Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 2716881, 9 pages
https://doi.org/10.1155/2021/2716881

Research Article

An Effective Fault-Tolerant Intrusion Detection System under Distributed Environment

Bo Hong, Hui Wang, and Zijian Cao

School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China

Correspondence should be addressed to Bo Hong; hongbo123@xatu.edu.cn

Received 19 August 2021; Accepted 20 September 2021; Published 19 October 2021

Academic Editor: Deepak Gupta

Traditional intrusion detection system is limited to a single network or several hosts, which has been seriously unable to fulfill the growing information security problems. This paper uses the distributed technology to design and implement an intrusion detection system (IDS) based on the hybrid of Hadoop with some effective open-source technologies. On the one hand, it can efficiently realize the data acquisition and analysis under distributed environment. On the other hand, it can solve the problems of single-point fault-tolerant and the insufficient data processing capacity of the traditional intrusion detection system. In this IDS, RabbitMQ, Flume, and MongoDB are utilized to act as the middleware of this system to build the system environment which includes the collector, analyzer, and data storage. By detecting the CPU and memory usage of hosts, TCP connections, network bandwidth, web server operation logs, and the logs of user behavior, the proposed IDS especially focuses on monitoring the first four parts, which can better detect external distributed denial of service attacks and intrusions and send automatically alarm service information to the administrators.

1. Introduction

The wide application of Internet and the heterogeneity of the interconnection network are limited to a single host-based traditional intrusion detection technology that has become increasingly difficulty to meet the current security requirements. In addition, the forms of network intrusions and attacks have become very hidden and gradually tend to be distributed, coordinated, and diverse [1–4]. Therefore, intrusion detection system also needs to meet the new application requirements, such as easy expansion, reuse, cross-platform, and collaborative detection [3, 5–7]. Therefore, it is very urgent to study and utilize distributed technology to realize the distributed intrusion detection system. Hadoop is an open-source distributed system infrastructure based on cloud computing [4, 5, 8, 9] and can perform large-scale cluster distributed parallel programming calculations. Based on Hadoop, many cheap hardware devices can be formed into a cloud computing cluster and make full use of those cluster computing environment and fast storage advantages to develop a distributed system that handles large-scale data suitable for user needs. Hadoop is developed and implemented based on the object-oriented programming language Java, which has good fault tolerance, data analysis balance, and portability. Hence, the application of Hadoop in the intrusion detection of network data can effectively solve the problem of insufficient data analysis ability of the current intrusion detection system and realize a real distributed intrusion detection system. Based on the above analysis, this paper intends to design and implement a distributed intrusion detection system based on Hadoop and some open-source technologies.

Intrusion detection technology enables the security system to make real-time response to intrusion events and intrusion processes by studying the process and characteristics of intrusion behaviors. In terms of detection methods, there are roughly two kinds: misuse intrusion detection and anomaly intrusion detection [10–12]. In the former, it can be assumed that all intrusion behaviors and skills can be expressed as a pattern or feature; all known intrusion methods can be found by matching methods. The key idea of it is how to express the pattern of invasion and to distinguish the real invasion from the normal behavior, and its advantage is to identify the known attack false positives less;

the limitation is the unknown attack that can do nothing. In abnormal intrusion detection, all intrusion behaviors are assumed to be different from normal behaviors. Thus, if a trajectory of the normal behavior of the system is established, then theoretically, all system states that are different from the normal trajectory can be regarded as one suspicious behavior. For example, abnormal network traffic at unusual times is considered suspicious through traffic statistical analysis. The limitation of anomaly intrusion detection is that not all intrusion is abnormal, and the system trajectory is difficult to calculate and update.

By comparing the above two intrusion detection methods, it can be clearly seen that the abnormal detection is difficult to be quantitatively analyzed, and this detection method has inherent uncertainties. Unlike this, the misuse detection follows a defined pattern and can be detected by pattern matching on audit records or network real-time data streams, but only known intrusion methods can be detected. Thus, neither of these detection mechanisms is perfect. In terms of specific detection methods, there have been many intrusion detection methods, but any method has its limitations and cannot solve all intrusion problems [8, 13–19]. The main reason for choosing Hadoop is as follows. Spark is not applicable for applications that update the state asynchronously and finely, such as web service storage or incremental web crawler and index; that is, the application model of incremental modification is not suitable. Spark is also not suitable for processing super large amounts of data. The super large here is relative to the memory capacity of the cluster, because spark needs to store data in memory. In a word, Spark is mainly used for big data calculation, while Hadoop is mainly used for big data storage. Because the requirement of real-time calculation about IDS system is not high, Hadoop is the better choose. Therefore, the research of intrusion detection method is still a difficult point in the current intrusion detection research. Based on the above analysis, this paper designs a distributed intrusion detection system in combination with the structural characteristics of Hadoop cluster. The main contributions of this paper can be summarized as follows.

(1) To better detect external distributed denial of service attacks, Hadoop which has good fault tolerance, data analysis balance, and portability is used in the IDS system to solve the problem of insufficient data analysis ability of the current intrusion detection system and realize a real distributed IDS

(2) The open-source technologies, RabbitMQ, Flume, and MongoDB which, respectively, act as the collector analyzer and data storage, are utilized to construct the IDS system. It is not only free of copyright problems, but also it is efficient

The rest of this article is organized as follows. Section 2 represents the framework of IDS with effective techniques incorporated into Hadoop for different functions. The system implementation of the proposed IDS is given in Section 3. In Section 4, the system testing and analysis are discussed, followed by conclusions in Section 5.

2. The Framework of the Proposed IDS

Through the research on the process and characteristics of intrusion behavior, the intrusion detection technology enables the security system to make real-time response to intrusion events and intrusion process. There are two detection methods: misuse intrusion detection and abnormal intrusion detection. But these two kinds of detection mechanisms are not perfect. As for specific detection methods, there are many intrusion detection methods, but any method has its limitations and cannot solve all problems. Therefore, the research on intrusion detection methods is still a focus of current intrusion detection research. Based on the existing problems of intrusion detection methods, we design this intrusion detection system. The structure of the proposed IDS system is shown in Figure 1, which consists of data collector, data detector, data transceiver middleware, data analysis center, system monitoring, and alarm service.

In Figure 1, this system refers to the design idea of checklist (checklist is an online system operating status monitoring function, which can be a single-point monitoring or distributed monitoring) system and adds data detector, system monitoring page, alarm service, and other functions on the basis of the checklist system. In addition, some other functions of the system are realized by open-source software. The reason is that these open-source software are widely used in large enterprise applications and have undergone a lot of tests, which is conducive to the stability and long-term operation of the system. Based on the basic framework of checklist, this paper develops a distributed intrusion detection system based on Hadoop distributed computing framework.

2.1. Data Detector. The data detector is the system data acquisition and event analysis unit, distributed at the bottom of the system. A detector is a detection subject that runs independently on the host. The system has no restrictions on the detector; that is, the detector runs as a root user. According to the classification of data sources, the data detector can be divided into the detector of the host and the detector of the base network.

For this IDS system, the host-based detector is used to mainly detect the host CPU utilization rate, TCP connection number, MEM utilization rate, network bandwidth, user behavior log, web server operation log, which are the six indicators. CPU, TCP, MEM, network bandwidth are the four general indicators for each host, by fetching service to grab the four indicators, fetching the data for the unit with the minute, and writing the data directly into the data transceiver middleware; then, these data are feedbacked to the data center, for the user behavior and system log information through journal monitoring service time to grab the latest information and regular cleaning. For the network data detector, this system uses the host's own firewall and other protection procedures. The operation process of the data monitor is shown in Figure 2.

2.2. Data Collector. The data acquisition agent controls all detectors on the local host, and the detector does not send

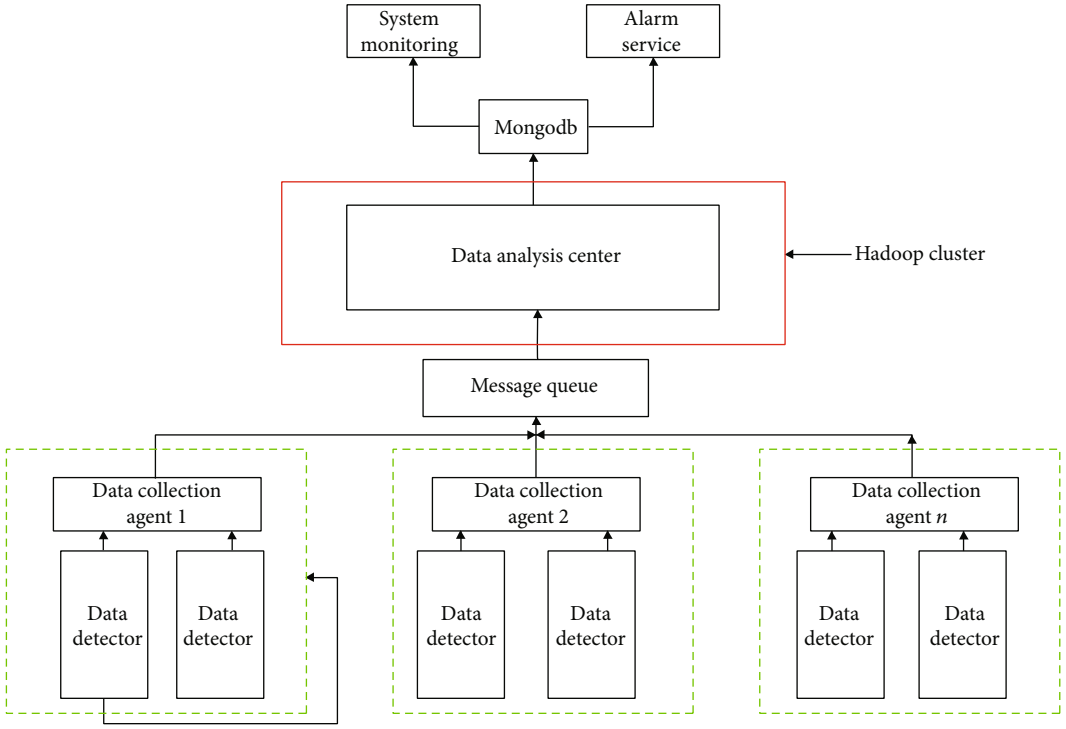

Figure 1: IDS structure diagram.

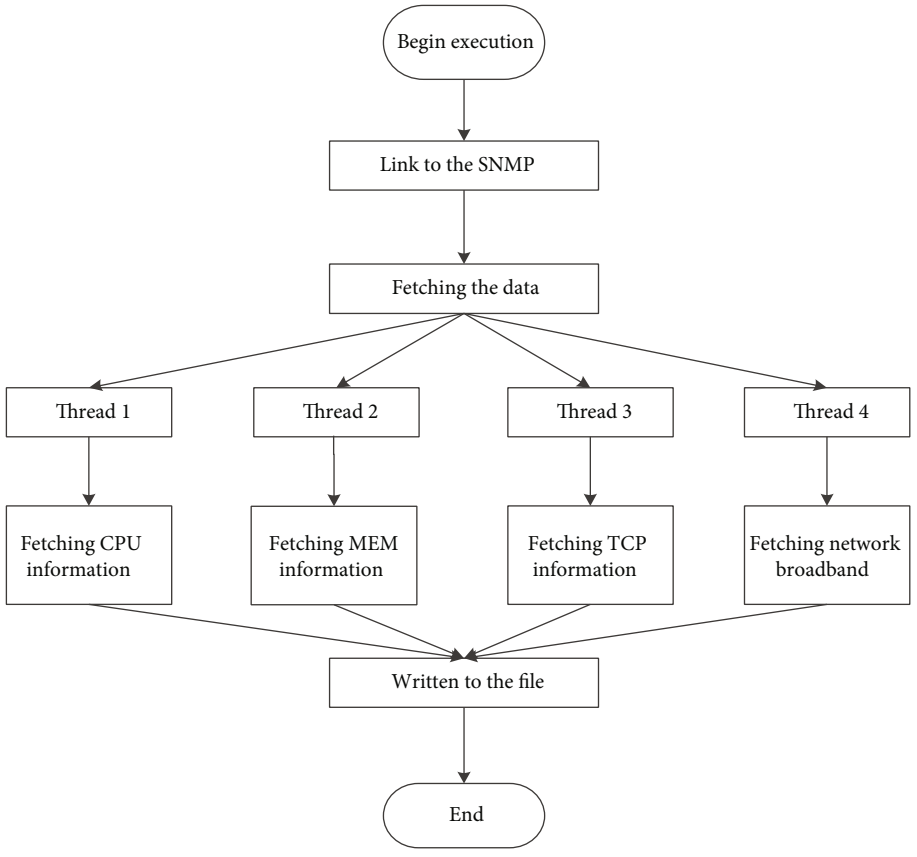

Figure 2: Flow chart of data detector.

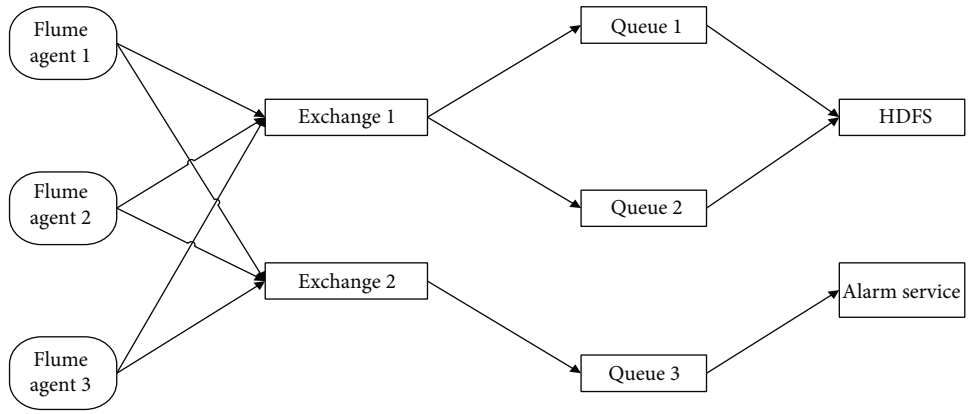

Figure 3: Data transceiver middleware structure diagram.

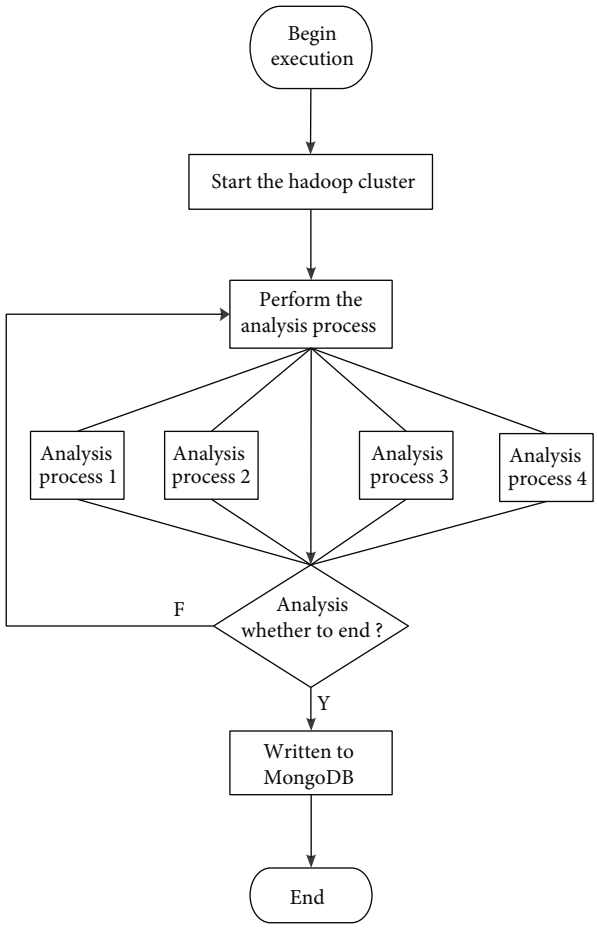

Figure 4: Flow chart of data analysis center.

data directly to the transceiver. When the detector wants to transmit data, the data acquisition agent connects to the transceiver to transmit. Flume is used as the data collector in this system. Flume listens for the data detector to write data to the file. If the file has new content, the new content is sent out on a line basis and sent to Flume's collector. Collector aggregates the collected data and writes it to RabbitMQ. The Flume agent listens to files such as CPU, MEM, TCP connection count, network bandwidth, web server access logs, and user behavior logs. If these files are updated, the agent sends the updates line by line to Flume's collector, which writes the aggregated logs to the responding RabbitMQ exchange.

2.3. Data Transceiver. The reason why this system uses the data transceiver middleware is that a monitoring system may need to monitor multiple regions or networks. Different types of tasks are processed in different regions, and the data generated are used for different purposes. The use of the data transceiver middleware can classify the unnecessary data and facilitate the analysis of data by the data analysis center. This system uses the RabbitMQ as the data transceiver middleware, the RabbitMQ is a kind of message queue and will be written to the log information of different queues, data collector as the data producer just writes different types of data into the corresponding queue, and data analysis center data read from the queue as a data consumer end, analysis, and calculation. The RabbitMQ mechanism used in this system is shown in Figure 3.

2.4. Data Analysis Center. As the core module of the system, the data analysis center adopts the mode of centralized storage, distributed detection, and centralized analysis. When the analysis of the data of detector is sent back according to different need, we can customize different data analysis process, and give full play of the Hadoop cluster computing ability. By analysing the collected data, we can also find the underlying detect intrusion behavior. Then, the feedback of the intrusion detection results is timely made to the system administrator, to adjust each host firewall policy. The analysis process in the data analysis adopts different analysis processes for different logs. Through the analysis of the web server access logs, we can get the number of IP visits to the monitored system per unit time, which the IP address has the highest access frequency in 24 hours, the IP source, and other information. By analyzing the analysis of the user behavior log, we can get the information about what a user did after entering the system, which services he visited, etc. Through this information, it can be analyzed out the illegal

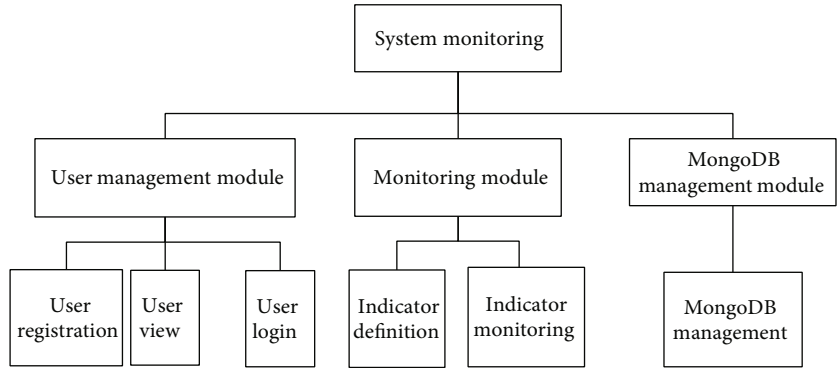

Figure 5: System monitoring function diagram.

operation of the user and the timely discovery of intrusion behavior. The analysis center uses the cluster of Hadoop as the basic framework, which is expansible. The MapReduce computing framework of Hadoop is sufficient to deal with massive data, which completely solves the problem that the single-point processing capacity of the traditional intrusion detection system is not out of the question. According to the actual situation, users define their own analysis process for the information to be analyzed and obtain the required analysis results. The basic flow of the data analysis center is shown in Figure 4.

2.5. System Monitoring. The system monitoring module is used to display the running state of each host of the monitored system, as well as the running state of the Hadoop cluster. By monitoring the running state of each host, traces of external intrusion can be found, discovered, and handled in a timely manner. Monitoring system is mainly to monitor each host CPU, MEM, TCP connection, network bandwidth, and other basic information; this information is every machine must have information; they can clearly reflect the running status of each host. The function diagram of system monitoring is shown in Figure 5.

In Figure 5, the user management module consists of three functions: user registration, user view, and user login. The user must log in before entering the monitoring system. If the user does not logged in, the user cannot enter the system. User registration function can only be made if the system administrator has this permission; ordinary users cannot be registered; it must be the administrator to add users; user view function can see how many users exist in the monitoring system. The monitoring module has two functions: indicator definition and indicator monitoring. The indicator definition defines the indicator to be monitored by the user, the name of the set stored in MongoDB, the indicator is discrete or continuous and other information. Metric monitoring shows only the collected data, in the form of a trend chart. It can be viewed the recent one hour, six hours, and the whole day trend chart; the system's operation status is intuitively displayed in front of the user. MongoDB management module is mainly designed for the convenience of online MongoDB management. In general, MongoDB is deployed in the production environment in enterprise-level applications, and the development machine cannot access MongoDB in the production environment. The problem that the development machine cannot access the online environment is solved by accessing the online environment through HTTP protocol.

2.6. Alarm Service. The alarm service is at the top of the system. On the one hand, the data distraction center analyzes the collected data to find out whether the system has been attacked and whether the operation is normal. When the data analysis center analyzes some information anomalies, it sends alarm information to the relevant administrators through the alarm service, including SMS alarm and email alarm. On the other hand, when the monitoring system monitors the abnormal running state of a host, it will alarm in time, such as high CPU utilization rate and other problems.

It is through the alarm service administrators can find the problems of the system in time and deal with them in time to ensure the long-term operation of the system. For the four indicators monitored by this system (CPU, MEM, TCP connection number, and network bandwidth), the alarm strategy adopted is that when the value of an indicator appears n times in a continuous m minutes, the alarm can be set at different levels. If the level is low, the alarm will be sent by email, and if the level is high, the alarm will be sent by SMS and mail. Considering that the staff cannot check the email in time after work, the use of short message alarm is beneficial after work. The flow chart of alarm service is shown in Figure 6.

2.7. Data Storage. This system uses MongoDB, an open-source NoSQL database. MongoDB is a database based on distributed file storage and has the ability of massive data processing [20, 21]. It has the following characteristics: set oriented and data is stored in the dataset. Schema-free means that users of the files we store in the database do not need to know the format of the data stored. Documents stored in a collection exist as key-value pairs for easy lookup. This system mainly processes data collector gathering up the log data, such as a large amount of data and format is not fixed; the traditional relational database is not good at dealing with huge amounts of data, so we use MongoDB as a database to store data analysis center after analysis of data, and it will facilitate system monitoring read data from the

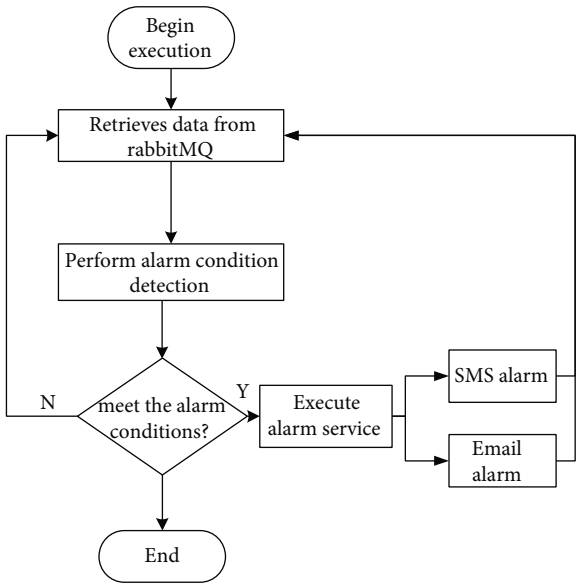

Figure 6: Alarm service flow chart.

```
{"_id": "49902cde5162504500b45c2c",
 "serverName": "server001",
 "quotaName": "CPU",
"quotaValue": 1,
 "currentTime": yyyy-MM-dd hh:mm:ss
}
```

Algorithm 1

```
Hadoop jar /xxx/xxx/xx.jar arg1 arg2
```

Algorithm 2

MongoDB. The data format of CPU, MEM, TCP connection number, and network bandwidth monitored by this system in MongoDB is as follows in Algorithm 1.

The corresponding set names of each index are as follows: quota_cpu, quota_mem, quota_tcp, and quota_network_width, respectively. _id: Produced by MongoDB is a unique identifier for each document: ServerName: Server name. QuotaName: Indicator name; the names of these four indicators are CPU, MEM, TCP_CONN, and network_width. quotaValue indicates the current value of this indicator. CurrentTime identifies the time when this piece of data was written to MongoDB, enabling MongoDB to be queried by time dimension.

3. System Implementation

3.1. Hadoop Cluster Operation. After the Hadoop cluster is set up, it will be executed all the time after starting the Hadoop cluster. Developers can submit their written Java programs into JAR packages to the Hadoop cluster for execution; it can commit multiple JAR packages for execution in Algorithm 2.

```
while(recVB.getOid().toString().indexOf(cpuOid)>=0){
   pdu.clear();
   pdu.add(new VariableBinding(recVB.getOid()));
   pdu.setType(PDU.GETNEXT);
   ResponseEvent respEvnt = snmp.send(pdu, target);
   if (respEvnt != null && respEvnt.getResponse() != null) {
   for (int i =0; i < recVBs.size(); i++) {
   recVB = recVBs.elementAt(i);
      if(recVB.getOid().toString().indexOf(cpuOid) >=0){
         cpuInfo+=recVB.getVariable().toInt();
         cpuCoreNum += 1;
         tcpConn += recVB.getVariable().toInt();
      }}
}
```

Algorithm 3

where xxx represents the path of the JAR package and arg1 and arg2 are the parameters of the execution jar package.

3.2. Data Collector Implementation. Data acquisition can collect a variety of data, such as the system operation of the log information and the system operation of the data. This paper mainly realizes the collection of CPU utilization rate, memory, TCP connection number, network bandwidth, and other information when the host is running. The following mainly uses CPU utilization and TCP connection number as an example to introduce the implementation; other information is the same principle. The data detector is implemented in JAVA language, and the protocol is SNMP protocol. SNMP is a network management standard based on TCP/IP protocol family. Its predecessor is Simple Gateway Monitoring Protocol (SGMP), which is used to manage communication lines. The data detector is connected to SNMP service through the SNMP driver package of Java, and it is connected to port 161. By sending the corresponding SNMP OID to the server to obtain the corresponding information, the OID of CPU utilization is 1.3.6.1.2.1.25.3.3.1.2. The method for SNMP to obtain data is GetNext method, which uses recursive query to query the corresponding values of all OIDs under the current OID tree.

Taking the development machine as an example, the development machine uses a 4-core CPU. Hence, there should be 4 subtrees under the OID tree. The GetNext method is used to recursively obtain the utilization rate of all cores for 4 times to find the average value. The OID of the number of TCP connections is 1.3.6.1.2.1.6.9. Since the number of TCP connections in the system has only one value, the OID tree has only one, and the current number of TCP connections in the system can be obtained only by recursing once. The core workflow of the data detector is shown as follows in Algorithm 3.

3.3. Realization of System Monitoring. System monitoring is developed based on B/S structure, and the entire system monitoring module can be configured using Spring MVC

```
<bean id="initDashBoard"
   class = "com.elong.dashboard2.dashboard.InitDashBoardController">
</bean>
   <bean id="velocityConfig"
      class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">
      <property name="configLocation">
         <value>/WEB-INF/views/velocity/velocity.properties</value>
      </property>
   </bean>
   <bean id="viewResolver"
   class="org.springframework.web.servlet.view.velocity.VelocityLayoutViewResolver">
      <property name="cache" value="false" />
      <property name="layoutUrl" value="/layout/main.vm" />
      <property name="prefix" value="/templates/" />
      <property name="suffix" value=".vm" />
      <property name="exposeSpringMacroHelpers" value="true" />
      <property name="contentType" value="text/html;charset=UTF-8" />
      <property name="viewClass"
         value="org.springframework.web.servlet.view.velocity.VelocityLayoutView" />
   <bean id="handlerMapping"
   class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">
      <property name="interceptors">
         <ref bean="localeChangeInterceptor" />
      </property>
   </bean>
```

Algorithm 4

```
#RABBITMQ_NODE_PORT=8088 //The port number
#HOSTNAME=http://www.ddsnort.com
RABBITMQ_NODENAME=mq
RABBITMQ_CONFIG_FILE=/etc/rab.conf //Configuration file path
RABBITMQ_MNESIA_BASE=/rabbitmq/data
// MNESIAPath to the database
RABBITMQ_LOG_BASE=/rabbitmq/log //log The path
RABBITMQ_PLUGINS_DIR=/rabbitmq/plugins//The plugin path
```

Algorithm 5

framework in servlet.xml file in Spring, which determines the functions of using the framework, as well as the combination of Spring MVC and Velocity [12, 22, 23]. The main functions of the system monitoring module include user management module, monitoring module, and MongoDB management module. Under normal conditions, the development machines cannot access the online database. Through the MongoDB management module, the inaccessible problem can be solved by using HTTP protocol access. The following code configures how Velocity parses the view page, and Velocity configures the path of the page. The servlet.xml file is the key of the entire MVC framework. If this file is configured incorrectly, a web application will not run in Algorithm 4.

3.4. Main Configuration of RabbitMQ. The data detector, system monitoring, and alarm service of the system need to be realized by ourselves. Other modules are realized by open-source software. The data collector uses the open-source mass log collection tool Flume to collect the data. The data delivery middleware uses RabbitMQ, where users write different data streams to different queues. Some of the common RabbitMQ environment variables are as follows in Algorithm 5.

The above section is the main configuration of the RabbitMQ. For more details, please refer to the RabbitMQ website.

4. System Testing and Analysis

DDoS (distributed denial of service) attack tool is used to test the system and launch a DDoS attack on the monitored. There are many TCP links in the machine system, and then, the number of TCP connections in the system monitoring can be seen to surge. As shown in Figure 7, under normal circumstances, the number of TCP connections in the system tends to be stable, and it can be seen from Figure 7 that the number remains between 0 and 50. When the system is attacked by DDoS, the number of TCP connections

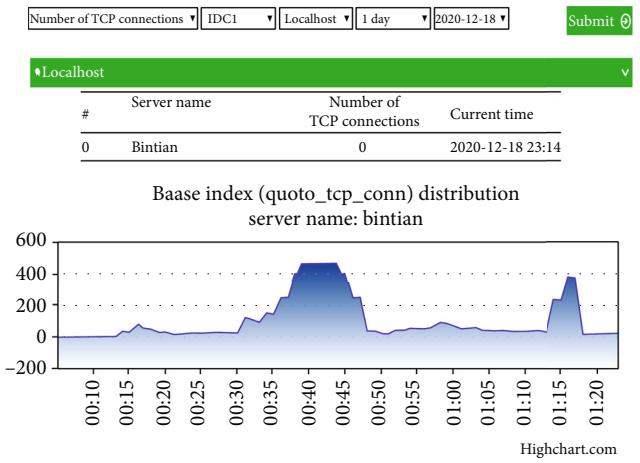

Figure 7: Schematic diagram of TCP network connection number when system is attacked.

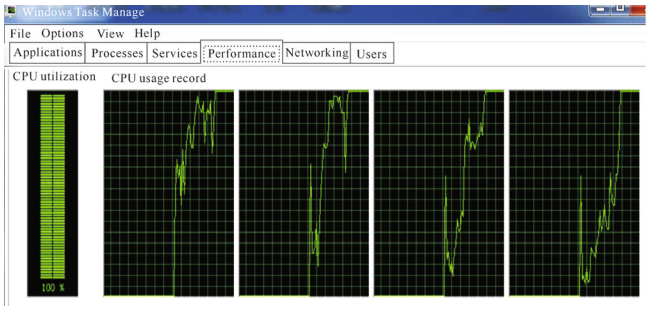

Figure 8: CPU utilization.

increases sharply to about 450. For external attack monitoring system, we can timely detect external attacks and intrusions. This is also shown in Figure 7.

To verify the correctness of the data detector, we intercepted the CPU utilization graph in the task manager of the Windows system at this time, as shown in Figure 8.

5. Conclusions

The distributed intrusion detection system based on Hadoop is a relatively practical system. This paper adopts distributed technology and open-source software to change the shortcomings of traditional intrusion detection system, such as insufficient computing power. Using distributed computing technology, through the network, the huge need to deal with the data will be automatically split and delivered by the server cluster consisting of a large system; after the search, calculation, and analysis, data mining, the optimized results, the corresponding treatment scheme, and reliable and stable update results stored in the database, through data detector of CPU, MEM, TCP, network bandwidth, and other four indicators test, can better find external DDoS attack and intrusion alarm and provide service function. Because the Hadoop is based on the distributed environment, the proposed IDS system with Hadoop and other effective open-source technologies is fault-tolerant according to the system testing and analysis.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

It is declared by the authors that this article is free of conflict of interest.

Acknowledgments

This research was partially funded by the Shaanxi Natural Science Basic Research Project (Grant No. 2020JM-565), the Project of Department of Education Science Research of Shaanxi Province (Grant No. 18JK0383), the Innovation and Entrepreneurship Training Program for College Students (Grant No. S202010702108), the Teaching Reform Research Project of Xi'an Technological University (Grant No. 20JGY38), and the General Cultivation Project of President Fund of Xi'an Technological University (Grant No. XGPY200207).

References

[1] R. Z. Yang, T. Y. Chen, and Y. P. Li, "Broad-scale distributed intrusion detection system," *Network and Information Security*, vol. 39, no. 7, pp. 31–35, 2020.

[2] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses," *IEEE Computer*, vol. 45, no. 1, pp. 39–45, 2012.

[3] S. Chen and V. P. Janeja, "Human perspective to anomaly detection for cybersecurity," *Journal of Intelligent Information Systems*, vol. 42, no. 1, pp. 133–153, 2014.

[4] S. Deng, A. H. Zhou, D. Yue, B. Hu, and L. P. Zhu, "Distributed intrusion detection based on hybrid gene expression programming and cloud computing in a cyber physical power system," *IET Control Theory & Applications*, vol. 11, no. 11, pp. 1822–1829, 2017.

[5] S. Deng, A. H. Zhou, and D. Yue, "Algorithms. Study data from Nanjing University update understanding of algorithms (distributed intrusion detection based on hybrid gene expression programming and cloud computing in a cyber physical power system)," *Journal of Engineering*, vol. 8, 2017.

[6] N. M. Ibrahim and A. Zainal, "A distributed intrusion detection scheme for cloud computing," *International Journal of Distributed Systems and Technologies*, vol. 11, no. 1, pp. 68–82, 2020.

[7] G. de la Torre Parra, P. Rad, K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, article 102662, 2020.

[8] S. Zhou and P. Yang, "Risk management in distributed wind energy implementing analytic hierarchy process," *Renewable Energy*, vol. 150, pp. 616–623, 2020.

[9] H. Han, H. Kim, and Y. Kim, "Evaluation of distributed intrusion detection system based on MongoDB," *KIPS Transactions on Computer and Communication Systems*, vol. 8, pp. 287–296, 2019.

[10] A. M. Riyad, M. S. I. Ahmed, and R. L. R. Khan, "An adaptive distributed intrusion detection system architecture using multi agents," *International Journal of Electrical and Computer Engineering*, vol. 12, 2019.

[11] D. L. Li, "Exploring the application of intrusion detection technology in computer database," *Digital Technology & Application*, vol. 37, no. 3, 2019.

[12] K. Xie, Y. X. Yang, Y. Xin, and G. Xia, "Cellular neural network-based methods for distributed network intrusion detection," *Mathematical Problems in Engineering*, vol. 2015, Article ID 343050, 10 pages, 2015.

[13] J. F. Colom, D. Gil, H. Mora, B. Volckaert, and A. M. Jimeno, "Scheduling framework for distributed intrusion detection systems over heterogeneous network architectures," *Journal of Network and Computer Applications*, vol. 108, pp. 76–86, 2018.

[14] J. C. McEachen and C. Wai Kah, "An analysis of distributed sensor data aggregation for network intrusion detection," *Microprocessors and Microsystems*, vol. 31, no. 4, pp. 263–272, 2007.

[15] T. E. Simos and C. Tsitouras, "Evolutionary derivation of Runge–Kutta pairs for addressing inhomogeneous linear problems," *Numerical Algorithms*, vol. 87, no. 2, pp. 511–525, 2021.

[16] M. A. Medvedev, T. E. Simos, and C. Tsitouras, "Explicit, eighth-order, four-step methods for solving $Y''=F(X,Y)$," *Bulletin of the Malaysian Mathematical Sciences Society*, vol. 43, no. 5, pp. 3791–3807, 2020.

[17] Y. Sarac and S. S. Sener, "Identification of the initial temperature from the given temperature data at the left end of a rod," *Applied Mathematics and Nonlinear Sciences*, vol. 4, no. 2, pp. 469–474, 2019.

[18] Y. X. Wang, S. R. Behera, J. Wong et al., "Towards the automatic generation of mobile agents for distributed intrusion detection system," *The Journal of Systems & Software*, vol. 79, no. 1, pp. 1–14, 2006.

[19] A. Fagiolini, G. Dini, and A. Bicchi, "Distributed intrusion detection for the security of industrial cooperative robotic systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7610–7615, 2014.

[20] J. Manan, A. Ahmed, I. Ullah, L. Merghem-Boulahia, and D. Gaïti, "Distributed intrusion detection scheme for next generation networks," *Journal of Network and Computer Applications*, vol. 147, article 102422, 2019.

[21] M. Himi, J. Tapias, S. Benabdelouahab et al., "Geophysical characterization of saltwater intrusion in a coastal aquifer: the case of Martil-Alila plain (North Morocco)," *Journal of African Earth Sciences*, vol. 126, pp. 136–147, 2017.

[22] H. Tang, L. Chen, J. Lu, and C. K. Tse, "Adaptive synchronization between two complex networks with nonidentical topological structures," *Physica A*, vol. 387, no. 22, pp. 5623–5630, 2008.

[23] Z. Sui, "Large-scale network intrusion detection based on multi-point and mulit-task decomposition mapping technology," *Computer Simulation*, vol. 31, no. 2, 2014.