDOS2_dmz	DDOS2_inside
1	10	13	7.8	8.3
2	9.8	14.2	7.5	8.5
3	10	12.8	6	8.1
4	9.8	13	5.9	8.3
5	9.8	13.4	6	8.5
6	9.2	14.3	5.9	8.1
7	10	14.2	5.9	8.2
8	10.1	13	5.7	8.1
9	10.1	14.2	5.8	8.4
10	9.9	14.3	5.8	8.2
AVG	9.87	13.64	6.23	8.27

Table 6 is the result that network packets are parsed and processed for ten times in DMZ and the internal network of two attack scenarios by using this architecture. The results demonstrate that the average memory utilization rate of DDoS1_dmz is 9.87, the average memory utilization rate of DDoS1_inside is 13.64, the average memory utilization rate of DDoS2_dmz is 6.23, and the average memory utilization rate of DDoS2_inside is 8.27.

4. Analysis on the overall performance: the network packets, the average packet rate, the average data rate, the average memory utilization rate and the average processing time by using the architecture to process the network data packets of DMZ and the internal network of the two attack scenarios are illustrated in Figure 6.

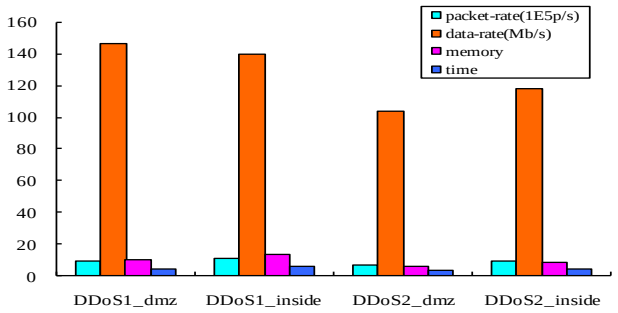

Figure 6. Overall performance.

Figure 6 shows that when the architecture is used for processing the network packets in DMZ and the internal network of two attack scenarios under the same network environment, if the packet number is smaller, the processing time is shorter, and the packet rate and memory utilization are smaller. Conversely, if the packet number is bigger, the processing time is longer, and the packet rate and memory utilization rate become bigger. Because the data rate is not only related to the number of packets, but also related to the size of each packet, hence, the changing trend of the data rate is inconsistent with the changing trends of other three performances.

5. Analysis on the regularity: this architecture can detect the events matching the event pattern. Taking the TCP three- handshake protocol as an example, the connection number the three-handshake packet number and the three-handshake packet rate of the network data packet in DMZ and internal network of the two attack scenarios detected according to the connection request event pattern and the full-connection event pattern established by the architecture are shown in Table 7.

Table 7. Three-handshake network packets.

	DDOS1_dmz	DDOS1_inside	DDOS2_dmz	DDOS2_inside
Number of connections	9559	9401	7034	7102
3-way-handshake packets	9462	9183	7019	7089
3-way-handshake packet rate (p/s)	0.8	0.79	1.14	1.15

It can be seen from the Table 7 that the three-handshake packet rate of DDoS1_dmz is 0.8, the three-handshake packet rate of DDoS1_inside is 0.79, the three-handshake packet rate of DDoS2_dmz is 1.14, and the three-handshake packet rate of DDoS2_inside is 1.15.

6. Attack detection results: The number of attackers for the network packets of the two attack scenarios detected according to the half-open connection event pattern of the architecture is illustrated in Table 8.

Table 8. Attacker.

	DDOS1	DDOS2
Number of attackers	33909	51148

When the attack is detected according to the architecture, all attackers are detected, and the detection rate reaches 100%. The attackers will be listed in the blacklist so as to prohibit their access, so that it can achieve the purpose of threat detection and defense.

5. Conclusions

In order to promote the healthy and safe development of the Internet+, this paper constructs a collaborative detection system for detecting the threats to the cyber security in big data by using the technology of big data.

Firstly, it introduces the involved technologies - the format of pacp packets, the log collection components of Flume, the characteristics and data collection components of Kafka middleware, the memory database, the event stream query mechanism and the pattern matching mechanism of Esper, and the attack type of DDoS and the attack principle of TCP Synchronize Sequence Numbers (SYN) Flood. Secondly, it designs the data flow for parsing the network packets, the one-to-many log collection model of Flume, the consistent data caching of Kafka and the CEP data processing of Esper, and also designs the overall architecture model of the collaborative threat detection. Then, it implements the parsing of the network packets, the configuration of Flume as the Producer component of Kafka for log collection, and realizes the Esper engine by adopting a method combining the event stream query and the event pattern matching, and utilizes the Esper engine as the Consumer of Kafka to process the data. Finally, it tests the datasets, and analyzes the results from six aspects: reliability, processing efficiency, memory performance, overall performance, rules performance, and attack detection results. The results show that the system is reliable, high efficient and accurate; and moreover, the system has the advantages of low cost and flexible operation.

With the development of Internet+, the big data technology and the big data infrastructures, the research on the collaborative detection of cyber security threats in big data will become a hotspot. Especially as the emergence of the online transaction such as the E-bank and the online payment, due to vast number of users, vast amount, and a huge number of transactions, it is very important to perform the collaborative detection of cyber security threats in big data. Therefore, the data collection content, the data processing methods and the data storage forms will still be the future research direction.

1. The data collection content: besides the collection of the network packets, the business and system log of hosts, and the threat intelligence of Network (the report of APT event, the attack situation, the vulnerability, the public opinion on the security) should also be used as the event source of the collaborative threat detection.
2. The data processing methods: the engine esper provides a real-time collaborative threat detection method in big data; however, the off-line detection of threats should also be carried out in order to detect potential threats as early as possible. In addition, the engine esper can be embedded into storm, so that it can provide the distributed collaborative detection of the security threats.
3. The data storage form: in the big data environment, the data acquired in the big data environment and the processed result data not only include the structured data, but also include a great number of the unstructured data. These unstructured data need to be effectively and reasonably stored into the file system or a database, thereby facilitating the indexed searching.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Nos. 61201220, 61309018) and

the National Basic Research Program of China (No. 2012CB315901).

References

[1] Aniello L., Baldoni R., Chockler G., Laventman G., Lodi G., and Vigfusson Y., "Agilis: An Internet-Scale Distributed Event Processing System for Collaborative Detection of Cyber Attacks," *MIDLAB Technical Report*, 2011.

[2] Aniello L., Luna G., Lodi G., and Baldoni R., *Collaborative Inter-domain Stealthy Port Scan Detection Using Esper Complex Event Processing*, Springer, 2012.

[3] Apache Flume, available at: http://flume.apache.org/, Last Visited 2016.

[4] Apache Kafka, available at: http://kafka.apache.org/, Last Visited, 2016.

[5] Apache Software Foundation, available at: http://hadoop.apache.org/, Last Visited, 2016.

[6] Armbrust M., Bateman D., Xin R., and Zaharia M., "Introduction to Spark 2.0 for Database Researchers," *in Proceedings of 16th International Conference on Management of Data*, San Francisco, pp. 2193-2194, 2016.

[7] Baldoni R. and Chockler G., *Collaborative Financial Infrastructure Protection*, Springer, 2012.

[8] Critical Infrastructure in the Age of Cyber War, http://www.mcafee.com/us/resources/reports/rp-in-crossfire-critical-infrastructure-cyber-war.pdf, Last Visited, 2010.

[9] Cutting D., "Hadoop: Industrial Strength Open Source for Data Intensive Supercomputing," *in Proceedings of 17th Conference on Information and Knowledge Management*, Napa Valley, 2008.

[10] DARPA Intrusion Detection Scenario Specific Data Sets, available at: http://www.ll.mit.edu/ideval/data/2000data.html, Last Visited, 2016.

[11] Esper Reference (Version 5.2.0), *EsperTech Inc.*, pp. 253-260, 2015.

[12] EsperTech: Event Series Intelligence, available at: http://www.espertech.com/, Last Visited, 2016.

[13] Global Fraud Report-Annual Edition 2011-2012, Kroll,http://www.krollconsulting.com/fraud-report/2011-12/press-only/, Last Visited, 2011.

[14] Hunt P., Konar M., Junqueira F., and Reed B., "Zookeeper: Wait-free Coordination for Internet-Scale Systems," *Usenix Annual Technical Conference*, Berkeley, 2010.

[15] Lamport L., "Fast Paxos," *Distributed Computing*, vol. 19, no. 2, pp. 79-103, 2006.

[16] Lodi G., Aniello L., Luna G., and Baldoni R., "An Event-based Platform for Collaborative Threats Detection and Monitoring," *Information Systems*, pp. 175-195, 2014.

[17] Luna G., A Collaborative Processing System for Cyber Attacks Detection and Crime Monitoring, Theses, Sapienza University, 2010.

[18] Marz N. and Warren J., *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, Manning, 2015.

[19] Sergio C., Andrew P., and Christopher B., "The Diamond Model of Intrusion Analysis," Technical Report, Center for Cyber Threat Intelligence and Threat Research, 2013.

[20] Singh J., Kaur L., and Gupta S., "A Cross-Layer Based Intrusion Detection Technique for Wireless Networks," *The International Arab Journal of Information Technology*, vol. 9, no. 3, pp. 201-207, 2012.

[21] Staheli D., Mancuso V., Harnasch R., Fulcher C., Chmielinski M., Kearns A., Kelly S., and Vuksani E., "Collaborative Data Analysis and Discovery for Cyber Security," *in Proceedings of 12th Symposium on Usable Privacy and Security*, Santa Clara, 2016.

[22] Zuech R., Khoshgoftaar T., and Wald R., "Intrusion Detection and Big Heterogeneous Data: a Survey," *Journal of Big Data*, vol. 2, no. 1, 2015.

Jiange Zhang received the M.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2007, and is currently pursuing the Ph.D. degree at the State Key Laboratory of Mathematical Engineering and Advanced Computing. Her research interests include network security, big data and situation awareness.

Yuanbo Guo received his Ph.D. degree in computer science and technology from Xidian University, Xi'an, China. His research interests include network security, network protocol design and analysis, threats detection, and situation awareness. He is currently a full Professor of computer science.

Yue Chen received his Ph.D. degree in computer science and technology from Zhengzhou Information Science and Technology Institute, Zhengzhou, China. His research interests include network security, network protocol design and analysis, and advanced computing. He is currently a full Professor of computer science.