

Early Detection of LDDoS Attacks in IoT Utilizing Locality Sensitive Incremental TSVM Method

Xiaochun Yin*, Zengguang Liu**, Deyong Liu***, Zhengze Liu****

*Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science & Technology, Shouguang, Weifang, Shandong, China

**College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong, China

***Shandong Software Engineering Technology Center, Weifang University of Science & Technology, Shouguang, Weifang, Shandong, China

****School of Software & Microelectronics, Peking University, Beijing, China

xiaochunyin@wfust.edu.cn, st.lzg@163.com, gm@univalsoft.com, 1931087307@qq.com

Abstract—The Mirai botnet and its variants has made Internet of Things (IoT) devices a powerful amplifying platform for Low-rate Distributed Denial-of-Service (LDDoS) attacks. In this paper, we firstly propose a new low-rate variant, which is a sophisticated crossfire low-rate DDoS attack mechanism. At the same time, we investigate and develop a novel semi-supervised Locality Sensitive Incremental Transductive Support Vector Machine (LS-ITSVM) method. The proposed method maximizes the margins of different network flows by incorporating local frequency-domain features from the autocorrelation sequence of network flow into the regularization time-domain framework of TSVM. And it saves training and detecting time by incremental training support vectors and new added samples. In experiments, we verify the proposed crossfire LDDoS is more concealed and harmful firstly. Then, the result in public dataset proves the proposed method can distinguish abnormal network flows with higher detection accuracy, faster training and response time, and prevent abnormal network flow groups with less impact. At last, the results in private testbed prove LS-ITSVM is still available to new LDDoS variants.

Keyword—LDDoS, Internet of Things, Locality Sensitive Incremental TSVM, Frequency-Domain Features, Semi-Supervised Clustering

I. INTRODUCTION

DISTRIBUTED Denial-of-Service (DDoS) attacks, characterized by brute-force, sustained high rate or specifically designed to explore the protocol limitations or software vulnerabilities in services, are well-known. The difficulty of traditional internet DDoS is how to control as many as desktop computers, which has strong password, updated anti-virus software regularly, and unexpected off-line. With the emergence of IoT, things change. IoT devices have weak security configurations, lacked computational capabilities to run anti-virus, and constantly connected to Internet. As a result, it becomes soon a powerful amplifying platform for DDoS attacks. The recent prominent examples are Mirai botnet. According to the French webhost and cloud service provider OVH, a DDoS attack using Mirai malware hit them with 1.1 Tbps at peak by about 145K hacked cameras in September 2016 [1]. Moreover, hackers announced they could rent a Mirai botnet of 400K Bots only two months later [2]. Thus, researches tried to understand Mirai botnet operation and communication model [3], and proposed many feasible methods.

In order to dodge existing detection proposals, Mirai's variants with low-rate attack mechanisms start to obfuscate their DDoS activity and are becoming a serious threat to Internet. As we know, a successful DDoS attack needs to tune a tuple (T, R, L) of TCP at least. T is an integer multiple coincides with minRTO value of TCP; R is the peak rate and L is the burst length. In order to cause TCP packet loss and begin slow-start, R should be large enough and L should be longer than RTT value. As Fig. 1 shown, we simulated a LDDoS attacks of 4 Mirai Bots with the R of 50Kb/sec. For the purpose of drawing them clearly in Fig. 1, we start to attack one by one with a delay of 10ms. Obviously, we can see that the LDDoS attack stream hides itself among normal traffic by making its peak rate even lower than the normal

Manuscript received Dec. 14, 2020. This research was supported by Scientific Talent Fund Project of Weifang University of Science & Technology of China (Grant number: KJRC2021002) and was also supported by the Scientific Fund Project of Facility Horticulture Laboratory of Universities in Shandong of China (Grant number: 2018YY016) and Key Research and Development Program of Shandong, (Grant number: 2019GNC106034)., and a follow-up of the invited journal to the accepted & presented paper entitled " Early Detection of LDDoS Attacks in IoT Utilizing Locality Sensitive Incremental TSVM Method" of the 22th International Conference on Advanced Communication Technology (ICACT2020).

Xiaochun Yin is with the blockchain laboratory of agricultural vegetables, Weifang university of science & technology, China. (e-mail: xiaochunyin@wfust.edu.cn).

Zengguang Liu is with college of computer science and engineering, Shandong university of science and technology, China. (Corresponding author, phone: 18853688750; e-mail: st.lzg@163.com).

Deyong Liu is with Shandong software engineering technology center, Weifang university of science & technology, China. (e-mail: gm@univalsoft.com).

Zhengze Liu is with school of software & microelectronics, Peking university, China. (e-mail: 1931087307@qq.com).

traffic rate. All in all, compared with traditional DDoS attacks, LDDoS ones have three mainly characteristics: hard to detect, because it has same flow classified features with normal flow; low-cost and multi-targets, because the attacks can be finished in a few nodes with small flow data, they can be orchestrated to attack multiple targets simultaneously; long term attacked-target insensitive attacks, because attacked-target has self-adapt mechanism to adjust network flow. Thus, the aim of this paper is to have a method with high detection accuracy, fast response time to detect LDDoS attacks.

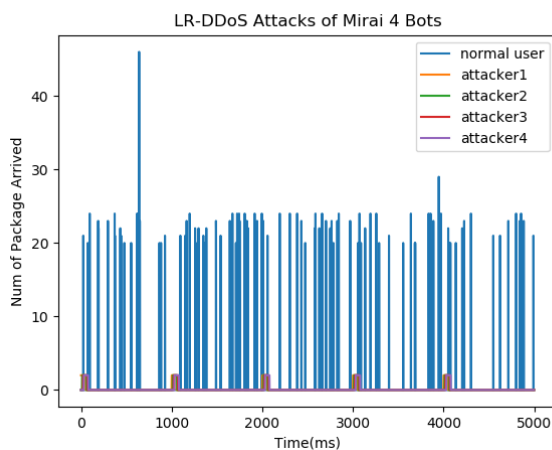


Fig. 1. LDDoS Attacks of Mirai 4 Bots

This paper is organized as follows: we firstly review the related state-of-the-art work in section II. Next, we work out a Mirai's variant: crossfire LDDoS by adding sophisticated low-rate TCP attack mechanisms and multiplexing algorithm to IoT DDoS in section III. And then, a novel semi-supervised Locality Sensitive Incremental Transductive Support Vector Machine (LS-ITSVM) method is proposed and described to address the new crossfire LDDoS in section IV. Next, simulation results are presented to show the effectiveness of defending against LDDoS attacks in Section V. Finally, Section VI concludes the paper with suggestive future enhancements.

II. RELATED WORK

After Mirai botnet, IoT botnets had received spread attention. Including traditional detection methods, researchers began to protect IoT with the skill of machine learning (ML) and Artificial Intelligence (AI) [4-7]. When LDDoS attacks came into the world, above methods were not applied to these new type attacks. As a result, researchers had to focus on new attacks, and a few defensive solutions had been existed in the literature. G. Kaur et al. [8] worked out a variant of the Shiryayev-Roberts's procedure to detect and deter these stealthy low-rate DoS attacks dynamically, using Checkpoint Anomaly Detection techniques. Z. Wu et al. [9] studied complex multifractal structure for developing concise mathematical models, and proposed an algorithm of multifractal detrended fluctuation analysis (MF-DFA), which is used to explore the change in terms of multifractal characteristics over a small scale of network traffic due to LDDoS attacks. ML and AI also were applied in LDDoS. Z. G. Liu et al. [10] proposed a new method to take frequency-domain characteristics from the autocorrelation sequence of network flow as clustering feature to group end-user flow data by BIRTH algorithm, and re-merge these

clustering results into new groups by overcoming the deficiency of BIRTH algorithm. M. Yue et al. [11] investigated a new identification approach based on wavelet transform and combined neural network to classify normal network traffic and LDDoS attack traffic. Z. J. Wu et al. [12] used the propagation back (BP) model of neural network to establish the nonlinear model of network traffic, and proposed a method of LDDoS attacks detection based on particle filter according to the mechanism of LDDoS attack. In this method, the difference between the estimated value of the particle filter and the one step prediction is used as the detection basis, and a detection threshold is designed to determine the initiation and termination of the LDDoS attacks for the purpose of detecting LDDoS attacks. P. Nagar et al. [13] provided a comparative study between the ANN and the optimizer-based ANN technology. They concluded that the convolution neural network with SVM show effective analysis providing accurate forms of IDS, thereby improving its detection based on individual class along with maintaining its results fundamentally. X. Peng et al. [14] optimized a Mahalanobis distance by perturbing continuous features and discrete features of DDoS samples respectively, and proposed an improved boundary-based method to craft adversarial DDoS samples. M. Begli et al. [15] designed an intrusion detection system (IDS) using SVM algorithm, and proved its efficiency. P. Rivas et al. [16] implemented and trained a non-probabilistic binary linear attack pattern classifier, and trained a support vector machine and a convolutional neural network using a supervised learning model with labelled data sets. Experimental results suggested that the models can detect DDoS attacks with high accuracy rates. T. A. Tuan et al. [17] performed an experimental analysis of the machine learning methods (including Support Vector Machine, Decision Tree, Naive Bayes, and ANN) for Botnet DDoS attack detection. The evaluation is done on the UNBS-NB 15 and KDD CUP-99 which are well-known publicity datasets for Botnet DDoS attack detection. Results showed SVM has higher accuracy. Although these methods have high accuracy, they need more training and detecting time when new samples were labelled.

Based on above insights and the truth of massive untagged packages in IoT, we will propose and develop a novel semi-supervised LS-ITSVM method for higher detecting accuracy and fast training and response time. Additionally, we try to present a sophisticated IoT-Bots Multiplexing Method for crossfire LDDoS Attack, which is more powerful and stealthy.

III. CROSSFIRE LDDOS ATTACKS MODEL

In IoT era, there are huge number of Mirai bots. But lots of bots' attack capability are wasted, because there are many unused gaps between bursts, which can be well-organized to have another attack. Thus, hackers could be exactly leveraged on their large number of Mirai bots and divide them into several subsets to low individual traffic rates for hiding their attack behaviors further. At the same time, the bots' capability can be used fully to fire orchestrated crossfire attack [18, 19].

Crossfire attack [18] is a powerful attack that degrades and often cuts off network connections to a variety of selected server targets by flooding only a few network links. In its attack-flow assignment step, each bot can try its best to send attack flow without worrying about being discovered. Because there is an assumption that the middle routers only have failure

detection mechanism, instead of attack detection mechanism. However, it is not true in network world, especially in IoT world. For example, some attackers set the goal of bots multiplexing to form desired waves with fewest bots [20]. But these kind of attacking flow were detected easily by statistical approach in middle router's idle CPU circle. More researchers are hiring Software-Defined Networking (SDN) to control switches and routers placed at the network edges, which helps prevent LDDoS in time. If we use random ultra-low peak rate, existing detecting methods are hard to find these flows out. Thus, the question is how to work out an attacking plan based on a simple multiplexing method for the attackers. Fortunately, we can have lots of hacked bots in IoT, that is, we have chance to patch the gap on the bursts and make an efficient bot multiplexing scheme. Therefore, we use dynamic random fewest bots to fire attacks alternately. According to this idea, we propose the bots multiplexing low-rate algorithm as following algorithm 1. In this algorithm, one key point is to split bots' capability into random small ones, which can utilize much less bandwidth and resources. This will be harder to detect.

ALGORITHM 1: BOTS MULTIPLEXING ALGORITHM

Input: DelayMatrix = mat(NBOTS, NTARGETS); CapMatrix = mat(NBOTS, 1); TargetIPMatrix = mat(NTARGETS, 1);
Output: AttackPlan, which includes TargetBots, BeginTime and TargetIP.
Notes: NBOTS is the number of bots, while NTARGETS is the number of targets; MAXALLOWCAP is the max allowed capability of ultra-low-rate attack.

Procedure:

```

1: Initialize DelayMatrix based on detected delay time from bots to targets;
2: Initialize CapMatrix based on bots' max bandwidth;
3: for cap in range(CapMatrix):
4:   if cap >= MAXALLOWCAP:
5:     split into multiple parts, and insert them into CapMatrix random;
6: Set attacking count, attacking time(L) and peakrate(R);
7: for i in range(attackcount):
8:   for j in range(NTARGETS):
9:     random k in range(NBOTS):
10:    if bots has been used in current attacking:
11:      continue;
12:    end if
13:    if bots ever used to attack this target:
14:      continue;
15:    end if
16:    if attacking peak rate has been met:
17:      break;
18:    end if
19:    if DelayMatrix[TargetBots[j]][-1][j]-DelayMatrix[k][j]>=L:
20:      update TargetBots by targetBots[j].append(k);
21:      update corresponding DelayMatrix line by
      DelayMatrix[k][eachTarget] = DelayMatrix[k][eachTarget] + L;
22:    end if
23:  end random
24: end for
25: refresh TargetBots matrix when time up or most bots has been used;
26: end for
27: generate AttackPlan based on TargetBots, DelayMatrix and TargetIPMatrix;
```

IV. LOCALITY SENSITIVE TSVM METHOD

In LDDoS attacks scenarios, attack flows almost have same classified features with normal flows, and they can fire long term attacked-target insensitive attacks. As a result, the traffic volume analysis method cannot detect such a stealthy attack any more. Recently, some researchers begin to introduce the semi-supervised TSVM methods [21-25]. The pros are that these methods can take use of the underlying

geometric structure of unlabelled samples to train the classifier, while the cons are that they are not suitable for incremental learning and their training speed is too slow to meet IoT's requirement. In order to address above issues, we propose a novel locality sensitive incremental TSVM method in this paper. "Locality sensitiveness" helps to quickly locate which cluster the unlabelled data belongs to and speed up training. "Incremental TSVM" can help to do incremental learning by support vectors and prior classified unlabelled data.

A. Locality Sensitive Features Extraction

ALGORITHM 2: LOCALITY SENSITIVE FEATURES EXTRACTION ALGORITHM

Input: NetworkDataSet = mat(labeled dump network items);

EntropyThresh;

Output: priComps = array(features). HashTable = array(hashBuckets)

Procedure:

```

1: Initialize FeatureList based on NetworkDataSet;
2: Set well-tuned EntropyThresh;
3: Calculate frequency-domain features based on time-domain features (PSD) based on DFT/FFT, and add it into FeatureList;
4: for features in FeatureList:
5:   Calculate its entropy based on (3);
6: end for
7: Extract features in ascending order by probability value based on EntropyThresh;
8: for vector in NetworkDataSet:
9:   Cast vector with extracted features by Hamming min-Distance algorithm, and generate h(vector);
10: end for
11: Generate hash buckets with same hash values;
12: Generate HashTable based on buckets.
```

The idea of locality sensitive is for solving the approximate in high dimensional spaces. It can help to map similar items to the same buckets with high probability. With it, the problem of "searching the similar samples in a huge data set" is turned to "searching them in a small bucket". Thus, the searching can speed up. In order to use locality sensitive, we need to satisfy the following conditions for any $x_1, x_2 \in M$:

- if $d(x_1, x_2) \leq d_1$, then $h(x_1) = h(x_2)$ with probability at least P_1 ;
- if $d(x_1, x_2) \geq d_2$, then $h(x_1) \neq h(x_2)$ with probability at most P_2 ;

Here, M is data set space; d is a distant function in M ; x_1, x_2 are vectors in M ; d_1 and d_2 is the threshold of distance; h is a function used to map elements from M to a bucket $s \in S$.

As to x_1, x_2 , we should choose them from training dataset well. In our case, if we review the Fig. 1 carefully, we find that there are always periodic low-rate rectangle shape waves. Generally, what exactly happens is that more power of the autocorrelation function is distributed in the lower frequency band if there is shrew stream contained in the traffic. Discrete Fourier Transform (DFT) and Fast Fourier Transformation (FFT) is exactly the mathematics method to convert a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform. Thus, we add frequency-domain representations of time-domain features Power Spectrum Density (PSD) converted by DFT/FFT as new features.

After that, features subset is chosen by enhanced information entropy. Information entropy is the average rate at which information is produced by a stochastic source of data. The measure of information entropy associated with

each possible data value is the negative logarithm of the probability mass function for the value:

$$S = -\sum_i P_i \log P_i \quad (1)$$

When the data source produces a low-probability value, the event carries more “information” than when the source data produces a high-probability value. In order to use entropy in feature extraction, conditional probability is introduced into entropy definition. Now entropy is calculated based on event probability. Every feature value is treated as event. Thus, event entropy means the distribution of value. The re-definition of (1) is as following:

$$S(i, l) = \int P(x|\omega_i) \log P(x|\omega_i) dx \quad (2)$$

Here, $S(i, l)$ is the entropy of i^{th} feature value in single classification of l . Thus, the expectation of all classification would be $S(i)$, it is a reference point of different features. The formula can be defined as

$$S(i) = E(S(i, l)) = \sum_i S(i, l) P(\omega_i) \quad (3)$$

As to h , which aims to maximize the probability of a “collision” for similar items, there are several open-sourced and validated methods [26-28]. We can use any one of them directly. Here, we use minimal distance algorithm family in Hamming space [26]. Since it has a huge improvement compared to other algorithms.

B. Incremental TSVM Algorithm

TSVM extend SVMs in that they could treat partially labelled data in semi-supervised learning by following the principles of transduction. The key idea of TSVM is that it begins with a labelling of the test/predict data based on the classification of an inductive SVM. Then it improves the solution by switching the labels of test/predict examples so that the objective function decreases. In other words, TSVM can be treated as seeking an optimal solution in given labelled dataset: $\{(x_i, y_i)\}, i = 1, 2, \dots, n; x_i \in R_m; y_i \in \{+1, -1\}$; and unlabelled dataset: $x_1^*, x_2^*, \dots, x_k^*$, and it can be described as following:

$$\begin{aligned} \min V(y_1^*, \dots, y_k^*, \omega, b, \varepsilon_1, \dots, \varepsilon_i, \varepsilon_1^*, \dots, \varepsilon_k^*) \\ = \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^n \varepsilon_i + C^* \sum_{j=1}^k \varepsilon_j^* \\ \text{s.t.} \\ \forall_{i=1}^n: y_i [\omega \cdot x_i + b] \geq 1 - \varepsilon_i \quad (4) \\ \forall_{j=1}^k: y_j^* [\omega \cdot x_j^* + b] \geq 1 - \varepsilon_j^* \\ \forall_{i=1}^n: \varepsilon_i \geq 0 \\ \forall_{j=1}^k: \varepsilon_j^* \geq 0 \end{aligned}$$

In (4), parameter C is the impact factors of hyperplane of labeled samples, while C^* is the impact factors of unlabeled samples; ξ_i and ξ_i^* are slack variables; $\langle \omega, b \rangle$ is the hyperplane. For obtaining the optimal solution of (4), the first step is to set well-tuned C and C^* by historical experiment, and to learn the initialized SVM classifier based on labeled samples through induction method. The second step is to use this SVM classifier to classify all unlabeled samples, and mark only pairwise labels of positive and negative samples in support vectors nearby. The last step is to re-train SVM based on new chosen samples and labeled samples. Training procedure is repeated between step 2 and 3, until all unlabeled samples have their own classification. From above procedure, there are two disadvantages of transduction method. The one is that TSVM builds no predictive model. And the other one is if a previously unknown point is added to the set, the entire transductive algorithm would need to be repeated with all of the points in order to predict a label.

In real IoT environment, we need not only train a predictive model on limited labeled samples, but also adjust the model by new coming packages in real-time. Obviously, TSVM can be computationally expensive in this kind of case. Further, this might cause the predictions of some of the old points to change, which is unexpected in early warning system and is treated as false positive. After an in-depth study of TSVM, we find that “not all samples” play the same important role during learning the optimal solution of (4). These samples that meets the support vector conditions contribute a lot to find out optimal hyperplanes and decision functions. In general, the x_i whose lagrange multiplier’s value α_i is between 0 and C ($0 < \alpha_i < C$) is defined as normal vector, while the x_i whose lagrange multiplier’s value α_i equals C ($\alpha_i = C$) is defined as support vector. The later represents classified features of most samples, and can be learned to get the final classification. In other words, the set of support vectors can fully describe the features of the whole training set, learning in the set of support vectors is equivalent to learning in whole training set. Obviously, the scale of support vectors set is far smaller than training set. In every iterable training, only the support vectors are mentained and combined with the new added samples for new classifier. With this, TSVM has the capacity of incremental learning. The incremental TSVM algorithm can be described as following algorithm 3.

ALGORITHM 3: INCREMENTAL TSVM ALGORITHM

Input: NetworkDataSet = mat(labeled dump network items);
NewNetworkDataSet=mat(unlabeled network items);
Output: ITSVMClassifiers, LabeledNewNetworkDataSet;
Notes: NetworkDataSet $\cap$ NewAddedNetworkDataSet= $\emptyset$

Procedure:

- 1: Initialized NetworkDataSet;
- 2: Use TSVM algorithm to train NetworkDataSet, and get the base ITSVMClassifier, and SupportVecSet;
- 3: **while** (NewNetworkDataSet != NULL):
- 4: get k samples from NewNetworkDataSet; // k can be any pre-set value.
- 5: Generate new training set combined SupportVecSet and k, marked as NewTrainingSet;
- 6: Use TSVM algorithm to train NewTrainingSet to get NewITSVMClassifier, and NewSupportVecSet;
- 7: **end while**

ALGORITHM 4: INTEGRATED ALGORITHM OF LOCALITY SENSITIVE AND ITSVM

Input: Ω_o, Ω_k ;
Output: classifier $f(x)$, predictedResults

Procedure:

- 1: create HashTable H_o and $\Omega_{o,sv}$ for Ω_o per Algorithm 1 & 2;
- 2: **while** ($\Omega_i \neq \text{NULL}$ && $i \leq k$):
- 3: calculate hash vectors $h(\Omega_i)$;
- 4: **for** each hash vectors $h(\Omega_i)$:
- 5: **if** (vector exists in HashTable H_{i-1}):
- 6: pre-mark the vector per the correlated label of HashTable H_{i-1} ;
- 7: pick up this vector into $\Omega_{i,sh}$;
- 8: **end if**
- 9: **end for**
- 10: merge $\Omega_{i-1,sv}$ into $\Omega_{i,sh}$ for **new** $\Omega_{i,sh}$;
- 11: **do**
- 12: train classifier $f(x)$ based on Ω_o , **new** $\Omega_{i,sh}$ per Algorithm 3;
- 13: classify $\Omega_{i,sh}$ based on $f(x)$;
- 14: **if** (pre-marked values != predicted values)
- 15: unmark these values, and treat them as unlabeled samples;
- 16: **end if**
- 17: **while** (unlabeled samples != NULL)
- 18: report predictedResults for Ω_i ;
- 19: generate new support vector set $\Omega_{i,sv}$;
- 20: create new HashTable H_i ;
- 21: **end while**

From algorithm 3, the biggest distinguished difference between incremental TSVM and SVM is that the learning samples of incremental SVM consist of only the support vectors of the learned samples and the new learning samples, while the SVM consists of all the learned samples and the new learning samples. Incremental TSVM discards some samples without losing classification accuracy. At the same time, its training time is speeded up.

C. Integrated with Locality Sensitive and ITSVM

Now that we have worked out locality sensitive features extraction method for locating and pre-labelling examples, at the same time, we have incremental TSVM for quickly incremental learning. Thus, we further integrate them together for semi-supervised learning quickly more and trying to improve classification accuracy by a small margin.

Assumed that there is a small labeled dataset Ω_o , the new sequenced added unlabeled dataset $\Omega_k (k = 1, \dots, n)$. The support vector set correlated Ω_k is $\Omega_{k,sv}$. The hash table of Ω_k is H_k . $h(\Omega_k)$ is used to represent hashed values vector. Samples set filtered by locality sensitive hash function is $\Omega_{k,ish}$. $f(x)$ is the classifier of ITSVM. With these, integrated algorithm of locality sensitive and ITSVM is detailed described as algorithm 4.

V. EXPERIMENTS AND RESULTS

A. Crossfire LDDoS Attacks Experiments

A.1 Experiment Environment

In order to verify that crossfire LDDoS attack has more concealment on the premise of maintaining the same attack effect. A crossfire LDDoS attack test platform is designed and implemented as shown in Fig. 2.

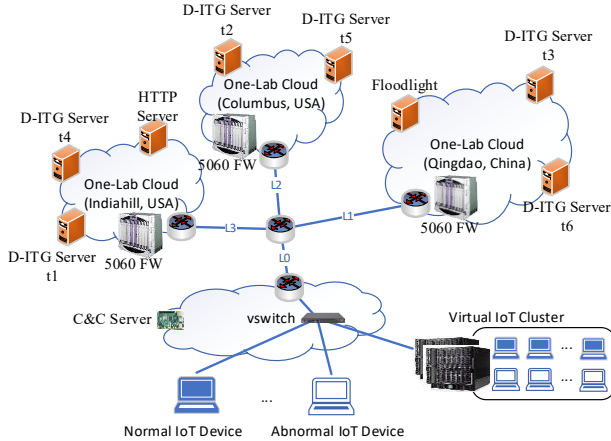


Fig. 2. LDDoS Attack Test Bed

In the IoT part of the test platform, Lenovo ThinkPad T470 laptop is used to simulate IoT device, run D-ITG tool to generate normal traffic and LDDoS attack traffic, and run slowhttptest tool to launch HTTP LDDoS attack. In order to simulate massive IoT devices, 10 HP C7000 servers are used to simulate 4,000 IoT devices through virtualization technology at max. Nokia one-lab cloud is used to deploy D-ITG server, HTTP server and Nokia 5060 FW, respectively. The link bandwidth between different nodes is as following. the link bandwidth in cloud is 10,000 Mbps, the link bandwidth of L0 and L1 is 50Mbps, the link bandwidth of L2 and L3 is 10Mbps, and the local bandwidth is 100Mbps.

A.2 The Experiment of Attack Concealment

We assume that we have 10 bots (says b_0 - b_9) for 3 targets (says t_1 - t_3). These bots belong to heterogeneous bots group, which includes 4 random size sub groups: 1Mbps, 2Mbps, 3Mbps and 4Mbps. As a result, the capability matrix of bots is as Fig. 3. The targets is D-ITGs who locate in Indiahill, Columbus and Qingdao.

C&C server expects to use enough distributed bots to generate 7Mbps peak rate with 500ms of burst length. Per algorithm 1, we random choose the first bot (says b_0), and recalculate its delay times to other targets as shown in Fig. 4-Delay Matrix- b_0 . And then, the sequenced bots (b_2) is chosen based on its CapMatrix and its interval, since b_1 cannot meet “concurrent attack time” with b_0 . The part of final attacking plan are shown in Fig. 4-Attacking Plan. We can see that attack targets of bots are totally different in the given time (or in the given the utilization rate of bots). Take b_0 and b_2 as an example, b_0 begins to attack t_1 in “relative time” 0ms, and then to attack t_2 in 500ms. while b_2 begins to attack t_2 in “relative time” 5ms, and then to attack t_1 in 507ms. Obviously, this algorithm can enhance attack ability effectively.

	b0	b1	b2	b3	b4	b5	b6	b7	b8	b9
R	1M	4M	4M	2M	1M	3M	2M	3M	3M	4M

Fig. 3. The Capability Matrix (CapMatrix) of Bots

Delay Matrix:

Original Matrix				Matrix after two bots chosen			
	t1	t2	t3		t1	t2	t3
b0	72	78	6	b0	572	578	506
b1	86	70	44	b1	86	70	44
b2	30	80	0	b2	572	622	542
...				...			
b9	47	24	20	b9	47	24	20

Attacking Plan:

epochs	bots	Begin Time	Target IP
1	b0	0	t1
2	b0	500	t2
...	...	...	...
1	b3	5	t2
2	b3	507	t1
1	b4	52	t1
2	b4	578	t2
2	b5	513	t2

Fig. 4. The Case of Crossfire LDDoS Algorithm of Bots

It can be seen from the attacking plan that bots attack multiple targets in turn. Taking b_0 as an example, b_0 attacked the target t_1 in a time of 0ms, and then attack the target t_2 in 500ms. Here, we draw the attacking plans of 1000 epochs (including the enlarged view of the attacking plans of the first 10 epochs) into a 3D graph as shown in Fig. 5. From the perspective of the attacker, each bot b_i constantly attack two or more targets. This shows that bots' attack capability has been fully utilized. From the perspective of the attacked target, each target t_i is attacked by a very wide range of b_i . The bots do not repeatedly attack the same target for a long time. This shows that the attack is more concealed. With the increase of attack epochs, as shown in the 3D graph of 1000 epochs of attacking plan in Fig. 5, all bots' attack capabilities are fully invoked to attack the targets as concealed as possible.

If the number of bots is enough, the attack behaviour will be more concealed. Here, we increase the number of controlled bots to 4,000, and draw the 3D attacking plan from

the perspective of the attack target. The experimental results are shown in Fig. 6. It is clear that there is no duplicate b_i appeared from any t_i 's view.

In conclusion, the proposed attack algorithm has good concealment, and can make full use of the ability of bots to launch low-rate attacks that are more difficult to detect.

A.3 The Experiment of Attack Performance

This section designs crossfire attack experiments to verify the attack performance of the algorithm through the indices of throughput.

In the experiment, we start 10Mbps aggregation traffic to D-ITG t_2 , and start 20Mbps aggregation traffic to D-ITG t_3 , which lead to L_0 blocked. We also initiate normal access of 5Mbps to D-ITG t_1 . The sampling period is set to 100ms, and the sampling length is set to 120s. The throughput of L_0 and t_1 is shown in Fig. 7. Per L_0 , the throughput reaches the peak bandwidth of the link, which shows that the proposed attack algorithm can generate effective attacks and block the link. But per t_1 , It is seen that the normal access traffic has been at a low level, indicating that the proposed algorithm successfully downgrades the normal access. Therefore, the experiment proves that the proposed crossfire LDDoS attack algorithm has more low-rate attack ability.

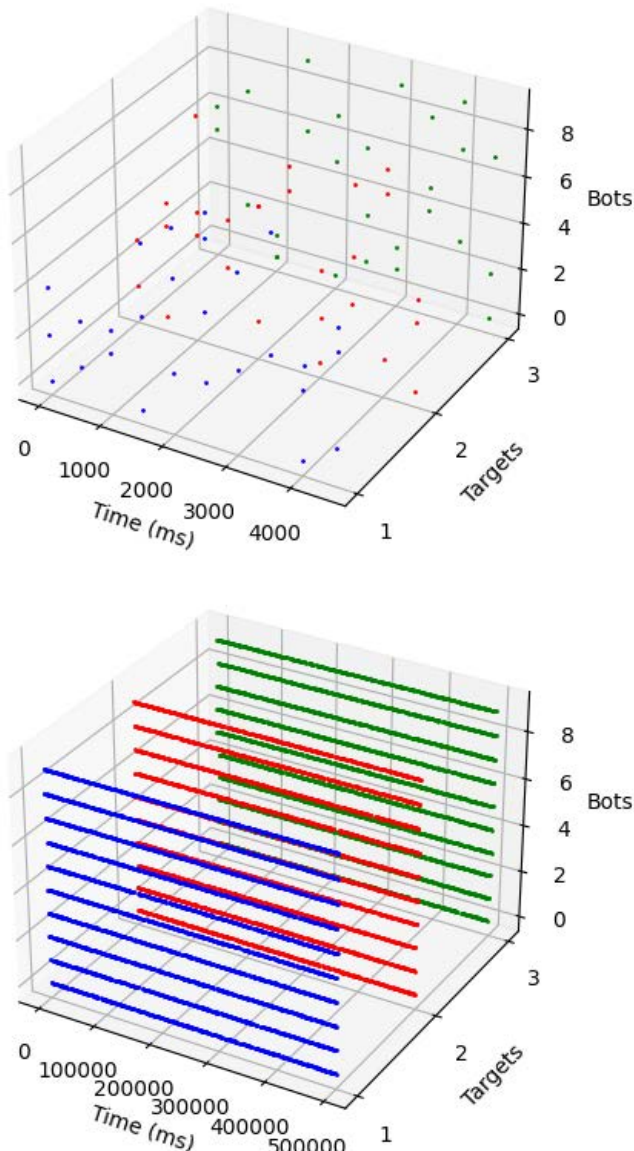


Fig. 5. The 3D Graph of Attacking Plan with 10 Bots

B. LS-ITSVM Defense Algorithm Experiments in Public Dataset

The proposed LS-ITSVM method should be a general and universal method, which has high detection accuracy and fast training and response time. In order to verify these, we carried out the following experiments, and made the comparison with Label propagation algorithm (LPA), and ISVM.

In the experiment, we apply our algorithm in KDD CUP 1999 dataset, which can prove the algorithm is general and universal method, and it has a good performance than others.

B.1 Experiment Environment and Dataset Description

The experiment is conducted on a laptop with Intel(R) Core (TM) i5-7200U CPU. GPU is not used here. The library Keras with TensorFlow as the backend is imported into the proposed model implementation.

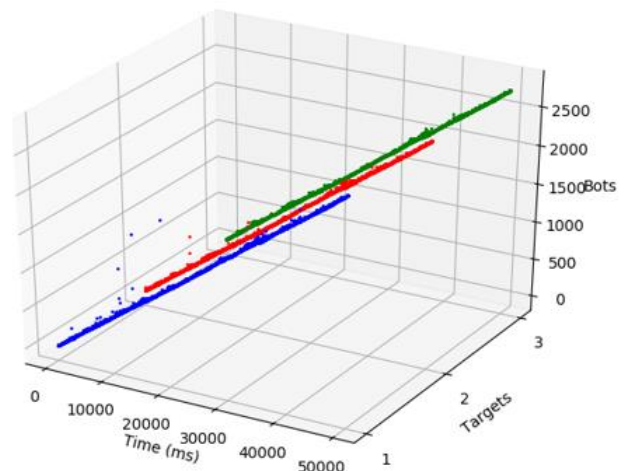


Fig. 6. The 3D Graph of Attacking Plan with 4,000 Bots

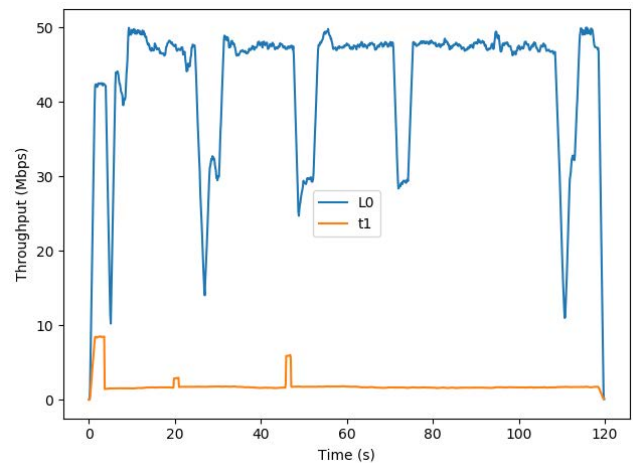


Fig. 7. The Throughput of Services and Links under Crossfire Attacks

KDD CUP-99 dataset was provided by Lincoln Labs, who set up an environment to acquire nine weeks of raw TCP dump data for a local-area network simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. The raw training data was about four gigabytes of compressed binary TCP dump data. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. In order to verify the above algorithm quickly, we use kddcup.data_10_percent as data source. We randomly choose 50,000 data items as test set. We split the to-be-trained data set into labeled and unlabeled parts by 20%:80%.

B.2 Features extraction

Each of those samples in “KDD CUP-99 10% dataset” has 41 features. 38 features of them are character and number related features. Thus, we pre-process them by reflecting and normalization. E.g. there are 3 choices in protocol_type: tcp, udp, icmp, which are reflected into 0, 1, 2. And then, we generate PSD by FFT and add them as new dimensions. E.g. in Fig. 8, the normalized cumulative amplitude spectrum (NCAS) value of dst_bytes are shown. What exactly happens is that more power of the autocorrelation function of dst_bytes is distributed in the lower frequency band when there is LDDoS contained in the traffic. After above operations, dimensions of features are bumped to 110.

In order to avoid the influence of the dimension of feature attribute on the experiment, it is necessary to unify the dimension of the experimental data. Thus, we use (5) to normalize the feature attributes to [0, 1]. In (5), x is the feature vectors. MIN is the minimal value of x , while MAX is its maximal value.

$$x = \frac{x - MIN}{MAX - MIN} \quad (5)$$

After above necessary dimensionality expand and normalize. The entropies of them are calculated for feature extraction. We well tune the threshold to 50, whose entropy values are bigger than 0.1, and chose them as training features.

B.3 Evaluation Indicators Analysis

In order to explain that the proposed method is better than others, we compare two indicators: detection accuracy score, training and response CPU time.

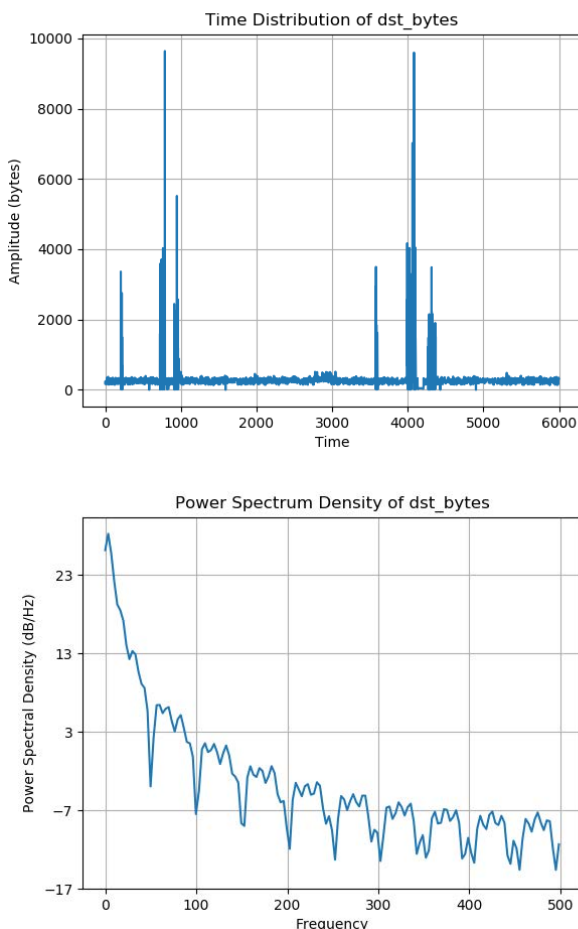


Fig. 8. The Time and Frequency Distribution of dst_bytes

Detection accuracy refers to how closely a prediction comes to measuring a "true value", since prediction are always subject to error. Here, we use the ratio between predicted value and the corresponding true value detailed as following, where $1(x)$ is the indicator function.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1(\hat{y}_i = y_i) \quad (6)$$

As Fig. 9 shown, the proposed locality sensitive ITSVM almost has the same accuracy with ISVM, even if we bump the semi-supervised increase set size to 100k. but locality sensitive ITSVM method has far better detection accuracy than LPA method. The accuracy of LPA is stick to 80% or so. And even worst, LPA is a huge memory-consumed method. It reports memory is not enough when it calculates label. Thus, there is not statistical data when size is bigger than 15k.

The training and detecting CPU time is the other critical parameter to evaluate the performance of locality sensitive ITSVM method. We define it as the time when method trains with semi-supervised incremental data and detects whether malicious flows exist or not. As Fig. 10 shown, the CPU time of ISVM is increased continually with the growth of increase set size, and bumps to almost 100s from 60s. Same to accuracy, LPA method is stopped when size is bigger than 15k. Compared with them, our proposed method training and detecting time is stable in 50s.

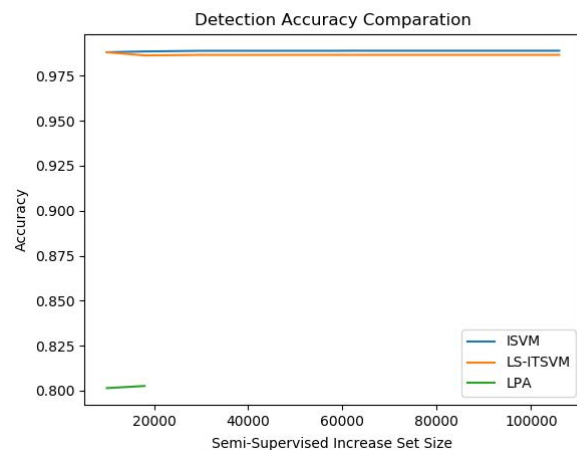


Fig. 9. The Detection Accuracy Comparison with KDD-CUP Dataset

Thus, take both detection accuracy score and CPU time into account, locality sensitive ITSVM, which has better performance, can detect DDoS with higher accuracy and shorter CPU time.

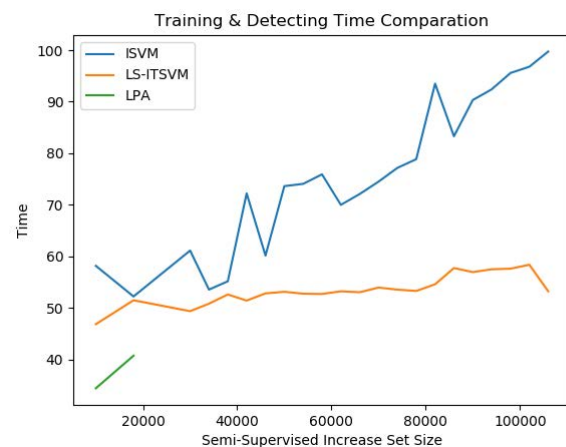


Fig. 10. The Training and Detecting Time Comparison with KDD-CUP Dataset

C. LS-ITSVM Defence Algorithm Experiments in Simulated IoT Dataset

C.1 Dataset Description

We hacked randomly 1k endpoints/users as bots, and start to attack our cloud data center by algorithm 1 with the peak rate of 50Kb/sec and the attacking period of 500ms. This can lead to network traffic jam of whole cloud data center. We collect all raw TCP dump data with same dimensions and with extra source and destination IP. Thus, our detection target becomes to not only detect abnormal flows, but also detect which bots fire the attacks.

C.2 Features extration

We do the same things with “KDD CUP-99 10% dataset”, and find out the top 50 features with sorted entropy values. Especially, we reserve IP addresses as features for the purpose of detecting bots.

C.3 Evaluation indicators Analysis

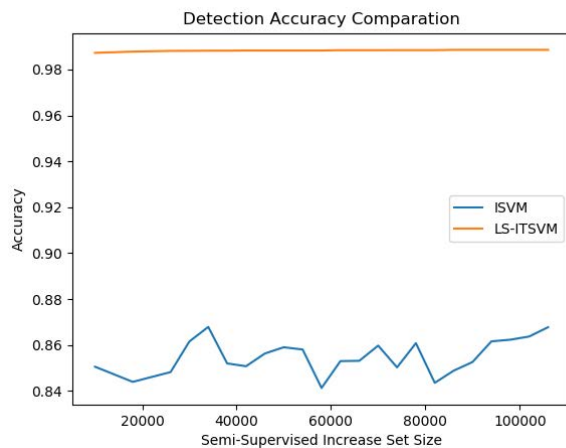


Fig. 11. The Detection Accuracy Comparison with IoT Dataset



Fig. 12. The. Training and Detecting Time Comparison with IoT Dataset

If we remove IP addresses temporarily, the comparisons are almost the same with the experiment result of “KDD CUP-99 10% dataset”. That is, both LS-ITSVM and ISVM have higher detection accuracy when we do FFT transform. whereas when we merge IP info into features, the accuracy of ISVM is down to 86% or so as Fig 11 shown. Notes: we don’t compare with LPA, since it has been approved that it has worst performance in above experiment.

In CPU time facet, they have the same trend. With the growth of the size of data set, ISVM CPU time is increased

continually, while the CPU time of LS-ITSVM keeps stable from time to time as Fig. 12 shown.

VI. CONCLUSION

In the paper, we firstly enhance IoT DDoS attack mechanism as a kind of new Mirai botnet variants: crossfire LDDoS attacks. And then, we develop a novel semi-supervised locality sensitive incremental TSVM method for resolve Mirai botnet and this new variant. At last, the results in KDD CUP-99 prove LS-ITSVM has better performance, and the indicators in simulated IoT lab prove LS-ITSVM is still available to new LDDoS variants. Per above experiments, we can conclude that the proposed method can differ bots from normal users. As known, LDDoS is a complex large-scale network attack behavior. The KDD CUP-99 is a good dataset, but it is too old. In order to improve the fidelity and ensure the scale of LDDoS attack and defense technology emulation, we will use Long Short-Term Memory Networks (LSTM) to extract time-domain characteristics of LDDoS as the specific generating conditions of GAN, and establish a Condition Generative Adversarial Networks (CGAN) model to generate mimicking behaviors of bots in the near future. Meanwhile, we will also spend more time in SDN and work out a method to prevent these bots into the data center.

ACKNOWLEDGMENT

This research was supported by Scientific Talent Fund Project of Weifang University of Science & Technology of China (Grant number: KJRC2021002) and was also supported by the Scientific Fund Project of Facility Horticulture Laboratory of Universities in Shandong of China (Grant number: 2018YY016) and Key Research and Development Program of Shandong, (Grant number: 2019GNC106034).

REFERENCES

- [1] D. Goodin, “Record-Breaking DDoS Reportedly Delivered by > 145K Hacked Cameras”, *Ars Technica*, Sept. 2016.
- [2] C. Cimpanu, “You Can Now Rent a Mirai Botnet of 400,000 Bots”, *BleepingComputer.com*, Nov. 2016.
- [3] C. Kolias, G. Kambourakis, et al., “DDoS in the IoT: Mirai and Other Botnets”, *Computer*, vol. 50, no. 7, pp. 80-84, Jul. 2017.
- [4] A. Mehmood, M. Mukherjee, et al., “NBC-MAIDS: Naïve Bayesian Classification Technique in Multi-agent System-enriched IDS for securing IoT against DDoS attacks”, *J Supercomput*, May. 2018, DOI: 10.1007/s11227-018-2413-7
- [5] E. Hodo, X. Bellekens, et al., “Threat Analysis of IoT Networks Using Artificial Neural Network Intrusion Detection System”, presented at 2016 Int. Symp. on Net., Comp. & Comm., Yasmine Hammamet, Tunisia, May 11-13. 2016.
- [6] C. Zhang, R. Green, et al., “Communication Security in Internet of Thing: Preventive Measure and Avoid DDoS Attack over IoT Network”, presented at the 18th Symp. on Comm. & Net., Alexandria, Virginia, Apr. 12-15, 2015.
- [7] J. W. Su, V. D. Vasconcellos, et al., “Lightweight Classification of IoT Malware Based on Image Recognition”, presented at 2018 IEEE 42nd Annu. Comp. Soft. & App. Conf., Tokyo, Japan, Jul. 2018.
- [8] G. Kaur, P. Agrawal, “Detection of LDDoS Attacks Using Variant of CUSUM and Shiryayev-Roberts’s Algorithm”, presented at the 4th Int. Conf. on Para., Dist. & Grid Comp., Wanknaghat, India, Dec. 22-24, 2016.
- [9] Z. Wu, L. Zhang, et al., “Low-rate DoS Attacks Detection based on Network Multifractal”, *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 5, pp. 559-567, Sept. 2016.
- [10] Z. G. Liu, X. C. Yin, et al., “A New Network Flow Grouping Method for Preventing Periodic Shrew DDoS Attacks in Cloud Computing”, presented at the 18th Int. Conf. on Advanced Comm. Tech., Pyeongchang, Korea, Jan. 31-Feb. 3, 2016.

- [11] M. Yue, L. Liu, et al., "Identifying LDoS Attack Traffic based on Wavelet Energy Spectrum and Combined Neural Network", *Int J Commun Syst.*, Sept. 2017.
- [12] Z. J. Wu, J. Jiang, et al., "A Particle Filter-Based Approach for Effectively Detecting Low-rate Denial of Service Attacks", presented at the *Int. Conf. on Cyber-Enabled Dist. Comp. & Know. Disc.*, Chengdu, China, Oct. 13-15, 2016.
- [13] P. Nagar, H. K. Menaria, M. Tiwari, "Novel Approach of Intrusion Detection Classification DeepLearning Using SVM", presented at the *1st Int. Conf. on Sust. Tech. for Comp. Intel.*, Jaipur, India, Mar. 20-30, 2019.
- [14] X. Peng, W. Q. Huang, Z. X. Shi, "Adversarial Attack Against DoS Intrusion Detection: An Improved Boundary-Based Method", presented at *2019 IEEE 31st Int. Conf. on Tools with Art. Intel.*, Portland, USA, Nov. 4-6, 2019.
- [15] M. Begli, F. Derakhshan, H. Karimipour, "A Layered Intrusion Detection System for Critical Infrastructure Using Machine Learning", presented at *2019 IEEE 7th Int. Conf. on SEGE*, Oshawa, Canada, Aug. 12-14 2019.
- [16] P. Rivas, C. DeCusatis, M. Oakley, A. Antaki, N. Blaskey, S. LaFalce, S. Stone, "Machine Learning for DDoS Attack Classification Using Hive Plots", presented at *2019 IEEE 10th Annual UEMCON*, New York City, USA, Oct. 10-12, 2019.
- [17] T. A. Tuan, H. V. Long, L. H. Son, R. Kumar, I. Priyadarshini, N. T. K. Son, "Performance Evaluation of Botnet DDoS Attack Detection Using Machine Learning", *Evolutionary Intelligence*, pp. 1-12, Nov. 2019.
- [18] M. S. Kang, S. B. Lee, V. D. Gligor, "The Crossfire Attack", presented at *2013 IEEE Sym. on Secu. and Pri.*, Berkeley, USA, May. 19-22, 2013.
- [19] H. S. Li, J. H. Zhu, "The New Threat to Internet: DNP Attack with the Attacking Flows Strategizing Technology", *Int. J. Commun. Syst.*, vol. 28, no. 6, pp.1126C1139, 2014.
- [20] H. S. Li, J. H. Zhu, "LAAEM: A Method to Enhance LDoS Attack", *IEEE Comm. Letters*, vol. 20, no. 4, pp. 708-711, Apr. 2016.
- [21] Y. Yang, Z. Z. Li, et al., "An Adaptive Semi-supervised Clustering Approach via Multiple Density-based Information", *Neuro Computing*, vol. 257, pp. 193-205, Sept. 2017.
- [22] X. Z. Gao, Q. S. Sun, et al., "Multiple Instance Learning via Semi-supervised Laplacian TSVM", *Neural Proc. Letters*, vol. 46, no. 2, pp. 219-232, Aug. 2017.
- [23] H. L. Du, Y. Zhang, "Transductive Support Vector Machine Algorithm Based on Cluster and Cooperative Labeling", *HeNan Sci.*, vol. 35, no. 1, Jan. 2017.
- [24] N. Piroonsup, S. Sinthupinyo, "Analysis of Training Data Using Clustering to Improve Semi-supervised Self-training", *Knowledge-Based Systems*, vol. 143, pp. 65-80, Mar. 2018.
- [25] K. J. Zhang, M. Xian, "Research on Hybrid Intrusion Detection Model Based on DBN and TSVM", *Comp. App. and Software*, vol. 35, no. 5, May. 2018.
- [26] A. Andoni, P. Indyk, et al., "Beyond Locality-Sensitive Hashing", presented at *25th Ann. ACM-SIAM symp. On dis. Alg.*, New York, 2014.
- [27] G. Aluç, M. T. Özsu, et al., "Building self-clustering RDF databases using Tunable-LSH", *The VLDB Journal*, <https://doi.org/10.1007/s00778-018-0530-9>, Dec. 2018.
- [28] F. Wu, X. Y. Jing, Q. H. Huang, "Uncorrelated Locality-Sensitive Multi-view Discriminant Analysis", *Natl. Acad. Sci. Lett.* (2020), doi. 10.1007/s40009-019-00864-4, Jan. 2020.



Xiaochun Yin (M'12) was born in Shouguang, China in 1980. She received the B.S. degree in Education and Technology from Qufu Normal University in 2004, and received the M.S. degree in Education and Technology from Nanjing Normal University in 2007, and received the Ph.D. from Dongseo University in 2015.

From 2007 to now, she is an associate professor in Weifang University of Science & Technology. Her research interests include network security, IoT security, blockchain and AI areas.

Professor Yin was a recipient of the Science Research Innovation Award of Weifang University of Science & Technology in 2017 and 2018.



Zengguang Liu was born in Shouguang, China in 1982. He received the B.S. and M.S. degrees in computer science from the University of Shanghai for Science and Technology in 2005 and 2008 respectively. He is currently pursuing the Ph.D. degree in computer science at the University of Shandong for Science and Technology.

From 2008 to 2016, he was a Senior Architecture Engineer in Core Network Department of Alcatel-Lucent. He had been a Principal AI Engineer of Hewlett Packard Enterprise since 2016. His research interests include big data, AI and network security areas.

He was a recipient of the Innovation Award of Shandong for Excellence in 2011.



Deyong Liu received the B.S. degree in computer science and technology from Shandong Normal University in 1996 and the M.S. degree in computer technology from Ocean University of China in 2010.

He is currently a Professor with the Weifang University of Science & Technology. He is focusing on Cloud Computing, IoT, and Smart city big data research. At present, he is the tutor of Shandong Normal University, the tutor of the Torch High-tech Industry Development Center of the Ministry of Science and Technology, the director of Shandong Software Industry Association.



Zhengze Liu received the B.S. degree in QingDao University in 2020, majoring in accounting. He is now studying in Peking University for software engineering.

He is now engaged in research cloud computing, IoT, and machine learning algorithm for paired financial trading.