Combined Forest: a New Supervised Approach for a Machine-Learning-based Botnets Detection

Christophe Maudoux, Selma Boumerdassi, Alex Barcello, Eric Renault

Combined Forest: a New Supervised Approach for a Machine-Learning-based Botnets Detection

Christophe Maudoux* 0000-0001-5215-9046
Selma Boumerdassi* 0000-0003-2603-2433
Alex Barcello† Eric Renault†

Abstract

Nowadays, botnet-based attacks are the most prevalent cyber-threats type. It is therefore essential to detect this kind of malware using efficient bots detection techniques. This paper presents our security anomalies detection system, based on a model that we named Combined Forest. Our approach consists of merging some pre-processed Decision Trees to highlight different kinds of botnet by detecting their intrinsic exchanges. Using a supervised data approach, each tree is built from a labelled dataset. In order to achieve this, we aggregate the IP-flows into Traffic-flows to extract key features and avoid over-fitting. Then, we tested different machine learning algorithms and selected the most suitable one. After that, many experiments have been done to determine the best parameters and design the most accurate, adaptative and efficient model.

Index terms – botnets, anomalies detection, machine learning, supervised & meta algorithms, network, cybersecurity

I – INTRODUCTION

The most serious threat against networks and a major challenging topic within cybersecurity nowadays are bot proliferation and botnet detection [1]. Botnets are an unusually large collection of devices compromised by malware and under the control of a remote attacker. Unlike other more conventional malware such as viruses, trojans or worms, bots exchange data with each other or with an attacker through an open communication channel using protocols such as IRC, HTTP or Peer-to-Peer (P2P).

Some Supervised and Unsupervised approaches have been developed to detect anomalies by using flow-based network traffic analysis like [2, 3]. The first supervised approach requires extracting *thirty-nine different statistical features* for every traffic flow defined by source and destination IP/port and protocol knowing that the *source port can evolve* and the *system needs to analyse each payload*. The second study based on the *Principal Component Analysis* unsupervised algorithm uses the CTU13 dataset described in section V. Each scenario here is *split into two parts* for training and testing purpose that can *introduce a bias*. Furthermore, difficulty here is to *tag each obtained clusters* as normal or malicious depending on *statistical analysis* that can lead to misclassification. In this paper, we focus on supervised Machine Learning Algorithms (MLAs) and we propose a new approach based-on a *boosted forest* built by combining several *pre-processed Decision Trees*.

The article is organized as follows. First, we introduce the supervised MLAs employed for our research. After explaining botnets and their architectures, we detail the CTU13 dataset that we employed for training and testing our Anomalies Detection System (ADS). Then, we present our *Combined Forest* approach and evaluate the solution in terms of accuracy and computing time. Future works consist in transposing our detection system to mobile network.

II – MACHINE LEARNING ALGORITHMS

MLAs are programs that can learn from data and improve from experience. Learning tasks may include defining the best function that maps input to output, finding out the hidden structure of unlabelled data or grouping samples such that objects within the same cluster are more similar to each other than to the objects from another cluster [4].

The most common type of machine learning process is to learn the mapping function: $Y = f(X) + e$ to make predictions of Y for new X with an irreducible error e based on the lack of attributes to sufficiently characterize the best mapping. This is called predictive modelling. The goal here is to make the most accurate possible predictions, irrespective of the function form.

Different MLAs make different assumptions about the shape and structure of the function. Assumptions can greatly simplify the learning process, but can also limit what can be learned. This is why it is very important to try out different algorithms on a machine learning problem.

2.1 Decision Tree (DT) or CART

Classification and Regression Trees (CART) is a supervised algorithm that can be used for classification or regression predictive modelling problems but are better at solving classification ones. DTs are composed of two nodes: *Decision* and *Leaf* nodes. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches are the decision rules and each leaf node represents the outcome. Decision nodes are used to make choice and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. Decisions are performed on the basis of features from the given dataset. It is a graphical representation for getting all possible solutions to a problem based on given conditions. It is called a DT because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure depicted by figure 1.

*Cnam Paris – Cedric/Networks and IoT Systems, 75003 Paris, France. Email: {firstname.name}@cnam.fr
†ESIEE Paris – LIGM/UMR CNRS 8049, 93162 Noisy-le-Grand, France. Email: {firstname.name}@esiee.fr

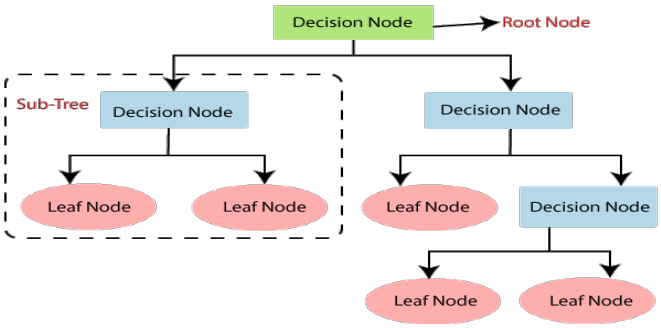

Fig. 1. CART diagram

2.2 Random Forest (RF)

This model is based on a multitude of DTs as shown in figure 2. It is one of the most popular and powerful MLAs. They select which variable to split on using a greedy algorithm that minimizes error. The DT can have a lot of structural similarities and in turn result in high correlation of their predictions. Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or weakly correlated. With *Random Forest*, the learning algorithm is limited to a random sample of features on which to search. The number of features that can be searched at each split point must be specified as a parameter to the algorithm.

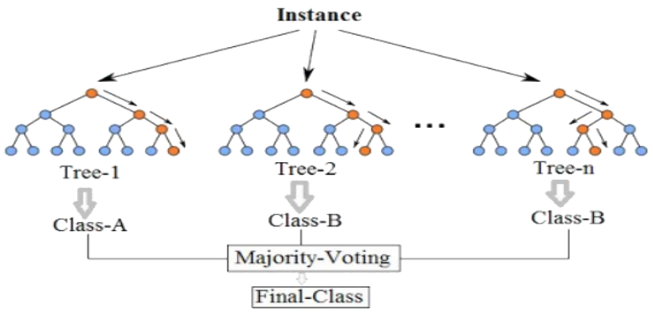

Fig. 2. Simplified RF

III – WHAT IS A BOTNET?

Cybersecurity is a major issue today. Over the last ten years, botnets represent the latest growing threat. They are becoming the preferred media for launching various attacks. During the 2000's, botnets evolved to become more and more sophisticated and malicious like `Zeus`, `Stuxnet`, `Emotet`, `Retadup`, ...

Botnet topology is essentially built on four components: *(i)* the compromised machines also known as the *Bots* or *Zombies (ii)* the *BotMaster* who controls the botnet, developed *(iii)* the *bot code (iv)* and the *Command and Control server* (C&C server) used for leading the botnet. A generic botnet topology is illustrated by figure 3.

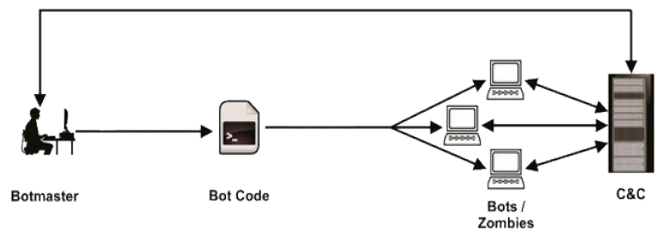

Fig. 3. Botnet topology

Control architecture is the key aspect of any botnets. Bots interact with the BotMaster to receive commands or send logs, to each other or to exchange data over a network using legitimate communication channels, in order to become part of a wider network of infected devices [5].

Botnet resilience and also its efficiency lies in the structure of its *Command and Control* architecture. The C&C channel is the main weak point and without it, a botnet is just a collection of useless infected devices.

IV – DETECTION PROCESS

Many researches have been done by monitoring network traffic, analysing messages or parsing their payload contents with induced privacy issues [2, 3]. In this paper, we will focus on machine learning techniques used for extracting patterns and detecting anomalous behaviours that could point out an existing C&C channel between malware and consecutively a botnet.

The main assumption of machine learning based ADS is that botnets generate specific traffic patterns that could be efficiently detected by MLAs. This approach affords a flexible detection method that does not require any prior knowledge of botnet traffic characteristics. It is independent and isolated from communication technologies and resilience strategies employed by botnets. Different kinds of MLAs have been developed and implemented in several detection schemas [6, 7].

Three main steps are required to achieve a good anomaly detection rate by using machine learning. First, all *IP-flows* must be aggregated into *Traffic-flows*. During the second step, specific features are extracted from aggregated flows to be used for computing main characteristics and modelling decisions. Finally, an MLA is trained and implemented to differentiate malicious activities from benign traffic.

4.1 Traffic Flow Aggregation

Huge amount of data can be collected during a network dump process. To decrease the data volume for analysis and be able to highlight exchange patterns, IP-flows are merged into a Traffic-flow corresponding to an exchange transaction between a source and a destination. This communication is mainly defined by source and destination IP address or port.

4.2 Features Selection & Patterns

Many features are available and can be extracted to define patterns. The main ones are; transaction duration, source or destination port, IP address, direction flow, number of transmitted packets, packet size, payload content, connection state, and protocol.

By using these characteristics, it is possible to define and detect patterns. The aim is to reduce key information to extract and analyse but keep the ability to detect new or unknown malicious behaviours and therefore not miss attacks. Selecting the right set of features for data modelling improve the performance learning process and reduce computational costs such as training time or required resource.

4.3 Training & Classification

Supervised MLAs training is based-on previously defined patterns and by providing a labelled dataset. A dataset can be created by capturing and tagging legitimate traffic from a well-known network then merging it with generated malicious traffic. In other hand, some open source labelled datasets are publicly available. For our researches, we chose to experiment the CTU13 dataset.

V – CTU13 DATASET

Czech Technical University 13 is a commonly applied dataset for research like [3, 8] provided by Malware Capture Facility Project of Czech Technical University [9]. It was captured in 2011 and consists of thirteen different botnet activity captures called scenarios listed in table 1.

A scenario is a specific botnet traffic sample generated and labelled as malicious data. Each malware uses several protocols and performed different actions. Traffic from known and controlled network endpoints like a router or switch is tagged as normal.

TABLE 1
CTU13 Scenarios

Nmr/Id	Dur (hrs)	Bots	Bot	Size (GB)
42/1*	6.15	1	Neris	52
43/2	4.21	1	Neris	60
44/3*	66.85	1	Rbot	121
45/4	4.21	1	Rbot	53
46/5*	11.63	1	Virut	37.6
47/6*	2.18	1	Menti	30
48/7*	0.38	1	Sogou	5.8
49/8*	19.5	1	Murlo	123
50/9	5.18	10	Neris	94
51/10	4.75	10	Rbot	73
52/11	0.26	3	Rbot	5.2
53/12	1.21	3	NSIS.ay	8.3
54/13	16.36	1	Virut	34

*Scenarios used for training

Training process requires specific and labelled data to be able to learn some botnet features and define a signature based on intrinsic behaviour. Therefore, we chose the scenarios marked with '*' to train different MLAs and leave out the other ones. Each training scenarios contains just one bot activity and only one infected host with IP address *147.32.84.165*.

Other scenarios have been used for testing the *forest* built by combining the DTs learned from the six training scenarios.

VI – COMBINED FOREST

We adopt a nonconformist approach to botnets and their C&C channel detection which is not based on conventional tools like `C` and `Python/Jupiter`. To construct our model and reach our aim, we went through the following points. *(i)* Very large IP 'binetflow' files provided by the CTU13 dataset have been aggregated into smaller Traffic-flows files by our advanced `Perl` parser to be processed by `Weka`. *(ii)* Several supervised MLAs have been tested to determine the most suitable one and four Decision Trees have been extracted from training scenarios using *J48* and a *Confidence Factor* (CF) of '0.35'. Increasing the CF involved a more accurate resulting tree but with more nodes and leaves. These deeper DTs are offset by the next step. *(iii)* Each pre-processed DT has been improved with the *AdaBoost* meta-classifier. *(iv)* All boosted trees have been combined to build our *forest* with the *Stacking* method.

Typically, meta classifiers have been developed to boost nominal classifiers like *AdaBoost* or *ClassifierOptimizer*. We have implemented the *AdaBoost* method [10]. This meta-algorithm can be used in conjunction with other MLAs to improve performance. Output of the other 'weak learners' is combined into a weighted sum that represents the final output of the boosted classifier. *AdaBoost* is adaptive in the sense that subsequent weak learners are tweaked in favour of those instances misclassified by previous classifiers.

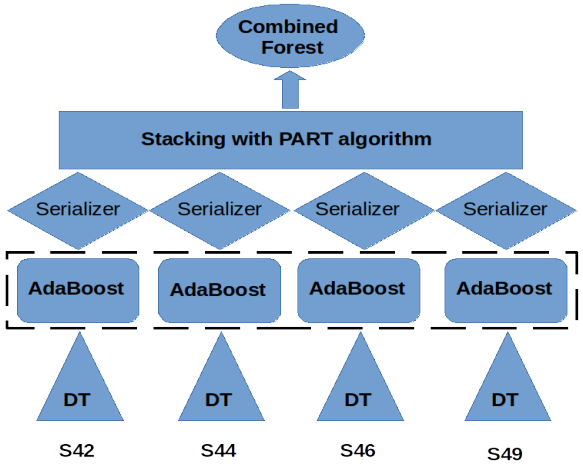

Fig. 4. Our Combined-Forest model

Some meta algorithms like *Vote*, *Bagging* or *Stacking* exist for gathering base class classifiers. To build our model, we chose to employ the *Stacking* one with PART to combine our pre-processed DT models [11]. It allows us to merge our DTs into a model we named *Combined Forest* depicted by figure 4.

Bagging meta-algorithm did not increase performances and *Voting* could not be used here because the result was always false. For each testing scenario, only one tree is able to detect botnet activity!

Pre-built models must be wrapped to be loaded into `Weka` before using. To do so, a generic connector named *Serializer* is required. This classifier loads a serialized model and calls it to make predictions.

VII – TOOLS & METRICS

7.1 Workbench for Machine Learning

We conducted our studies with the *Weka Toolbox*[1] to *(i)* find out which is the best algorithm able to detect security anomalies, *(ii)* determine the most suitable tuning parameter values to improve performances and *(iii)* validate our proposed ADS.

Waikato Environment for Knowledge Analysis is a freely available open source software developed at the University of Waikato in New Zealand. Overall, data must be converted into an ARFF (Attribute-Relation File Format) file. This is an ASCII text file that describes a list of instances sharing a set of attributes.

7.2 Basic Features

Algorithm performances can be evaluated by using multiple metrics defined and explained below. We can define the following four basic features:

True Positives (TP) correctly predicted to be true.

True Negatives (TN) correctly predicted to be false.

False Positives (FP) predicted to be true, but incorrect.

False Negatives (FN) predicted to be false, but incorrect.

[1] http://www.cs.waikato.ac.nz/ml/weka

7.3 Performance Metrics

Therefore, main performance metrics are calculated based-on the previous basic features:

Precision $\frac{TP}{TP+FP}$ ⇒ expresses what fraction of predictions as a positive class were actually positive.

True-Positive Rate (TPR) or Recall or Sensitivity $\frac{TP}{TP+FN}$ ⇒ fraction of all positive correctly predicted.

True-Negative Rate (TNR) or Specificity $\frac{TN}{TN+FP}$ ⇒ fraction of negative cases correctly predicted.

False-Positive Rate (FPR) $\frac{FP}{TN+FP} = 1 - \text{TNR}$

False-Negative Rate (FNR) $\frac{FN}{TP+FN} = 1 - \text{TPR}$

F-Measure (FM) or F1 Score $2\frac{\text{Recall.Precision}}{\text{Precision+Recall}}$ ⇒ is the harmonic mean of *Precision* and *Recall*.

Matthews Correlation Coefficient (MCC)

$$\frac{TP.TN - FP.FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Sensitivity and *Specificity* are inversely proportional to each other. So when we increase sensitivity, specificity decreases and vice versa.

VIII – EXPERIMENTS AND RESULTS

8.1 Feature Extraction

IP-flows must be aggregated into Traffic-flows to extract specific patterns and detect malicious activities. To highlight this important step, we experimented with two opposite approaches: *(i)* a basic parsing tool to directly analyse IP-flows and *(ii)* a parser to aggregate IP-flows, both written in `Perl`.

8.1.1 Basic Parser

A first naive approach could just consist in transposing 'binetflow' files provided by the CTU13 dataset into ARFF files without any modifications[2].

We tested this basic parsing with S42 scenario and the J48 algorithm [12]. First results exposed by table 2 seem very impressive with **TPR, Recall, Precision and F-Measure = 1 and FPR = 0**. These very good performances can be explained by the fact that J48 algorithm over-fits the data. Leaves are based on source port and bot IP address (147.32.84.165).

8.1.2 Advanced Parser

Our advanced approach aims to aggregate some IP-flows into Traffic-flows defined by a 4-tuple primary key. Client-Server exchanges are identified by source and destination IP address and port (socket), where the source port is versatile and can evolve. We chose to build the Aggregation Key (AgK) with the transport protocol [13, 14].

The 4-tuple aggregation key is set like this: **AgK = SrcAddr_DstAddr_Dport_Protocol** by using our advanced parser[3]. Results obtained with the same J48 algorithm from the S42 scenario pre-processed with our advanced parser are detailed by table 3. DT learned from aggregated data is composed of '18' leaves with a size equal to '35'. Model metrics are very good with **TPR and Recall = 0.935, FPR = 0, Precision = 0.959, F-Measure and MCC = 0.947**.

Most important feature here is that the advanced model *does not refer to any IP address or source port*. Table 4 summarizes total number of flows after basic or advanced parsing and relative number of flows marked as malicious with corresponding ratios.

TABLE 2
BASIC-PARSING CONFUSION MATRIX

Classified as	True	False
True	28,122	4
False	2	2,584,844

99.9998% Correctly classified instances!!!

TABLE 3
ADVANCED-PARSING CONFUSION MATRIX

Classified as	True	False
True	2,154	150
False	91	694,143

99.9654% Correctly classified instances

TABLE 4
TOTAL IP & TRAFFIC-FLOWS

Scenario	IP-flows (Ratio ‰)	Traffic-flows (Ratio ‰)
S42	2,612,972/28,126 (10.65)	695,538/2,305 (3.3)
S44	4,156,939/2,556 (0.61)	684,175/2,464 (3.6)
S46	121,994/749 (6.1)	43,116/65 (1.5)
S47	520,225/230 (0.442)	119,375/4 (0.034)
S48	106,298/59 (0.555)	40,725/13 (0.32)
S49	2,762,306/1,414 (0.512)	462,824/63 (0.14)

Total flows/Labelled as bot (Malicious/Total ratio)

8.2 Experiments

Algorithm Selection

`Weka` provides many MLAs. We chose to test one Neural Network i.e. Multilayer Perception (MLP), the Naive Bayes (NB) algorithm and some DTs. Available DT algorithms are *J48* and some derived: *J48Consolidate*, *DTGraft* and *BFTree*. From tables 5 to 7, we can deduce that *(i)* the J48 algorithm provides the best MCC with the minimum time testing and the maximum specificity (see section 7.3) *(ii)* S47 and S48 can not be used for building a DT because these scenarios do not provide enough malicious data with only 4 (0.034‰) and 13 (0.32‰) botnet Traffic-flows respectively.

TABLE 5
ALGORITHMS TPR/TNR/MCC

Scenario	J48	NB	MLP
S42	0.93/1.00/**0.95**	**0.99**/0.89/0.16	0.54/1.00/0.79
S44	1.00/1.00/**0.99**	1.00/0.96/0.33	0.00/1.00/X
S46	0.30/1.00/**0.58**	0.11/0.98/0.03	0.00/1.00/X
S48	0.00/1.00/X	**0.68**/0.95/?	0.00/1.00/X
S49	0.52/1.00/**0.68**	**0.93**/0.80.02/?	0.00/1.00/X

[2] https://bitbucket.org/cmaudoux/parsers/src/master/basic.pl
[3] https://bitbucket.org/cmaudoux/parsers/src/master/advanced.pl

TABLE 6
ALGORITHMS ELAPSED TIME TRAINING/TESTING

Scenario	J48	NB	MLP
S42	34.8/**0.04**	**2.5**/?	?/?
S44	20.2/**0.05**	**2.5**/0.36	708.6/0.12
S46	0.5/**0.00**	**0.1**/0.02	44.8/0.01
S47	2.0/**0.01**	**0.3**/0.06	127.6/0.02
S48	0.2/**0.00**	**0.1**/0.02	43.9/0.01
S49	13.8/**0.02**	**1.9**/0.19	500.5/0.09

TABLE 7
WEKA DT ALGORITHMS TPR/MCC

Scen.	BF	J48	Consolidate	Graft
S42	0.89/0.93	0.93/**0.95**	**0.97**/0.73	0.93/0.94
S44	1.00/0.99	1.00/**0.99**	**1.00**/0.95	0.99/0.99
S46	0.04/0.48	0.31/**0.72**	**0.68**/0.19	0.26/0.52
S49	?	0.52/**0.68**	**0.65**/0.47	0.50/0.69

All DTs reach a TNR equal to '1'

Algorithm Optimisation

CF is used for pruning. Smaller values incur more pruning and MCC variations (table 8). The tree is created according to the implemented algorithm. The additional pruning process looks at what nodes can be removed without affecting performance. This step reduces the risk of over-fitting the training data. To over-fit means that the model is able to classify the training data perfectly, but nothing else because instead of learning the underlying concept, the model learned intrinsic and specific properties of the training data, making it worthless for real data.

TABLE 8
MCC DEPENDING ON CF PARAMETER

Scen.	0.20	0.23	0.25	0.28	0.30	0.35	0.38
S42	0.95	0.95	0.95	0.95	0.95	**0.95**	0.94
S44	0.99	0.99	0.99	0.99	0.99	**0.99**	0.99
S46	0.57	0.57	**0.57**	0.55	0.55	0.56	0.55
S49	0.68	0.70	0.70	0.70	0.70	**0.72**	0.70

Confidence Factor modulation from 0.20 to 0.38

8.3 Results

The first step was to build our DT models from each training scenario. From the previous experiments we can conclude that the best results, presented in table 9, are obtained with the *J48* algorithm and a *Confidence Factor* equal to '0.35'. Table 10 exposes results reached after using the *J48* algorithm boosted with *AdaBoost* which improves the MCC compared to table 9. Finally, tables 11 and 12 summarize the overall efficiency.

Our *Combined Forest* model performances are interesting and encouraging with the following Mean/Standard Deviation values computed from table 12: **TPR = 0.374/0.123, Precision = 0.976/0.054, FM = 0.528/0.121 and MCC = 0.595/0.089**. These results could be improved by providing scenarios with higher Botnet/Total Traffic-flow ratio.

TABLE 9
BASE-MODELS

Scen.	Size*	TPR	FM	MCC	ETT (s)	Root
S42	35/18	0.932	0.945	0.945	36.5	Dport53
S44	25/13	0.997	0.993	0.993	17.6	Dport22
S46	33/17	0.308	0.455	0.517	0.5	Dport6774
S49	17/9	0.532	0.673	0.698	13.9	TotBytes

*Nodes/Leaves – J48 algorithm with CF = 0.35

TABLE 10
BOOSTED-MODELS

Scen.	TPR	FM	MCC	Precision	ETT (s)
S42	0.931	0.947	0.947	0.962	625
S44	0.991	0.991	0.991	0.992	329
S46	0.385	0.543	0.596	0.926	22
S49	0.597	0.747	0.772	1.00	188

J48 with CF = 0.35 and `AdaBoost` – ETT = Elapsed Time Training

TABLE 11
COMBINED FOREST RESULTS *without* ADABOOST

Scen.	TPR	Precision	FM	MCC	Bot/Total flows
S43	0.356	0.965	0.520	0.586	309/427,810
S45	0.133	0.667	0.222	0.298	15/201,329
S50	0.471	0.979	0.636	0.678	3,303/436,460
S51	0.395	0.941	0.557	0.610	81/236,082
S52	0.333	1.0	0.5	0.577	9/43,330
S53	?	?	?	?	977/92,482
S54	0.414	0.954	0.577	0.628	302/370,614

TABLE 12
BOOSTED COMBINED-FOREST RESULTS

Scen.	TPR	Precision	FM	MCC	ETT (s)
S43	0.324	1.00	0.489	0.569	5
S45	0.200	1.00	0.333	0.447	2.5
S50	0.558	0.866	0.678	0.693	5
S51	0.457	1.00	0.627	0.676	3
S52	0.333	1.0	0.5	0.577	0.5
S53	?	?	?	?	2
S54	0.374	0.991	0.54	0.609	5

ETT = Elapsed Time Testing

IX – DISCUSSION & FURTHER WORKS

Aggregate IP-flows into Traffic-flows by creating a 4-tuple primary key constructed of; source and destination IP, destination port and protocol allowed the highlighting of the botnet C&C channel as exposed in section 4.1. The last column of table 9 named 'Root' specifies the first node of each model extracted from the training data. With figure 5, It highlights the fact that *Neris* (S42) exchanges over TCP/6667 or UDP/53 (DNS) and *Rbot*'s (S44) C&C channel is based on ssh (TCP/22). We can suppose that *Neris* and *Rbot* obfuscate or mix their exchanges with `ssh` or DNS traffic to by-pass firewalls and also prevent C&C channel detection. Same graphics also show that *Virut* exchanges by using TCP on ports 80, 443 and 6667 – a well-known backdoor – or UDP on port 53. It required more digging and analysing bot code to confirm this.

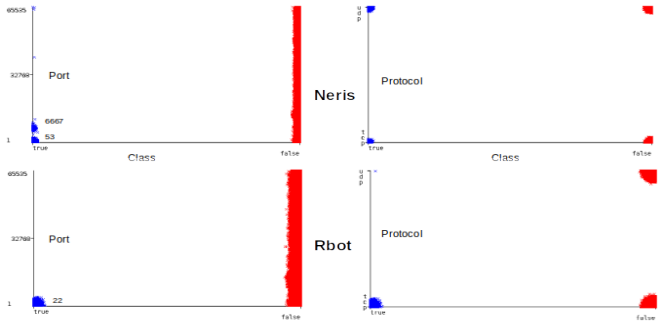

Fig. 5. *Neris* and *Rbot* activities (plotted with jitter)

A first model implementation with the boosting algorithm just after the stacking one did not increase the performances significantly. Therefore, the boosting phase must occur after the DT modelling as depicted by figure 4.

Our model is able to highlight a botnet family's C&C channel if, of course, a corresponding DT has been extracted and combined. Tables 11 and 12 show that botnet activities in S53 have not been detected by our *Combined Forest* because no relative training scenario has been provided.

We plan to transpose our ADS to mobile network and test our *Combined Forest* model with `Android` malware or mobile botnets. This will require to define a new aggregation key and an adaptation of the parser to extract useful features.

X – CONCLUSION

Botnets must be effectively detected to be defeated. This article explores and describes how network traffic flow analysis and machine learning can be employed to provide an efficient ADS. We developed and exposed a new botnet detection system that relies on Traffic-flows aggregation and supervised MLAs for extracting patterns and modelling botnets C&C channel. Our system is able to accurately highlight botnet activity using network flow features and multi Decision Trees merged into a *Combined Forest*. This approach that consists in detecting C&C exchanges is more efficient because of pointing out attack activities is often too late and more difficult! Additionally, our ADS can be easily improved to detect new botnet families by appending some corresponding previously learned DTs without building all the model again.

We can deduce from our researches that this selected approach provides high precision of traffic classification by analysing a few key features, and it will also be suitable for real-time application. We are planning further studies to test and adapt this approach to implement a real-time system able to detect mobile botnet activities.

References

[1] A. Guirakhoo. *FBI IC3 2019: Cybercrime Results in over $3.5 Billion in Reported Losses — Digital Shadows*. en-US. Mar. 2020. URL: https://www.digitalshadows.com/blog-and-research/fbi-ic3-2019-cybercrime-results-in-over-3-5-billion-in-reported-losses/ (visited on 11/12/2020).

[2] M. Stevanovic and J. M. Pedersen. "An Efficient Flow-Based Botnet Detection Using Supervised Machine Learning". In: *ICIN*. Honolulu, HI, USA: IEEE, Feb. 2014, pp. 797–801. ISBN: 978-1-4799-2358-8. DOI: 10.1109/ICCNC.2014.6785439. URL: http://ieeexplore.ieee.org/document/6785439/ (visited on 02/25/2021).

[3] E. S. C. Vilaça, T. P. B. Vieira, R. T. de Sousa, and J. P. C. L. da Costa. "Botnet Traffic Detection Using RPCA and Mahalanobis Distance". In: *WCNPS*. Brasilia, Brazil: IEEE, Oct. 2019, pp. 1–6. DOI: 10.1109/WCNPS.2019.8896228. URL: https://ieeexplore.ieee.org/document/8896228/.

[4] J. Brownlee. *Master Machine Learning Algorithms: Discover How They Work and Implement Them*. en. Machine Learning Mastery, Mar. 2016.

[5] M. Kumar. *French Police Remotely Removed RETADUP Malware from 850,000 Infected PCs*. en. Article. Aug. 2019. URL: https://thehackernews.com/2019/08/retadup-botnet-malware.html (visited on 11/15/2020).

[6] S. García, M. Grill, J. Stiborek, and A. Zunino. "An Empirical Comparison of Botnet Detection Methods". en. In: *Computers & Security* 45 (Sept. 2014), pp. 100–123. ISSN: 0167-4048. DOI: 10.1016/j.cose.2014.05.011. URL: http://www.sciencedirect.com/science/article/pii/S0167404814000923 (visited on 03/18/2020).

[7] B. Abraham, A. Mandya, R. Bapat, F. Alali, D. E. Brown, and M. Veeraraghavan. "A Comparison of Machine Learning Approaches to Detect Botnet Traffic". In: *IJCNN*. Rio de Janeiro: IEEE, July 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489096. URL: https://ieeexplore.ieee.org/document/8489096/ (visited on 02/25/2021).

[8] A. Blaise, M. Bouet, V. Conan, and S. Secci. "BotFP: FingerPrints Clustering for Bot Detection". In: *IEEE/IFIP Network Operations and Management Symposium*. Budapest, Hungary: IEEE, Apr. 2020. DOI: 10.1109/NOMS47738.2020.9110420. URL: https://hal.archives-ouvertes.fr/hal-02501912 (visited on 04/10/2021).

[9] *The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background Traffic*. en-US. URL: https://www.stratosphereips.org/datasets-ctu13 (visited on 06/14/2020).

[10] Y. Freund and R. E. Schapire. "Experiments with a New Boosting Algorithm". en. In: *ICML*. Morgan Kaufmann Series in Data Management Systems. San Francisco, 1996, pp. 148–156.

[11] D. Wolpert. "Stacked Generalization". en. In: *Neural Networks* 5.2 (Dec. 1992), pp. 241–259. ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80023-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608005800231.

[12] D. N. Bhargava, G. Sharma, D. R. Bhargava, and M. Mathuria. "Decision Tree Analysis on J48 Algorithm for Data Mining". en. In: *IJARCSSE* (2013), p. 6. ISSN: 22776451, 2277128X.

[13] *Stratosphere Testing Framework*. en-US. URL: https://www.stratosphereips.org/stratosphere-testing-framework (visited on 01/17/2021).

[14] *Example of Using STF for Detecting C&C Channels. An Analysis of Pushdo Malware*. en-US. URL: https://www.stratosphereips.org/blog/2014/03/9/example-of-using-stf-for-detecting-cc-channels (visited on 01/15/2021).