




Article

# Detecting Mixed-Type Intrusion in High Adaptability Using Artificial Immune System and Parallelized Automata

Fu-I Chou <sup>1</sup>, Wen-Hsien Ho <sup>2,3</sup> , Yenming J. Chen <sup>4,\*</sup> , Jinn-Tsong Tsai <sup>5,\*</sup>  and Chia-Wen Chang <sup>4</sup>

<sup>1</sup> Department of Automation Engineering, National Formosa University, Yunlin 632, Taiwan; alvis.cfi@gmail.com

<sup>2</sup> Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, Kaohsiung 807, Taiwan; whho@kmu.edu.tw

<sup>3</sup> Department of Medical Research, Kaohsiung Medical University Hospital, Kaohsiung 807, Taiwan

<sup>4</sup> Department of Logistics Management, National Kaohsiung University of Science and Technology, Kaohsiung 824, Taiwan; garven0313@gmail.com

<sup>5</sup> Department of Computer Science, National Pingtung University, Pingtung 900, Taiwan

\* Correspondence: yjchen@nkust.edu.tw (Y.J.C.); jttsai@mail.nptu.edu.tw (J.-T.T.)

Received: 28 December 2019; Accepted: 17 February 2020; Published: 25 February 2020



**Abstract:** This study applies artificial immune system and parallelized finite-state machines to construct an intrusion detection algorithm for spotting hidden threats in massive number of packets. Existing intrusion detections are mostly not focused on adaptability for mixed and changing attacks, which results in low detection rate in new and mixed-type attacks. Using the characteristics of artificial immune and state transition can address the attacks in evolutionary patterns and track the anomalies in nonconsecutive packets. The proposed immune algorithm in this study is highly efficient based on a selection step in multi-island migration. Result shows that the algorithm can effectively detect mixed-type attacks and obtains an overall accuracy of 95.9% in testing data.

**Keywords:** intrusion detection system (IDS); artificial immune system (AIS); finite-state machine (FSM)

## 1. Introduction

Intrusion detection systems (IDSs) aim to identify and isolate all types of intrusion inside a computer or communication system [1]. Anomaly and misuse detections are popular in the field of detection. Existing systems result in high false alarm in multiple anomalies interlaced together. Despite the sophisticated algorithms in the history of development, an effective method discovering mixed-type attacks is still needed for security administrators.

Mixed-type attacks refer to the attacking pattern in which the suspicious packets are interlaced with normal or abnormal packets. Hackers often attack by sending abnormal packets, and some attacks can be identified by a single packet [1]. However, attackers might use useless or massive normal packets in a mixed mode in order to cheat autodetection algorithms [1]. Thus, a simple analysis of a single packet will fail to define the attack, and subsequent packet contents have to be analyzed through a specialized method [2]. In this study, we are no longer restricted to one single packet or analytical moment, we use finite-state machines (FSMs) to associate non-single packet attacks to internal tracking states. A state in the FSM can signal an attack. We apply parallelized FSMs to capture multiple suspicious attacks on the same time. Subsequently, an artificial immune system (AIS) is

used to learn the condition of state transition from packet information and generate a complete state transition diagram.

The immune algorithm or AIS is derived from imitating biological immune process with unique learning and memory features and superior identification capability in system activity log for detecting unknown attack modes [3,4]. When a virus enters an organism, the immune system will perform a chain of reactions to kill the foreign enemy. AIS possesses specificity, discreteness, adaptability, and learning and memory functions; hence, it can detect more new attacks than conventional methods and retain effective protection [5–8]. AIS is widely applied in various knowledge disciplines, particularly in the information technology sector. However, most previous applications in intrusion detection are restricted in static model analysis and lack adaptability to external environments. Although recent deep learning in neural networks produce excellent results in IDS [9], insufficient works concentrate on unknown types of attack. Immune system can well adapt to unknown attacks through its massive learning and adaptive capability [10,11]. Additionally, an FSM is a mathematical model of computation with finite number of states at any given time, initial state, and input triggering state transition with a predefined transition probability [3,12]. The advantage of this model is to keep tracking events effectively in a directed graph. Therefore, this model can be used to discern multiple attacks among massive mixed packets.

## 2. Literature Review

Software and hardware devices should be able to detect network behavior and provide relevant information to IDS. These data reports about computer network systems contain audit records in the operating system and header files for network packets [13,14].

IDS is normally divided into anomaly, misuse, and hybrid detections [1,15]. Anomaly detection alerts anything that deviates from normal actions, whereas misuse detection matches activities with registered abnormal patterns. Hybrid detection is the mixed of the two detections, producing a complementary role to perform improved tasks [2,16].

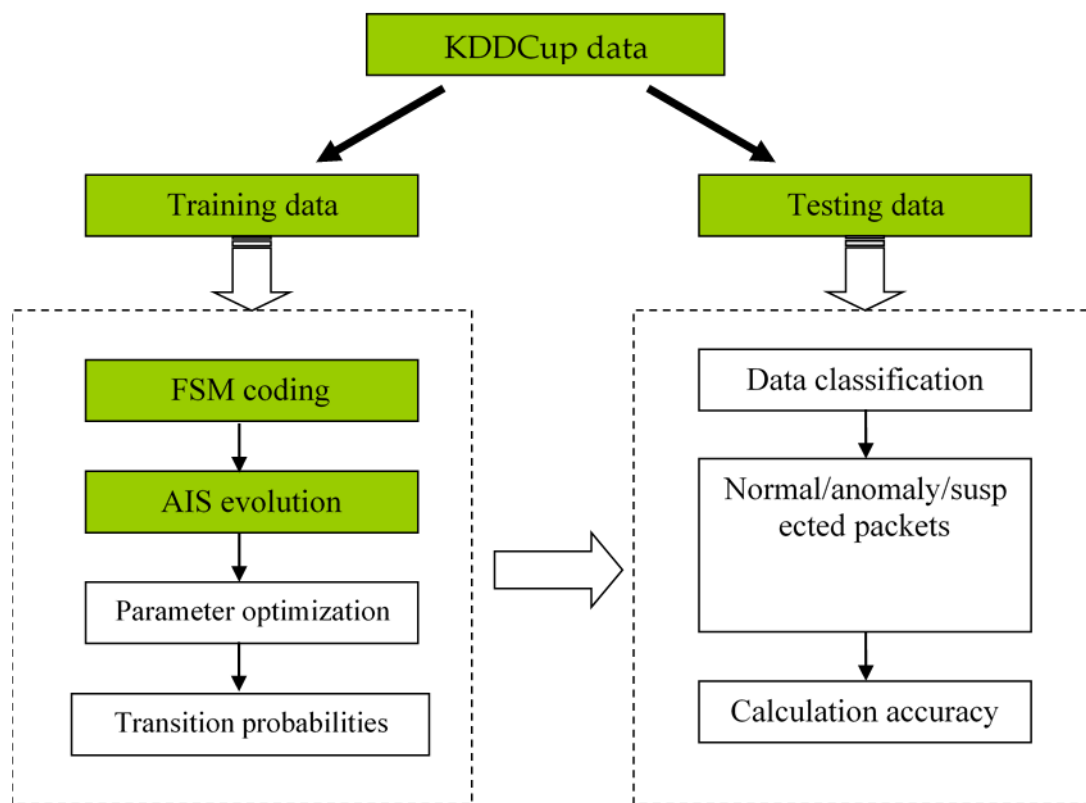
Bharati and Kumar [3] used statistical methods on an audit log as a data source. They also used normal packet data as a benchmark to establish a baseline and then compared it with the test material. If the baseline establishes standard values, then anything deviates from the baseline is considered abnormal. This method calculates the time to process large amounts of data; however, setting a baseline that can be effectively distinguished is difficult. Al-Khaleefa et al. [17] simulated the operation of the human brain, trained the weights of intermediate nodes with infinite term memory, and classified the packets into normal or abnormal patterns. This method is complex and time-consuming; security administrators cannot obtain clear information of the attacks through the trained weightings of nodes.

Bradley and Tyrrel [12] applied immune algorithm and a single FSM to monitor an electronic hardware system for potential operation failures. Bharati and Kumar [3] integrated state transition models and statistical methods into rule-based analysis methods and established nominal behavior based on frequent system calls, resource use cases, and file access pattern. Rule-based methods compare test data and identify abnormal access or use of resources. However, such methods cannot recognize new attacks. Fu et al. [18] classified the test data and then used string matching and FSM in constructing a high-speed IoT network IDS to improve the efficiency and increase the speed and quantity of network packet inspection. However, the inspection is limited to a single package, and the correlation between the envelopes cannot be considered fully. Hwang et al. [19] analyzed network audit logs to detect abnormal behavior using a data mining approach, such as a combination of associate rules and frequent episode.

### 3. Model Analysis

#### 3.1. Description of Simple Attack

This study uses the evolutionary mechanism of artificial immunology to estimate the transition probability of finite automata, as shown in Figure 1, i.e., FSM is to distinguish attacks and AIS is to learn the transition probability in detecting suspects. The probabilities are coded into the antibodies of AIS and the coding scheme of antibodies is based on the operation of automata. The number of possible attack states, which are an arbitrary number representing the limiting capacity of the learning system, is preset to meet the requirement of the experiment. An orthogonal matrix is used for the experimental design to reduce the number of trials for fine tuning the parameters for antibody optimization. The process is shown in Figure 1. The completed IDS will classify data packets into normal, known attack, undetermined (no fit to the output of the classifier), and conflict (fit all types) packets. The system performance is accessed to reflect its capability and stability.



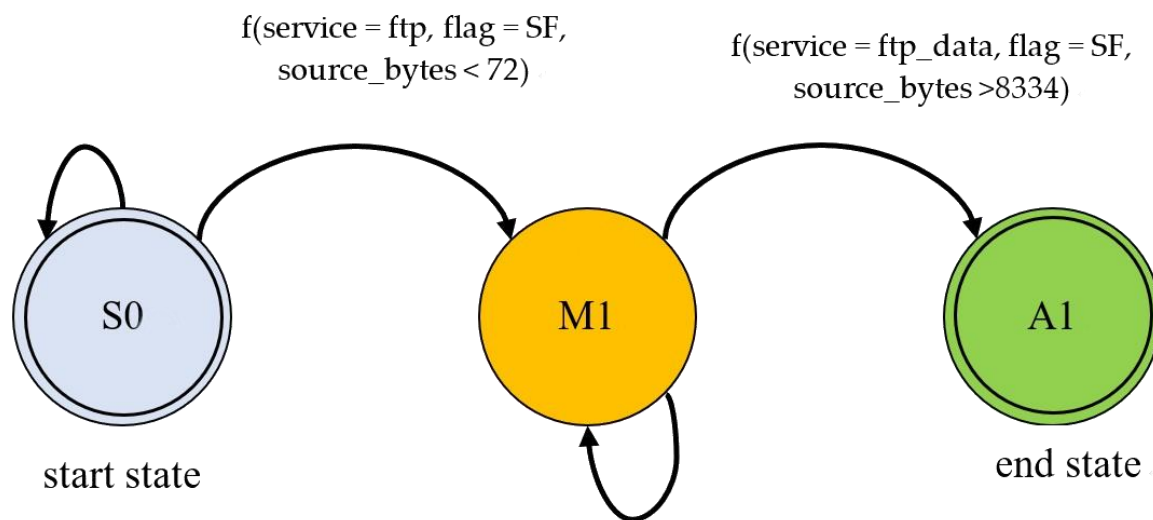
**Figure 1.** Procedures for the intrusion detection experiment. A dataset is divided to training and testing sets. The training set is used to adjust transition probabilities of FSMs via AIS according to known attack types. The testing set is used to calculate classification accuracy of trained FSMs by pretending that the attack types were unknown.

#### 3.2. Mixed-Type Attacks

The intrusion attack contains a chain of associated programs, and multiple packets form a complete attack process, which may be mixed with some normal packets in nonlinear time. For example, two warezmaster attacks [1] (Table 1) separated by one normal packet (Figure 2). This mixing packet could fool some detection algorithms. In the experiment, the correlation of non-consecutive attacks is captured by FSM, describing the attack process with a chain of state transition.

**Table 1.** A sample of data log in which a warezmaster attack was interlaced by a normal packet.

Packet	Duration	Protocol Type	Service	Flag	Src Bytes	Dst Bytes	Land	Wrong Fragment	Urgent	Hot	Label
76,627	0	tcp	http	SF	210	542	0	0	0	0	normal
76,628	1	tcp	Smtpt	SF	965	328	0	0	0	0	normal
76,629	0	tcp	Smtpt	SF	1191	368	0	0	0	0	normal
76,630	1	tcp	Smtpt	SF	1291	325	0	0	0	0	normal
76,631	0	tcp	Smtpt	SF	14,081	337	0	0	0	0	normal
76,632	12	tcp	ftp	SF	72	300	0	0	0	1	warezmaster
76,633	0	tcp	http	SF	185	635	0	0	0	0	normal
76,634	0	tcp	FTP data	SF	8334	0	0	0	0	0	warezmaster
76,635	1	tcp	smtpt	SF	1640	344	0	0	0	0	normal
76,636	0	tcp	http	SF	307	354	0	0	0	0	normal
76,637	0	tcp	http	SF	219	5014	0	0	0	0	normal
76,638	0	tcp	http	SF	212	3902	0	0	0	0	normal
76,639	0	tcp	http	SF	347	5320	0	0	0	0	normal
76,640	0	tcp	http	SF	327	365	0	0	0	0	normal
76,641	0	tcp	http	SF	296	7129	0	0	0	0	normal

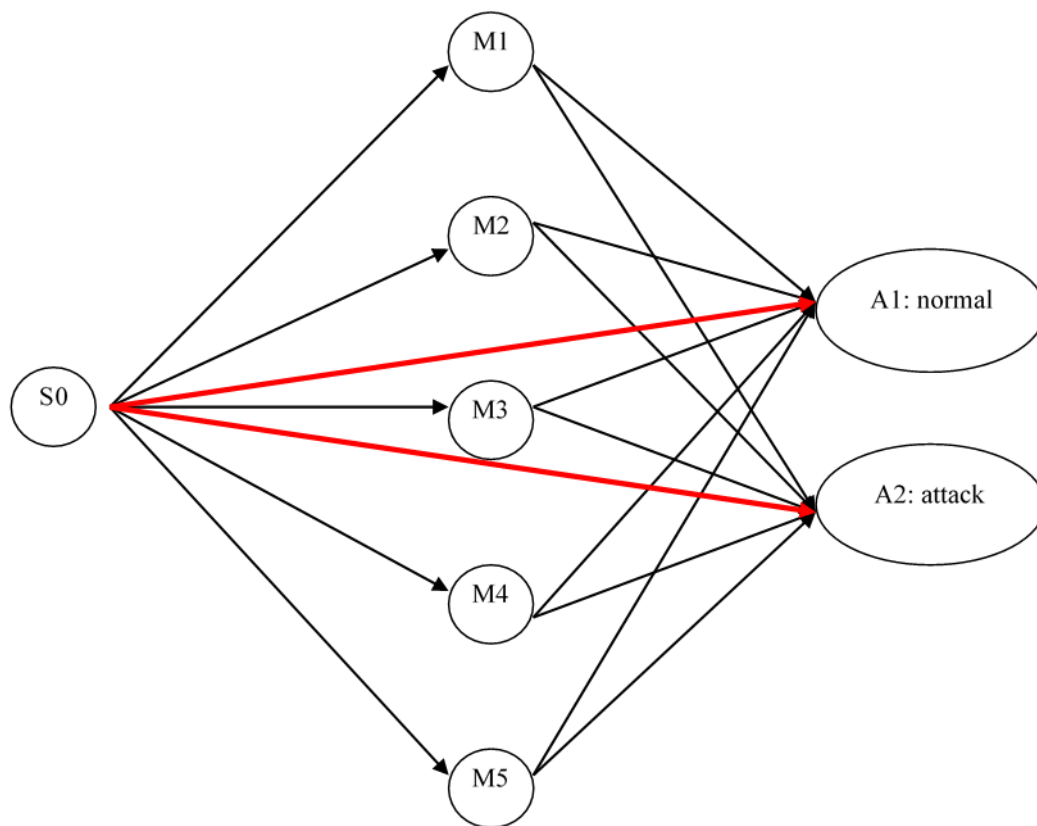


**Figure 2.** An illustration explaining the mechanism of state transition for detecting warezmaster among irrelevant packets. At beginning, a suspicious attack is detected,  $f(\text{service} = \text{ftp}, \text{flag} = \text{SF}, \text{source bytes} < 72)$ , but the attack activity cannot be certain completely. Therefore, state M1 is entered and the next state will be determined by further information. If sufficient evidence is observed,  $f(\text{service} = \text{ftp data}, \text{flag} = \text{SF}, \text{source bytes} > 8334)$ , state A1 is entered and the attack is captured completely.

In the state transition diagram designed for attack incidents (Figure 2), suppose that the state transition probability can be represented by a function containing three parameters, i.e.,  $P(S_i|S_j) = f(X, Y, Z)$ , where X, Y, and Z denotes service, flag, and source\_bytes, respectively. When the system begins to receive packet, it is at the initial state S0 with normal packets or not conforming to the first state transition function of attack. Given as an example, when the 76,632 nd packet is received, the parameters of the packet are service = ftp, flag = SF, and src bytes = 72. Then, the state transfers to M1. When two cases exist, normal packets are initially received continuously; then, they are kept under state M1 over a certain time t (we use number of packets in this study), wherein time t can be set by users with experience. The attacker might stop further attacks of network activities for certain reasons, or the network packet is lost. Hence, in order to limit the size of memory consumption, the FSM resource will be released, if no additional intrusive packets are received, and the system returns to state S0 for further detection. In the illustration, we manually set that the receiving packet has parameters of service = ftp data, flag = SF, and src bytes > 8334. Then, the state transfers to A1. After being calculated by the output function, the warezmaster attack can be determined.

### 3.3. Build Intrusion Detection Model Using FSM

In this study, a system model is constructed using FSM. As shown in Figure 3, there are three kinds of states, which are initial, possible attacks, and attack states. Lines in the graph represent state transitions. The red lines denote the attacks can be simply determined by a single packet. The black lines denote that a malicious attack cannot be determined by a single packet, and will be decided only after sufficient information is arrived. Therefore, we use intermediate states (M1–M5) to denote the status that the next states will be determined by other packets. The automata start from an initial state S0, transits to a suspected state M1–M5 (The number of states depends on the system resources), and finally reaches attack states (A1: normal, A2: attack). S0 can reach any suspected and attack state (antibodies 1–7), whereas each suspected state can conclude an attack state, for a total of 17 antibodies (Table 2).

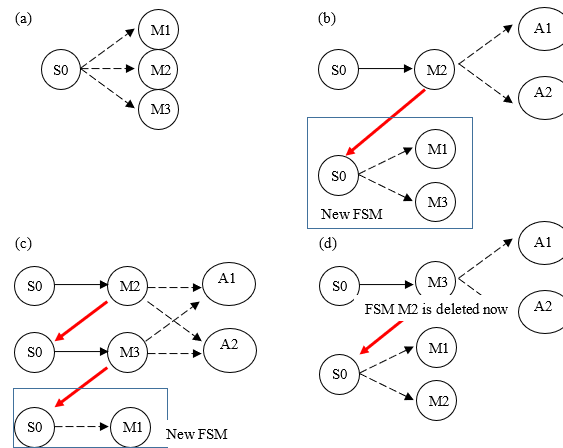


**Figure 3.** State and state transitions for the FSM of intrusion detection. Some attacks are easy to be captured, and, therefore, the red lines represent direct transitions to final results. Some attacks are elaborate. We have to wait for subsequent information to make the final decision. Each internal state M1–M5 corresponds to one suspicious attack. The number of states depends on the system resources.

**Table 2.** Assignment of state transition to antibodies.

Current State	S0									
Next state	M1	M2	M3	M4	M5	A1	A2			
Antibody#	1	2	3	4	5	6	7			
Current State	M1		M2		M3		M4		M5	
Next state	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2
Antibody#	8	9	10	11	12	13	14	15	16	17

The detection of suspicious attacks over an intermediate state is shown in Figure 4. The dash lines in the FSMs present possible future actions, which are not a part of standard notation and are only for explanatory purpose. When one possible attack state is detected, the system will request to start another state machine to detect multiple attacks simultaneously.



**Figure 4.** (a) An FSM tracking M1–M3 is initialized; (b), (c) new FSMs spawned when other intermediate attack types are encountered; (d) the FSM tracking M2 is abandoned after packet count  $t$ .

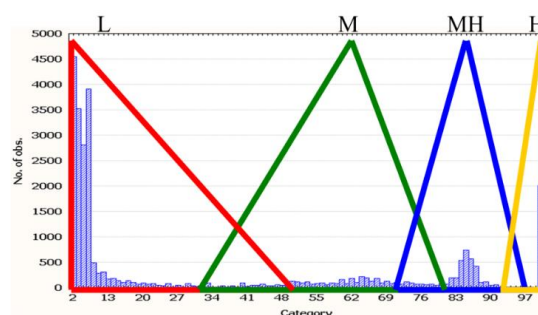
When the state machine detects suspicious packets compliant with M2, the system goes to a suspicious attack state and continues subsequent detection while verifying whether an attack is occurring.

When the state machine is in a suspicious attack state, it will count its life time  $t$  (number of packets) simultaneously. In this manner, system load can be reduced, and actuating excessive or overdue state machines can be avoided. If no associated packets in time  $t$  can prove the occurrence of an attack, then this state machine is abandoned to secure system resource operation.

The process is as follows.

- (a) Start detection.
- (b) When a suspicious attack is detected, a state machine tracking M2 will be spawned to detect the potential attacks by keeping monitoring the possibility of consecutive anomaly.
- (c) Upon detecting of another potential attack, a new state machine M3 will be spawned to keep tracking the consecutive suspicious pattern.
- (d) M2 will be abandoned after examining  $t$  packets to avoid system overload.

The study extracts 41 features from the provided packets information, including categorical and quantity values. We code the categorical features into antibody with Arabic numbers (Table 3), whereas the quantity values are converted to fuzzy numbers [20] (Figure 5).

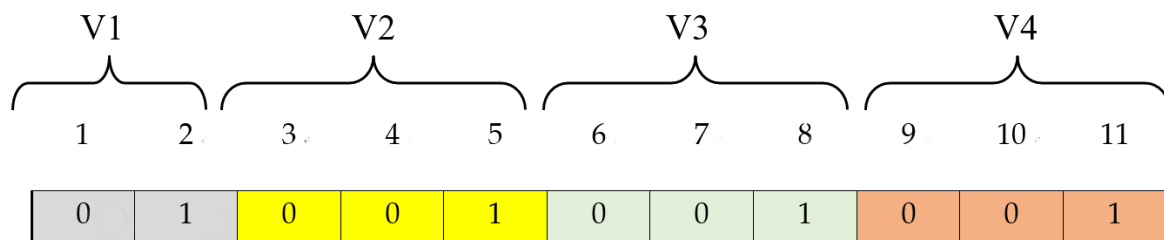


**Figure 5.** Quantity features need to be quantized before coded to AIS antibodies. We use histogram to help fuzzy quantizing the quantity values to four categories.

**Table 3.** Coding scheme for assigning values to categorical features.

Feature			
V1	V2	V3	V4
0: 1	L: 3	ftp: 6	tcp: 9
1: 2	M: 4	telnet: 7	udp: 10
	H: 5	http: 8	icmp: 11

Assume that the content of a certain packet is V1 = 1, V2 = H, V3 = http, and V4 = icmp. Then, the encoding for the categorical features is as Figure 6: The state transitions are assigned to antibodies according to the arrangement in Table 3. Antibodies 1–7 represent the transitions starting from state S0, whereas antibodies 8–17 represent the rest transitions that end to attack states A1–A2.



**Figure 6.** The encoding scheme for the features in an antibody.

Our algorithm is expressed as Figure 7. The affinity, as in Equation (1), is calculated as the fitness of antibody to pathogen. For example, two antibodies have affinity values 3 and 4 with respect to a probable attack A as shown in Table 4 ([21]). The unit of evolution is antibody set. Therefore, the actual affinity in (1) will add up all antibodies in an antibody set. The algorithm will keep high-affinity antibody sets in the memory.

$$Affinity = \sum_{i=1}^M \text{sum}(\text{notXOR}(\text{Attack}(i), \text{Antibody}(i))) \tag{1}$$

M: length of the antibody.

**Table 4.** An example for calculating the affinities of two antibodies with respect to attack A.

	Antibody Codes						The Affinity of Antibody to Attack A
Attack A	1	0	1	0	1	0	
Antibody 1	0	0	1	1	0	0	3
Antibody 2	1	1	0	0	1	0	4

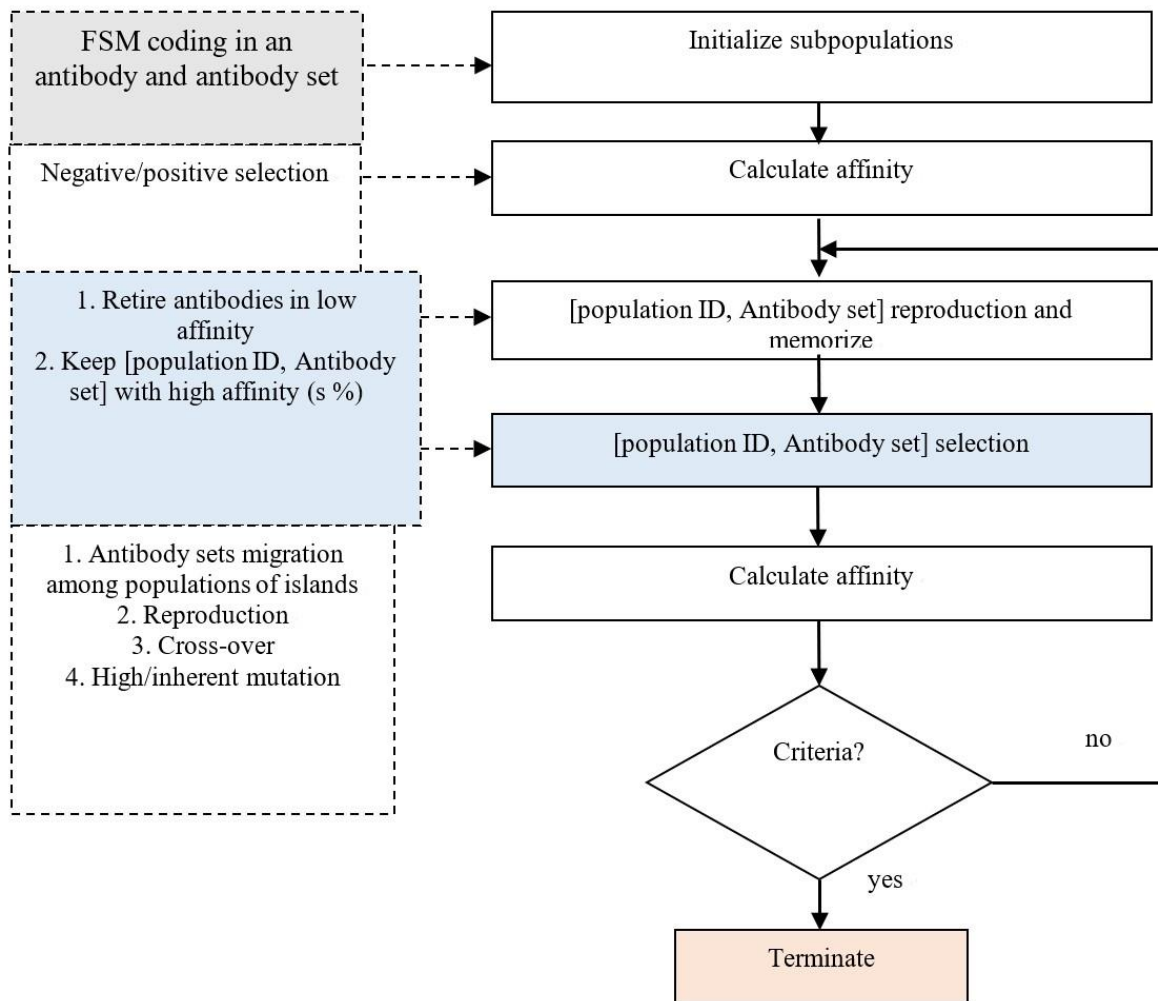


Figure 7. The computation flow of our AIS training algorithm.

### 3.4. Antibody Generation and Migration

As the antibody design, each learning unit, i.e., an antibody set, contains 17 antibodies, based on current illustrative design. If intermediate states have 10, the antibody set will contain 22 antibodies. Affinity is calculated according to various antibody sets. For example, in Figure 8,  $n$  antibody combinations exist, or an initial random population generates  $n$  recombinations at a time. Every antibody set contains 17 antibodies, each antibody training according to different data without interference. Affinity is calculated on the basis of all antibody sets. Thus, 17 antibodies are arranged in serial sequence.

Antibody Set 1	Antibody 1	Antibody 2	...	Antibody 17
Antibody Set 2	Antibody 1	Antibody 2	...	Antibody 17
...		...		
Antibody Set $n$	Antibody 1	Antibody 2	...	Antibody 17

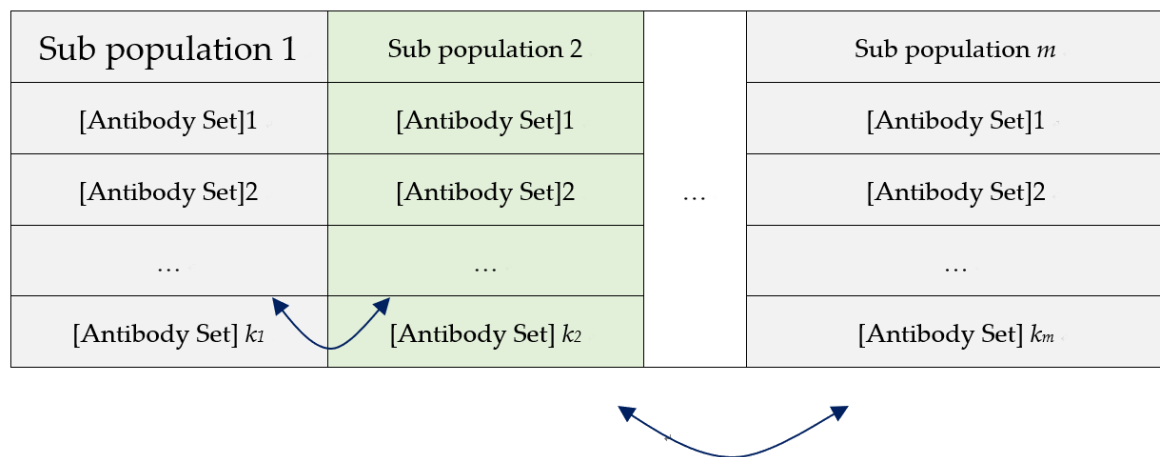
Figure 8. Allocation of antibodies for the transition functions of an FSM (an antibody set presents an FSM; a population will evolve to select a best antibody set).

The study adopts multiple random populations at one time (Figure 9) to simulate island genetic algorithm (IGA) migration, in which each population will evolve independently in their resident island [22]. IGA involves parallel involution in each island of population to avoid stuck into local optimum and provide opportunities to seek the best solution. The island structure is shown in



Figure 9. Many studies indicated that IGA outpaces conventional genetic algorithm ([22]). The best solution is rendered by keeping migration with good antibody sets in different islands to exchange and substitute with part of antibody sets in other populations over time, as shown by the arrows in Figure 7. The migration will be selected according to a predefined probability (migration rate). The size of slowly evolved population decreases, and the decreased numbers are passed to population with better parameters to increase the number of populations with better evolution parameters and find the best solution effectively.

After completion of the system state machine modeling, an affinity test of packet against each antibody is performed. If a packet’s antibody affinity is greater than a pre-calculated lower bound, this packet will be preliminarily determined to be a suspicious attack. However, the final conclusion should be obtained after acquiring the affinities of this packet with other antibodies.



**Figure 9.** The antibody sets migration between populations of islands. The best solution is rendered by keeping migration with good antibody sets in different islands to exchange and substitute with part of antibody sets in other populations over time. The migration will be selected according to a predefined probability (migration rate).

## 4. Results

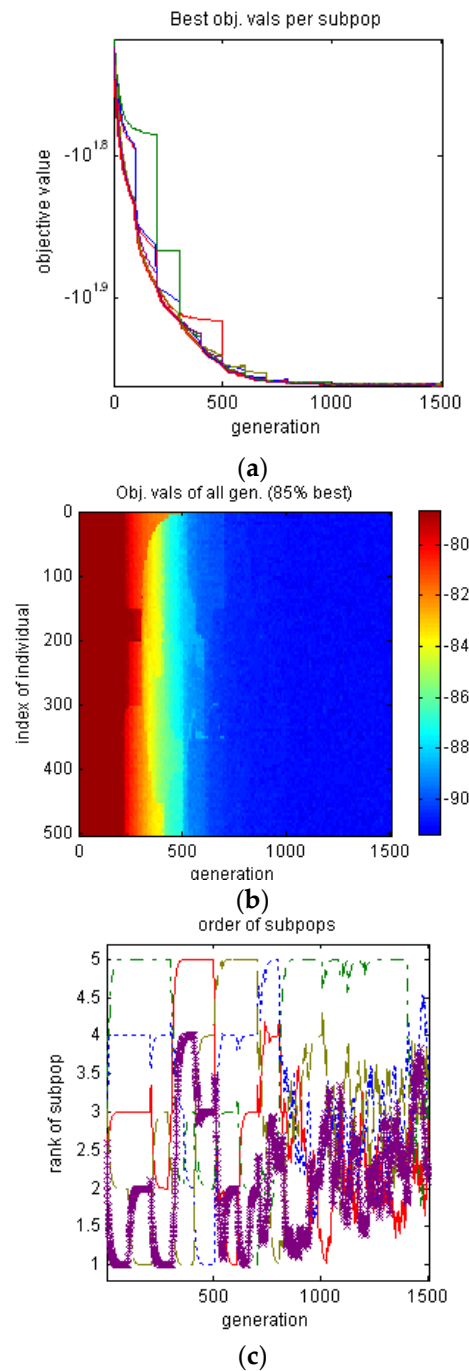
### 4.1. Experimental Design

To compare the performance our algorithm to existing studies, we follow the choice of the most the adopted dataset in the Third International Knowledge Discovery and Data Mining Tools Competition, [23]. The data used in the experiment were randomly divided into training and testing sets. The data contained 41 features in three categories, namely TCP header, content, traffic features. The KDDCup packet of the data is shown in Table 5.

Among the 23 types of attack in KDDCup data, this study screened nine types of common attacks plus one normal activity for training data, and each type contained 10,000 randomly selected packets. For testing, five additional types of attack were selected from the KDDCup data as testing data. The novel data were used as proof for the stability and adaptability of our method.

The algorithm initially used the parameter setting in Table 6 and created 50 populations with 100 k antibodies in each population. After the evolution, the convergence result is shown in Figure 10. The changes of the best affinity in selected five populations among the 1500 generations are shown by the different colors in Figure 10a (the thick red line is the best affinity for all populations). Evidently, affinity has gradually converged after 1000th generations. The best affinity changes of antibodies are shown in Figure 10b. The affinity changes among the populations are shown in Figure 10c. The evolution before 500th generations is fast, and the rate becomes moderate after 500. Slowly converged population may

also produce the best affinity antibody due to the IGA migration mechanism with antibody exchange. After 1000 generations, the affinity between populations become indifferent.



**Figure 10.** The convergence of the IGA evolution. (a) The changes of the best affinity in selected five populations among the 1500 generations are shown by the different colors (the thick red line is the best affinity for all populations). Evidently, affinity has gradually converged after 1000th generations. (b) The best affinity changes of antibodies are displayed in 3D heat map. (c) The affinity changes among the populations before 500th generations are fast, and the rate becomes moderate after 500. Slowly converged population may also produce the best affinity antibody due to the IGA migration mechanism with antibody exchange. After 1000 generations, the affinity between populations become indifferent.

**Table 5.** KDDCup data log.

Duration	Protocol Type	Service	Flag	Src Bytes	Dst Bytes	Land	Wrong Fragment	Urgent	Hot	Label
0	udp	private	SF	105	146	0	0	0	0	normal
0	udp	private	SF	105	146	0	0	0	0	normal
0	udp	private	SF	105	146	0	0	0	0	normal
0	udp	private	SF	105	146	0	0	0	0	snmpgetattack
0	udp	private	SF	105	146	0	0	0	0	snmpgetattack
0	udp	private	SF	105	146	0	0	0	0	snmpgetattack
0	udp	domain	SF	29	0	0	0	0	0	normal
0	tcp	private	SF	105	146	0	0	0	0	normal
0	udp	private	SF	105	146	0	0	0	0	snmpgetattack
0	tcp	http	SF	223	185	0	0	0	0	normal
0	udp	private	SF	105	146	0	0	0	0	snmpgetattack
0	tcp	http	SF	230	260	0	0	0	0	normal

Data source (<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>).

**Table 6.** Parameter setting in the IGA AIS.

Populations	5				
Population number	1	2	3	4	5
Antibody sets in a population	100	100	100	100	100
Migration interval	100				
Random samples	200				
Migration rate	0.1				
Selection type	Threshold selection				
Selection pressure	1.7				
Reproduction rate	0.1				
Cross-over rate	0.3	0.4	0.5	0.6	0.7
Mutation rate	0.02	0.05	0.02	0.05	0.02
Stopping criteria	Stopped at 1500th generation				

Threshold = 1/selection pressure (SP); Antibody value = antibody affinity/max affinity in a population at a generation; For example, a particular antibody will be selected if it must surpass the threshold ( $1/1.7 = 0.588$ ); antibody affinity = 1000; max affinity in a population at a generation = 3000;  $1000/3000 = 0.3 < 0.588$ ; thus, this antibody will not be selected.

#### 4.2. Optimization of Metaparameters

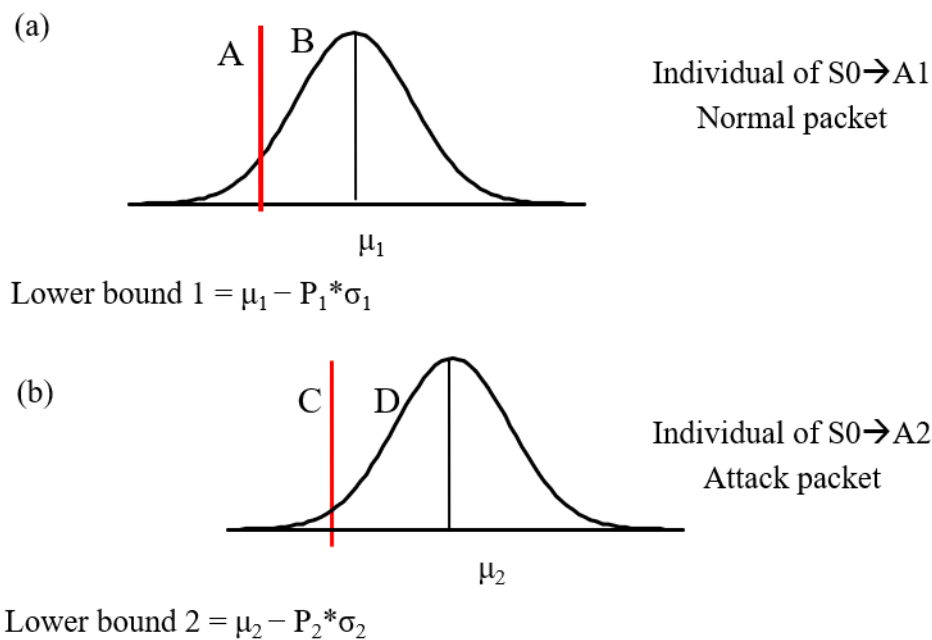
The lower bound of affinity matching must be calculated for each antibody to perform anomaly matching. Only those packets that the affinity is greater than the lower bound will be treated as an attack.

Two types of antibody are taken as examples in Figure 11. In Figure 11a, whether the antibody belongs to a normal packet is identified. In Figure 11b, whether the antibody is attacking packet is detected. The red lines are the lower bounds of antibodies. When the affinity between the packet and antibody is greater than that of the lower bound (or the affinity lies on the right of the red line), the antibody manifests a receiving state. If the packet is only received by the antibody of a normal packet, as detected, then this packet can be determined to be a normal packet; otherwise, if the packet is only received by the antibody of an attack packet, as detected, then it is determined to be an attacking packet. If the packet is rejected simultaneously by normal and attacking antibodies, then it is an unknown packet. If the packet is received by more than two types of antibody, then it is a conflict packet. The details are illustrated in Table 7.

**Table 7.** Illustration of the determination ranges from Figure 9.

Antibodies to Be Normal Packets		Antibodies to Be Attack Packets		Determination	
Less than A	Greater than B	Less than C	Greater than D	Unknown Conflict	Suspicious
yes	yes	yes	Yes		
yes	yes		yes		Attack
	yes	yes			Normal

To find most fit lower bound of each antibody (i.e., the best position of the red line in Figure 8), each type of experimental factor is classified as three levels of 0.5, 1.0, and 1.5 standard deviations. Five antibodies in the center state adopt one level, whereas the other 12 antibodies have different levels. Hence, a total of 13 experimental factors exist, with  $3^{13} = 1,594,323$  combinations in total. This study adopted the Taguchi method with an orthogonally designed array to reduce the number of tests in obtaining the desired result and determining the best level of each experimental factor [24]. The experiment needs 13 best levels in total. Thus,  $L_{27}(3^{13})$  orthogonal array was adopted.



**Figure 11.** Attack judgment for antibody affinity. The red lines are the lower bounds of antibodies. (a). When the affinity between the packet and antibody is greater than that of the lower bound 1 (B) and less than that of the lower bound 2 (C), the packet can be determined to be a normal packet. (b) If a packet is (A) and (D) on the same time, it is an attack packet.

The calculation of the lower bound is shown in Equation (2). In each set, the lower bound of each antibody was calculated according to different level combinations and then compared with the training data to obtain the value of  $D$  of each set (Equation (3)). Finally, the best parameter of each antibody was calculated from  $D$ . Only 27 tests in the experiment could have the best factor and level combination, according to the signal-to-noise ratio ( $S/N$ ) expressed in Equation (4).

$$Lbound_i = \mu_i - P_i \times \sigma_i \tag{2}$$

$Lbound_i$ : lower bound of  $i$ th antibody

$\mu_i$ : mean similarity of  $i$ th antibody and training data

$\sigma_i$ : standard deviation of similarity of  $i$ th antibody and training data

$P_i$ : experimental factor of  $i$ th antibody.

$$D = (\text{number of unknown packets} + \text{number of controversial packets}) \tag{3}$$

$$\frac{S}{N} = -10 \log \left[ \frac{1}{n} \sum_{i=1}^n y_i^2 \right] \tag{4}$$

After IDS modeling and level optimization in Equation (4) of each antibody, comparison of model and test data is conducted. In the experiment, the chosen ten types of attack in original training data (left half of Table 8), then screen them through secondary stage identification, leave attacks of normal packet and single packet to detect (right half of Table 8), and verify identification capability of direct detection and secondary detection; it can be known from table that the accuracy of this model in judging compound packet (packet including first and second identifications) (0.984) is very similar to that of single packet (0.989), therefore it can be concluded that this model can effectively detect compound attack packet, and regardless of single packet or compound packet attacks, high accuracy and precision can be obtained.

**Table 8.** Detection performance for single-type attacks.

Single-Type Attack (378,067 Records)			
		True condition	
		Normal	Attack
Predicted condition	Normal	93,394 (true positive)	0 (false positive)
	Attack	1916 (false negative)	280,519 (true negative)
Indeterminant	Conflict	167	0
	unknown	1895	176

tp = true positive; tn = true negative; fp = false positive; fn = false negative; Accuracy = (tp + tn)/total = (93394 + 280519)/378067 = 0.989; Recall rate = (tp)/(normal true conditions) = (93394)/(93394 + 1912 + 167 + 1895) = 0.959; Precision rate = (tp)/(normal predicted conditions) = 93394/(93394 + 0) = 1.

### 4.3. Experimental Results

After training, five additional types of attack in the original data were regarded as new attacks for the testing phase, with a total of 548 cases in total. The experimental results are shown in Tables 8–10. As shown in Tables 8 and 9, the performance of single-type and mixed-type attacks is superior. By adding new and unknown patterns of attacks, the performance remains good, as shown in Table 10. For the detection results of unknown or conflict, we treat them as attacks to avoid miss capturing.

**Table 9.** Detection performance for mixed-type attacks.

Mixed-Type Attack (491,438 Records)			
		True Condition	
		Normal	Attack
Predicted condition	Normal	93,394 (true positive)	3456 (false positive)
	Attack	1912 (false negative)	390,371 (true negative)
Indeterminant	Conflict	167	38
	Unknown	1899	201

tp = true positive; tn = true negative; fp = false positive; fn = false negative; Accuracy = (tp + tn)/total = (93394 + 390371)/491438 = 0.984; Recall rate = (tp)/(normal true conditions) = (93394)/(93394 + 1912 + 167 + 1899) = 0.96; Precision rate = (tp)/(normal predicted conditions) = 93394/(93394 + 3456) = 0.964.

**Table 10.** Detection performance for new-type attacks, which are completely unseen for the system.

New-Type Attacks (548 Records) Plus Previous Mixed-Type Attacks (491,438)					
		True condition			
		Normal	Attack	New Attack	
Predicted condition	Normal	81,300	2367	109	
	Attack	3623	390,265	318	
	Indeterminant	Conflict	376	423	40
		Unknown	423	1011	81

Accuracy = 0.959; Recall rate = 0.835; Precision rate = 0.897.

The overall detection accuracy was 95.9%. This study produced sufficient recall rate in terms of single and mixed-type intrusive packets. Although our algorithm has different goal to the KDD (Knowledge Discovery and Datamining) competition, we are still interested in a comparison to single-type attacks. The winner of KDD cup and a recent research had an overall accuracy of 92.7% and 96.5%, respectively [25] compared with the rate of 95.9% in this study. For the massive unknown variants from intrusion hackers, a small difference in the accuracy number may not be meaningful in performance comparison. For other performance metrics, we have a lower recall rate but have high precision, i.e., we have less type I error. Type I error implies labeling malicious packets to normal packets. Detection problems always face the trade-off between false alarming and miss capturing. We think a good security administrator cannot easily allow suspicious activities to exist under security supervision. Therefore, the decision trade-off will toward to high precision, instead of high recall. Thus, the proposed algorithm contributes to attack detection in presenting mixed-type and new attacks.

## 5. Conclusions

This study is the first to define multiple possible states using FSM theory and the evolution property of AIS to establish the transition function between varied states. Then, this study conducts a non-consecutive packet analysis, which cannot be found in a single packet. Furthermore, the detection of new intrusion is not a superficial direct analysis; rather, this study uses an intermediate state to extract new attacks similar to known attacks and has an identification capability for new intrusions, attaining stability of intrusion detection algorithm. This study attributes unidentifiable and controversial packets to one type, providing reference in analysis for administrators of user organizations, and the high stability relieves administrators of labor load caused by excessive erroneous messages.

**Author Contributions:** Conceptualization, Y.J.C. and J.-T.T.; Methodology, Y.J.C.; Software, Y.J.C.; Validation, Y.J.C. and F.-I.C.; Formal Analysis, Y.J.C.; Investigation, Y.J.C. and W.-H.H.; Resources, C.-W.C.; Data Curation, C.-W.C.; Writing-Original Draft Preparation, Y.J.C.; Writing-Review & Editing, W.-H.H.; Visualization, Y.J.C.; Supervision, Y.J.C.; Project Administration, Y.J.C. and J.-T.T.; Funding Acquisition, F.-I.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Ministry of Science and Technology in Taiwan grant numbers MOST 107-2410-H-992-014-MY2, MOST 108-2221-E-037-007 and MOST 108-2218-E-992-304, and in part by the "Intelligent Manufacturing Research Center" (iMRC) from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

**Conflicts of Interest:** The authors have no conflicts of interest to declare.

## References

1. Bragg, R.; Strassberg, K.; Rhodes-Ousley, M. *Network Security: The Complete Reference*. McGraw-Hill/Osborne: New York, NY, USA, 2004; p. 854.
2. Chan, A.; Ng, W.; Yeung, D.; Tsang, C. Refinement of rule-based intrusion detection system for denial of service attacks by support vector machine. In *Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2004; Volume 7, pp. 4252–4256.*

3. Bharati, T.S.; Kumar, R. Intrusion detection system for manet using machine learning and state transition analysis. *Int. J. Comput. Eng. Technol.* **2015**, *6*, 2.
4. Hao, Y.; Sheng, Y.; Wang, J. A graph representation learning algorithm for low-order proximity feature extraction to enhance unsupervised ids preprocessing. *Appl. Sci.* **2019**, *9*, 4473. [CrossRef]
5. Kim, J.; Bentley, P. Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; Volume 2, pp. 1244–1252.
6. Dozier, G.; Brown, D.; Hurley, J.; Cain, K. Vulnerability analysis of AIS-based intrusion detection systems via genetic and particle swarm red teams. *Congr. Evol. Comput.* **2004**, *1*, 111–116.
7. Zhang, Y.; Wang, L.; Sun, W.; Green, R.C.; Alam, M. Artificial immune system based intrusion detection in a distributed hierarchical network architecture of smart grid. In Proceedings of the 2011 IEEE Power and Energy Society General Meeting, Detroit, MI, USA, 24–28 July 2011; pp. 1–8.
8. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **2018**, *25*, 152–160. [CrossRef]
9. Asghar, M.Z.; Abbas, M.; Zeeshan, K.; Kotilainen, P.; H'am'al'ainen, T. Assessment of deep learning methodology for self-organizing 5g networks. *Appl. Sci.* **2019**, *9*, 2975. [CrossRef]
10. Chen, M.H.; Chang, P.C.; Wu, J.L. A population-based incremental learning approach with artificial immune system for network intrusion detection. *Eng. Appl. Artif. Intell.* **2016**, *51*, 171–181. [CrossRef]
11. Saurabh, P.; Verma, B. An efficient proactive artificial immune system based anomaly detection and prevention system. *Expert Syst. Appl.* **2016**, *60*, 311–320. [CrossRef]
12. Bradley, D.; Tyrrell, A. Immunotronics—novel finite-state-machine architectures with built-in self-test using self-nonsel self differentiation. *IEEE Trans. Evol. Comput.* **2002**, *6*, 227–238. [CrossRef]
13. Sultan, Z. Multiple simultaneous threat detection in Unix environment. *Int. J. Comput. Sci. Netw. Secur.* **2009**, *9*, 65–75.
14. Shin, Y. A vm-based detection framework against remote code execution attacks for closed source network devices. *Appl. Sci.* **2019**, *9*, 1294. [CrossRef]
15. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
16. Kabir, E.; Hu, J.; Wang, H.; Zhuo, G. A novel statistical technique for intrusion detection systems. *Future Gener. Comput. Syst.* **2018**, *79*, 303–318. [CrossRef]
17. Al-Khaleefa, A.S.; Ahmad, M.R.; Isa, A.A.M.; Esa, M.R.M.; Al-Saffar, A.; Hassan, M.H. Feature adaptive and cyclic dynamic learning based on infinite term memory extreme learning machine. *Appl. Sci.* **2019**, *9*, 895. [CrossRef]
18. Fu, Y.; Yan, Z.; Cao, J.; Kon'e, O.; Cao, X. An automata based intrusion detection method for internet of things. *Mob. Inf. Syst.* **2017**. [CrossRef]
19. Hwang, K.; Cai, M.; Chen, Y.; Qin, M. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *IEEE Trans. Dependable Secur. Comput.* **2007**, *4*, 41–55. [CrossRef]
20. Wang, C.-N.; Huang, Y.-F.; Chai, Y.-C.; van Thanh, N. A multi-criteria decision making (MCDM) for renewable energy plants location selection in Vietnam under a fuzzy environment. *Appl. Sci.* **2018**, *8*, 2069. [CrossRef]
21. Stibor, T.; Timmis, J.; Eckert, C. On the appropriateness of negative selection defined over Hamming shape-space as a network intrusion detection system. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 2, pp. 995–1002.
22. Ho, W.-H.; Chiu, Y.H.; Chen, Y.J. Multi-Objective Pareto Adaptive Algorithm for Capacitated Lot-Sizing Problems in Glass Lens Production. *Appl. Math. Model.* **2018**, *53*, 731–738. [CrossRef]
23. KDD1999. Available online: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (accessed on 28 February 2020).

24. Lan, T.-S.; Chuang, K.-C.; Chen, Y.-M. Optimization of machining parameters using fuzzy Taguchi method for reducing tool wear. *Appl. Sci.* **2018**, *8*, 1011. [[CrossRef](#)]
25. Behdad, M.; Barone, L.; French, T.; Bennamoun, M. On XCSR for electronic fraud detection. *Evol. Intell.* **2012**, *5*, 139–150. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).