Efficient SVH2M for information anomaly detection in manufacturing processes on system call

Chao-Hsien Hsieh[1], Fengya Xu[2*], Qingqing Yang[2], and Dehong Kong[2]
[1] College of Engineering, Xi'an International University
Xi'an Shaanxi 710077, China
[e-mail: george_hsieh@qq.com]
[2] School of Cyber Science and Engineering, Qufu Normal University
Qufu Shandong 273165, China
[e-mail: 822215180@qq.com]
[*]Corresponding author: Fengya Xu

Received April 25, 2024; revised July 17, 2024; accepted September 29, 2024; published October 31, 2024

Abstract

With the integration of the manufacturing process in the Internet, cybersecurity becomes even more important in the process of factory operations. Because of the complexity of data traffic in the manufacturing industry, the identification and classification of anomalous behavior is an important direction of current research. System calls are made at the operating system level. Therefore, the use of system call sequences can detect potential threats much earlier. So, this paper chooses system call information as the research object. System call orderliness is an ideal property for analysis of using hidden Markov model. In terms of methodology, the SVH2M model improves the performance and efficiency of attack detection in manufacturing systems. The SVH2M model combines pSVM with mHMM. The pSVM and mHMM models use SVMPSA and PATA. pSVM is first used to initially categorize the system call sequences into normal and abnormal categories. The classification of pSVM can reduce the amount of data. This reduces the error rate of mHMM processing. Next, mHMM is built for different types of known anomalies. The SVH2M model in the false positive rate is lower than that of hidden Markov model. The experimental results show that the AUC of the improved model is increased by 17%. The average Mismatch Rate is reduced by 16%. The performance and efficiency of detecting anomalous information are improved in manufacturing systems.

Keywords: Hidden Markov model, support vector machine, system call, intrusion detection.

1. Introduction

Cyber attacks have become increasingly common over the past decade. Cyber attacks often make headlines on industrial companies [1]. However, intrusion detection systems (IDS) are used in manufacturing processes to secure various devices and networks from potential threats. With the popularity of industrially connected devices, cybersecurity is becoming increasingly important in manufacturing processes [2]. Manufacturing equipment is often connected through networks to achieve automation and maximize efficiency. However, this also gives hackers plenty of attack surface [3]. In the manufacturing process, attack surfaces can arise in several situations. As more manufacturing equipment are connected to the network, every device can be a target for hackers. Many manufacturing companies still be using outdated software and hardware. Intruders have accessed through unencrypted data transmissions. They also exploit software vulnerabilities in devices to carry out attacks. This not only threatens the security of manufacturing equipment, but also lead to production data leakage. Therefore, the use of IDS to enhance network security is essential for manufacturing processes.

There are diverse cybersecurity threats in the manufacturing environment [4], for example, malware attacks, permission attacks, cloud service attacks, distributed service denial attacks, and so on. When managers understand the attack behavior, they can effectively fix the defects. There are generally two methods. First, many research papers on IDS use network traffic as the source of data analysis. The dataset of DARPA [5-6] was used as the basis for test verification. It is suitable for training the dataset of IDS. But DARPA's data takes a long time to collect and process. This dataset is not representative of the behavior patterns of existing network traffic. Second, one approach of intrusion detection is to use system calls [7-9] as the source of training and testing. System call sequences are recorded data sequentially over time [10]. Compared with the amount of real-time production information, it is easier for system calls to find the characteristic behavior patterns of manufacturing process.

The hidden Markov model (HMM) is a random process with an underlying finite state structure. But HMM has some disadvantages such as poor classification ability and poor pattern recognition ability. One approach is to integrate artificial neural networks into the HMM architecture to improve performance. Artificial neural network can reduce the number of parameters. It encounters some problems such as local minimum, slow estimation process, and weak generalization. Support vector machine (SVM) shows its powerful classification performance. SVM is a general-purpose learning machine based on structural risk minimization (SRM). It is also a universal learning machine based on limit sample data. SVM has been applied in many fields such as classification, time series estimation, and function approximation.

While the traditional SVM model boasts powerful classification capabilities, it performs poorly when dealing with nonlinear and non-Gaussian distributed data, and struggles to capture temporal information within the data. On the other hand, while the HMM model excels at processing temporal data, its modeling capabilities are limited when confronted with high-dimensional and complex data structures. In rapidly evolving manufacturing environments, the demand for real-time anomaly detection is increasing. Traditional anomaly detection methods often fail to meet those requirements which is both leading to detection delays and rising false alarm rates.

The integration of pSVM and mHMM fully leverages the strengths of both. By incorporating a probabilistic framework, pSVM enhances its ability to model complex data and partially mitigates the complexity of parameter tuning. Meanwhile, mHMM, through its multimodal or modified approach, reinforces its capacity to process temporal data, thereby improving detection real-time performance and accuracy. By fusing the classification capabilities of pSVM with the temporal modeling prowess of mHMM, our method can more accurately identify anomalous patterns in the manufacturing process. Thereby it significantly reduces false alarm rates. Compared to single models, the integrated model demonstrates superior efficiency when dealing with complex data.

In this paper, the improved SVM model by the Particle Swarm Optimization algorithm is referred to as the pSVM for brevity. The HMM model specifically constructed for different types of known anomalies is referred to as the mHMM. pSVM and mHMM are combined to construct a SVH2M model for anomaly detection. This hybrid model can effectively improve the training efficiency of Markov model. At the same time, it can realize abnormal detection in the manufacturing process. To compare with the traditional hidden Markov model, the hybrid model has a lower mismatch rate. By applying this model, the performance and efficiency can be improved in manufacturing system. And the attack behavior can be accurately identified in manufacturing process. The contributions of this paper are as follows.

1) Design a simplified architecture with the industrial DMZ in the center. This architecture separates IT from OT. This approach reduces the likelihood of external attackers gaining access to the internal network. It enhances overall network security.

2) Develop both novel pSVM and mHMM. pSVM is an improvement of SVM with PSO algorithm. mHMM constructs a multiple HMM models for different types of exceptions.

3) Combine pSVM and mHMM to create a robust framework for pattern recognition and classification. This combination leverages for the strength of pSVM in fast filtering to reduce data volume for analysis. mHMM is applied to delve into the temporal characteristics.

This paper mainly combines pSVM and mHMM to realize information anomaly detection on system call. The first section mainly introduces the background of this paper. It is emphasized that there are cybersecurity issues in the manufacturing environment. The second section summarizes the related research of intrusion detection. The third section proposes the overall framework and algorithm. The fourth section verifies the advantages of SVH2M model by experiments. The fifth section is the summary of this paper and the description of the future research content.

2. Related Research

2.1 Intrusion detection system (IDS)

Intrusion detection is an active security defense technology. It detects and prevents potential attacks by monitoring network traffic and system behavior in real time. Also, it protects the security of enterprises, individuals, and data. Intrusion detection technology is widely used in various fields, such as enterprise network, cloud computing, Internet of Things, etc. **Table 1** summarizes the different methods used for intrusion detection.

SVM, as an effective classifier, has been widely used in pattern recognition and machine learning. By extracting the features of network traffic and using SVM to classify, the intrusion behavior can be accurately detected [11]. To improve the performance of SVM in intrusion detection, researchers have carried out optimization of the algorithm. For example, by combining feature selection techniques such as principal component analysis (PCA), the dimension of features can be reduced [12]. In addition, there are studies which combine SVM with other algorithms to improve detection accuracy and efficiency, such as ant colony optimization (ACO) [13]. Although SVM-based intrusion detection system has made some achievements in the research, there are still some challenges and future research directions.

For example, there is relatively little research on mixed abuse and anomaly detection systems [14].

Table 1. Methods of intrusion detection

Paper	Application Field	Methods	Solved Problem
[11]	Internet of Things	SVM	Improve the intrusion detection system
[12]	Intrusion detection	SVM	Select the appropriate kernel function for the SVM
[13]	Intrusion detection	SVM	Analyze network traffic
[15]	Interconnected network physical systems	HMM	Effectively detect hidden attacking scenarios
[16]	Vehicle CAN bus network	HMM	Attack on exception
[17]	Network security	HMM	Detect the occurrence of multiple MSA
Our work	Intrusion detection	pSVM +mHMM	Improve the intrusion detection system

In recent years, intrusion detection systems based on HMM have been favored for their advantages in processing sequence data [15]. For example, Dong et al. [16] proposed a CAN bus intrusion detection system based on multiple observation HMM. It is used for the security protection of the vehicle network. Similarly, Shawly et al. [17] used HMM to design a detection architecture for cyber-physical systems. The study verifies the effectiveness of HMM in intrusion detection.

Intrusion detection can be divided into two main categories, misuse detection and anomaly detection. Misuse detection [18] is based on a set of negative behaviors. By comparing the database of negative behavior, it can determine an intrusion behavior. Anomaly detection [19] uses a positive behavior data. This paper combines the misuse detection and anomaly detection. The initial classification using by SVM is part for the principle of misuse detection. Timing analysis using by HMM is part for the principle of anomaly detection.

2.2 Intrusion detection system using system calls

In 1996, Forrest et al. [20-21] of the University of New Mexico adopted system call as the data source of IDS. The test data is compared to this positive behavior database during detection. When the test data does not exist in the database, an exception should be occurrence. Module creation is simple in this way.

Intrusion detection systems based on machine learning have attracted much attention for their excellent data analysis. Especially at the system call level, researchers can detect abnormal behavior more effectively [22]. System calls can be considered as a characteristic behavior of a program. Moreover, these calls have a sequential behavior along a timeline with causal relationships between them. Therefore, system calls have the advantages of smaller data volume and more stable behavior to compare with network traffic as the data source for an anomaly detection system. This is beneficial for establishing a simple anomaly detection system.

In a related study, Wunderlich et al. [23] compared the impact of different system call representation methods on intrusion detection. Rosenberg and Gudes [24] pointed out existing technologies that focus too much on the frequency or transformation relationships of system calls. Therefore, they proposed a method to improve the detection effect by using global information. In terms of dataset, ADFA-LD is favored for its ability to reflect the characteristics of modern computer systems. Xie et al. [25] applied a single-class support vector machine algorithm which was combined with a short sequence model to evaluate ADFA-LD.

This paper employs a Markov model to categorize system calls into states. It then uses probabilistic statistics to train a finite-state machine model. The approach effectively utilizes the sequence and causal relationships inherent in system calls for anomaly detection.

2.3 Symbol table

Table 2 provides an explanation of the symbols used in this article.

Table 2. Applications in each class

Symbol	Meaning
Q	The set of all possible states
V	The set of all possible observations
N	The number of possible states
M	The number of possible observations
O	Corresponding observation sequence
A	State transition probability matrix
B	Observation probability matrix
π	Initial state probability vector
λ	Hidden Markov model
λ_i	The i-th HMM parameter
Q_{train}	Each system call sequence in train dataset
dis_{train}^{hy}	The distance of Q_{train} to the decision hyperplane in SVM
Q_{test}	System call sequence in test dataset
Q_{test}^{score}	A classification score for Q_{test} indicating the probability of being normal
HMM_i, $i = 1, 2, \ldots$	The HMM parameters based on λ_i
$num_{\text{particles}}$	Particle number
dim_{size}	Parameter dimension
$Data_{\text{pre}}$	Sample data after preprocessing
w	Inertia weight
w_{decay}	Weight decline factor
t_{max}	Maximum iterations
$particle_{\text{po}}$ $particle_{ve}$	Particle swarm position and velocity
fit	Particle fitness (the correct classification rate of SVM)
P_{best_fit} [i]	The individual optimum fitness value of the i-th particle
P_{best_po} [i]	The individual optimal position of the i-th particle (the corresponding parameter value)
g_{best_fit}	Global optimal fitness value
g_{best_po}	Global optimal position
$\gamma_t(i)$	The probability of being in state q_i at time t, given model λ and observation O
$\Sigma_t(i,j)$	The probability of being in state q_i at time t and in state q_i at time t+1, given model λ and observation O

3. System Architecture and Methodology

3.1 System architecture

As shown in **Fig. 1**, this is a simplified architecture for intrusion detection of manufacturing process. The architecture consists of three main components, Information Technology (IT), Operational technology (OT), and Industrial DMZ (Demilitarized Zone). The Industrial DMZ is an isolated network.

In this architecture, the industrial DMZ plays a key role. It contains an IDS which is used to monitor and detect potential intrusion threats in the network. It monitors network traffic and abnormal behaviors in the DMZ in real time. Take appropriate measures in time to improve network security. In the IDS, the service data transmitted in the industrial DMZ is preprocessed by normalization. It provides a high-quality feature set for intrusion detection methods based on SVH2M model. This can optimize the detection performance and efficiency of intrusion detection algorithms. Then, the intrusion detection model is constructed by SVH2M model. The machine learning method is applied to the intrusion detection of industrial Internet. It makes IDS more intelligent and efficient.

IT and OT represent the fields of information technology and operational technology, respectively. Information technology covers the internal computer networks and systems of enterprises, including office networks, data centers, etc. For example, IT is used in manufacturing industry for production planning management, inventory management, supply chain management, and so on. Operational technology relates to industrial control systems (ICS) and automation equipment in the manufacturing industry. For example, OT in the manufacturing industry is mainly used for the control of the production process and the monitoring of equipment. This architecture is designed to isolate the IT and OT domains.

In summary, this simplified architecture is centered around the industrial DMZ which places for the IDS. This provides cybersecurity protection for the manufacturing industry. Through reasonable network isolation and security measurement, network security can be improved in manufacturing. It can ensure the confidentiality of sensitive data of production system.

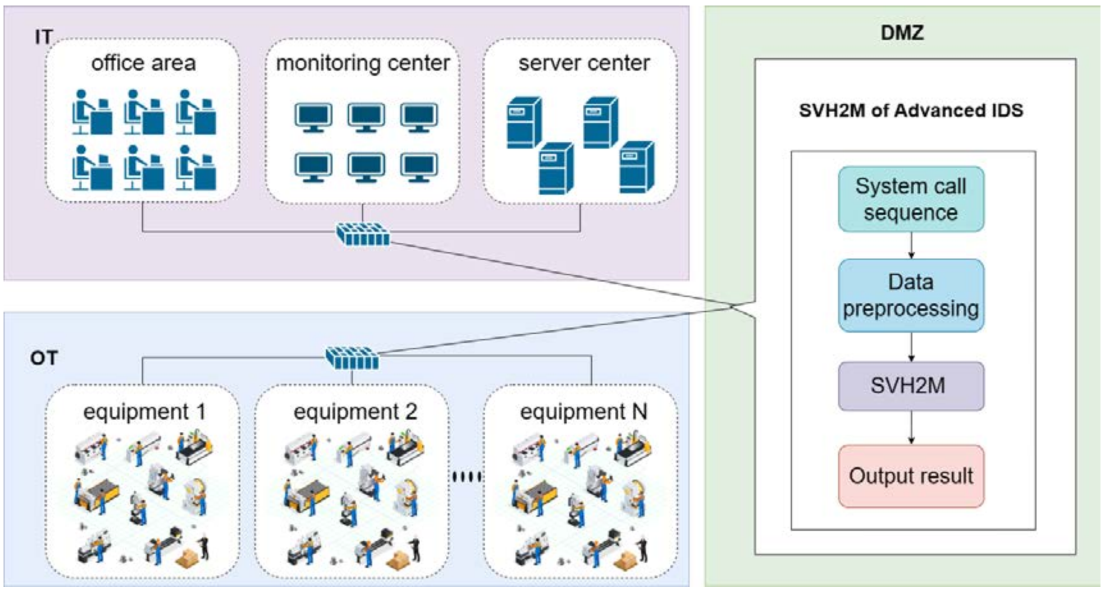


Fig. 1. Simplified architecture for intrusion detection of manufacturing process.

3.2 SVH2M of Advanced IDS

This part mainly explains the architecture for SVH2M of Advanced IDS as shown in Fig. 2.

System call sequence. Each program in the operating system has its own unique sequence of system calls. SVH2M will use these feature sequences to collect the sequence of system calls for different programs in use. They are compared to real-time tracking in the system log. This way can train the model to recognize normal and abnormal patterns in each program.

Data preprocessing. The TF-IDF values of n-gram entries are used to extract system call features. The generated data set is saved as a csv file. The truncated SVD is used to reduce the dimensionality of csv files.

SVH2M. After dimensionality reduction, SVH2M model is used to construct a classification model. The architecture is divided into two main stages, the preliminary classification in stage 1 and the timing analysis in stage 2.

In stage 1, pSVM training is the system first to classify the data series. This step mainly depends on the Support Vector Machine based on Particle Swarm Algorithm (SVMPSA).

Feature extraction. First, the system extracts feature which is suitable for SVM. These features include statistics-based features such as frequency, distribution, etc. These features can reflect patterns and regularities in the data series. This can provide a basis for the subsequent classification.

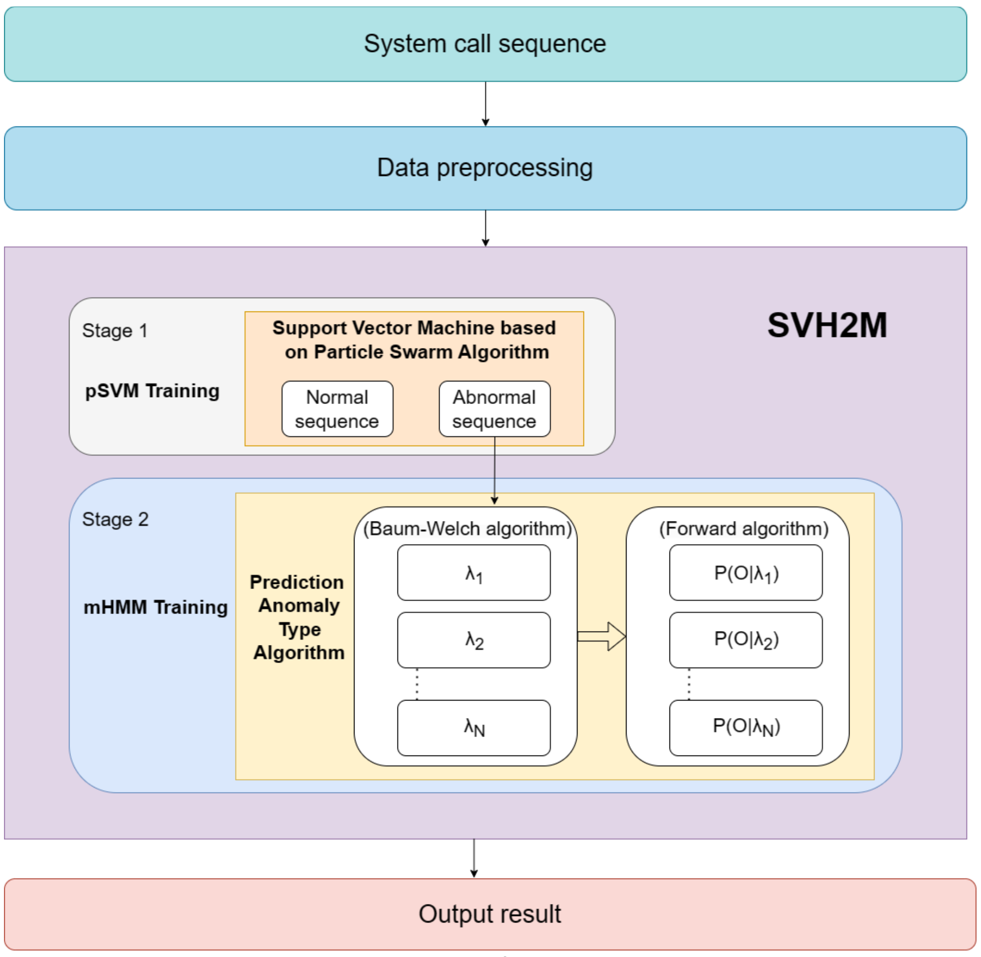


Fig. 2. SVH2M of advanced IDS.

pSVM classification. The trained pSVM model is used to classify the extracted features. SVMPSA is a powerful supervised learning algorithm. It can roughly classify data as "normal" and "abnormal" characteristics. At this stage, the goal of the pSVM is to roughly classify the data to provide a basis for subsequent time series analysis. Because of the stage 1, the results of pSVM classification are further analyzed in the time sequence analysis stage.

In stage 2, mHMM training can describe the relationship between time points in a sequence. Especially for sequences labeled as exceptions, mHMM can better describe their temporal properties. This step mainly depends on the Prediction Anomaly Type Algorithm (PATA).

Baum-Welch algorithm. This algorithm is used to calculate HMM parameters. It is an iterative algorithm which is used to estimate HMM parameters including state transition probability and observation probability. By adjusting these parameters, HMM can better fit the time characteristics of the abnormal sequence.

Forward algorithm. This algorithm is used to calculate the likelihood of a given observation sequence. It observes the probability of a particular sequence. It is used to evaluate the significance of abnormal sequences.

In general, the pSVM is first used for preliminary classification of new production process data. For sequences classified as exceptions, further timing analysis is performed to use mHMM to identify the exception type. This is an effective two-stage approach. It is suitable for complex system call sequence analysis and exception detection. After these two phases, the system can be more accurately detect anomalies and identify their type.

Output result. This module is responsible for displaying the results of SVH2M module analysis. Thus, it helps users get the required information quickly and accurately.

In addressing the challenge of massive and complex data generated by large-scale manufacturing systems, the SVH2M model demonstrates scalability across different manufacturing system sizes through its innovative data processing mechanisms, flexible parameter adjustment strategies, and the integration of distributed and parallel processing technologies.

The SVH2M model, by integrating pSVM and mHMM, leverages pSVM for initial classification to reduce data volume, facilitating subsequent mHMM processing. This data preprocessing mechanism enables the model to more effectively manage data and reduce computational burden when dealing with large-scale manufacturing systems. As system size increases, although data volume significantly grows, the preliminary screening effect of pSVM becomes more crucial in maintaining the overall model performance. Both pSVM and mHMM in the SVH2M model contain adjustable parameters that can be tuned according to the actual size of the manufacturing system. For instance, in large-scale systems, increasing the complexity of pSVM may enhance classification accuracy, or adjusting the number of states and transition probabilities in mHMM can better capture subtle changes in system call sequences. This flexibility allows the model to adapt to systems of varying sizes. To further enhance the scalability of the SVH2M model in large-scale manufacturing systems, distributed and parallel processing techniques can be employed. By segmenting the dataset into subsets and running pSVM and mHMM in parallel across multiple computing nodes, this can significantly reduce processing time.

In complex network environments, system call sequences may be influenced by various external factors such as network latency and packet loss. The SVH2M model can partially mitigate these external disturbances by extracting deep features from system call sequences and leveraging the sequence modeling capabilities of mHMM, can partially mitigate these external disturbances. To address the ever-changing network environment, the SVH2M model needs to possess dynamic adaptability. This can be achieved through online learning or incremental learning approaches. This allows the model to update its parameters and structure in real time. Thus it adapts to new network environment and attack mode. By continuously absorbing new data samples and adjusting model parameters, the SVH2M model can maintain its effectiveness in complex network environments. In highly complex network environments, different manufacturing systems may have close interactions and dependencies. To improve the overall performance of the SVH2M model, cross-domain and cross-system collaborative detection can be considered. By sharing anomaly information and detection results, different systems can collaborate to enhance the identification capability of potential threats.

In summary, the SVH2M model demonstrates good scalability across different manufacturing system sizes and network complexities. By optimizing data processing mechanisms, adjusting model parameters, adopting distributed and parallel processing techniques, and implementing dynamic adaptability and cross-domain collaborative detection, the model's performance and efficiency in complex manufacturing environments can be further improved.

3.3 pSVM classification

SVM is a powerful classification tool. It is good at processing high-dimensional data. The sequence of system calls is usually high dimensional. SVM can effectively identify abnormal patterns in these data. The pre-classification of SVM can reduce the amount of data. An initial SVM classification can help rule out obviously normal or abnormal cases. This reduces the error rate of HMM processing. SVM focuses on static feature classification. Also, HMM focuses on the time dependence of serial data. This combination provides a more comprehensive perspective on categorizing anomalies.

SVM makes predictions by finding a hyperplane that maximizes the edge between two classes. In the case of linear non-divisible, it seeks the best possible dividing plane in a space with more dimensions. To do this, it introduces certain variables known as relaxation variables. These variables help in managing instances that are hard to separate. Additionally, it uses a technique of nonlinear mapping. This mapping transforms the original and simple data into a more complex space. In this complex space, it becomes easier to separate the data. Its main principle is shown in (1) of convex quadratic programming problem.

$$\begin{cases} \min\limits_{\alpha} \dfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i \\ s.t. \sum_{i=1}^{N} \alpha_i y_i = 0, 0 \le \alpha_i \le C, i = 1,2,\dots,N \end{cases} \tag{1}$$

α_i and α_j are Lagrange factors of the i and j samples respectively. C is the penalty parameter. x and y are vector values of samples and classes. $K(x_i, x_j)$ is the kernel function. The optimal solution α^* is obtained from (1). The displacement term b^* is further obtained from (2).

$$b^* = y_j - \sum_{j=1}^{N} \alpha_i^* y_i K(x_i, x_j) \tag{2}$$

Support vector machine adopts Gaussian kernel function. Its classification decision function is shown in (3).

$$f(x) = sign\left[\sum_{i=1}^{N} \alpha_i^* y_i \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right) + b^*\right] \tag{3}$$

As a factor affecting the performance of intrusion detection models, model parameter optimization plays a crucial role in the process of model training. In the process of solving the support vector, a penalty factor C is introduced to deal with the deviation of individual data samples. In addition, the parameter σ of the kernel function also plays an important role. Particle Swarm Optimization (PSO) is an intelligent optimization algorithm based on population. It is especially suitable for searching the optimal parameters of a model in a specified space. In PSO, each particle represents a potential solution. Its position and velocity are continuously updated during the iterative process. The position of these particles represents the combination of values for C and σ. Through continuous search and optimization, it strives to find the best combination of parameters. In each iteration, each particle updates its velocity and position based on its individual best position (P_{best}) and the group best position (g_{best}). In this way, the entire particle swarm gradually converges towards the optimal solution. And it eventually finds the optimal parameter combination for the model. The mathematical expression is as shown in (4) and (5).

$$v_i^{k+1} = wv_i^k + C_1R_1\left(P_{best} - x_i^k\right) + C_2R_2\left(g_{best} - x_i^k\right) \tag{4}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{5}$$

Where w is the position weight, C_1 and C_2 are acceleration factors, R_1 and R_2 are random numbers uniformly distributed over the interval [1,0]. P_{best} is the individual optimal position. And g_{best} is the group optimal position. v_i^kandx_i^k are the velocities and positions of the i-th particle at the k iteration.

3.3.1 Implementation process

In the training phase, the process begins with initialization. Then, system call sequence gathers samples which include both normal and abnormal behaviors in the system. Next, the process performs feature extraction and evaluation. N-gram items and TF-IDF values are used to transform the raw data. This transformation makes the data suitable for machine learning. The most relevant features are then evaluated and selected. Subsequently, these features are used to train the pSVM multi-classifier. An appropriate kernel function is chosen based on (3).

After reading the pre-processed sample data, the position and velocity of the particle swarm are initialization. And the initial position and velocity of the particle are generated randomly. The optimal position of all particles is taken as the penalty constant C and the kernel function parameter σ. And then, the decreasing rule of the inertia weight w is set. The maximum iteration number t_{max} of the particle swarm is set. In this method, the corresponding fitness of particles is the correct classification rate of SVM through cross-verification. The individual extremum and population extremum are subsequently updated. Based on the population extremum, the speed and position of each particle are adjusted. As following this situation, a check is made to see if the algorithm has attained the maximum number of iterations; if not, the process returns to the previous step for further iteration. The best parameters are selected by comparing the population extremum from each iteration. Finally, these optimal parameters are substituted into the support vector machine.

The pSVM is trained to distinguish between normal and various types of abnormal behaviors. Once training is complete, then store into the pSVM. It is placed in a model repository. This repository is for quick future deployment or further refinement. At the end of the training phase, the pSVM model classifies the training data. It classifies them as normal or abnormal.

The testing phase starts with the initialization process. Unknown system call sequence gathers test samples which are used to evaluate the model's performance on new data. Feature extraction is then performed on these test samples. This extraction maintains consistency with the training phase. The extracted features are input into the trained pSVM. The model then classifies each test sample as normal or abnormal. The classification results evaluate the accuracy and validity of the model. Common metrics include precision, recall, and F1-score. This approach is depicted in **Fig. 3**.

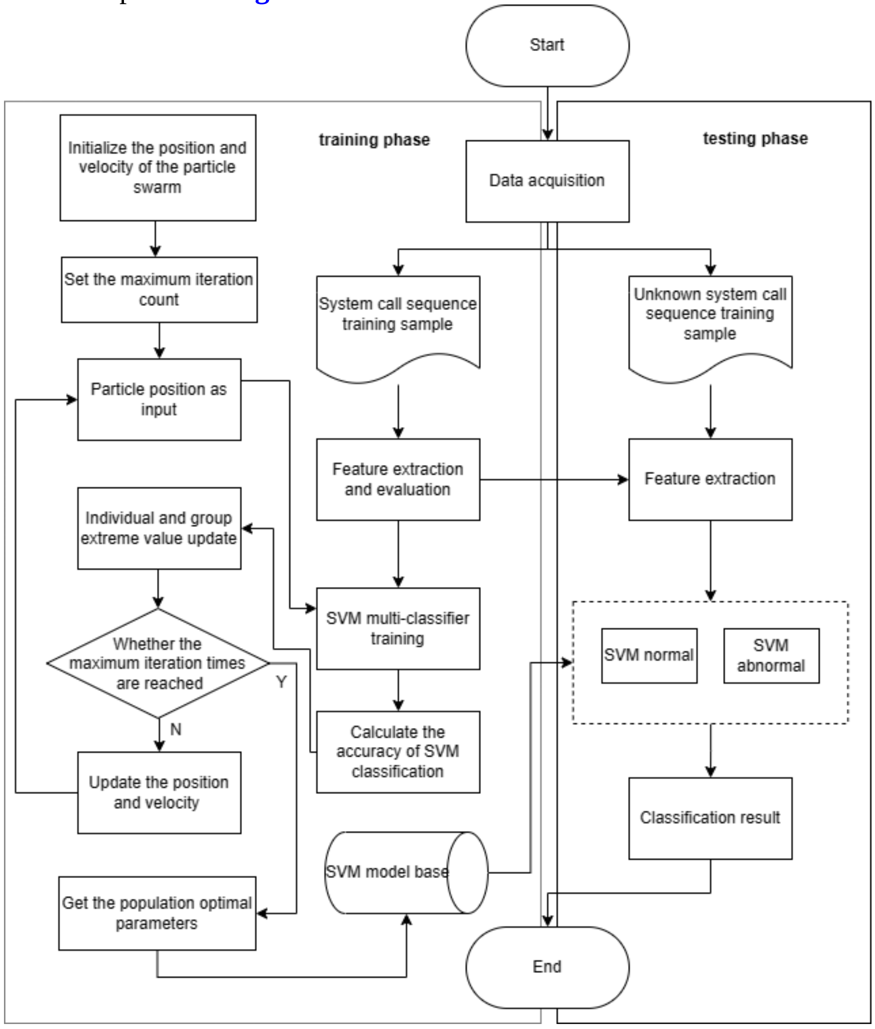

Start
training phase
testing phase
Initialize the position and velocity of the particle swarm
Data acquisition
Set the maximum iteration count
System call sequence training sample
Unknown system call sequence training sample
Particle position as input
Feature extraction and evaluation
Feature extraction
Individual and group extreme value update
SVM multi-classifier training
Whether the maximum iteration times are reached
Y
N
SVM normal
SVM abnormal
Calculate the accuracy of SVM classification
Update the position and velocity
Classification result
SVM model base
Get the population optimal parameters
End

Fig. 3. pSVM training process diagram.

3.3.2 Support Vector Machine based on Particle Swarm Algorithm (SVMPSA)

Fig. 3 introduces the method to retrieve a set of abnormal state sequences using SVMPSA. First, the initial pSVM is established through the initial particle swarm location. And the model accuracy is verified by cross-validation method with the training set data. The model accuracy is returned as the fitness of the particle swarm. The velocity and position of the particle swarm are updated to find the optimal parameters. And it establishes the optimal SVM detection model.

Next, the pSVM is trained to use the input set of system call sequences (denoted as S). For each sequence in S, the pSVM assigns a class label, either "normal" or "abnormal". The pSVM calculates a classification score for each sequence in the set S', indicating the probability of the sequence belonging to the normal category. If the classification score of a sequence in S' is higher than the testing threshold V, it is labeled as abnormal. Otherwise, it is labeled as normal.

Algorithm 1: **S**upport **V**ector **M**achine based on **P**article **S**warm **A**lgorithm **(SVMPSA)**

```
Input: Set of system call sequences S, Testing threshold V
Output: Classification of each sequence in S as normal or anomalous
1： Initialization parameters $num_{\text{particles}}$ , $w_{\text{decay}}$ , $t_{max}$
2： Initializes $particle_{\text{po}}$ and $particle_{\text{ve}}$
3： Building pSVM
4： for t in range ($t_{max}$) do
5：     for i in range ($num_{\text{particles}}$) do
6：         Calculate $fit$
7:         if fit > $P_{best\_fit}$ [i] then
8:             Update their individual optimal position$P_{best\_po}$
9:             Update their individual optimal fitness$P_{best\_fit}$
10:        if max($P_{best\_fit}$) > $g_{best\_fit}$ then
11:            Update global optimal position $g_{best\_po}$
12:            Update global optimal fitness $g_{best\_fit}$
13:        Update $particle_{\text{po}}$ using formula (4)
14:        Update $particle_{\text{ve}}$ using formula (5)
15:        if t ==$t_{max}$ – 1 then
16:            break
17:    end for
18:    if t ==$t_{max}$ – 1 then
19:        break
20: end for
21: Decreasing inertia weight $w_{\text{decay}}$
22: Update the pSVM
23: for $Q_{train}$ in S do
24：     Compute $dis_{train}^{hy}$
25：     Assign a label to $Q_{train}$
26：  end for
27： for $Q_{test}$ do
28：     Compute $Q_{test}^{score}$ for $Q_{test}$
29：     if $Q_{test}^{score}$ < V then
30：         Flag $Q_{test}$ as anomalous
31：     else
32：         Flag $Q_{test}$as normal
33：     end if
34： end for
```

3.4 mHMM classification

HMM models are good at recognizing patterns in time series data. It uses these patterns to predict the future direction of the data. When analyzing a sequence of system calls, the occurrence of an event may depend on previous events. HMM is good at understanding these kinds of dependencies. SVM focuses on the location of data points in space. It's not so focused on how those points change over time. HMM complements the analysis by focusing on changes in time series.

The HMM is determined by the initial probability distribution, the state transition probability distribution, and the observed probability distribution. HMM is defined as follows.

$$Q = \{q_1, q_2, \ldots, q_N\} \tag{6}$$

$$V = \{v_1, v_2, \ldots, v_M\} \tag{7}$$

As shown in (6) and (7), N is the number of possible states. M is the number of possible observations. As shown in (8) and (9), I is a sequence of states of length T. O is the corresponding observation sequence.

$$I = \{i_1, i_2, \ldots, i_T\} \tag{8}$$

$$O = \{o_1, o_2, \ldots, o_T\} \tag{9}$$

A is the state transition probability matrix, as shown in (10) and (11).

$$A = [a_{ij}]_{N*N} \tag{10}$$

$$a_{ij} = P\left(i_{t+1} = q_j \middle| i_t = q_i\right), i = 1,2, \ldots, N; j = 1,2, \ldots, N \tag{11}$$

a_{ij} is the probability of transitioning to state q_j at time t+1 with state q_i at time t. As shown in (12) and (13), B is the observation probability matrix.

$$B = [b_j(k)]_{N*M} \tag{12}$$

$$b_j(k) = P\left(o_t = v_k \middle| i_t = q_j\right), k = 1,2, \ldots, M; j = 1,2, \ldots, N \tag{13}$$

$b_j(k)$ is the probability of generating observed v_k at time t in state q_j. As shown in (14) and (15), π is the initial state probability vector.

$$\pi = (\pi_i) \tag{14}$$

$$\pi_i = P(i_1 = q_i), i = 1, 2, \ldots, N \tag{15}$$

π_i is the probability of being in state q_i at time t=1. The hidden Markov model is determined by the initial state probability vector π, the state transition matrix A, and the observation probability matrix B. Thus, the hidden Markov model λ can be represented by a ternary symbol, as shown in (16).

$$\lambda = (A, B, \pi) \tag{16}$$

3.4.1 Implementation process

As pSVM training, it has initially classified system call sequences as normal or abnormal. There is now a batch of sequences marked as abnormal. The aim is to determine the specific type and possible cause of the abnormalities. First, mHMM models are built for different types of known abnormalities, as shown in **Fig. 4**. For example, one model might represent sequences abnormal due to sensor faults. Another model might represent abnormalities caused by network delays. For each sequence marked as abnormal by pSVM, the mHMM is used to analyze its features. The final classification results are determined according to the majority voting principle. Based on the probability of the sequence under each HMM, it can determine the most likely type of abnormality for the sequence. This approach is depicted in **Fig. 4**.

Use sequences marked as abnormal by pSVM to train the mHMM. The abnormal system call sequences serve as observation sequences. HMM models sequential dependencies in the training data. The HMM is trained with the observation sequence $O = \{o_1, o_2, \ldots, o_T\}$. First, initialize the HMM parameters $\lambda = (A, B, \pi)$. Then, use the Baum-Welch algorithm to re-estimate the HMM parameters. The Baum-Welch algorithm includes the following three steps.

Step 1. Define forward probabilities. Define the partial observation sequence up to time t as $o_1, o_2, \ldots, o_t$. And the probability of being in state q_i at time t as the forward probability, denoted as in (17). The forward probability $\alpha_t(i)$ and the probability of observation sequence

P (O|λ) can be recursively calculated through several inductive steps, as shown in (18), (19), and (20).

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, i_t = q_i|\lambda) \tag{17}$$

a) Initialization.

$$\alpha_1(i) = \pi_i b_i(o_1), i = 1,2, \dots, N \tag{18}$$

b) Recursion.

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i(o_{t+1}),$$
$$i = 1,2, \dots, N. t = 1,2, \dots, T-1. \tag{19}$$

c) Termination.

$$P\,(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{20}$$

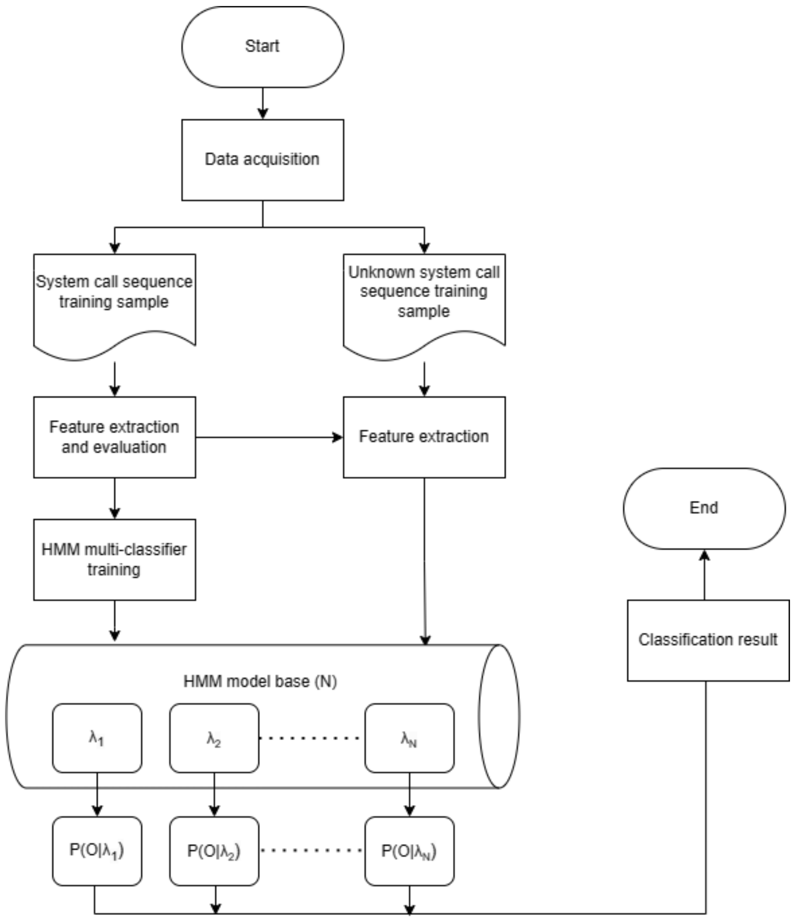


Fig. 4. mHMM training process diagram.

Step 2. Define backward probabilities. Define the probability of the partial observation sequence from t+1 to T. Given the condition that the state at time t is q_i. This probability is known as the backward probability, denoted as in (21).

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T, i_t = q_i|\lambda) \tag{21}$$

The backward probability $\beta_t(i)$ and the observed sequence probability P $(O|\lambda)$ can be retrieved by recursion method. The calculation can be summarized by the following three steps, as shown in (22), (23), and (24).

a) Initialization.

$$\beta_t(i) = 1, i = 1,2,\ldots,N \tag{22}$$

b) For t=T-1, T-2, …,1

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(j), i = 1,2,\ldots,N \tag{23}$$

c) Calculate P $(O|\lambda)$.

$$P\,(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i) \tag{24}$$

Step 3. Calculate λ parameter. Specifically, the probability variables $\gamma_t(i)$ and $\varepsilon_t(i,j)$ can be calculated based on the variables α and β, as shown in the (25) and (26).

$$\gamma_t(i) = P\big(O, i_t = q_i \big| O, \lambda\big) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)} \tag{25}$$

$$\varepsilon_t(i,j) = P\big(i_t = q_i, i_{t+1} = q_j \big| O, \lambda\big) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{26}$$

The variables γ and ε in the formula for estimating the HMM parameters are followed by three steps, as shown in the (27), (28), (29), and (30).

a) Initialization. For n=0, model $\lambda^{(0)} = (A^{(0)}, B^{(0)}, \pi^{(0)})$

b) Recursion. For n=1,2, …,

$$a_{ij}^{(n+1)} = \frac{\sum_{t=1}^{T-1}\varepsilon_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{27}$$

$$b_j(k)^{(n+1)} = \frac{\sum_{t=1,o_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{28}$$

$$\pi_i^{(n+1)} = \gamma_1(i) \tag{29}$$

c) Termination.

$$\lambda^{(n+1)} = \big(A^{(n+1)}, B^{(n+1)}, \pi^{(n+1)}\big) \tag{30}$$

3.4.2 Prediction Anomaly Type Algorithm (PATA)

The algorithm 2 for classifying abnormal state sequences is described as follows. It first uses the Baum-Welch algorithm to train the mHMM parameters including A, B, and π. And it trains multiple HMM models for different types of known anomalies. For each sequence labeled as abnormal by the pSVM, the trained mHMM is used to analyze its features. This usually involves calculating the probability of the sequence under each model. Based on the probability of the sequence under each HMM, the most likely type of abnormality corresponding to the sequence is determined. For example, as a sequence has a higher probability under a specific type of HMM, it can be inferred that the anomaly may belong to that type. The Viterbi algorithm is used to generate each hidden state sequence. The sequence determines the attacking type which it belongs to. Each HMM classifies the input sequence of system calls. And each HMM outputs a classification result. In accordance with majority voting, this will count the votes of each category to obtain the final classification.

Algorithm 2: **P**rediction **A**nomaly **T**ype **A**lgorithm (**PATA**)

Input: Set of system call sequences S, labeled as abnormal by SVM
Output: Classification of each sequence in S with specific type of anomaly

```
1:  Initialize $\lambda = \{\pi, A, B\}$ with small random values
2:  while $\lambda$ is not convergence do
3:    Train the $HMM_1$ and S using Baum-Welch algorithm
4:    Train the $HMM_2$ and S using Baum-Welch algorithm
5:    Train the $HMM_3$ and S using Baum-Welch algorithm
6:  end while
7:  for $Q_{train}$ in S do
8:    Compute the likelihood $P(Q|\lambda)$
9:  end for
10: For $Q_{train}$ in S do
11:   If $P(Q|\lambda1) > P(Q|\lambda2)$
12:     Label sequence as " Abnormal 1"
13:   else $P(Q|\lambda1) < P(Q|\lambda2)$
14:     Label sequence as " Abnormal 2"
15:   else
16:     Label sequence as "Unknown Anomaly"
17: end for
18: Statistical classification result
19: Output results according to voting principles
```

4. Experiment

4.1 Experimental setup and dataset

The experimental software is carried out on Windows10 operating system. The hardware environment uses a 12th Gen Intel(R) Core (TM) i9-12900KF 3.19GHz CPU and a NVIDIA GeForce RTX 3080Ti GPU with 12G memory. The ADFA-LD dataset [26] and MAWI dataset [27] were used in this experiment.

In a system call-based intrusion detection system, the detected anomaly types are typically related to deviations from the system call sequences generated by programs during normal execution. These anomalies may include but are not limited to the following types:

Unusual System Call Sequences: When a program executes system call sequences that deviate from its normal behavior, it may indicate an anomaly. For example, if a program typically does not perform write operations to sensitive files (such as system configuration files), but such operations are recorded in the system logs, it is likely a sign of intrusion.

Abnormal System Call Frequencies: Even if the system call sequences themselves appear normal, a sudden increase or decrease in the frequency of certain calls may also indicate an anomaly. For instance, excessive read or write operations could signify data leakage or an attempt by malware to steal or modify data.

Unexpected Parameter Values: Certain system calls require specific parameter values, and if these values exceed normal ranges or expectations, they may also signify an anomaly. For example, a network program suddenly attempting to connect to an illegal or unknown IP address.

Chained Anomalies: A series of system calls that individually seem normal but, when combined, constitute anomalous behavior. These actions may appear harmless in isolation but, when taken together, may reveal malicious activity.

In operating systems, each program possesses a distinctive system call signature sequence that uniquely differentiates it from other programs. These sequences serve as the program's fingerprint, allowing for its identification amidst the myriad of processes running concurrently. To harness this distinguishing feature, we will embark on a process that involves capturing the system call sequences generated by various programs during their normal usage periods. These collected sequences will then serve as the benchmark for what constitutes normal behavior for each program. By leveraging these established signatures, we will train a model that meticulously compares the real-time traces extracted from system logs against the pre-recorded normal patterns. This comparison process is designed to empower the model with the capability to discern the normal operational mode of each program, effectively distinguishing it from any anomalous behavior that may arise. Our model differentiates between the aforementioned anomaly types through the following approaches:

Feature Extraction: First, the system call sequences are converted into 6-ngram sequences, enhancing data richness and enabling the model to learn more complex patterns. Each 6-ngram serves as a feature, representing specific relationships and orders between system calls.

Pattern Recognition: During training, the model learns patterns of normal and anomalous behavior. By comparing 6-ngram sequences from real-time system logs with learned patterns, the model can identify sequences that deviate from normal patterns, i.e., anomalous sequences.

Classification and Threshold Setting: The model employs machine learning algorithms to classify input system call sequences as normal or anomalous. Additionally, thresholds are adjusted to control the model's sensitivity and specificity, minimizing false positives and false negatives.

Anomaly Pattern Description: For detected anomalies, the model can generate descriptions of anomaly patterns by analyzing common features within the anomalous sequences. For instance, if multiple anomalies involve write operations to sensitive files, the model can infer a specific anomaly pattern and issue corresponding warnings.

4.2 Experimental setup and dataset

In this paper, confusion matrix, accuracy, precision, recall rate, false positive rate, specificity, and mismatch rate are used to evaluate the performance of the classification model.

Confusion matrix. The confusion matrix is a table. It displays the relationship between the predictions and actual labels. It divides the predictions into four different categories. These categories are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Accuracy. It is the ratio of the number of correctly classified samples to the total number of samples, as shown in (31). It measures the overall classification accuracy of a model for all samples. The higher the accuracy, the better the performance of the model.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{31}$$

Precision. It represents the proportion of prediction accuracy among all the positives predicted by the model, as shown in (32). Also, it represents the percentage of predicted positive situations.

$$Precision = \frac{TP}{TP + FP} \tag{32}$$

Recall. It represents the positive proportion of all true positives that the prediction model is correction, as shown in (33). It is used to measure the number of predicted positive situations for rightness of actual value and predicted value. Generally, this formula of recall is the same as the formula of sensitive.

$$Recall = \frac{TP}{TP + FN} \tag{33}$$

Specificity. It refers to a specific negative proportion of all true negatives that the model was correct, as in (34). It is mainly used in the recognition ability of the model to the negative example.

$$Specificity = \frac{TN}{TN + FP} \tag{34}$$

Mismatch Rate. The proportion of sequences in the test set that are incorrectly identified as anomalies by a trained model, as in (35).

$$MR = \frac{number\ of\ abnormal\ sequences}{total\ number\ of\ sequences} \tag{35}$$

4.3 Experimental results and analysis

4.3.1 System call sequence analysis

Fig. 5 shows a stacked bar chart. It is used to compare the frequency of system calls between normal data and intrusion data. The blue bar chart shows the distribution of system calls under normal conditions. The red bar chart shows the distribution of system calls in the case of intrusion. They are used to show the proportion of system call numbers. In this diagram, the Y-axis is labeled "Proportions". It represents the relative frequency of different system calls. The X-axis is labeled "Syscall num". It represents a different system call number. With this visualization, it can clearly show the difference in system call frequency between normal and intrusion data.

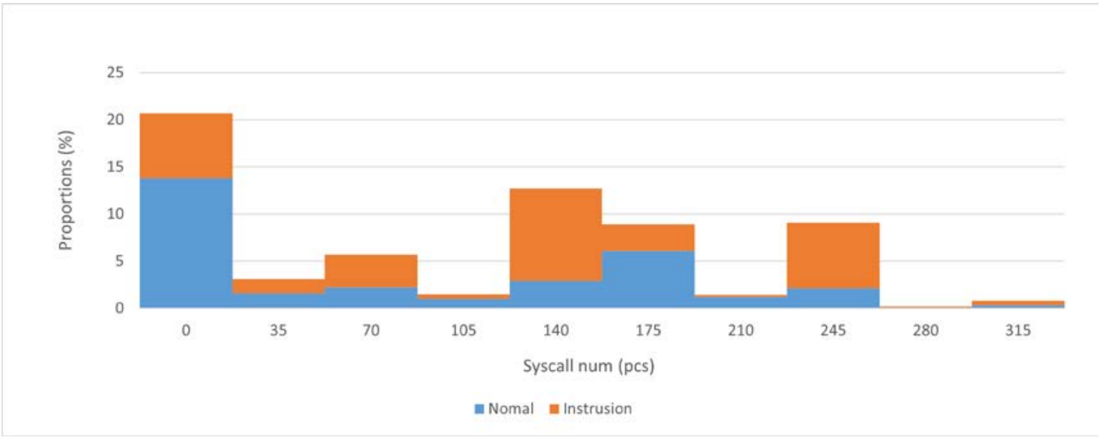
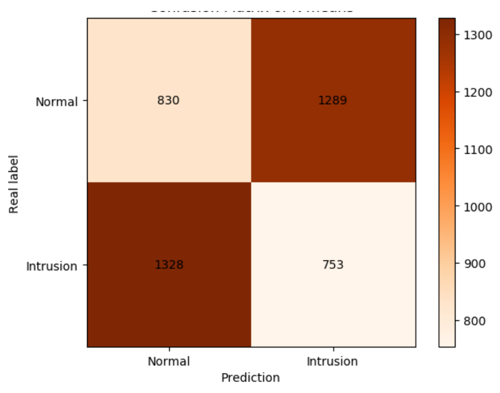


Fig. 5. Stacked bar chart of the proportion of system calls in normal data and intrusion data.

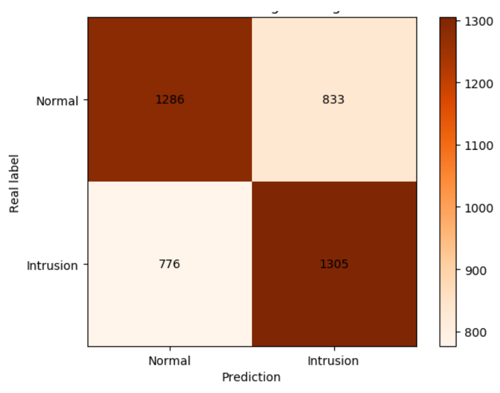
4.3.2 Comparison of classification models

This paper uses four different machine learning methods to build the model. Specific methods include K-means, Logistic Regression, Neural Network, and SVH2M.

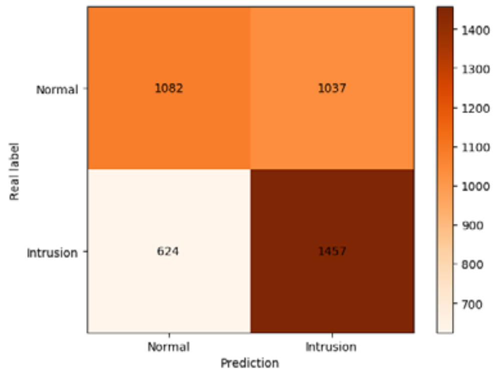
Fig. 6 shows the confusion matrix generated by four machine learning methods. It represents a summary of the predicted results of the classification problem. It summarizes the predicted quantities of normal and attack sequences using count values. Additionally, it provides analysis of the predictions for each category. The upper left corner of the picture is TP, the upper right corner is FP, the lower left corner is FN, and the lower corner is TN. In **Fig. 6 (a)**, **(b)**, **(c)**, and **(d)**, the number of normal sequences that can be correctly identified is 830, 1286, 1082, and 1710, respectively. Also, the number of attack sequences that can be correctly identified is 753, 1305, and 1457. For the K-means model in **Fig. 6 (a)**, most normal sequences are correctly identified as normal, but a portion of attack sequences are misclassified as normal. For the Logistic Regression model in **Fig. 6 (b)**, both normal and attack sequences are recognized with relatively high accuracy compared to the models shown in **Fig. 6 (a)**, **Fig. 6 (c)**, and **Fig. 6 (d)**. It shows that the Logistic Regression model has good performance. For the Neural Network model in **Fig. 6 (c)**, the recognition accuracy for attack sequences is the highest in the four models. But the accuracy for normal sequences is the lowest in the four models. Thus, the model is more inclined to classify sequences as attacks. Finally, for the SVH2M model in **Fig. 6 (d)**, the recognition accuracy for normal sequences is the highest in the four models. The accuracy for attack sequences is not as high as **Fig. 6 (c)**, but it is still better than the models in **Fig. 6 (a)** and **Fig. 6 (b)**. So, the model is more prone to classifying sequences as normal.



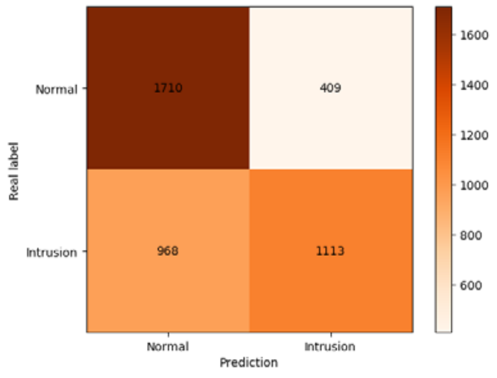
(a) K-means



(b) Logistic Regression



(c) Neural Network

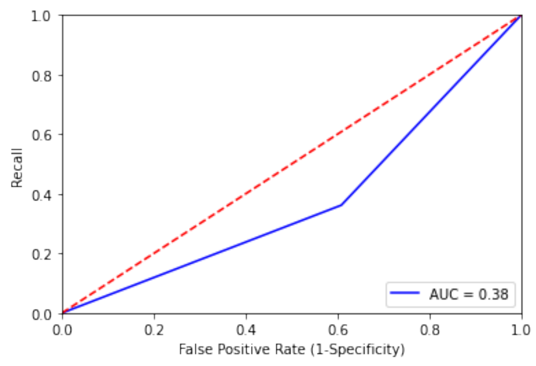


(d) SVH2M

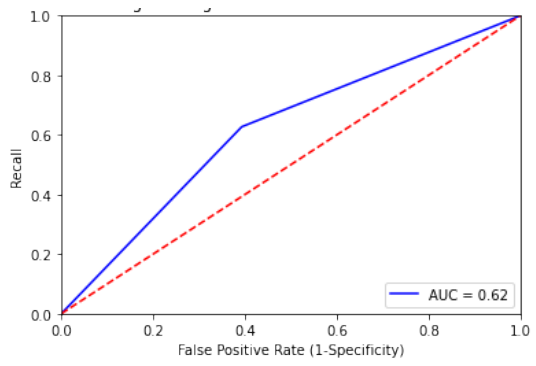
Fig. 6. Confusion matrix diagram of different models.

Fig. 7 shows the average ROC curves for different models. ROC curves and AUC (Area Under Curve) assess classification model performance. The ROC curve shows the true positive rate versus the false positive rate. It measures the area between average false positive and true positive rates.

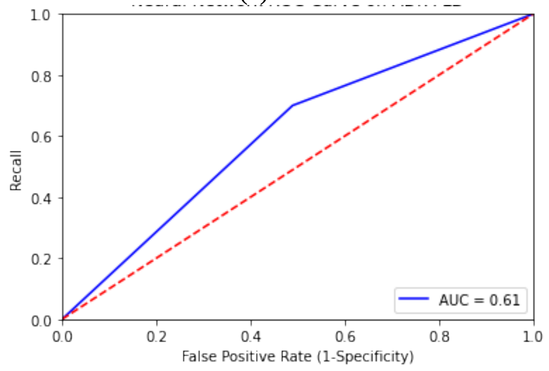
In **Fig. 7**, the K-means model's AUC is 0.38. The Logistic Regression model's AUC is 0.62. The SVH2M model's AUC is 0.67. The Neural Network model's AUC is 0.61. Therefore, the SVH2M model exhibits the best classification performance among the four models. The K-means model performs the worst due to its inapplicability for classification tasks. The Logistic Regression and Neural Network models show similar performance. But the Logistic Regression and Neural Network are slightly inferior to SVH2M. Thus, the SVH2M has superior classification performance.



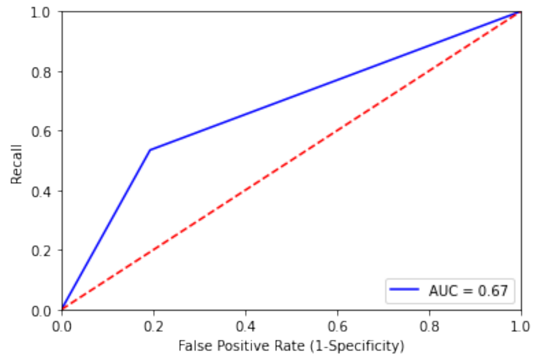
(a) K-means



(b) Logistic Regression



(c) Neural Network



(d) SVH2M

Fig. 7. ROC diagrams for different methods.

In **Table 3**, the K-means has the lowest performance across all metrics. This means that K-means has a lot of trouble to distinguish between normal and attack sequences. To compare K-means, Logistic Regression performs better in all metrics. But it's still not the best model of all. From the experimental results, the neural network performs the best in terms of recall. This means that it rarely misses the real attack sequence. However, its accuracy and specificity are lower than Logistic Regression and SVH2M. So, it mistakenly identifies some normal sequences as attack sequences. In the accuracy comparison, SVH2M shows a significant advantage. Its accuracy is 0.73. The lowest K-means method is only 0.37. SVH2M is twice as accurate as K-means. In terms of precision, SVH2M reaches a high value of 0.67. However, SVH2M performs slightly worse than other methods in terms of recall rates. In addition, the specificity of SVH2M is up to 0.81. These results show that SVH2M has high accuracy, precision, and specificity in classification tasks.

In **Table 4**, the SVH2M model performed the best across all evaluation metrics, particularly with a high precision rate of 0.93 and a recall rate of 0.95, indicating that the model is adept at accurately identifying genuine malicious samples (high precision) while also covering a significant portion of them (high recall). Additionally, the high specificity of 0.91 demonstrates the model's low misjudgment rate for normal samples. SVH2M, specifically designed for handling system call sequences, effectively utilizes temporal dependencies and contextual information within the sequences, achieving high classification performance. The Neural Network and Logistic Regression models exhibited similar performance in terms of precision, accuracy, and specificity, but the Neural Network slightly outperformed in accuracy. However, the Neural Network's lower recall rate of 0.54 suggests that it may miss a considerable number of malicious samples. While Neural Network and Logistic Regression performed well in some metrics, their deficiencies in recall indicate that they may not fully capture the characteristics of malicious samples. The K-means clustering algorithm performed relatively poorly in this classification task, notably with significantly lower precision and recall rates compared to other supervised learning models. This could be attributed to K-means being a distance-based clustering method that does not directly optimize classification performance and is sensitive to the initial centroid selection.

From **Table 3** and **Table 4**, the SVH2M model demonstrated the best performance on both datasets, indicating that it may be a strong choice for similar tasks and datasets.

Table 3. Applications in each class comparison performance of classification models

Models	Precision(%)	Accuracy(%)	Recall(%)	Specificity(%)
K-means	0.37	0.38	0.36	0.39
Logistic Regression	0.61	0.62	0.63	0.61
Neural Network	0.58	0.60	0.70	0.51
SVH2M	0.73	0.67	0.53	0.81

Table 4. Comparison of classification performance of different models on the MAWI dataset

Models	Precision(%)	Accuracy(%)	Recall(%)	Specificity(%)
K-means	0.62	0.63	0.57	0.66
Logistic Regression	0.71	0.78	0.59	0.83
Neural Network	0.70	0.79	0.54	0.86
SVH2M	0.93	0.92	0.95	0.91

4.3.3 Comparison experiment of algorithms

This experiment mainly tests the performance of SVMPSA in intrusion detection. And the experimental results are shown in **Table 5**.

The BPNN (Backpropagation Neural Network) algorithm achieves an accuracy of 82.32%. Its running time is 28.94 seconds. This indicates that BPNN is not the best choice in the fast response scenario. The GRNN (Generalized Regression Neural Network) algorithm shows an accuracy rate of 85.56%, which is slightly higher than the BPNN. At the same time, its runtime is 22.46 seconds. GRNN is less than BPNN about 6 seconds. The performance is better than BPNN. The accuracy of SVM algorithm is only 67.42%, which is significantly lower than BPNN and GRNN. In addition, its running time reached 31.82 seconds, which is longer than the BPNN. In the present experimental environment, the traditional support vector machine algorithm is not the best choice for intrusion detection.

Table 5. Algorithm comparison result

Algorithm	Accuracy(%)	Time(s)
BPNN	82.32	28.94
GRNN	85.56	22.46
SVM	67.42	31.82
SVMPSA	94.69	15.89

SVMPSA shows accuracy rate of 94.69%, which is much higher than other algorithms. In addition, its running time is only 15.89 seconds, which is the fastest of all the algorithms. This shows that SVMPSA not only performs well in accuracy, but also has a clear advantage in processing speed.

Each HMM is trained to recognize a specific type of cyberattack. For example, HMM1 detects Hydra-FTP (Hyd) attacks. HMM2 detects Webshell (Web) attacks. And HMM3 detects Adduser (Add) attacks. Hyd is an attempt to brute-force the FTP protocol to use Hydra. Hydra is a powerful tool for login to crack. Add is used to add a new user to the system. In a cyber attack, the attacker creates a new user account on the victim's machine to gain access. Web is a malicious script. An attacker usually implants a Webshell into the victim's web server. A total of 100 samples are collected for the experiment. Three sample results are shown in **Table 6**. For example, sample 1 is classified as a Webshell attack by HMM1 and HMM2 when HMM3 is classified an Adduser attack. According to the majority voting principle, the final classification result is also a Webshell attack. It matches the actual label. So, the sample is classified correctly.

Table 6. Comparison result

Sample number	HMM1	HMM2	HMM3	PATA	Actual label
1	Web	Web	Add	Web	Web
2	Hyd	Web	Hyd	Hyd	Hyd
3	Add	Add	Add	Add	Add

The average accuracy of HMM1, HMM2, and HMM3 classification is 80.31%, 82.64%, and 81.47%, respectively. The average accuracy of PATA is 94.26%. It is an effective strategy to adopt the majority voting principle to determine the final classification result. Because it can use the multiple models to improve the robustness of classification. In the experiment, when multiple models give different classification results for the same sample, the majority voting principle can provide a more reliable final classification result. PATA algorithm can enhance classification performance by introducing additional decision logic. In addition, the multiple models can also play a key role in improving overall performance. So PATA can improve the accuracy of intrusion detection.

4.3.4 performance evaluation of SVH2M model

The SVH2M model combines Support Vector Machines with hidden Markov models. This combination classifies normal and abnormal behaviors. The AUC is the chosen key indicator. It assesses the model's performance. The HMM has an AUC of 0.71 as shown in **Fig. 8**. The SVH2M model's AUC is 0.88. The AUC is significantly increased in the SVH2M model. Therefore, the combined model enhances classification performance.

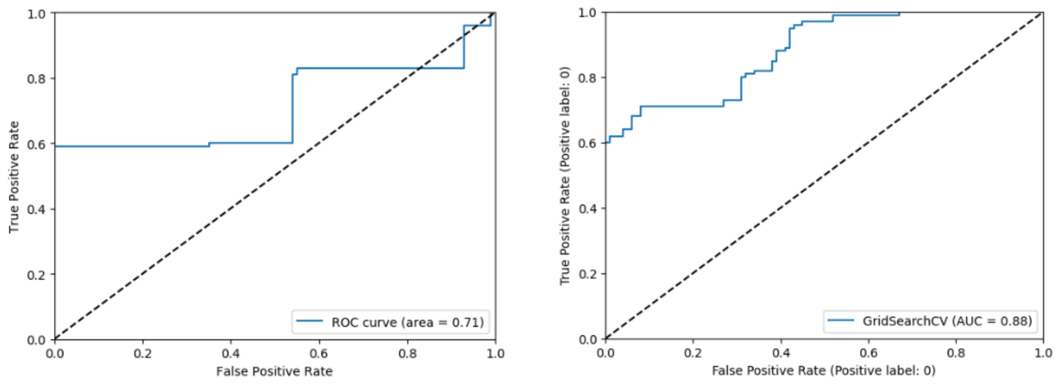


Fig. 8. HMM vs SVH2M.

The SVH2M method outperforms the original hidden Markov model as shown in **Table 7**. The mismatch rate for HMM decreases as the amount of data increases. While using 50% of the data, the mismatch rate is 3.89%. And, while using 80% of the data, the mismatch rate drops to 1.98%. Similarly, the mismatch rate for SVH2M also decreases as the amount of data increases. When using 50% of the data, the SVH2M mismatch rate is 2.14%. And when using 80% of the data, the SVH2M mismatch rate is 1.24%. When using 80% of the data, SVH2M's mismatch rate is lower than HMM. Data usage is dropped slightly from 70% to 80%. This superiority is evident in the mismatch rate on various proportions of datasets. When the training dataset is large, the mismatch rate significantly decreases. The mismatch rate will stabilize around 80%. It also suggests that the SVH2M model's performance has potential for further improvement with more training data. But adding more data beyond a certain point don't improve the model's performance. Therefore, in practical scenarios, people should expand the training set as much as possible. This ensures the model can fully learn and adapt to various behavior patterns.

Table 7. Mismatch rate

DATASET(%)	HMM(%)	SVH2M(%)
50	3.89	2.14
60	2.52	1.88
70	1.99	1.24
80	1.98	1.24

In summary, by adopting the SVH2M model, this model achieves lower mismatch rate across datasets of different sizes. Moreover, as the training dataset increases, there is potential for further improvement in performance.

5. Conclusion

This paper conducts an in-depth study of the system call process in manufacturing workflows. It classifies normal and abnormal intrusions. To enhance the performance and efficiency of detecting attacks in manufacturing systems, the SVH2M model is used to establish behavior models. This study compares pSVM with several other common machine learning methods. The results show that pSVM significantly outperforms in accuracy, precision, and specificity. Hence, pSVM is combined with mHMM. The SVH2M model exhibits superior false alarm rates on datasets of varying normal proportions. By adopting the SVH2M model approach, the performance and efficiency of manufacturing systems are effectively improvement. This research provides a viable method for enhancing the security of manufacturing systems. It guides further improvement and optimization of model performance. However, this study still has some limitations. First, the research mainly focuses on the system call process without an in-depth analysis of other types of network traffic. Second, the study has not yet covered all possible types of attack behavior. Therefore, future research can further expand the models and algorithms to improve the comprehensiveness and accuracy of the detection system.

Acknowledgement

This work was supported by the Initiation Funds for High-level Talents Program of Xi'an International University (grant no. XAIU202411).

References

[1] N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya and Z. Tari, "Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions," *IEEE Communications Surveys & Tutorials*, vol.25, no.3, pp.1775-1807, thirdquarter 2023. Article (CrossRef Link)

[2] M. Nuaimi, L. C. Fourati and B. B. Hamed, "Intelligent Approaches Toward Intrusion Detection Systems for Industrial Internet of Things: A Systematic Comprehensive Review," *Journal of Network and Computer Applications*, vol.215, Jun. 2023. Article (CrossRef Link)

[3] J. Qian, X. Du, B. Chen, B. Qu, K. Zeng and J. Liu, "Cyber-Physical Integrated Intrusion Detection Scheme in SCADA System of Process Manufacturing Industry," *IEEE Access*, vol.8, pp.147471-147481, Aug. 2020. Article (CrossRef Link)

[4] S. Alem, D. Espès, L. Nana, E. Martin and F. De Lamotte, "A Novel bi-anomaly-based Intrusion Detection System Approach for Industry 4.0," *Future Generation Computer Systems*, vol.145, pp.267-283, 2023. Article (CrossRef Link)

[5] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, M.A. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proc. of DARPA Information Survivability Conference and Exposition, DISCEX'00*, vol.2, pp.12-26, SC, USA, Jan. 2000. Article (CrossRef Link)

[6] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation," in *Proc. of third International Workshop, Recent Advances in Intrusion Detection*, LNCS, vol.1907, pp.162-182, France, 2000. Article (CrossRef Link)

[7] Z. Liu, N. Japkowicz, R. Wang, Y. Cai, D. Tang, and X. Cai, "A statistical pattern based feature extraction method on system call traces for anomaly detection," *Information and Software Technology*, vol.126, Oct. 2020. Article (CrossRef Link)

[8] F. Yu, C. Xu, Y. Shen, J.-Y. An, and L.-F. Zhang, "Intrusion detection based on system call finite-state automation machine," in *Proc. of 2005 IEEE International Conference on Industrial Technology*, pp.63-68, Hong Kong, China, Dec. 2005. Article (CrossRef Link)

[9] X. Zhang, Z. Zhu and P. Fan, "Intrusion detection based on cross-correlation of system call sequences," in *Proc. of 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pp.7-283, Hong Kong, China, Nov. 2005. Article (CrossRef Link)

[10] S. Lv, J. Wang, Y. Yang and J. Liu, "Intrusion Prediction with System-call Sequence-to-sequence Model," *IEEE Access*, vol.6, pp.71413-71421, Nov. 2018. Article (CrossRef Link)

[11] A. Al-Saleh, "A balanced communication-avoiding support vector machine decision tree method for smart intrusion detection systems," *Scientific Reports*, vol.13, no.1, Jun. 2023. Article (CrossRef Link)

[12] M. A. Almaiah, O. Almomani, A. Alsaaidah, S. Al-Otaibi, N. Bani-Hani, A. K. Al Hwaitat, A. Al-Zahrani, A. Lutfi, A. B. Awad, T. H. H. Aldhyani, "Performance Investigation of Principal Component Analysis for Intrusion Detection System Using Different Support Vector Machine Kernels," *Electronics*, vol.11, no.21, Nov. 2022. Article (CrossRef Link)

[13] A. A. Alqarni, "Toward support-vector machine-based ant colony optimization algorithms for intrusion detection," *Soft Computing*, vol.27, no.10, pp.6297-6305, May 2023. Article (CrossRef Link)

[14] M. Hosseinzadeh, A. M. Rahmani, B. Vo, M. Bidaki, M. Masdari, and M. Zangakani, "Improving security using SVM-based anomaly detection: issues and challenges," *Soft Computing*, vol.25, pp.3195-3223, Feb. 2021. Article (CrossRef Link)

[15] T. Shawly, "A Detection and Response Architecture for Stealthy Attacks on Cyber-Physical Systems," *JOIV International Journal on Informatics Visualization*, vol.7, no.3, pp.801-807, Sep. 2023. Article (CrossRef Link)

[16] C. Dong, H. Wu and Q. Li, "Multiple Observation HMM-Based CAN Bus Intrusion Detection System for In-Vehicle Network," *IEEE Access*, vol.11, pp.35639-35648, Apr. 2023. Article (CrossRef Link)

[17] T. Shawly, M. Khayat, A. Elghariani and A. Ghafoor, "Evaluation of HMM-Based Network Intrusion Detection System for Multiple Multi-Stage Attacks," *IEEE Network*, vol.34, no.3, pp.240-248, May/Jun. 2020. Article (CrossRef Link)

[18] R. Agarwal and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining (a Case-Study in Network Intrusion Detection)," in *Proc. of the 2001 SIAM International Conference on Data Mining*, pp.1-17, 2001. Article (CrossRef Link)

[19] E. Nikolova and V. Jecheva, "Some similarity coefficients and application of data mining techniques to the anomaly-based IDS," *Telecommunication Systems*, vol.50, no.2, pp.127-135, 2012. Article (CrossRef Link)

[20] S. Forrest, S.A. Hofmeyr, A. Somayaji and T.A. Longstaff, "A sense of self for Unix processes," in *Proc. of 1996 IEEE Symposium on Security and Privacy*, pp.120-128, 1996. Article (CrossRef Link)

[21] S. A. Hofmeyr, S. Forrest and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol.6, no.3, pp.151-180, 1998. Article (CrossRef Link)

[22] P. Khandelwal, P. Likhar and R. S. Yadav, "Machine Learning Methods leveraging ADFA-LD Dataset for Anomaly Detection in Linux Host Systems," in *Proc. of 2022 2nd International Conference on Intelligent Technologies (CONIT)*, pp.1-8, Hubli, India, 2022. Article (CrossRef Link)

[23] S. Wunderlich, M. Ring, D. Landes and A. Hotho, "Comparison of System Call Representations for Intrusion Detection," in *Proc. of International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019)*, AISC, vol.951, Springer, Seville, Spain, pp.14-24, May. 2020. Article (CrossRef Link)

[24] I. Rosenberg and E. Gudes, "Bypassing system calls–based intrusion detection systems," *Concurrency and Computation: Practice and Experience*, vol.29, no.16, Aug. 2017. Article (CrossRef Link)

[25] M. Xie, J. Hu and J. Slay, "Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD," in *Proc. of 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp.978-982, Xiamen, China, 2014. Article (CrossRef Link)

[26] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. of 2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp.4487-4492, Shanghai, China, Apr. 2013. Article (CrossRef Link)

[27] K. Cho, K. Mitsuya and A. Kato, “Traffic data repository at the WIDE project,” in *Proc. of the annual conference on USENIX Annual Technical Conference (ATEC '00)*, USENIX Association, USA, 2000. Article (CrossRefLink)

CHAO-HSIEN HSIEH received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2015. He is currently an Associate Professor with the College of Engineering, Xi'an International University, Xi'an 710077, Shaanxi, China. His research interests include cloud computing, blockchain, and database.

FENGYA XU is currently pursuing a master’s degree at the School of Cyber Science and Engineering, Qufu Normal University, Qufu, Shandong, China. Her research interests include load balancing direction in cloud computing and network security.

QINGQING YANG is currently pursuing a master's degree at the School of Cyber Science and Engineering, Qufu Normal University. Her research interests is cloud computing and information processing.

DEHONG KONG is currently pursuing a master's degree at the School af Cyber Science and Engineering, Qufu Normal University, Qufu, Shandong, China. Her research interests include cloud computing and big data.