Towards Continuous Enrichment of Cyber Threat Intelligence: A Study on a Honeypot Dataset

Arnolnt Spyros, Angelos Papoutsis, Ilias Koritsas, Notis Mengidis, Christos Iliou, Dimitris Kavallieros, Theodora Tsikrika, Stefanos Vrochidis, and Ioannis Kompatsiaris
Information Technologies Institute, *CERTH*, Thessaloniki, Greece
{aspyros,apapoutsis,ilias.koritsas,nmengidis,iliouchristos,dim.kavallieros,theodora.tsikrika,stefanos,ikom}@iti.gr

***Abstract*—Cyber Threat Intelligence helps organisations make the right decisions in their fight against cyber threats and strategically design their defences by continuously providing information regarding the cyber threat landscape. In this context, honeypots are a widespread solution for gathering intelligence about threat actors. However, honeypots do not inherently provide information about the origin of threat groups, their resources, capabilities and their impact. Thus, we propose an approach that classifies threats, as highly or less abusive, based on their behaviour characteristics using four ensemble machine learning algorithms applied on security incidents identified in a rule-based manner on a deployed honeypot. After prepossessing and hyper-tuning of the parameters, the four models, Adaptive Boosting Classifier (AdaBoost), Random Forest Classifier (RFC), Light Gradient Boosting Machine (LGBM) and Extreme Gradient Boosting (XGBoost), achieve good results, with RFC and LGBM achieving the best recall (84%, 83%) and LGBM and XGB the best AUC (91%, 90%).**

***Index Terms*—cyber threat intelligence, machine learning, honeypots, ensemble methods, Wazuh**

I. Introduction

Due to the continuously increasing cyber-attacks in terms of both volume and sophistication, organisations and critical infrastructures are susceptible to a wide range of diverse threats which can severely damage their day-to-day operations. Moreover, such cybersecurity incidents could also have, depending on their impact, legal liabilities for the affected organisations (e.g., GDPR sanctions). Therefore, organisations need to implement a variety of security mechanisms to enhance the cybersecurity of their infrastructures; to this end, gaining intelligence about the threat landscape, and specifically about the severity and nature of the attacks, is of utmost importance to an organisation. In this context of developing effective defence mechanisms, the cybersecurity community is increasingly sharing Cyber Threat Intelligence (CTI) in order to assist organisations be better protected against modern threats. While CTI contains valuable information, additional actionable information could also be added to further increase the value and usefulness of the available CTI. An important example of such information is the potential level of abusiveness of the threat, as the knowledge of such information could facilitate the proper allocation of resources, as well as improve the overall Incident Response (IR) procedure.

To this end, this work proposes an approach which enables the characterisation of the origins of a cyber-attack (i.e., the IP addresses), based on their abusiveness. In particular, this work proposes to classify the attacks as "highly-abusive" or "less-abusive", by employing Machine Learning (ML) algorithms on the security incident logs generated by a deployed honeypot. In this way, apart from the typical CTI information, such as the Indicators of Compromise (IoCs), that is extracted from security incidents in an organisation, we propose to further enrich such actionable CTI with information regarding the severity of the incident that has occurred. Therefore, the added value is that, apart from leveraging the initial CTI information, organisations would be able to further utilise and act upon the additional information, in order to facilitate the adoption and implementation of appropriate security measures (e.g. threat modelling, allocation of the security resources, etc.) and thus properly prioritise the containment of a threat based on its level of abusiveness.

Specifically, the proposed method first identifies security incidents in a rule-based manner and then classifies them by using ML algorithms and in particular ensemble ML methods that typically exhibit less bias and less variance; to this end, the Adaptive Boosting Classifier (AdaBoost), Random Forest Classifier (RFC), the Light Gradient Boosting Machine (LGBM) and Extreme Gradient Boosting (XGBoost) algorithms are considered. The experiments performed on a dataset gathered from the deployed honeypot achieve promising results, indicating that the studied ensemble ML models are highly effective concerning the classification of the severity of the incidents, based on the level of abusiveness, even when trained with a small amount of data.

Moreover, further to the aforementioned novelty of our work and contrary to most existing work that utilises only publicly available static datasets [1] that pose the risk of not being relevant to the continuously evolving threat landscape, this work utilises a new dataset gathered during the summer of 2021 based on a honeypot solution deployed on the cloud in order to gather data related to cyber-attacks. The data is constantly updated with new information from modern attacks that are subsequently analysed to extract features that are provided as input to the ML algorithms. The dynamic nature of the data has the potential to enable the improvement of the models as more data become available for their training.

II. Related Work

Honeypots are decoy systems that are used to attract attackers by exposing vulnerable services. They are considered a

valuable source for gathering information regarding the tactics and techniques, as well as the attack patterns used against the exposed services, thus providing useful and actionable CTI. Additionally, honeypots are utilised in conjunction with other security components (e.g., IDS and Security Information and Event Management (SIEM) systems) in order to improve and enhance their detection performance [2], [3]. For their deployment, the majority of approaches utilise docker containers; low and medium-interaction honeypots are preferred, with Dionaea[1] and Cowrie[2] being the most popular ones [2]–[5].

Various research efforts have tried to address the problem of CTI classification. For example, a framework for gathering CTI from Twitter posts has been developed [6]; this framework is trained based on the features extracted from CTI using threat information from public repositories, such as Common Vulnerabilities and Exposures (CVE), and classifies unseen tweets as normal or abnormal.

In [7], a framework was proposed for CTI extraction and categorization using social media data. Particularly, a Convolutional Neural Network (CNN) was used to identify CTIs' targeted domain. In addition, IoC extraction approaches were used to identify unseen types of IoCs. In the end, the generated IoC and its domain tag (identified with CNN) were used to create a categorised CTI with a specific domain.

In [8], a classification of CTI data originating from hacker forums was performed using two different variants of Recurrent Neural Networks (RNNs). The data were classified, using specialised web crawlers, into relevant or irrelevant. Two variants of RNN, namely Gated Recurrent Unit(GRU) and Long Short-Term Memory (LSTM), achieved a high accuracy.

In [9] an ontology-based threat assessment system was proposed that utilises System Windows logs to classify system processes, in real-time, as high-threat, medium-threat, low-threat or unknown, based on the identified characteristics. Automate responses to threat were utilised based on the indicators of compromise. As a result a continuously updated threat intelligence was generated.

Contrary to aforementioned research, we focus our analysis on extracting more information regarding the attack source. Specifically, the approach proposed in this work aims to expand the IoCs extracted from incident logs by combining them with ML methods, in order to provide additional intelligence about the level of abusiveness of the identified incident origin and enrich the extracted CTI. Moreover, this work utilises a newly collected dataset that contains actions of real attackers. This aspect is considered quite important in the modern cybersecurity environment, due to the dynamic nature of the cyber-attacks landscape which results in new attacks on a daily basis.

III. Methodology

This section describes the methodology of the approach proposed in this work; its high-level architecture is depicted in Figure 1. By considering attackers targeting a deployed honeypot to exploit vulnerabilities in the exposed services, an agent deployed on the honeypot monitors the various log paths and identifies any new entry that is added to these logs. These are then analysed in a rule-based manner to identify the security incidents and subsequently extract CTI-related data (e.g., IP addresses, ports). Once the CTI is extracted, the relevant information is analysed in order to extract features which are provided as input to the ML-based models in order to distinguish a security incident source as "highly-abusive" or "less-abusive", i.e., to distinguish whether an IP is the origin of a high-severity incident or not.

A. Data Collection and Information Extraction

The most appropriate honeypot solution for the purposes of this work was selected on the basis of the following criteria: (i) the services that each honeypot provides and how useful these services are, (ii) the maturity level of the honeypot solution, and (iii) the complexity of the deployment and configuration of the honeypot. Upon extensive research, we decided to deploy the Dionaea honeypot on a cloud Virtual Machine (VM) located on the Amazon Web Services (AWS) platform[3]. Dionaea is considered as a low-interaction honeypot that supports several common services, including, but not limited to, SMB, HTTP, and FTP.

Dionaea has been configured to store only security incidents (i.e., cyber attacks), and not extensive logs, in order to reduce noise. Dionaea has also been configured to store the logs in JSON format to facilitate further processing, while a cron job has been developed to compress and delete bitstreams (i.e., uploaded malware files).

The data collected from the honeypot are then stored in a central database (namely Elasticsearch) for the extraction of

[1]https://github.com/DinoTools/dionaea

[2]https://github.com/cowrie/cowrie

[3]https://aws.amazon.com/

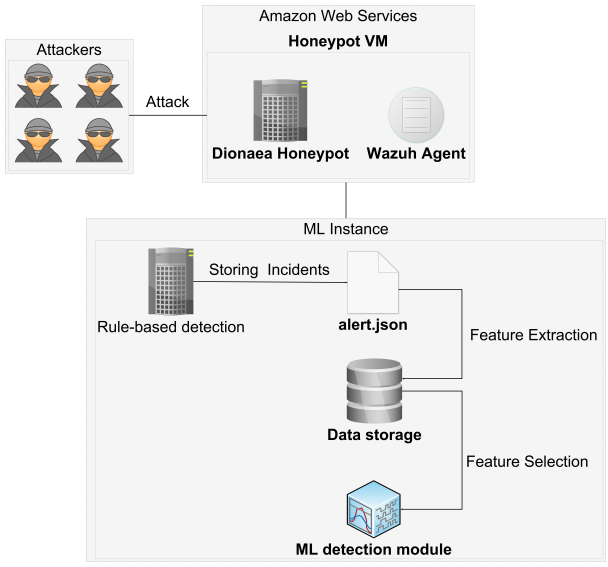

Fig. 1. High-level architecture

attacks by utilising the Wazuh[4] security platform. Wazuh was selected since it is a free, open source, and enterprise-ready security monitoring solution for threat detection, integrity monitoring, incident response, and compliance, which uses the Elasticsearch-Logstash-Kibana (ELK) stack as its back-end. Concerning threat detection, Wazuh implements a rule-based approach. In this work, a Wazuh agent is installed on the honeypot VM and periodically monitors the honeypot logs and forwards back to the Wazuh manager any new log entries found. Therefore, the Wazuh manager is able to analyse the gathered logs as soon as a security incident occurs in near real time, through an encrypted and authenticated channel based on Advanced Encryption Standard (AES).

To extract the information from the attacks, a signature-based approach that employs a set of rules via regular expressions is used. The extracted information includes a variety information such as IoCs (e.g., IPv4 address, source port, browser agent version) and the attack on the honeypot (e.g., destination IPv4, destination port, requested URL) which are included in the CTI that is produced in a later stage. While Wazuh currently contains more than 3000 predefined rules for various services[5], these rules do not cover all services exposed by Dionaea and, therefore, 17 additional custom rules were developed in order to expand the amount of the security incidents that can be identified on our Dionaea honeypot, such as connections to services (e.g., SQL, MySQL, SMB), malware uploads, and SQL injections. The Wazuh manager identifies any new attack found at the honeypot in real time and stores the incident log in a *alerts.json* file that is then parsed to extract CTI. The extracted CTI is subsequently used as input by our ML models.

B. Feature Extraction and Selection

Feature extraction aims to extract features considered to be useful in the context of intelligence enrichment by analysing the logs generated from Dionaea. First, the total collected requests are grouped into sessions by following a similar approach used in the web bot detection problem [10]. Specifically, the logs are initially split per IP; when an IP stays idle (i.e., performs no request) for a period greater than 30 minutes, a new session is created upon a new request. Then, the closed session is analysed so as to extract IoCs and compose the CTI.

The employed features (see Table I for the full list) were selected after (i) enumerating all protocols and request types by Dionaea, (ii) examining a wide range of relevant works [11]–[13] to assess the features extracted when similar protocols are available in use cases relevant to the approach proposed in this work, and (iii) considering features deemed useful towards effective ML-based incident severity classification in terms of reducing computational costs, eliminating data redundancy, and reducing false alarms. Concerning the case of protocols for which we did not find relevant work in the literature, we defined new features (e.g., Port_con_count) based on the functionalities supported in the deployed honeypot for those protocols.

TABLE I
FEATURES EXTRACTED FROM THE GENERATED CTI

Feature	Description
Con_count	# of Connections
Num_of_logins	# of logins
Connection_duration	Connection duration
Port_con_count	# of connections from a specific port
File_uploads	# of file uploads
URL_length	Length of requested URLs
URL_length-max	Length of the longest requested URL
URL_length-min	Length of the shortest requested URL
URL_length-average	Average length of requested URLs
HTTP_requests	# of HTTP requests
Bytes_sent	Total bytes sent
HTTP_GET_requests	# of HTTP GET requests in a session
HTTP_POST_requests	# of HTTP POST requests in a session
HTTP_3xx_percentage	% of HTTP requests that led to an HTTP 3xx code response
HTTP_4xx_percentage	% of HTTP requests that led to an HTTP 4xx code response
Image_requests	% of HTTP image requests
css_requests	% of HTTP css file requests
js_requests	% of HTTP JavaScript file requests
HTML_to_image_ratio	# of requested HTML files divided by # of requested image files in a session

After feature extraction, we used various techniques to perform feature selection on the collected dataset. First, we removed features that depict high correlation with each other, using the corr() method of SKLearn, a method based on the Pearson correlation coefficient. We set the threshold to 0.80, i.e., we remove features that depict correlations above this value. Also, as another feature selection technique, we used the mutual information method [14] to remove features that did not contribute to finding the target value.

C. ML-Based Threat Level Classification

The ML-based models aim to enrich the CTI that is extracted from the original data. To this end, this work examines the use of several ML methods and focuses on methods that fall under the notion of ensemble methods, i.e., techniques where different or similar models contribute in cooperation to the final prediction. Different research efforts have highlighted the superior predictive power of ensemble methods in classification tasks [15], compared to simple models, including categorisation tasks in the cyber-security domain [16], [17], given that they exhibit less bias (i.e. are more "comprehensive" models) and less variance (i.e. less chance of over-fitting).

The main types of ensemble methods are as follows [16]: (i) *Bootstrap aggregation (Bagging)* [18]: models are trained in parallel, with each model being trained separately from the others, while a majority vote is used for prediction; (ii) *Boosting* [19]: models are trained in sequence, with each model learning from the prediction mistakes of the previous model, while a weighted vote is used for prediction; and (iii)

[4]https://wazuh.com/

[5]https://documentation.wazuh.com/current/user-manual/ruleset/getting-started.html

Stacking [20]: the predictions of well-performing base models are combined to build a new model and provide better results

This work considers four different ensemble models: the Adaptive Boosting Classifier (AdaBoost) [21], the Random Forest Classifier (RFC) [22], the Light Gradient Boosting Machine (LGBM) [23] and the Extreme Gradient Boosting (XGBoost) [24]. These fall into the bagging (RFC) and boosting (Adaboost, XGB and LGBM) types of ensemble learning.

In particular, RFC uses Decision Trees (DTs) as a base, with its significant difference from a plain DT being that each node is split using the best among a subset of predictors randomly chosen at that node. Moreover, RFC uses only a sample of the features each time during the bagging procedure, which reduces the generated variance, while RFC can also highlight the importance of the features in terms of their predictive power [16], thus resulting in high performance in classification tasks [25]. On the other hand, AdaBoost is a powerful boosting algorithm that also uses DTs at its base. AdaBoost uses the mistakes of the previous models by increasing the weight of the misclassified data points and finally turning weak models into strong models [16]. XGBoost and LGBM, known as Gradient Boosting Decision Trees (GBDTs) [26], directly learn from the models' prediction mistakes, instead of updating the weights of data points. XGBoost uses gradient descent to minimise the error of the DT that operate on its base and also regularisation methods such as LASSO (L1) and Ridge (L2) to reduce over-fitting and improve performance [24]. Finally, LGBM also use DT on its base. However, in contrast to XGBoost, LGBM splits the trees leaf-wise rather than depth or level-wise, which leads to better results as more loss is minimised on each leaf [23].

Overall, the notion of ensemble learning and the aforementioned representative algorithms are good candidates for our problem, as they can lower the bias and the variance during training, which is particularly important in cases of relatively small datasets, similar to the ones used in this work.

IV. Evaluation

A. Experimental Setup

Following the procedure described in Section III-A, we deployed a Dionaea honeypot for a month and collected a dataset that consists of 345 security incidents that occurred on the deployed honeypot, with each security incident considered as a sample. Subsequently, the 18 features listed in Section III-B were extracted and used as input to the ML algorithms described in Section III-C.

To build our ML models, we performed several preprocessing steps. First, we used the train/test split method from sklearn to split our dataset. Due to the relative small size of the dataset, we split it with a ratio of 90% into the train data and 10% into the test data [27], in order not to lose valuable training samples. The samples were shuffled and also stratified across the class data so that the test split maintained the proportion of both classes.

To evaluate the performance of our models, we used cross-validation and, more specifically, stratified k-fold that ensures that the dataset class proportion is preserved on every split, which can result in low bias and variance rate [28]. During the stratified k-fold, we used the random search technique [29] to find the best parameters for our models using the train set. Afterwards, the models were trained on the same set and tested on the separate holdout test set, to evaluate their performance on previously unseen data.

Finally, to account for values away from the normal distribution (since most ML algorithms perform better when feature values are in a specific range), we used the robust scaler method [30]. We also used the SMOTE (Synthetic Minority Over-sampling Technique) method [31] and, more specifically, the Adaptive Synthetic Sampling approach (ADASYN) [32] to deal with unbalanced classes.

To build the ground truth and categorise an extracted IP address as origin of either highly-abusive or less-abusive behaviour, we utilised two relevant resources: the AbuseIPDB[6] and the VirusTotal[7] projects. In particular, the extracted IP addresses were queried against the database of AbuseIPDB which returns a result that includes a confidence of abuse rating (scaled 0-100) indicating, based on user reports, the confidence on whether the IP address is involved in malicious activity. A rating of 100 means that we are certain an IP address belongs to a threat actor, while a rating of 0 means that they were unreported or undetected by other parties. We set the threshold to 75, i.e, IP addresses with confidence of abuse rating lower than 75 were categorised as less-abusive, while those with confidence equal or greater than 75 were categorised as highly-abusive. Moreover, we also considered an additional ground truth resource in order to increase the robustness of our evaluation. To this end, we scanned the test set IP addresses using VirusTotal to obtain information about their behaviour; if the IP address was found abusive in VirusTotal, we considered it as highly-abusive. In this case, the evaluation was performed by comparing our experimental results with the VirusTotal results on the basis of their accuracy.

B. Evaluation Metrics

In this work, threat categorisation is viewed as a binary classification problem where negative stands for the less-abusive class and positive for the highly-abusive one. The overall performance of the four models was measured using [33]: (i) *precision* (i.e., the ratio of True Positives (TP) to the sum of TP and False Positives (FP)), (ii) *recall* (i.e., the ratio of TP to the sum of TP and False Negatives (FN)), (ii) the *F1-measure* (i.e., the harmonic mean of precision and recall), and (iv) the *Receiver Operator Characteristic* (*ROC*) which depicts on a plot the relationship between the True Positive (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis; for a numeric representation of this relationship, the *Area Under the Curve* (*AUC*) is calculated.

[6]https://www.abuseipdb.com/

[7]https://www.virustotal.com/

TABLE II
RESULTS OF THE EVALUATION EXPERIMENTS

	Precision	Recall	F1 score	AUC
RFC	0.59	0.63	0.59	0.82
ADA	0.66	0.78	0.60	0.78
LGBM	0.69	0.80	0.70	0.91
XGB	0.67	0.78	0.67	0.90
Results after threshold adjustment				
RFC	0.70	0.84	0.69	0.82
ADA	0.66	0.78	0.60	0.78
LGBM	0.69	0.83	0.67	0.91
XGB	0.71	0.81	0.73	0.90

TABLE III
RESULTS EVALUATION WITH VIRUSTOTAL

	Higly-abusive	Less-abusive	Accuracy with VT
ADA	11 (31.43%)	24 (68.57%)	80.00%
RFC	15 (42.86%)	20 (57.14%)	85.71%
LGBM	17 (48.57%)	18 (51.43%)	85.71%
XGB	17 (48.57%)	18 (51.43%)	85.71%
VirusTotal	12 (34.28%)	23 (65.71%)	

TABLE IV
MOST COMMON ATTACKS PER CATEGORY

Attack	Less-abusive	Highly-abusive
mssqld	**76**	2
Blackhole	20	**145**
upnpd	0	8
smbd	0	6
http (bruteforce)	15	0

C. Results

Table II presents the evaluation results on our dataset for the four ML models. Our main goal is to correctly identify as many highly-abusive attacks as possible (high recall) and avoid wrongly categorising highly-abusive attacks as less-abusive (low false negative rate).

First, we consider the default probability threshold for binary classification (i.e., 0.5). In this case, the RFC algorithm achieved 59% precision and 63% recall, which reveal a relative significant rate of false negatives. The other algorithms achieved higher recall scores, but with room for improvement.

To this end, we applied a threshold adjustment technique [34] to adjust the predefined threshold of 0.5 and lower the false negative predictions of the positive class. The new decision thresholds after the adjustment were 0.32 for RFC, 0.58 for ADA, 0.38 for LGBM, and 0.65 for XGB. After threshold adjustment, all ensemble algorithms achieved better results for all metrics compared to the results before threshold adjustment, except for ADA that remained the same.

More specifically, the results show that our models achieved a relatively high recall, where RFC and LGBM had the best scores (84% and 83%, respectively). ADA achieved the same results as before, which is not unexpected given the minor adjustment in its threshold. One important observation is that while we still classified as highly-abusive some samples that are in fact less-abusive (relative low TN and high FP), this was something that we could tolerate, given that we had a high recall which is our main goal. The F1 scores were not so high, but acceptable, showing a good balance between recall and precision on the test set.

Even though CTI is extracted from the original security incidents, we do not have information about the severity of these incidents, and all of them are considered of equal importance, which, in most occasions, is not the case. By utilising the ML models we are able to classify these incidents according to their severity (i.e., abusiveness), which offers additional information about the attack. Notably, ADA predicted 19 sessions as highly-abusive, RFC 15, LGBM 16, and XGB 11. Three of the models had 0 FN, except for XGB that had 1, which means that all highly-abusive sessions were correctly classified as such by our models.

The AUC score shows how well an algorithm can recognise the classes given a specific threshold. More specifically, it represents the relationship between TPR (i.e. abusive predicted as highly-abusive) and FPR (i.e. less-abusive predicted as highly-abusive) in different thresholds where higher TPR and lower FPR result in better scores. LGBM and XGB had the best AUC scores (91% and 90%, respectively).

It should also be added that we also implemented the stacking technique, but there was no improvement compared to the boosting and bagging methods that we studied (results not shown due to space limitations).

Next, we evaluate our results by comparing them with the results of VirusTotal project for the test set (IP addresses) in terms of level of abuse. The results (Table III) show that three of our models achieve similar results to Virus Total by 85.71%, with ADA being less similar than the other three models, achieving accuracy of 80%. This shows that our models (particularly RFC, XGB, LGBM) have good predictive capabilities, in terms of the abusive behaviour of an IP address.

Table IV lists the most common attacks per category. With regard to the less-abusive IPs, the most common attacks are SQL Injections (mssqld), DoS (Blackhole), and Bruteforce (http bruteforce); such Bruteforce attacks were performed using XML-RPC requests and have been identified in logs by the entries that include the *POST /xmlrpc.php HTTP/1.0* HTTP Request. Concerning the highly-abusive IPs, the most common attacks are DoS (Blackhole, upnpd) and malware uploads (smbd). An important observation is that DoS attacks are included in the most common attacks in both categories; this could be due to easiness of performing such attacks as they do not require much sophistication and can be performed from both advanced or less advanced attackers. The observations stemming from Table IV could facilitate an organisation to conduct threat modelling based on their assets and employ proper security measures, while the knowledge of which attacks are usually related to each category could be considered an important advantage during an IR procedure.

Overall, the different preprocessing techniques we applied, along with the fine tuning of the selected algorithms with the random search technique, allowed us to achieve relatively good results. Although the relatively small dataset that was used did

not prevent the models from achieving good results, we argue that bigger datasets can lead to more robust results without the need of the extensive preprocessing used in our work. In this line, our dataset is constantly updated with new threat information, that will improve our models in terms of their predictive capabilities.

V. Conclusions

In this work, we utilised ensemble ML methods and a honeypot deployed on the cloud to distinguish the security incidents that were identified in the honeypot (i.e., Dionaea). The RFC and LGBM models achieved the best results regarding the recall metric (84%, 83%) while LGBM achieved the best AUC results (91%) following by XGB (90%). The results showed that ensemble ML models are highly effective concerning the categorisation of security incidents, based on their level of abusive behaviour. The dataset that we utilise is constantly updated, thus enabling the continuous update of the generated CTI and increasing the amount of the extracted features. Therefore, as a future work we plan to add more observations as input to our ML algorithms, while also implement more advanced methods such as the utilisation of deep neural networks. Finally, we plan to deploy more honeypots as well as VMs which will provide various web pages and web applications in order to gather different data in terms of diversity and intelligence; we consider that the T-POT platform is an appropriate tool to this end.

Acknowledgment

This work was supported by the FORESIGHT (H2020-833673) and ECHO (H2020-830943) projects, funded by the European Commission.

References

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[2] S. Kumar, B. Janet, and R. Eswari, "Multi platform honeypot for generation of cyber threat intelligence," in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. IEEE, 2019, pp. 25–29.

[3] Z. Zhang, H. Esaki, and H. Ochiai, "Unveiling malicious activities in lan with honeypot," in *2019 4th International Conference on Information Technology (InCIT)*. IEEE, 2019, pp. 179–183.

[4] A. Kyriakou and N. Sklavos, "Container-based honeypot deployment for the analysis of malicious activity," in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–4.

[5] S. Bistarelli, E. Bosimini, and F. Santini, "A report on the security of home connections with iot and docker honeypots." in *ITASEC*, 2020, pp. 60–70.

[6] B. D. Le, G. Wang, M. Nasim, and A. Babar, "Gathering cyber threat intelligence from twitter using novelty classification," *arXiv preprint arXiv:1907.01755*, 2019.

[7] J. Zhao, Q. Yan, J. Li, M. Shao, Z. He, and B. Li, "Timiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data," *Computers & Security*, vol. 95, p. 101867, 2020.

[8] A. S. Gautam, Y. Gahlot, and P. Kamat, "Hacker forum exploit and classification for proactive cyber threat intelligence," in *International Conference on Inventive Computation Technologies*. Springer, 2019, pp. 279–285.

[9] V. Mavroeidis and A. Jøsang, "Data-driven threat hunting using sysmon," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 2018, pp. 82–88.

[10] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and I. Kompatsiaris, "Detection of advanced web bots by combining web logs with mouse behavioural biometrics," *Digital Threats: Research and Practice*, vol. 2, no. 3, pp. 1–26, 2021.

[11] J. Jabez and B. Muthukumar, "Intrusion detection system (ids): Anomaly detection using outlier detection approach," *Procedia Computer Science*, vol. 48, pp. 338–346, 2015.

[12] J. Song, H. Takakura, Y. Okabe, and K. Nakao, "Toward a more practical unsupervised anomaly detection system," *Information Sciences*, vol. 231, pp. 4–14, 2013.

[13] Q. Cao, Y. Qiao, and Z. Lyu, "Machine learning to detect anomalies in web log analysis," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 519–523.

[14] L. F. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 9–16, 1987.

[15] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020.

[16] N. Lower and F. Zhan, "A study of ensemble methods for cyber security," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 1001–1009.

[17] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," in *Proceedings of the Australasian computer science week multiconference*, 2018, pp. 1–6.

[18] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[19] P. Bartlett, Y. Freund, W. S. Lee, and R. E. Schapire, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[20] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[21] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[22] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[24] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[25] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[26] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[28] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[30] M. M. Ahsan, M. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, p. 52, 2021.

[31] S. Mishra, "Handling imbalanced data: Smote vs. random undersampling," *Int. Res. J. Eng. Technol*, vol. 4, no. 8, pp. 317–320, 2017.

[32] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[33] L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats," in *4th International Conference on Internet of Things, Big Data and Security (IoTBDS)*, 2019, pp. 125–136.

[34] G. Collell, D. Prelec, and K. Patil, "Reviving threshold-moving: a simple plug-in bagging ensemble for binary and multiclass imbalanced data," *arXiv preprint arXiv:1606.08698*, 2016.