Concept Paper Submission: ICCRTS Conference on Multi-Domain C2 for Topic 7, Human Information Interaction

Using Semi-Supervised Learning for Flow-Based Network Intrusion Detection

Nandi O. Leslie

U.S. Army Research Laboratory, Adelphi Laboratory Center

2800 Powder Mill Rd

Adelphi, MD 20783, USA

Cell: (202)528-0770

Email: Nandi.O.Leslie.Ctr@mail.mil

Abstract

In a distributed and semi-autonomous environment, network breaches must be detected prior to reaching the highly-valued targets or networked devices—this requires proactive adversarial modeling that is behavior or anomaly-based and capable of operating in a high-speed network environment. Using an intelligent agent architecture and machine learning, I propose a network intrusion detection system (NIDS) that is flow-based to produce alerts on malicious and/or anomalous traffic. With this proposed semi-supervised learning approach, I detect botnet traffic and distinguish it from the normal and background network traffic in the network session or flow datasets (i.e., NetFlow files). I evaluate the prediction performance and computational resource utilization results for the flow-based NIDS algorithms and compare these results with signature-based NIDS that are reactive by design. With this approach, I show an improvement in detection accuracy and NIDS efficiency when compared with traditional signature-based NIDS and other probabilistic modeling approaches examined on these network traffic datasets. In addition, the model improvements reduce the burden on the human analysts to sift through NIDS alerts that are often riddled with false alarms.

Keywords: Network security, intrusion detection, machine learning

1 Introduction

Since the late 1980s, researchers have studied and developed intrusion detection systems (IDS) using anomaly and behavior-based detection models that leverage techniques from machine learning and more broadly statistics (Denning, 1987; Smaha, 1988; Vaccaro & Liepins, 1989; Wang & Stolfo, 2004; Tsai et al., 2009; Harang & Mell, 2016; Leslie et al., 2018). Although there have been many successes in detecting network breaches in data confidentiality, integrity, and availability, these network IDS (NIDS) are still plagued with high false-positive alerts — these notifications misclassify benign network activities as malicious which also has adverse impacts on the personnel who must generate security incident reports: they may disregard the IDS tools generating above some tolerable threshold of false-positive alerts which, in turn, can result in true-positive alerts going unreported (Denning, 1987; Tsai et al., 2009; Leslie et al., 2017; Shearer et al., 2018).

IDSs are generally categorized in terms of whether they use signature or behavior-based detection models (e.g., anomaly detection models), and their location on a host or network (Wang & Stolfo, 2004; Hu et al., 2009). Organizations face a variety of information security vulnerabilities and threats, and though signature-based IDS are needed to identify violations of security policies, they are insufficient for guarding against previously unidentified vulnerabilities or new exploits that are also called "zero-day attacks." Whereas, anomaly or novelty-based detection algorithms for IDS are capable of detecting network breaches where no signature exists (Yeung & Chow, 2002). However, these models can also be subverted.

Semi-supervised or unsupervised learning algorithms for anomaly-based network IDS (NIDS) provide greatly needed solutions for network and information security in the face of previously

unidentified vulnerabilities. They provide the security analyst with required tools to further investigate and protect an organization's networks and its proprietary data. To address these research challenges, there are two main contributions in this paper to intrusion detection research.

1. I develop a novel anomaly-based NIDS algorithm called Semi-Supervised Learning of Exploits and Exploit Kits (SSLEEK) to detect network intrusions with various characteristics, including performing distributed denial of service (DDoS), and port scanning. SSLEEK's pre-processing module is based on a distance metric for IPv4 addresses — these are categorical features requiring conversion to real-valued features for implementation in the machine learning algorithms.
2. Using K-means clustering, Gaussian Mixture Model (GMM), and the k-nearest neighbors (k-NN) algorithms, I examine and compare SSLEEK's prediction performance results. These modeling techniques can be applied to both host-based systems and networks of various types, ranging from tactical mobile ad hoc networks (MANETs) to commercial enterprise networks.

The remainder of this paper is divided in the following sections: in Section 2, related work is presented; in Section 3, the datasets for training and testing the machine learning algorithms for the NIDS are presented; in Section 4, the modeling approaches are further described; and in Section 5, the results are presented; and in Section 6, the concluding remarks are presented.

2 Related Work

Over the past ten years, various machine learning algorithms have been used to enhance NIDS detection accuracy, and limited comparison studies and surveys exist to help determine which

algorithms are best suited for NIDS given the networks' vulnerabilities (Gu et al., 2007; Tsai et al., 2009; Garcia et al., 2014). One of several taxonomies for IDS that leverage machine learning techniques is defined by the following: single classifiers or clustering methods characterized by using one machine learning algorithm (e.g., k-NN, K-means, support vector machines); hybrid algorithms consist of two functional components, where the first takes raw input data and performs some intermediate processing, and the second component takes as input the intermediate step's results and using some computational algorithm yields the final results; and ensemble algorithms are based on using boosting or bagging to vote on the final predictions of multiple machine learning algorithms (Tsai et al., 2009).

For example, BotHunter is a hybrid algorithm based on the Snort IDS — this is a commercially available signature-based detection model — combined with two proprietary statistical methods to detect stages of the malware process or spread over the network: Statistical Scan Anomaly Detection Engine (SCADE), and Statistical Payload Anomaly Detection Engine (SLADE) that uses n-grams to assess the packet payload (Gu et al., 2007). Additional examples of hybrid-based intrusion detection algorithms include BClus and CAMNEP which are both behavior-based models: BClus uses the Expectation-Maximization (EM) algorithm to cluster NetFlow data; and CAMNEP is a collaborative anomaly detection model (Gu et al., 2007; Garcia et al., 2014). The modeling approach for SSLEEK defined in this paper differs from those approaches in that SSLEEK is a behavior-based algorithm that analyzes network sessions processed from packet headers (i.e., NetFlow files) as the raw input data, and it uses a single classifier or clustering approach.

Wang and Paschalidis (2017) develop an additional hybrid model for botnet detection that leverages the K-means algorithm — this two-stage approach uses an IPv4 distance metric on NetFlow data for the feature extraction step of implementing the K-means algorithm: (1) the first stage is an anomaly detection algorithm; and (2) the second stage detects the botnets using concepts from social network community detection that are based on graph-based approaches that approximate the graph degree distribution to capture node correlations over time. Here, in this paper, SSLEEK differs from this approach and others in that although I also introduce a novel IPv4 distance metric, I apply a single algorithm approach which uses semi-supervised and supervised learning algorithms — these are k-NN, K-means clustering, and GMM approaches — for botnet detection. I compare the performance of these machine learning algorithms on the CTU-13 botnet datasets, where the IPv4 address distance metric is either considered, or the IP addresses are excluded from the feature space.

There is limited previous research on using these machine learning algorithms for detecting malicious traffic in network flow or sessions data (Eskin et al., 2002; Bouzida et al., 2004; Tsai et al., 2009; Om & Kundu, 2012; Celik et al., 2015). Nonetheless, it is difficult to compare these anomaly detection models for IDS because a valuable comparison requires that similar datasets and network configurations be used for validating each method. The modeling approach in this paper differs from other studies: the feature space consists of network session data derived from NetFlow files; the feature selection and extraction methods include converting categorical features (e.g., flags, protocols, IP addresses) to real-valued features with either a one-hot encoding method or a novel IP address distance metric. The network sessions or flow data consists of packet header information exclusively.

3 Network Traffic Datasets

The Czech Technical University (CTU)-13 botnet datasets consist of packet capture (pcap) and NetFlow files that define thirteen diverse scenarios with distinct botnet characteristics summarized in Table 1 (Garcia, 2013). Garcia (2013) creates NetFlow files from the pcap files for each CTU-13 scenario labelled based on ground truth as "normal" for network traffic to and from a benign node, "botnet" for network traffic to and from a bot node, and "background" otherwise.

In Table 1, the number of flows in each category for the scenarios are presented (Garcia et al., 2014). The CTU-13 NetFlow data consist of categorical and real-valued features for the network flows or sessions: the real-valued features are duration (in seconds), source and destination ports, state, total packets sent, total bytes, and source bytes; and the categorical features are protocol (e.g., ICMP, TCP), and source and destination IP addresses.

Table 1. Details on flow records from CTU-13 (Garcia et al., 2014).

ID	Characteristic	Total flows	Botnet flows	Normal flows	Background flows
1	IRC, SPAM, Click Fraud (CF)	2,824,636	39,933	30,387	2,754,316
2	IRC, SPAM, CF, FTP	1,808,122	18,839	9,120	1,780,163
3	IRC, Port Scan (PS), US	4,710,638	26,759	116,887	4,566,992
4	IRC, DDOS, US	1,121,076	1,719	25,268	1,094,089
5	SPAM, PS, HTTP	129,832	695	4,679	124,458
6	PS	585,919	4,431	7,494	573,994
7	HTTP	144,077	37	1,677	142,363
8	PS	2,954,230	5,052	72,822	2,876,356
9	IRC, SPAM, PS, CF	2,753,884	17,880	43,340	2,692,664
10	IRC, DDOS, US	1,309,791	106,315	15,847	1,187,629
11	IRC, DDOS, US	107,251	8,161	2,718	96,372
12	PP	325,471	2,143	7,628	315,700
13	SPAM, PS, HTTP	1,925,149	38,791	31,939	1,854,419

4 A Flow-Based Model for NIDS: SSLEEK

To pre-process the feature space derived from the CTU-13 NetFlow data, one of the key steps is converting the categorical features to real-valued features. For the categorical features selected in the NetFlow data, I use one-hot encoding, except for the IPv4 addresses. Since there are numerous new IP addresses interacting with any given network on a regular basis, one-hot encoding for IP addresses in the network sessions would result in a high-dimensional feature space growing substantially with the ever-increasing number of new IP addresses interacting with a network. I implement a novel IPv4 address distance metric in Equation (1) that computes the distance between any pair of IPv4 addresses in the feature space. The distance between any two IPv4 addresses, $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ and $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$ is given by

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{4} a^{4-i}(x_i \neq y_i) \tag{1}$$

where $a > 1$ is a constant, and $x_k,\ y_k \in \{1, 2, \dots, 255\}$ for $k = 1, \dots, 4$.

To improve the efficiency of SSLEEK, I implement Principal Component Analysis (PCA) to reduce the dimensionality of the feature space while maintaining 99% of the variability in the data. Subsequently, I apply each the following three machine learning algorithms to detecting the various types of botnet traffic in the thirteen scenarios of the CTU-13. First, I use K-means clustering for each scenario — this is an unsupervised learning algorithm that clusters unlabeled data — where I set $K = 2$ representing two centroids which are initialized by the means of the malicious and benign training datasets, respectively. Second, I implement the GMM, a soft clustering algorithm, that is estimated by the Expectation-Maximization (EM) algorithm, where again the means for the GMM are defined as they are in the K-means centroid-initialization

phase. The GMM differs from the K-means clustering algorithm in that the EM step makes probabilistic cluster assignments for network traffic rather than deterministic as with the K-means algorithm (Hastie et al., 2008). Both the K-means algorithm and GMM are applied to the CTU-13 botnet datasets with semi-supervised learning techniques, where the labeled data is used only in the centroid initialization phase of training, as opposed to defining unsupervised learning implementations for these algorithms. Finally, I also apply the k-NN classification — this is a supervised learning method — to botnet detection for these network flow datasets. Although these machine learning algorithms have been used widely in the literature, there are limited studies that use these algorithms to detect botnet traffic (Celik et al., 2015; Wang & Paschalidis, 2017; Leslie et al., 2018). I implement the k-NN, K-means, and GMM algorithms with semi-supervised techniques in Python using the scikit-learn module (Buitinck et al., 2013).

To cross validate the NetFlow data of each CTU-13 scenario, I divide it into training and testing sets with K-fold cross validation, where I set $K = 5$. This cross-validation technique has been used widely in classical statistics and machine learning in a variety of domains (Kohavi, 1995). In addition, I normalize the training set from each scenario, and using the mean and standard deviation from the training data, I subsequently normalize the testing set for the same scenario.

5 Empirical Results

The k-NN algorithm is a simple, nonparametric technique to classify samples (Bishop, 1995; Manocha & Girolami, 2007). I examined different choices of *k*, the number of nearest neighbors, where $k = 2, \dots, 27$, and various L^p-norms for $p = 1, \dots, 9$. The prediction performance of k-NN varies drastically with different choices of *k* and *p*. For the *k*-NN prediction performance results, I set *k = 7* and *p = 3* and present the accuracy, precision, recall,

Table 2. The prediction performance results for SSLEEK using a semi-supervised learning implementation of the k-NN, K-means, and GMM algorithms. Best results are highlighted in bold font.

Scenario ID	ML algorithm	Accuracy	Precision	Recall	FPR%
1	**k-NN**	**0.992**	**0.8000**	**0.616**	**0.2270%**
	K-means	0.763	0.0320	0.518	23.3324%
	GMM	0.238	0.0092	0.482	76.5354%
2	**k-NN**	**0.996**	**0.9430**	**0.725**	**0.0570%**
	K-means	0.873	0.0820	0.980	12.8423%
	GMM	0.129	0.0003	0.020	86.9256%
3	**k-NN**	**0.998**	**0.8830**	**0.766**	**0.0580%**
	K-means	0.926	0.0000	0.000	5.5330%
4	k-NN	0.997	0.5100	0.006	0.0400%
	K-means	**0.998**	**0.9820**	**0.085**	**0.0004%**
	GMM	0.731	0.0002	0.025	26.7090%
5	**k-NN**	**0.990**	**0.6790**	**0.315**	**0.1770%**
	K-means	**0.721**	**0.0220**	**0.903**	**28.0142%**
	GMM	0.716	0.0217	0.903	28.5479%
6	**k-NN**	**0.999**	**0.9710**	**0.982**	**0.0480%**
	K-means	0.748	0.0320	0.991	25.4173%
	GMM	0.267	0.0001	0.009	73.0627%
7	**k-NN**	**0.999**	**0.5000**	**0.077**	**0.0080%**
	K-means	0.774	0.0010	0.375	22.5533%
	GMM	0.290	0.0004	0.563	70.9724%
8	**k-NN**	**0.998**	**0.5310**	**0.745**	**0.1370%**
	K-means	0.778	0.0080	0.817	22.1980%
	GMM	0.008	0.0021	1.000	99.4537%
9	**k-NN**	**0.965**	**0.8250**	**0.770**	**1.5880%**
	K-means	0.763	0.1880	0.502	21.1191%
	GMM	0.242	0.0583	0.498	78.2548%
10	**k-NN**	**0.999**	**0.9995**	**0.993**	**0.0040%**
	K-means	0.969	0.8050	0.817	1.7534%
	GMM	0.050	0.0064	0.069	95.1203%
11	**k-NN**	**0.999**	**0.9990**	**0.987**	**0.0100%**
	K-means	0.969	0.7890	0.818	1.8052%
	GMM	0.929	0.5169	0.984	7.5842%
12	**k-NN**	**0.994**	**0.5770**	**0.182**	**0.0900%**
	K-means	0.941	0.0290	0.244	5.4221%
	GMM	0.842	0.0272	0.653	15.6747%
13	**k-NN**	**0.990**	**0.8350**	**0.659**	**0.2760%**
	K-means	0.782	0.0500	0.525	21.2310%
	GMM	0.773	0.0478	0.525	22.2086%

and false positive rate (FPR). Surprisingly, although k-NN is a simple learning algorithm, it provides quite accurate (i.e., good precision and recall values) results with relatively low FPR for an anomaly-based model for a NIDS. In Table 2, the SSLEEK results with the k-NN, K-means,

and GMM algorithms are presented for the CTU-13 botnet datasets for comparison, where the IPv4 addresses were excluded from the NetFlow feature extraction.

Also, in Table 2 are the results of SSLEEK using the GMM algorithm. This implementation suffered the worst performance in comparison with the k-NN and K-means implementations in SSLEEK: highest FPR and lowest precision and recall values. For scenario ID 3, I did not test the GMM prediction performance on this dataset. However, since K-means consistently outperforms the GMM implementation in SSLEEK for the other twelve scenarios in CTU-13, I present K-means predictions along with those of the k-NN classifier which performs best in this case.

The prediction performance of SSLEEK with the k-NN algorithm far exceeds its effectiveness using the clustering algorithms, the K-means and the GMM for detecting these cyberattacks for all CTU-13 scenarios except two: IDs 4 and 5 (see Table 2). For the DDoS scenario ID 4, although K-means is the best-performing algorithm, its prediction performance is not ideal: the precision and recall values for K-means are 0.9820 and 0.085, respectively; the k-NN implementation resulted in precision and recall equal to 0.5100 and 0.006, respectively; and the GMM had the worst predictions for this scenario with precision equal to 0.0002, and recall equal to 0.006. For scenario ID 5, the GMM implementation in SSLEEK performs the worst, where precision is 0.0217, and recall is 0.903. The K-means predictions are slightly better with precision at 0.0220 and recall also equal to 0.903. Whereas, the k-NN algorithm has the following results: precision is 0.6790, and recall 0.315. It depends on the objectives of the Cyber Security Service Provider (CSSP) to determine whether the k-NN implementation is better than

that of K-means clustering in this scenario. That is, if the CSSP values a high recall value over precision, then K-means is the best-performing algorithm. However, if the CSSP values precision over recall, then the k-NN classifier is the best model for detecting botnet traffic.

In Table 3, I present predictions for selected scenarios, where the SSLEEK feature space includes the IPv4 addresses converted to real-valued features with the novel IP distance metric in Eq. (1). Although the prediction performance results for scenario IDs 10 and 11 in Table 3 are accurate, the results for other scenarios were poor and not presented.

The SSLEEK predictions for the CTU-13 scenarios are best, when the IP addresses are excluded from the feature space (see Table 2). In addition, the prediction performance of SSLEEK is quite impressive for many of the CTU-13 scenarios, including: (1) for ID 6, precision is 0.9710, and recall is 0.982; (2) for ID 10, precision is 0.9995, and recall is 0.993; and (3) for ID 11, precision is 0.9990, and recall is 0.987.

6 Concluding Remarks

In this paper, I develop SSLEEK, a novel modeling approach for anomaly-based NIDS that leverages machine learning algorithms to detect botnet traffic. The raw data input for SSLEEK

Table 3. The k-NN algorithm implementation in SSLEEK with the IPv4 distance metric for selected CTU-13 botnet scenarios.

Scenario ID	Precision	Recall	FPR%
3	0.7778	0.0013	0.0000%
10	0.9999	0.9886	0.0012%
11	0.9988	0.9786	0.0101%
12	0.0667	0.0023	0.0217%

are NetFlow files that include categorical features converted to real-valued features in a pre-processing step. I either convert the source and destination IP addresses from the NetFlow data to real-valued features with the novel IPv4 address distance metric introduced in Section 4 with results presented in Table 2 or exclude the IP addresses from the feature space with results presented in Table 3. For the other categorical features in the CTU-13 NetFlow files, I implement one-hot encoding for the conversion to real-valued features. This feature extraction and modification step is followed by PCA to reduce the dimensionality of the feature space. Subsequently, I apply three semi-supervised learning algorithms to SSLEEK for the network flow-based intrusion detection: k-NN classifier, K-means clustering, and GMM.

The advantages of SSLEEK include that it is effective at accurately detecting botnet traffic in the CTU-13 scenarios, especially with the k-NN algorithm. Moreover, the k-NN classifier performs best, if the IP addresses are excluded from the feature selection phase (see Tables 2 and 3). For scenario ID 4, one of the DDoS scenarios, SSLEEK has the best prediction performance with the K-means algorithm when excluding IP addresses from the feature space (see Table 2). For scenario ID 5, selecting k-NN or K-means as the best-performing algorithm depends on the CSSP's objectives for network security in the precision-recall tradeoff (see Table 2). That is, which NIDS classification error is most detrimental to security: false-positive alerts; or false negatives, where there are no NIDS alerts on malicious traffic.

Although the k-NN implementation in SSLEEK performs better than with the K-means and GMM, the k-NN algorithm has its limitations — labeled data is required. For the K-means and GMM algorithms, unlabeled or quite sparsely labeled network flow datasets are adequate. This is a strong advantage of these two clustering algorithms over the k-NN classifier for SSLEEK.

In the event that only a sparsely labelled dataset is available, the K-means clustering is the best-performing and best-suited algorithm for SSLEEK with the CTU-13 botnet scenarios.

References

1. Bouzida, Y., Cuppens, F., Cuppens-Boulahia, N., & Gombault, S. (2004, June). Efficient intrusion detection using principal component analysis. In *3éme Conférence sur la Sécurité et Architectures Réseaux (SAR), La Londe, France* (pp. 381-395).
2. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Layton, R. (2013). API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.
3. Celik, Z. B., Walls, R. J., McDaniel, P., & Swami, A. (2015, October). Malware traffic detection using tamper resistant features. In *Military Communications Conference, MILCOM 2015-2015 IEEE* (pp. 330-335). IEEE.
4. Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).
5. Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*, (2), 222-232.
6. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security* (pp. 77-101). Springer, Boston, MA.
7. García S. Malware capture facility project https://mcfp.felk.cvut. cz/; 2013 [accessed 06.03.14].
8. García, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, *45*, 100-123.
9. Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007, August). BotHunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium* (Vol. 7, pp. 1-16).

10. Harang, R., & Mell, P. (2016, October). Micro-signatures: The Effectiveness of Known Bad N-Grams for Network Anomaly Detection. In *International Symposium on Foundations and Practice of Security* (pp. 36-47). Springer, Cham.
11. Hu, J., Yu, X., Qiu, D., & Chen, H. H. (2009). A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection. *IEEE network*, *23*(1), 42-47.
12. Kott, A., Swami, A., & West, B. J. (2016). The internet of battle things. *Computer*, *49*(12), 70-75.
13. Leslie, N. O., Marvel, L. M., Edwards, J., Comroe, K., Shearer, G., & Knachel, L. (2017, May). Modeling approaches for intrusion detection and prevention system return on investment. In *Cyber Sensing 2017* (Vol. 10185, p. 1018502). International Society for Optics and Photonics.
14. Leslie, N. O., Martone, A., & Weisman, M. (2018, January). The Internet of Things (IoT): Computational Modeling in Congested and Contested Environments. In *Proceedings of the NATO STO IST-152 Task Group Workshop on Intelligent Autonomous Agents for Cyber Defence and Resilience, 18-20 October 2017, Prague, Czech Republic*. NATO.
15. Manocha, S., & Girolami, M. A. (2007). An empirical analysis of the probabilistic K-nearest neighbour classifier. *Pattern Recognition Letters*, *28*(13), 1818-1824.
16. Om, H., & Kundu, A. (2012, March). A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. In *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on* (pp. 131-136). IEEE.
17. Shearer, G., Leslie, N.O., Ritchey, P., Braun, T., & Nelson, F. (2017). IDS Alert Prioritization through Supervised Learning. In Proceedings of the NATO Specialists'

Meeting on Predictive Analytics and Analysis in the Cyber Domain, 10-11 October 2017, Sibiu, Romania: NATO.

18. Smaha, S. E. (1988, December). Haystack: An intrusion detection system. In *Aerospace Computer Security Applications Conference, 1988., Fourth* (pp. 37-44). IEEE.
19. Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994-12000.
20. Vaccaro, H. S., & Liepins, G. E. (1989, May). Detection of anomalous computer session activity. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on* (pp. 280-289). IEEE.
21. Wang, K., & Stolfo, S. J. (2004, September). Anomalous payload-based network intrusion detection. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 203-222). Springer, Berlin, Heidelberg.
22. Wang, J., & Paschalidis, I. C. (2017). Botnet detection based on anomaly and community detection. *IEEE Transactions on Control of Network Systems*, *4*(2), 392-404.
23. Yeung, D. Y., & Chow, C. (2002). Parzen-window network intrusion detectors. In *Object recognition supported by user interaction for service robots* (Vol. 4, pp. 385-388). IEEE.