Hindawi
Security and Communication Networks
Volume 2022, Article ID 3105291, 13 pages
https://doi.org/10.1155/2022/3105291

*Research Article*

# Task-Oriented Network Abnormal Behavior Detection Method

**Tao Li**,[1,2,3] **Wenzhe Dong**,[1] **Aiqun Hu**,[1,2,3] **and Jinguang Han**[1]

[1]*School of Cyber Science and Engineering, Southeast University, Nanjing 210000, China*
[2]*Purple Mountain Laboratories, Nanjing 210000, China*
[3]*Frontiers Science Center for Mobile Information Communication and Security, Southeast University, Nanjing 210000, China*

Correspondence should be addressed to Wenzhe Dong; 220194657@seu.edu.cn

Received 21 December 2021; Revised 18 May 2022; Accepted 3 June 2022; Published 30 June 2022

Academic Editor: Shahram Babaie



Since network systems have become increasingly large and complex, the limitations of traditional abnormal packet detection have gradually emerged. The existing detection methods mainly rely on the recognition of packet features, which lack the association of specific applications and result in hysteresis and inaccurate judgement. In this paper, a task-oriented abnormal packet behavior detection method is proposed, which creatively collects action identifications during the execution of network tasks and inserts security labels into communication packets. Specifically, this paper defines the network tasks as a collection of state and action sequences to achieve the fine-grained division of the execution of network tasks, performs Hash value matching based on random communication string and action identification sequence for packet authentication, and proposes a mechanism of action identification sequence matching and abnormal behavior decision-making based on a finite state machine, according to the fine-grained monitoring of task execution action sequence. Furthermore, to verify the validity of the anomaly detection method proposed in this paper, a prototype based on the FTP communication platform is constructed, on which the simulation experiments, including the DDOS attack and backdoor attack, are conducted. The experimental results show that the proposed task-oriented abnormal behavior detection method can effectively intercept network malicious data packets and realize the active security defense for network systems.

## 1. Introduction

Traditional network anomaly detection methods rely on the recognition of abnormal features which are abstracted from existing malware. Moreover, the existing assessment mechanisms only analyze network packets without considering related applications that generate the packets. Actually, network packets are one part of applications' behaviors. Thus, the existing anomaly detection methods always result in some limitations and inaccuracy [1].

Facing the lack of traditional anomaly detection technology, this paper proposes a task-oriented method that associates data packets with terminals' task running state. Traditional detection methods tend to detect anomalies through feature extraction, recognition, and matching. When an unknown attack is encountered, traditional methods are ineffective because there is no feature extraction for the unknown attack before it occurs. Under the proposed method, a network system model is designed with a highly integrated security mechanism. Different from current security defense mechanisms aiming at extracting packets' features of existing network attacks, the method proposed in this paper concentrates on the task's process state to classify and obtain the execution state and action identifications of each process. The acquired identifications can be inserted into data packets as security labels. Finally, security decisions will be made based on whether the packets are sent from normal tasks by analyzing the inserted labels.

Our proposed method separates the data layer and the control layer, which is similar to the architecture of Software-Defined Network (SDN) [2]. The network layer of the existing SDN has not been concretely implemented, while only its data layer and control layer are available.

Thus, the task-oriented anomaly detection method only utilizes the data layer and the control layer of SDN. On the data layer, the method realizes the transfer of security labels based on the basic functions of the network system; on the control layer, the controller makes security judgement by using security labels that are transferred by the data layer.

In order to combine task's behavior with network packet, our proposed method regards each task's state and action as application's behavioral characteristics, and a security label is used to associate data packet and application's behavior. Finite state machine (FSM) is used to record the normal behavior of a task. By comparing the information carried by packets with FSM, the proposed method can detect unknown attacks effectively and prevent attacks timely. In this paper, a network system model is also built according to the proposed method. Two related algorithms are designed to realize some basic functions. Algorithm 1 is designed to extract security labels from the data packets in the transponder. Algorithm 2 achieves the function of judging whether there is any abnormal behavior in the network, which works in the controller. Finally, this paper reforms the architecture of SDN to set up an experimental LAN. In this LAN, DDOS attack and backdoor attack are simulated to verify the validity of the proposed method.

This paper makes the following contributions:

(i) We propose a task-oriented method for detecting abnormal behaviors that associate data packets with applications and build a task-oriented model for the implementation of this method.

(ii) We insert security labels into data packets to carry the state and action information at the terminal and design an algorithm for extracting the security labels in the transponder.

(iii) We design an abnormal data packet decision mechanism that combines the task-oriented FSM model with the security labels carried by the data packets to judge whether there is any abnormal behavior.

(iv) We evaluate the security performance of the proposed method by simulating a DDOS attack and a backdoor attack to demonstrate the validity of this method.

The remainder of this paper proceeds as follows. Section 2 introduces typical existing network traffic anomaly detection methods. Section 3 presents some technologies used in the paper to abstract the working principles of our task-oriented model. Section 4 describes the method of generating a security label and shows how to extract the label from a data packet. Section 5 designs the algorithms used in the controller to judge whether the behavior of the task sending related packet is abnormal. Section 6 shows the experiments based on the constructed prototype platform, two different attacks are simulated, and protection results are analyzed. Section 7 concludes this paper and declares some future work.

## 2. Related Work

Since network security has been a major concern in computer networks and systems, many researchers have delved into areas of network security and proposed different detection mechanisms and technologies [3]. Data feature matching is a common and traditional method to detect abnormal network packets. Based on analyzing the packet characteristics under the normal and abnormal states, abnormal packet features are deposited in the feature library [4]. By extracting the characteristics of data packets and comparing them with the feature library, whether packets are normal or abnormal can be judged. The disadvantage of this method is that it relies too much on the feature library, which is constructed based on existing features.

At present, active detection technologies exploit numerous statistics, data mining, and Machine Learning- (ML-) based techniques to automatically detect attacks [5]. These technologies can be discriminated into shallow ML and Deep Learning (DL) according to the involved network architecture. ML-based anomaly detection approaches can be further classified as supervised [6], semisupervised [7, 8], and unsupervised [9, 10]. Supervised classification approaches have a high requirement for the training data, which should include as many as anomalous examples along with corresponding labels. Semisupervised anomaly detection methods extract training data from a sufficiently large amount of collected logs or network measurements to provide accurate estimates of the probability distribution of the normal and malicious classes. Unsupervised anomaly detection methodologies aim to automatically identify normal network behaviors from abnormal ones without exploiting labeled data [11]. Moreover, there are also three main anomaly detection methods based on Deep Learning, including Boltzmann Machine- (RBM-) based [12], Stacked Auto Encoders- (SAE-) based [13], and Convolutional Neural Network- (CNN-) based [14]. The RBM-based method uses the self-coding network method to reduce the feature dimension of the high-dimensional and nonlinear original data, and then the optimal low-dimensional feature vector obtained in the learning process can be identified by SVM. The SAE-based anomaly detection method extracts features by learning the traffic data layer by layer with high accuracy, but the robustness of feature extraction is poor. The traffic features extracted by the CNN-based approach have strong robustness and high detection performance. However, this method needs to convert network traffic into images first, which increases the burden of data processing [15, 16]. All these active detection technologies are greatly affected by the accuracy of training algorithms.

All the above anomaly detection methods are based on the analysis of network data itself in the present situation, which have no association with the state of the application. Thus, many researchers proposed that logs can be used to correlate application workflow and anomaly detection. Xiao et al. proposed a method to build an automaton for the workflow of each management task based on normal executions and then check log messages against a set of automata for workflow divergences in a streaming manner

[17] to detect anomalies. Du et al. proposed DeepLog, a deep neural network model to model a system log as a natural language sequence, which allowed DeepLog to learn log patterns from normal execution and detect anomalies when a log pattern deviates from the model trained from log data during normal execution [18]. Brown et al. presented recurrent neural network language models augmented with attention to anomaly detection in system logs [19]. Yao et al. proposed an anomaly detection method based on the multitask temporal convolutional network in cloud workflow, which relies on the feature extraction and recognition of event sequences and time sequences of tasks in the workflow [20]. Despite their contributions, these methods have several limitations. Firstly, the information recorded in logs is limited so that some anomalies can be missed. Secondly, all the methods are based on a distributed background, and it is difficult to aggregate logs of all components in a normal network. Furthermore, these methods are unable to detect abnormal packets in time as they do not combine the information of workflows with packets.

Thus, this paper combines network data packets with application behaviors to enhance the efficiency of network abnormal behavior detection and builds finite state machines from the source code level to record the normal workflows of applications.

## 3. Task-Oriented Model and definition

*3.1. Task-Oriented Architecture.* In order to achieve task-oriented implementation, our model divides the process of tasks into actions, states, and corresponding transitions. Meanwhile, security labels that include the sender's information of states and actions are generated and inserted into data packets. As we can see, the security labels carried by data packets are the essential bases for security decision-making. The processing of the information in data packet is shown in Figure 1.

Data layer of the task-oriented model consists of terminals, transponder, and switch, which are responsible for the transmission of packets. On the data layer, all data packets communicated between terminals must pass through the security label transponder, which extracts security labels and forwards them to the controller. The main component of the control layer is the controller that conducts the security decision-making. On the control layer, the controller makes the security decision on whether the data packet should be intercepted by the switch. According to the described functions, an example of the task-oriented network system and its workflow is depicted in Figure 2.

*Step 1.* Terminal 1 sends data packet with security label to terminal 2. Meanwhile, the encrypted state information of terminal 1 will also be sent to the controller.

*Step 2.* The packet sent in Step 1 must go through the transponder, which extracts the security label and forwards it to the controller. Then, the packet will be transported to the switch.

*Step 3.* After receiving the information sent in Step 1 and Step 2, the controller can make a security decision on whether the data packet should be intercepted and then send the decision to the switch.

*Step 4.* When the switch receives the decision from the controller, it can execute a routing strategy to block or transmit the packet.

*3.2. Formal Definition of Task-Oriented Model.* A task is executed by a certain device or program in accordance with a certain process. The form and state transitions of a program's task during the execution are called behaviors, which are embodied in the flow of actions performed on the entity. In order to successfully accomplish a task, some features should be complied with, such as safety, feasibility, normalization, and integrity [21]. We describe the task-oriented detection abnormal model as follows:

ACTION SET: $A = \{a\,1, a\,2, \ldots, an\}$ represents a series of actions executed by a task in a network system, and each action is a single step.

STATE SET: $s = \{s\,1, s\,2, \ldots, sn\}$ represents a collection of task states. A task can reach a certain state after it executes a certain set of actions, and there are continuity relations between states.

The transition between states requires a complete action set. The state transition function $T_{i\longrightarrow j}(A', s_i)$ represents that the state of task *si* turns to *sj* after all actions in the action set $A'$ are executed.

NORMAL DATA: it is assumed that the current state recorded in the controller is $s_i$. For the packet of communication between terminals, if the action $a_k$ and the state $s_j$ carried by the packet's security label satisfy that the value of $\mathrm{T}(a_k, s_j)$ is greater than 0, the data packet is normal. $\mathrm{T}(a_k, s_j)$ is defined as follows:

$$\mathrm{T}\left(a_k, s_j\right) = \begin{cases} 1, \text{if}\left(s_i == s_j\right)\text{or}\left(T_{i\longrightarrow j}\left(A', s_i\right) == s_j\right)\left(a_k \epsilon A'\right), \\ 0, \text{others}. \end{cases} \tag{1}$$

When $s_i$ is equal to $s_j$, that is, the current state does not change, the task remains in the same state. Besides, when the action $a_k$ triggers the normal transition $T_{i\longrightarrow j}(A', s_i)$, the packet is normal in the current state transition process. On the contrary, when the state is different from $s_j$ or not triggered by $a_k$, the related packet is abnormal. Therefore, we can define the abnormal data as follows:

ABNORMAL DATA: consistent with the above situation, if the value of $\mathrm{T}(a_k, s_j)$ equals 0, the data packet is labeled as abnormal.

NORMAL TASK: in the process of performing a complete task, the state moves from *s* 1 to *sn*. For any continuous task state $s_i$ and $s_j$, the following condition is satisfied: $s_j = \left\{T_{i\longrightarrow j}(A', s_i) | A' \subseteq A \text{ and } s_i \in S\right\}$. $A'$ represents the set of actions that triggers task's state

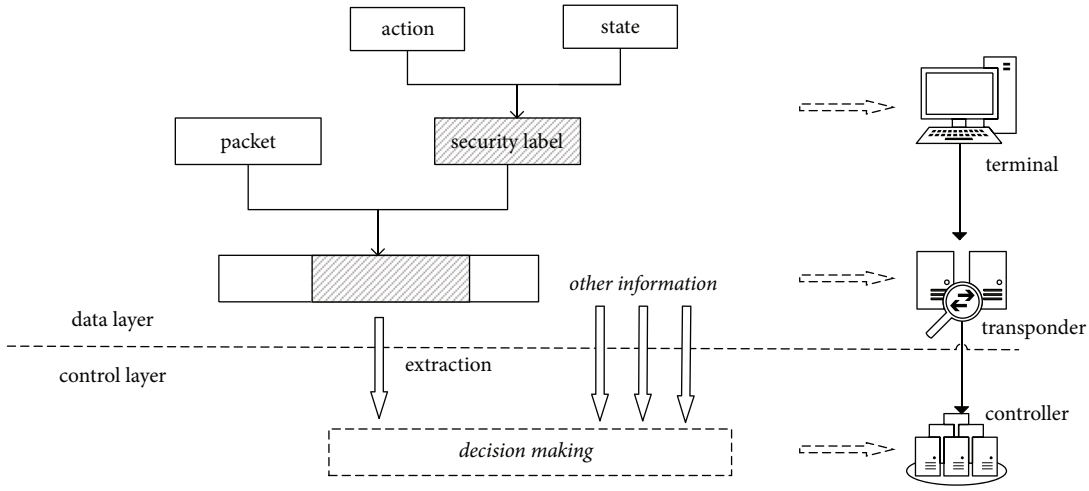



Figure 1: Task-oriented data packet processing flow.

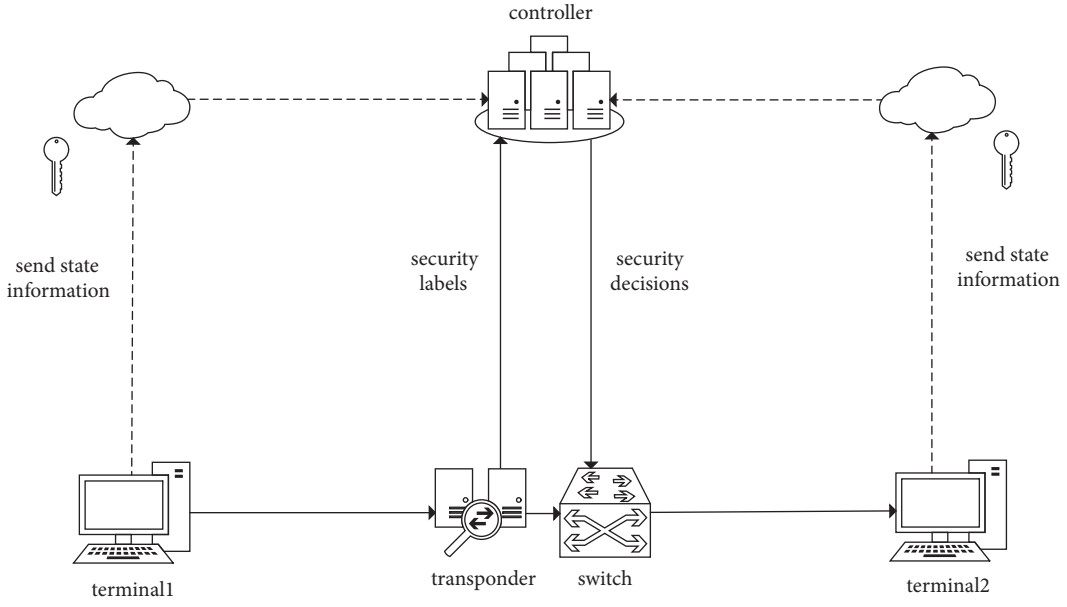



Figure 2: Task-oriented structure and workflow.

transition from state $s_i$ to state $s_j$. $s_j$ is not unique and $A'$ can be empty.

ABNORMAL TASK: consistent with the above situation, if the state transits under the condition that the action sequences are incorrect or the final state is not expected, the task is abnormal. That is, if there is $s_j = \{T_{i \longrightarrow j}(A'', s_i) | s_i \in S\}(A'' \neq A')$ or $s_k = \{T_{i \longrightarrow j}(A', s_i) | A' \subseteq A\}(s_k \neq s_j)$, the task will be judged as abnormal.

Based on the above formal descriptions, the abnormal behavior can be detected by the deduction of abnormal data packet. Besides, through the idea of task-oriented architecture, abnormal data packet is produced by abnormal task of related terminal. Hence, the relationship among these three concepts is shown in Figure 3.

## 4. Generation and Extraction of Security Label

*4.1. Security Label Generation.* In the task-oriented method, security judgement can be integrated into every step of data packet forwarding in the network. In order to associate current state and action information with network data packet, action sequence identification is designed in this paper. After being inserted into data packet, the action sequence identification becomes the security label of data packet and will be sent to the controller by the security label transponder.

By analyzing the structure of the IPv4 data packet, the action identification can be integrated into "Option field." As the Option field of IP packet is reserved for storing user-defined data, inserting related security information into this field will not affect other functions of data packet. In the inserting process, two key factors should be considered as follows.


Figure 3: The relationship among abnormal task, data, and behavior.

*4.1.1. The Max Length of Option Field Is 40 Bytes.* Option code field (1 byte): according to RFC791 [22], the 8-bit option code consists of a 1-bit copy flag, 2-bit option type, and 5-bit option number. As to option type, types "1" and "3" are suitable for customized identification. This paper selects the reserved type "3" as the customized type and the 11111 as the 5-bit customized option number. Therefore, the value of 8-bit option code field is 0x7F.

Action identification field (default 32 bytes): the Hash value of the action identification sequence is generated by the task of the terminal.

Communication random string field (default ≤ 7 bytes): the terminal's process and the corresponding controller process can obtain the communication random string Kcs through the key distribution algorithm.

If the length of Kcs is less than 7 bytes, adequate zeros can be used to fill it up. Based on the above considerations, the structure of the Option field can be designed as in Figure 4.

*4.1.2. Confidentiality and Antitampering of Option Field.* The steps for generating the option field are described in Figure 5:

(a) Generate the Hash value of the message actionM, which refers to the action identification: actionHash = Hash(actionM).

(b) Generate the communication random string as the salt, and get the data block: $M'$ = actionHash $||Kcs||$ padding.

(c) Generate the Hash value of the message $M'$: MHash = Hash($M'$).

(d) Fill (opType || MHash) in the Option field to construct a complete IP data packet.

As the above factors are considered, the security label can be inserted into original data packet. The Hash computation and *Kcs* in the generation steps can ensure the confidentiality and antitampering of the carried information that will be used in the future judgement mechanisms.

*4.2. Security Label Extraction Algorithm.* When data packet reaches the security label transponder, it is necessary to verify IP packet's header and extract the security label from the Option field. Then, the legal security label will be forwarded to the control layer. The procedure of processing a single packet can be described in Figure 6.

As shown in Figure 6, the header information can be extracted by analyzing and processing the IP data packet. If the length of the header is less than or equal to 0x05, the packet does contain an empty Option field and will not be forwarded. As we can see in Section 4.1.1, if the value of Option code field does not equal 0x7 F, the Option field is unreasonable and will not be forwarded either. Otherwise, the security label stored in the Option field will be extracted and forwarded to the controller. Meanwhile, the original IP data packet is normally transmitted to the data forwarding device. This paper proposes a security label extraction algorithm to implement the extracting process.

The input of Algorithm 1 is an array that consists of different elements. Each element corresponds to the value of a byte in a packet orderly. According to the structure of the IPv4 data packet, the ipStream[0] consists of the version number and the header's length. The version number is 4, and the header's length should be more than 5. So, the value of ipStream[0] should be greater than 0x45 (as shown in Line 2, Algorithm 1). Since ipStream [20] represents Option code field, the value should be equal to 0x7f (as shown in Line 3, Algorithm 1). As mentioned above, the version number is 4, so the header's length can be calculated by the formula in Line 4. The data from the 21st byte to the end is the security label (as shown in Line 5, Algorithm 1), which is the output of the algorithm.

## 5. Abnormal Behavior Judgement

*5.1. Task-Oriented FSM.* Finite state machine (FSM) is an abstract computing model [23], which is mainly used to model the behavior of objects and study the process of objects' life cycle. In order to abstract the process of a task, a basic model of FSM is established under the formal definitions in Section 3.2. Finite state machine is generally defined as a six-tuple $M$ = (S, $s$ 0, $A$, Tx, s′, $F$):

(i) S: finite state set.

(ii) $s$ 0: initial state, $s$ 0 ∈ S.

(iii) $A$: input action set.

(iv) Tx: state transition function. $T_{i\longrightarrow j}(A', s_i)$ represents that the state *si* turns to *s j* after all actions in $A'$ are executed ($A' \subseteq A$).

(v) s': the secondary state, s′ ∈ S.

(vi) $F$: end state set, $F \subseteq$ S.

The state transition of a finite state machine mainly involves four elements: the current state, conditions, the secondary states, and input actions. The conditions refer to the state transition rules. The input actions are the actions that should be actually executed, while the secondary state s′ is an entered state. $T_x$ refers to the transition function between two states. For example, $T_{i\longrightarrow j}(A', s_i)$ represents that the state of task *si* (initial state) turns to *s j* (secondary state) after all the actions in the action set $A'$ are executed.

*5.2. Abnormal Behavior Detection Mechanism.* The abnormal behavior detection mechanism in the proposed model is mainly divided into three parts: data packet authentication,

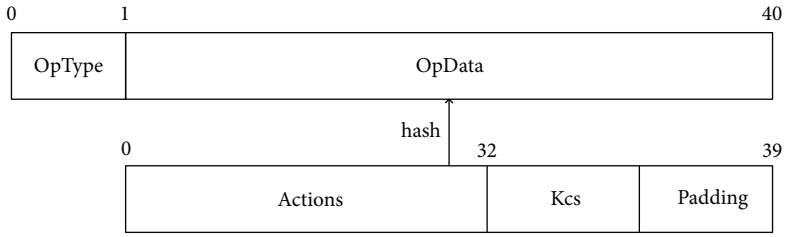



Figure 4: The structure of the option field.

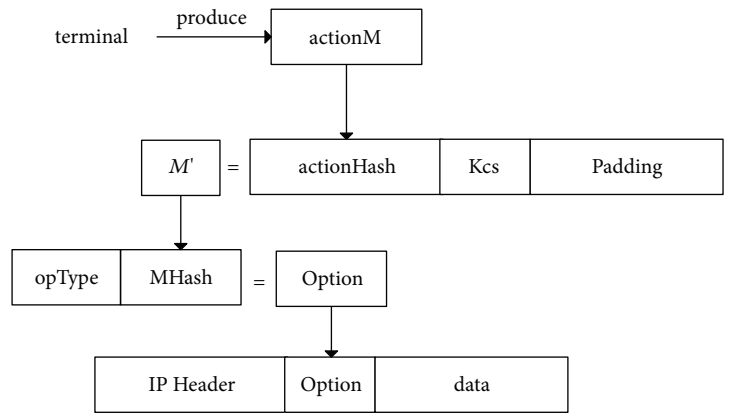



Figure 5: Generation of Option field.

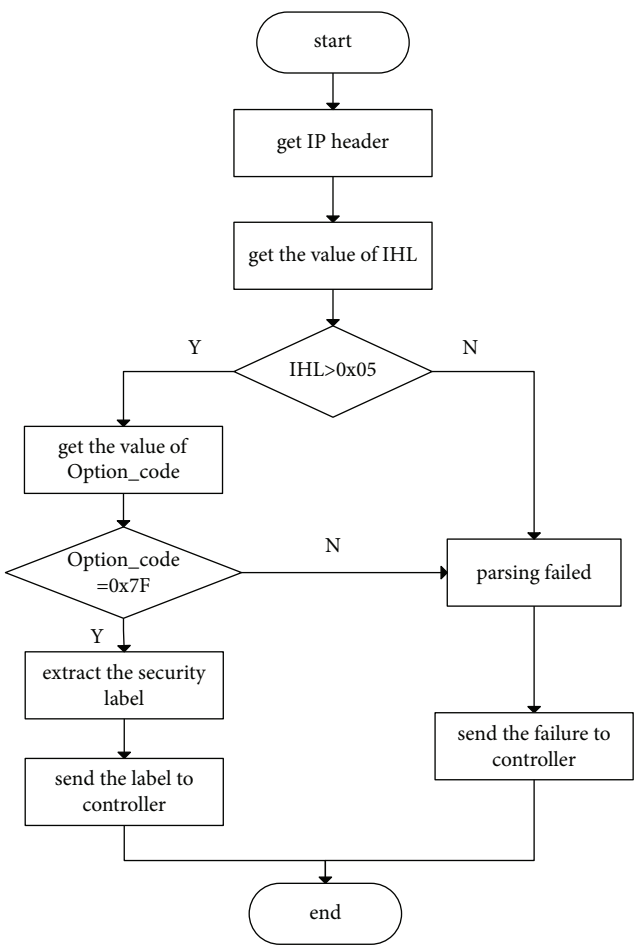



Figure 6: The procedure of processing a single packet.

action sequence identification matching, and security decision generation. The relationship between each part is shown in Figure 7.

The basis of data packet identity authentication is the communication random string Kcs which can only be obtained through the key distribution of the public key encryption system. Therefore, we can assume that the terminal process and the controller process can get the Kcs, while the attacker cannot obtain it. That is, even if an attacker can forge data packets, it cannot forge the security label in the Option field. The basis of action sequence identification matching is that if the action sequence of any data packet is changed by the attacker, the Hash value in the Option field cannot be matched successfully. The basis for generating safety decisions is the matching of actual states and actions with anticipant finite state machine. When the current state and action of the task are reasonable, the related data packet is forwardable. Otherwise, if any state transition is recognized as not within the scope of finite state machine, the related data packet will be intercepted. We can recognize incorrect state transitions through the abnormal behavior detection algorithm that is described as Algorithm 2.

The encrypted message from the terminal contains the state and action information that is encrypted by DES. The current state and action can be acquired by decrypting the message. The $D$ function is used to decrypt the message (as shown in Line 3, Algorithm 2). Then, the Hash value is computed by MD5 to verify the security label (as shown in Line 4, Algorithm 2). The $T$ function described in 3.2 is used to distinguish abnormal data from normal data. If the value of $T$ function equals 0, the behavior is judged as abnormal (as shown in Line 5, Algorithm 2). If the security label can match

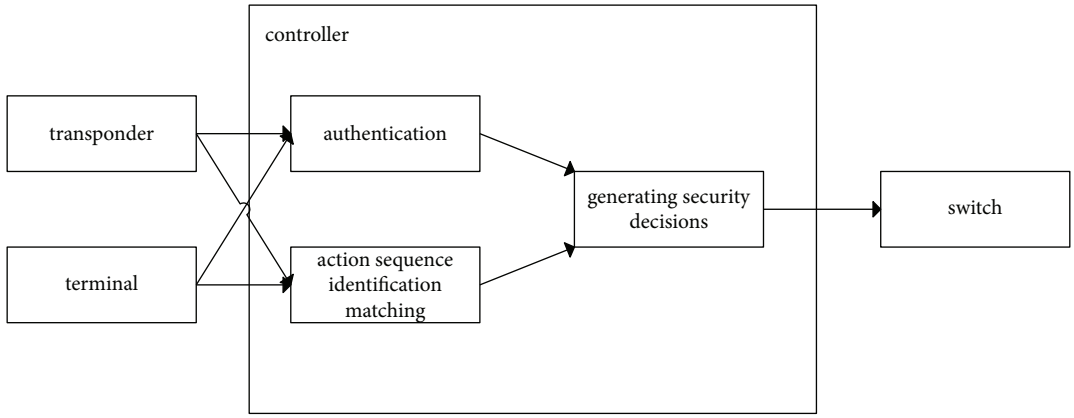



Figure 7: The logic of abnormal behavior detection.

with the action and the value of $T$ function equals 1, it means no abnormal behavior and the decision signal; that is, the output of the algorithm is true. Otherwise, the output is false (as shown in Line 6–12, Algorithm 2).

## 6. Evaluation

In our experiment, a network system is established based on FTP [24] to test the file transfer in both normal and abnormal environments. The prototype system is mainly composed of server, client, security label transponder, switch, and controller. The network topology diagram of the experimental system is shown in Figure 8.

*6.1. State and Action.* According to the division of the FTP workflow, the entire communication process can be divided into five states as shown in Table 1.

After the FTP process is started, the START state is entered. After inputting the user's name and password, a control connection and a data connection will be created, and the CONNECT state is entered. Meanwhile, the server will authenticate the user's name and password provided by the client, which is the AUTHENTICATION state. After the above procedures are all completed, the client can upload and download files, which is the TRANSFER state. After finishing the client process, the FSM will turn to the CLOSE state. The final finite state machine is constructed as M_FTP = (S, $s$ 0, $A$, Tx, s', $F$):

(i) Finite state set: $S$ = {CLOSE, START, CONNECT, AUTHENTICATION, TRANSFER}.

(ii) Initial state: $s$ 0 = START.

(iii) Input action set: $A$ = {Input_login, Input_fPath, ..., ftp_close}.

(iv) State transition function: Tx.

(v) The secondary state: s'.

(vi) End state set: $F$ = {CLOSE}.

After analyzing the source code of FTP, we can get the mapping relation between states and functions. The sequence of the functions is shown in Figure 9.

Take downloading a single file as an example. The main action identifications in the FTP process are divided into multiple types, such as human-computer interaction, memory (cache) reading, and memory (cache) writing. The human-computer interaction part can be further subdivided into keyboard inputting and mouse clicking. The action identifications of basic function are subdivided into specific action identifications such as establishing a socket connection and sending commands. The memory (cache) reading and writing identifications are subdivided into sending data to the system cache and reading data from the system cache. Table 2 shows all actions in the process of downloading a single file.

*6.2. Normal FTP Workflow.* In the process of file transmission under our architecture, the data packet is sent by the client and passes through the secure label transponder. The secure label transponder extracts the security label and forwards it to the controller. The security authentication module in the controller uses the corresponding communication random string Kcs to decrypt the action identification sequence sent by the FTP client. If the decryption result is empty, it means that the identity authentication fails due to the decryption error. Otherwise, the Hash authentication on the security label is executed. If the Hash value is matched correctly, action identification matching module is triggered to verify state and action information. The controller combines the current state of the task at the client with the single packet action identification sequence and compares related information with the finite state machine model to determine whether the behavior is abnormal. Under normal executing conditions, the running results on the controller are shown in Figure 10.

*6.3. DDOS Attack.* DDOS is an attack method that exhausts host resources and ultimately results in failure to respond to requests from normal users in a timely manner [25]. The attacker starts a daemon on client and uses the daemon to send data packets with port number 21 to simulate FTP data packets [26]. Some terminals are used to simulate zombies as shown in Figure 11. To simulate this attack on our experimental platform, a new process that can send the same packets to request a control connection continually should be created on these zombies. The port numbers of these packets are all 21 to masquerade themselves as sent by

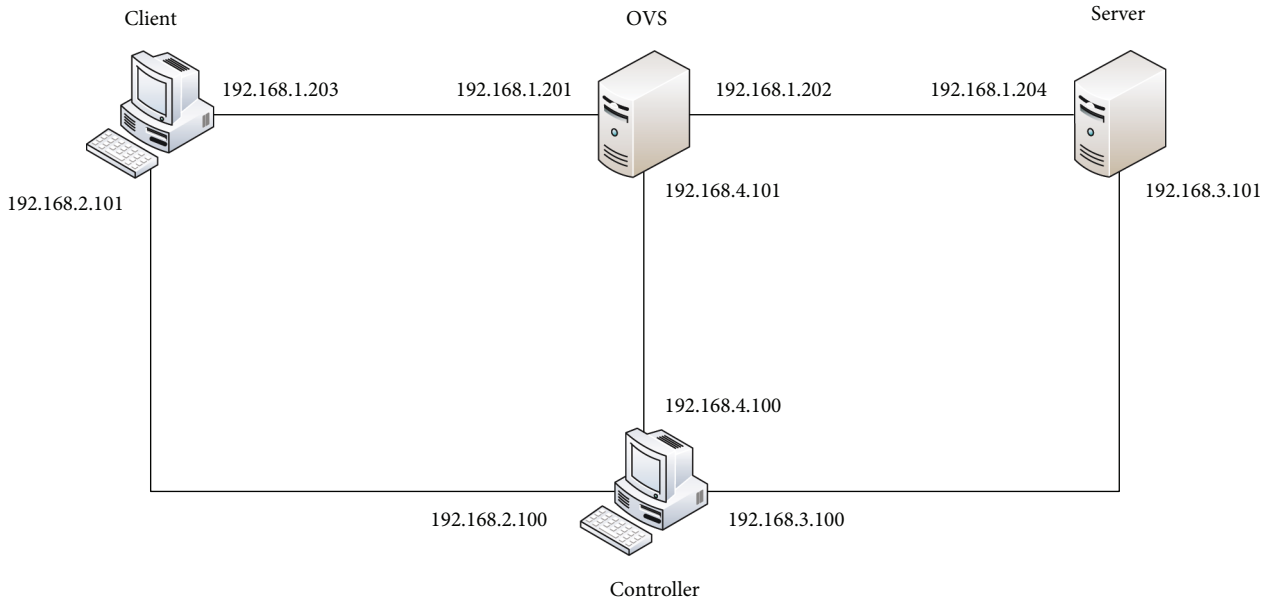



Figure 8: Network topology diagram of the experimental platform.

```
    Input: IPv4 data packet: ipStream
    Output: Security label: secLabel
(1) secLabel = []
(2) if ipStream[0] > 0x45 then
(3)    if ipStream [20] = = 0x7f then
(4)       headerLen = (ipStream[0] - 0x40) <<2
(5)       secLabel = ipStream[20, headerLen - 1]
(6)    end if
(7) end if
(8) return secLabel
```

Algorithm 1: Security label extraction algorithm.

the FTP process. However, the attacker does not know the random string Kcs of the communication between the real FTP client process and controller process. Therefore, compared with the normal security label, the data packet forged by the attacker is incomplete. The action identification sequence that is sent to the controller cannot be decrypted successfully by the controller using Kcs. It will be judged as abnormal in the authentication module.

In Figure 12(a), since DDOS attack uses an abnormal Kcs, an abnormal signal "The Kcs is false!" is generated on the controller in Line 4. The judgement is "Identify failed!" in Line 5. Because of the failure of identity authentication, action identification sequence matching is not performed. The result "Fail!" is directly generated as shown in Figure 12(b) and the packets generated by DDOS are intercepted.

*6.4. Backdoor Attack.* Backdoor attack is a secret way to access program and online service [27]. Attackers can install a backdoor to bypass the system's conventional security control rules, thereby gaining control over program or system [28]. In order to simulate a backdoor attack, we set up a backdoor username in FTP and initialize a global variable of Boolean type bd_flag to False. When the login succeeds with the backdoor username, bd_flag is changed to True. After the control connection is generated, the program can jump to a judgement. If the bd_flag is True, some files stored in the FTP server will be sent to a non-FTP port accessible to the attacker by the send() function. The flow of the simulated backdoor attack is shown in Figure 13.

The main purpose of the backdoor attack in this paper is to transmit information to a non-FTP port, causing information leakage as shown in Figure 14. The backdoor attack is mainly different from the normal transmission in the action identification sequence.

In Figure 15(a), as all operations are carried out in the FTP process, the judgement generates "Identify succeed!" in Line 46. Thus, the authentication module successfully passes the packet's identification. In the matching module of the action identification sequence, an action identification for initiating a socket connection to an unknown port is found in Line 48. This action identification does not exist in the action identification library. Thus, an action matching exception will occur, which leads to the final decision result "Fail!" as shown in Line 13 of Figure 15(b).

### 6.5. Performance Analysis

*6.5.1. Packet Transmission Delay.* Transmission delay refers to the total transmission time consumed by data packets that are sent from the client to the server. In this paper, Wireshark is used to capture network packets on the client and the server, respectively. $T_{\text{client}}$ indicates the time when the

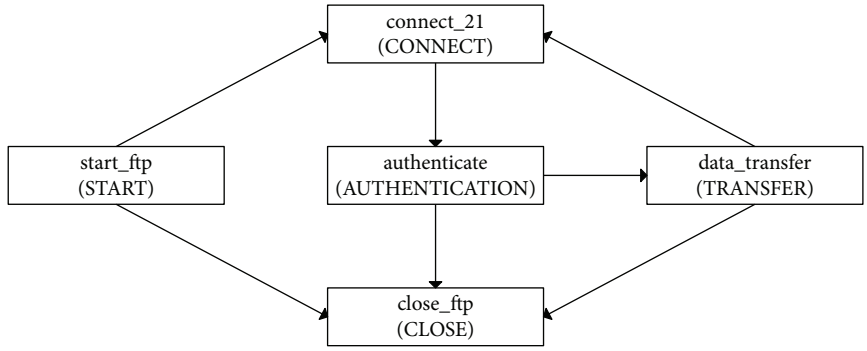
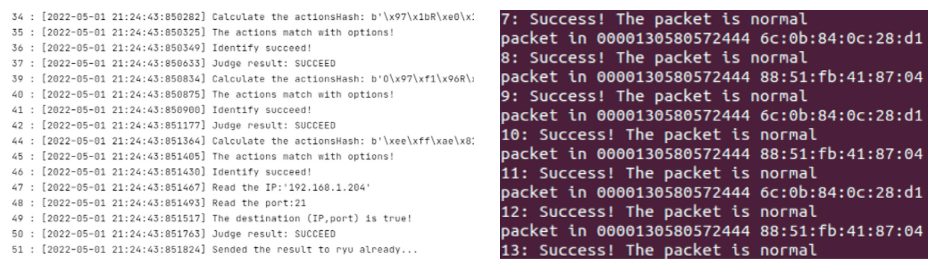



Figure 9: FTP function transition diagram.

Input: Security label: secLabel The encrypted message from the terminal: EncryptedMessage Communication random string: Kcs
Output: Abnormal behavior decision signal: signal
(1) authSig = True
(2) actionSig = True
(3) currentState, action = $D$(Kcs, EncryptedMessage)
(4) if Hash(Hash(action) || Kcs) = = secLabel then
(5) if T(action, currentState) = = 0 then
(6) actionSig = False
(7) end if
(8) else
(9) authSig = False
(10) end if
(11) signal = actionSig andand authSig
(12) return signal

Algorithm 2: Abnormal behavior detection algorithm.




. (a) (b)

Figure 10: Running results on the controller under normal conditions. (a) The process of judgement. (b) The result of judgement.

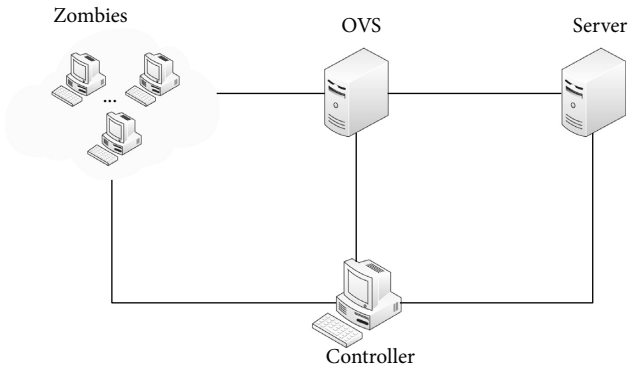
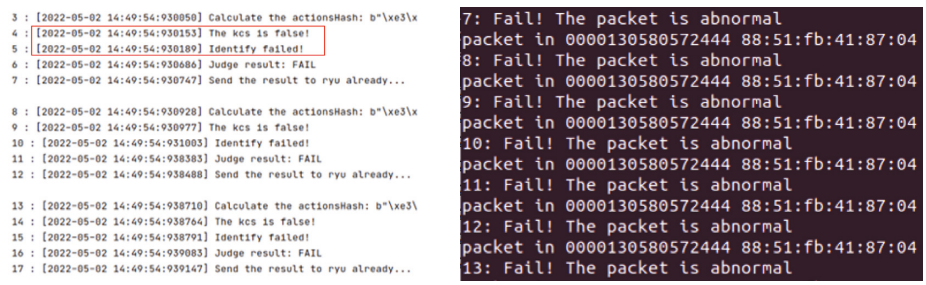



Figure 11: The simulation of DDOS attack.

client sends a data packet, while $T_{server}$ indicates the time when the server receives the same data packet. The delay time of file transfer on the network with and without our task-oriented network abnormal behavior detection method is compared. The delay time is computed as follows:

$$T_{delay} = T_{server} - T_{client}. \quad (2)$$

The Cumulative Distribution Function (CDF) [29] is used to describe the probability distribution of the transmission delay and is mainly used to visually express the occurrence probability of transmission delay in a certain range. The value of CDF can be calculated by the following formula:




(a) (b)

Figure 12: The running result of DDOS attack on the controller. (a) The process of judgement. (b) The result of judgement.

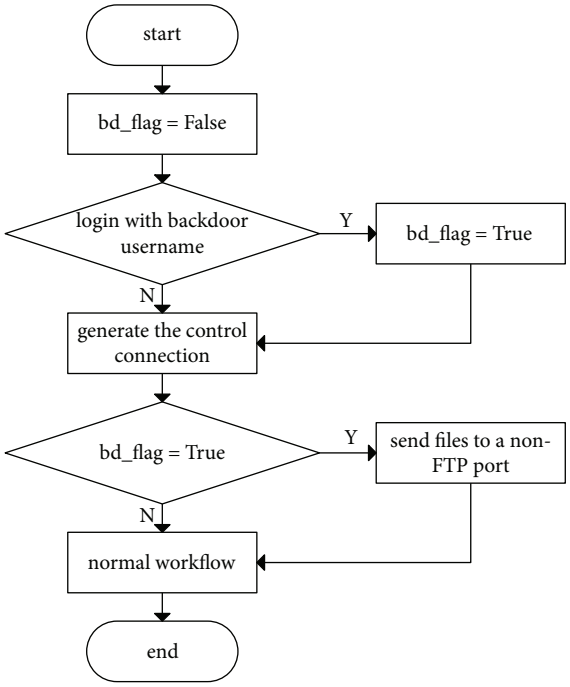



Figure 13: Flowchart of the simulated backdoor attack.

$$CDF = \frac{\#\text{ of packets\_in\_}T}{\#\text{ of all\_packets}}. \tag{3}$$

# of packets_in_$T$ represents the number of packets whose transmission delay is less than Tms. # of all_packets represents the number of the total sample packets.

The CDF comparison of the original network system and the modified network system with the proposed method is shown in Figure 16.

In the original network system, 82.6% of the packet transmission delay is less than 0.23 ms, while 71% of the packet transmission is less than 0.23 ms in the modified network system of this paper. The data packet transmission delay of the original network system mainly ranges from 0.18 ms to 0.53 ms, with an average of 0.34 ms, while that of the network system in this paper mainly ranges from 0.25 ms to 0.58 ms, with an average of 0.43 ms. Packet transmission latency increases by 0.09 ms on average or 26% on average.

The above experimental results show that the task-oriented abnormal behavior detection method can improve the defense ability of the network system and cause little transmission delay of data packets.

*6.5.2. Network Packet Detection Accuracy.* In our analysis, $C_{TP}$ denotes the number of correctly classified abnormal packets, $C_{TN}$ denotes the number of correctly classified benign packets, $C_{FN}$ denotes the number of incorrectly classified abnormal packets, and $C_{FP}$ denotes the number of incorrectly classified benign packets. The sum of all the collected packets can be denoted by $C_{all} = C_{TP} + C_{TN} + C_{FP} + C_{FN}$.

Thus, we calculate the detection accuracy (Accuracy), false positive rate (FPR), and false negative rate (FNR) by using the following formulas:

$$\text{Accuracy} = \frac{C_{TP} + C_{TN}}{C_{all}}, \tag{4}$$

$$FPR = \frac{C_{FP}}{C_{TN} + C_{FP}}, \tag{5}$$

$$FNR = \frac{C_{FN}}{C_{TP} + C_{FN}}. \tag{6}$$

We conduct experiments and collect data packets in three different scenarios to evaluate the performance of the proposed method. Firstly, in the absence of simulated attacks, we generate 1000 data packets to be sent by FTP client. Then, DDOS and backdoor attacks are triggered to generate abnormal packets. In the DDOS attack scenario, 500 packets are sampled, among which 371 are abnormal and 129 are benign. The detection results show that 373 of the sampled packets are judged as abnormal and 127 as benign. In the backdoor attack scenario, we also sample 500 packets, among which 92 are abnormal and 408 are benign. The detection results show that 87 of them are judged as abnormal and 413 as benign. According to the comparison between the detection results of the proposed method and the correct results, the values of $C_{TP}$, $C_{TN}$, $C_{FP}$, and $C_{FN}$ in different scenarios are shown in Table 3.

According to Table 3, the values of the above three parameters Accuracy, FPR, and FNR can be calculated as shown in Table 4.

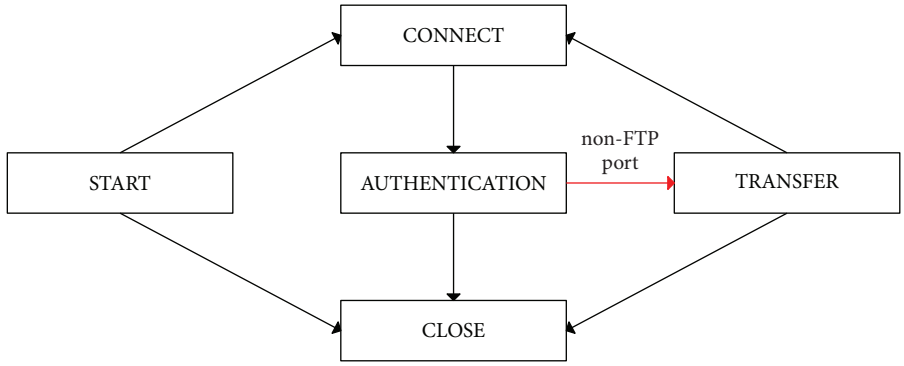
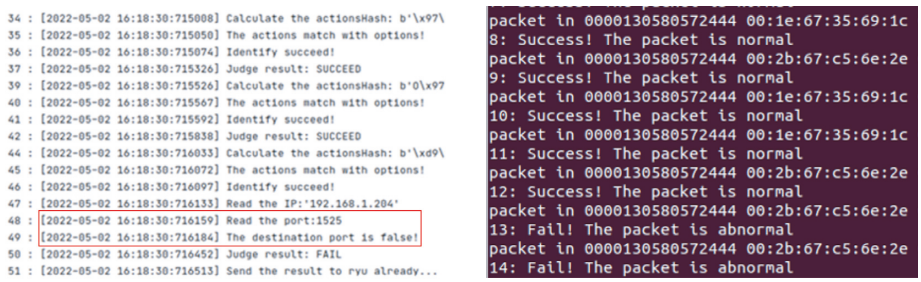



Figure 14: State transition diagram of backdoor attack.




. (a) (b)

Figure 15: The running result of backdoor attack on the controller. (a) The process of judgement. (b) The result of judgement.

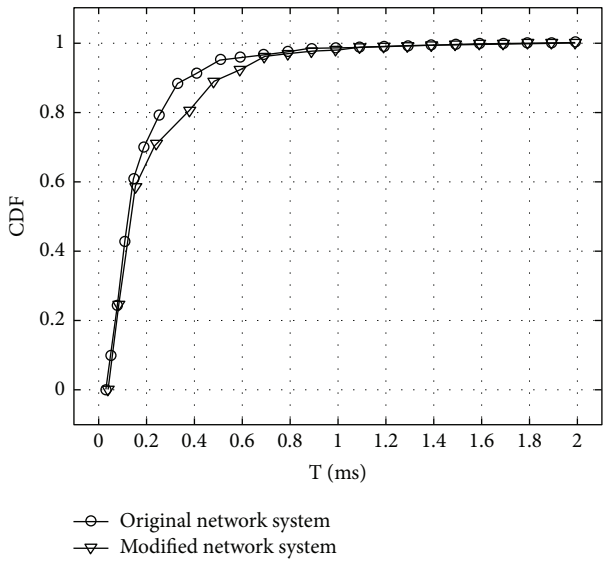



Figure 16: The CDF of transmission delay.

Table 1: Correspondence between state and function.

| State name | Function ID |
|---|---|
| START | start_ftp |
| CONNECT | connect_21 |
| AUTHENTICATION | authenticate |
| TRANSFER | data_transfer |
| CLOSE | close_ ftp |

Table 4 shows that the detection accuracy in normal communication can reach 95%, with a 5% false positive rate, meaning that the normal communication function of the network system is not affected. The accuracy of the network system can reach 80% in the case of DDOS attack, with a 12% false positive rate and a 4% false negative rate. As most attack packets are intercepted, the DDOS attack cannot be effective. In the case of backdoor attack, the proposed method can reach an accuracy of 87%, with a 1% false positive rate and an 11% false negative rate. Thus, the stolen files by the backdoor attack cannot be sent to the unknown port completely.

*6.5.3. Security Performance Analysis.* In this paper, if a packet is generated by normal behavior, it will be continuous with the state and action of the packet that was previously generated by the process. Therefore, it can successfully pass the authentication module and the abnormal behavior detection module without an abnormal signal. When the packet is generated by abnormal behavior, it can be divided into two cases. First, if the data packet is generated by the attacker disguising the target process, the identity authentication module of the controller will generate abnormal signal. Second, if the attacker deviates from the normal execution of the target process, the abnormal behavior detection module in the controller will output an abnormal signal.

Table 2: Actions of downloading a single file.

| Type | Action identification |
|---|---|
| Human-computer interaction Action identifications | Input user's information: $a_{1\_1}$: Input_login<br>Input file path: $a_{1\_2}$: Input_fPath<br>Click connect button: $a_{2\_1}$: click_connect_button<br>Click download button: $a_{2\_2}$: click_download_button<br>Click close button: $a_{2\_3}$: click_close_button<br>Double-click to start application: $a_{2\_4}$: double_click<br>Initialization socket_fd: $a_3$: socket_init |
| Basic function Action identifications | Create a socket connection: $a_4$: socket_create_connect_[PORT]<br>Initiate a control connection: $a_5$: ftp_connect_21<br>Send cmd command: $a_6$: ftp_sendcmd_[CMD]<br>Download a single file: $a_7$: ftp_retrbinary |
| Cache Action identifications | Send data to cache: $a_8$: send_[SIZE]_[ADDR]<br>Read data from cache: $a_9$: recv_[SIZE]_[ADDR] |
| State jump Action identifications | Socket connection successful: $a_{10}$: connect_success_[PORT]<br>Close socket connection: $a_{11}$: socket_close_[ADDR]<br>Log in successfully: $a_{12}$: login_success<br>Close the application: $a_{13}$: ftp_close |

Table 3: The values of CTP, CTN, CFP, and CFN.

| | CTP | CTN | CFP | CFN |
|---|---|---|---|---|
| Normal | 0 | 950 | 50 | 0 |
| DDOS attack | 357 | 113 | 16 | 14 |
| Backdoor attack | 82 | 403 | 5 | 10 |

Table 4: The values of Accuracy, FPR, and FNR.

| | Accuracy (%) | FPR (%) | FNR |
|---|---|---|---|
| Normal | 95 | 5 | 0 |
| DDOS attack | 80 | 12 | 4% |
| Backdoor attack | 87 | 1 | 11% |

## 7. Conclusion

This paper proposed a task-oriented abnormal behavior detection method, which realizes the linkage defense of the entire network system. By inserting security labels, the association of data packets and tasks is realized. We established a finite state machine under normal conditions and monitored the states and actions in real time to detect whether there is any abnormal behavior. Under the FTP experimental platform, two common attacks were carried out, which proved the effectiveness of this method.

Nevertheless, the method proposed in this paper still needs to be further improved. The generation of FSM for more complex application requires adding tags to the code to get the state transitions and action sequences precisely. The automation method is also required to generate FSM. Furthermore, the transponder can become a bottleneck of our method, which needs more efficient mechanisms for extracting and forwarding security labels. In addition, in the case of large data volume, the capacity of our method still needs to be improved. We leave these areas that can be improved as open problems and our future work.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This research was supported by the Fundamental Research Funds for the Central Universities (no. 2242022k30007), Purple Mountain Laboratories for Network and Communication Security, and National Science Foundation (no. 61601113).

## References

[1] Y. Jia, X. W. Wang, and W. H. Han, "YHSSAS: security situation awareness system for large-scale networks," *Computer Science*, vol. 38, no. 02, pp. 4–8, 2011.

[2] N. McKeown, "Software-defined networking," in *Proceedings of the INFOCOM Key Note*, Janerio, Brazil, April 2009.

[3] X. G. Li, "Survey and research on network traffic anomaly detection," *Henan Science and Technology*, no. 31, pp. 10–12, 2018.

[4] M. R. Wei, X. Qi, and J. L. Ren, "Analysis of host anomaly behavior based on stream data feature matching," *Information Security Research*, vol. 3, no. 5, pp. 469–476, 2017.

[5] A. L. Buczak, E. Guven, and G. Erhan, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[6] J. Ma, L. Sun, H. Wang, Y. Zhang, and U. Aickelin, "Supervised anomaly detection in uncertain pseudoperiodic data streams," *ACM Transactions on Internet Technology*, vol. 16, no. 1, pp. 1–20, 2016.

[7] M. Shao and N. Gu, "Anomaly detection algorithm based on semi-supervised collaborative strategy," *Journal of Physics: Conference Series*, vol. 1944, no. 1, Article ID 012017, 2021.

[8] M. E. Villa-Pérez, M. Á. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo, "Semi-supervised anomaly detection algorithms: a comparative summary and future research directions," *Knowledge-Based Systems*, vol. 218, Article ID 106878, 2021.

[9] C. Zhang, D. Song, Y. Chen et al., "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1409–1416, 2019.

[10] V. Papastefanopoulos, P. Linardatos, and S. Kotsiantis, "Unsupervised outlier detection: a meta-learning algorithm based on feature selection," *Electronics*, vol. 10, no. 18, p. 2236, 2021.

[11] K. Fotiadou, T.-H. Velivassaki, A. Voulkidis, D. Skias, S. Tsekeridou, and T. Zahariadis, "Network traffic anomaly detection via deep learning," *Information*, vol. 12, no. 5, p. 215, 2021.

[12] N. Gao, L. Gao, and H. Y. He, "A lightweight intrusion detection model based on autoencoder network with feature reduction," *Acta Electronica Sinica*, vol. 45, no. 3, pp. 730–739, 2017.

[13] A. Javaid, "A deep learning approach for network intrusion detection system," *EAI Endorsed Transactions on Security and Safety*, vol. 3, no. 9, pp. 1–6, 2016.

[14] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019.

[15] S. Q. Dong and B. Zhang, "Network traffic anomaly detection method based on deep features learning," *Journal of Electronics and Information Technology*, vol. 42, no. 3, pp. 695–703, 2020.

[16] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2019.

[17] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, "CloudSeer," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 489–502, 2016.

[18] M. Du, F. Li, and G. Zheng, "DeepLog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, TX, USA, November 2017.

[19] A. Brown, A. Tuor, and B. Hutchinson, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the 2018 1st Workshop on Machine Learning for Computer System*, p. 8, June 2018.

[20] J. Yao, C. L. Cheng, and J. Han, "Anomaly detection method based on multi-task temporal convolutional network in cloud workflow," *Computer Applications*, vol. 41, no. 6, pp. 1701–1708, 2021.

[21] Y. Zhao, J. Q. Liu, and Z. Han, "Research on access control model based on task," *Computer Engineering*, no. 5, pp. 28–30, 2008.

[22] J. Postel, "RFC 791: Internet Protocol," 1981, https://www.rfc-editor.org/info/rfc791.

[23] H. X. Sun, W. Xing, and L. Tao, "Research of model transformation approches based on finite state machine," *Computer Technology and Development*, vol. 22, no. 2, pp. 10–13, 2012.

[24] A. K. Bhushan, "RFC 114: File Transfer Protocol," 1971, https://www.rfc-editor.org/info/rfc114.

[25] W. Stallings, *Cryptography and Network Security*, Electronic Industry Press, Canada, 2006.

[26] Y. S. Dai, Y. P. Xiang, and Y. Pan, "Bionic autonomic nervous systems for self-defense against DoS, spyware, malware, virus, and fishing," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 9, no. 1, pp. 1–20, 2014.

[27] Q. Wang, *Network Attack and Defense Technology*, Tsinghua University Press, Beijing, China, 2019.

[28] S. Amit, "Study of the SSL and backdoor based attacks in network environments," *International Journal of Energy Sector Management*, vol. 6, no. 3, pp. 136–144, 2019.

[29] X. L. Bi, Y. M. Qiu, and B. Xiao, "Histogram equalization detection based on statistical features in digital image," *Chinese Journal of Computers*, vol. 44, no. 2, pp. 292–303, 2021.