A Novel Network user Behaviors and Profile Testing based on Anomaly Detection Techniques

Muhammad Tahir*[1], Mingchu Li*[2], Xiao Zheng[3], Anil Carie[4], Xing Jin[5], Muhammad Azhar[6], Naeem Ayoub[7]
Atif Wagan[8], Muhammad Aamir[9], Liaquat Ali Jamali[10], Muhammad Asif Imran[11], Zahid Hussain Hulio[12]
School of Software Technology, Dalian University of Technology, Dalian, 116620, China[1, 2, 3, 4, 5]
Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, 116020, China[1, 2, 3, 4, 5]
College of Computer Science, Shenzhen University, Shenzhen, 518060, Guangdong, China[6]
Department of Mathematics & Computer Science, University of Southern Denmark
Cam-pusvej 55, DK-5230 Odense M, Denmark[7]
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China[8]
College of Computer Science, Sichuan University, Chengdu, 610065, China[9]
College of Software Engineering, Nankai University, Jinnan District, Tianjin, 300350, China[10]
School of Chemical Engineering, Dalian University of Technology, Dalian, 116024, China[11]
School of Mechanical Engineering, Dalian University of Technology, Dalian, 116024, China[12]

***Abstract*—The proliferation of smart devices and computer networks has led to a huge rise in internet traffic and network attacks that necessitate efficient network traffic monitoring. There have been many attempts to address these issues; however, agile detecting solutions are needed. This research work deals with the problem of malware infections or detection is one of the most challenging tasks in modern computer security. In recent years, anomaly detection has been the first detection approach followed by results from other classifiers. Anomaly detection methods are typically designed to new model normal user behaviors and then seek for deviations from this model. However, anomaly detection techniques may suffer from a variety of problems, including missing validations for verification and a large number of false positives. This work proposes and describes a new profile-based method for identifying anomalous changes in network user behaviors. Profiles describe user behaviors from different perspectives using different flags. Each profile is composed of information about what the user has done over a period of time. The symptoms extracted in the profile cover a wide range of user actions and try to analyze different actions. Compared to other symptom anomaly detectors, the profiles offer a higher level of user experience. It is assumed that it is possible to look for anomalies using high-level symptoms while producing less false positives while effectively finding real attacks. Also, the problem of obtaining truly tagged data for training anomaly detection algorithms has been addressed in this work. It has been designed and created datasets that contain real normal user actions while the user is infected with real malware. These datasets were used to train and evaluate anomaly detection algorithms. Among the investigated algorithms for example, local outlier factor (LOF) and one class support vector machine (SVM). The results show that the proposed anomaly-based and profile-based algorithm causes very few false positives and relatively high true positive detection. The two main contributions of this work are a new approaches based on network anomaly detection and datasets containing a combination of genuine malware and actual user traffic. Finally, the future directions will focus on applying the proposed approaches for protecting the internet of things (IOT) devices.**

Keywords*—Network user behaviors; profile testing; anomaly detection techniques; datasets; anomaly detection algorithms; machine learning

I. Motivation and Introduction

Nowadays, detecting intruders and malware infections [1], in local networks is one of the most difficult and highest studied challenges in modern computer security. From the huge amount of detection methods proposed, a large majority used static rules or reputation methods for performing the detection; until more modern behavioral techniques were introduced. Although very useful, these static techniques were not enough to detect the majority of attacks and malware. In particular, it is believed that the more important limitations of the current techniques are that first, the detections are done per connection and not per user, second, the classifiers are trained and tested on "only normal" and "only infected" datasets and third, the types of attacks and infections evolve and make classifiers quickly less useful. Apart from the more traditional signature-based intrusion detection system (IDS), such as snort [2] and bro [3], there has been extensive research on behavioral detection methods during the last decade. From these new methods, the most used is anomaly detection techniques (ADTs) due to its easy implementation and understandability. Anomaly-based IDS detect anomalies by assuming that more than half the data is normal and then searching for some deviation from that normality. The main benefits of ADTs are its ability to detect previously unknown attacks and the identification of non-malicious problems within the network, such as corporate policy violation. Despite its extensive use, ADTs suffers from several issues that undermine its usefulness. First, it is hard to verify the results, leading to attacks being mixed with normal connections. Second, the anomalies found are not necessarily malicious, generating a high rate of false positives. Third, the nature of the network changes over time, making the original normal model obsolete and prone to more errors. Fourth, ADTs methods usually work with packets, making the methods a little less stable. The amount of errors

generated by these issues tends to be so large that researchers give the output of ADTs to other algorithms to improve the detection. In consequence, ADTs models tend to need constant maintenance and supervision even to achieve acceptable results. Shown below in the 2nd row of Fig. 1, is available from different sources in a multitude of forms.

To improve the current situation, it is proposed that a ADTs method that focuses on the high-level changes in the behavior of a user. The proposed anomaly detection method is analyzing time-fixed user profiles and how they change with regard to the past traffic of the same user. Each profile is composed of a set of features based on the flows received and sent during a time-window. Fig. 2 shows diverse data sources out of the box (i.e., packet, NetFlow, logs, files, 3rd-party alerts and threat feeds).

It is hypothesized that it is possible to detect the infected users and to have a small number of false positives by focusing on the high-level behavior in time. The detection approach consists of monitoring a user's computer, and it is collected flows during a fixed time-window. With the data collected in that time-window, profile is created for that user. The profile describes user's behavior during this time-window. With the data obtained during the time-window of each profile, twelve features are computed. These references are featured as profile features (for more details see Section III.C).

The features of a profile cover a wide range of possible changes in traffic to have enough representations of the behavior of the users. For example, one of the features analyzes the number of flows for all the transmission control protocol (TCP) connections from a host, and another feature describes the number of packets for all the TCP connections aggregated by destination ports. If a user is infected with a malware which tries to connect to a large number of different computers, this last feature will show a very distinctive pattern for the number of packets on each destination port. As another example, if malware steals personal information of a user the features of the profile might show a higher volume of packets sent from the user's computer.

The profiles generated for each time-window may allow us to detect these changes and variances and generate alarms that may prevent the actions of an infected computer.

To be able to detect changes in the behavior of users it is needed to define a baseline of their behaviors. Set of profiles are collected which covers different time intervals such as: working hours, night activity, weekends, etc. Within each profile, each feature describes the user's behavior from a different perspective. Fig. 3, showing data sources in a variety of detection vectors in order to detect the different attacks and stages.

Each perspective contributes a different type of information to the algorithm, and therefore obtaining detections for different behaviors. Each feature of individually is evaluated with the same feature in past profiles.

The initial phase of in anomaly detection method consists in preprocessing the collected data and preparing it for the algorithm. Dimensionality reduction technique is applied to the features of a profile. Each feature has to have the same dimensionality. Then the features of a profile are normalized. Data normalization might significantly improve the performance of anomaly detection algorithms [4]. The normalization is done for each feature individually with respect to the same feature in other profiles.

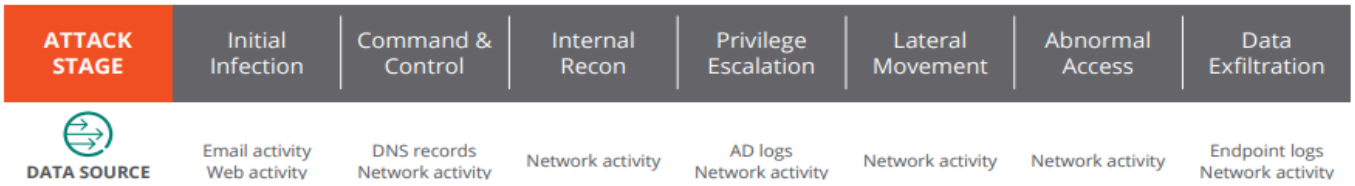


Fig. 1. Showing the Stages between Malware Infections and Data Losses as well as the Data Sources Needed for Analytics to Accurately Detect the Attack Stage.

Fig. 2. Showing the Comprehensive Subset of Data Sources to Build a Risk of Network user behaviors Profiling for Each Entity.

Type	Examples	Detection Vectors
Network activity	Firewall logs IDS/IPS logs Web Proxy logs Email logs Network traffic Network flows	Lateral movement Abnormal resource access Browser exploits Malware activity Suspicious file downloads Command and control activity Beaconing
Remote access activity	VPN logs	Credential theft, password sharing
Identity	AD logs DHCP logs	Credential violations Account takeover Privilege escalations
Infrastructure	DNS logs	Command and control activity Tunneling Exfiltration
3rd party alerts	FireEye alerts WildFire alerts	Incorporate alerts into user risk profiles
Threat intelligence feeds	Commercial & STIX feeds	Perform historical impact assessment
Logs	File integrity monitoring logs	Suspicious file activity USB, cloud based file exfiltration

Fig. 3. Showing the Variety of different Detection Vectors and there Examples.

After the preprocessing phase, the anomaly is trained to detect algorithms (see in Section III.E) by using the normal behavior of a user. An anomaly detection model is created for each feature of a profile, based on all the data of that feature across all the normal profiles of the user. Each profile has twelve different features. Therefore, each feature has its unique normality model.

Once the twelve normal models were trained, it is possible to evaluate the performance of the algorithms using new unseen profiles. The new unseen profiles contain both normal data and normal plus attack data, and they were generated in the same way as the training profiles used to create the normal models. The anomaly detection algorithms, trained in the normal profiles, are used to evaluate the unseen profiles and assign a label to each feature in these profiles. Each feature has a label, and all these labels are used to generate the final label of the profile.

After assigning the labels, the detection method uses majority voting to get the final decisions regarding a profile. If six out of twelve of the final labels assigned by the anomaly detection algorithms classify the profile as anomalous, then the final label of the profile is anomalous.

One of the most important drawbacks of using anomaly detection algorithms is the lack of verified and trusted data. Therefore, the datasets consist of normal profiles and anomalous profiles. A normal profile is a profile which was created before the computer was infected and an anomalous profile is a profile which was created after the infection of the computer. A clean virtual machine is used to create a packet capture for both datasets. The creation of the datasets involved several steps. The first step is to imitate a normal user doing standard things (e.g., checking emails). After some time, the machine is infected with a malware while at the same time it is capable to continue to perform normal actions.

To evaluate the performance of the algorithm it's necessary to have ground-truth labels. It has been assigned the normal label (label = 1) to all the profiles that are created before the infection of the virtual machine. The anomaly label (label = −1) has also been assigned to all the profiles that are created after the infection of the virtual machine. The reason to assign the anomaly labels in this way is the assumption that everything after an infection is worth detecting, and that malware produces changes in the behavior observed in the network. In fact, not all the attacks are anomalies, but this assumption helps us better evaluate our algorithm.

The datasets and labels are used during the experiments to evaluate our hypothesis and proposed approach. Each experiment uses one normal dataset for creating the normal model and one mixed dataset of normal plus attacks for evaluating and testing the performance of the model. Each dataset is split into three parts: train, validation, and test. The train set contains only normal profiles created before the infection. This set is used to create a model of normal network user behavior. The validation set and the test set contains a mix of normal profiles and anomalous profiles.

From all the anomaly detection algorithms that are available, those are selected and reported by the community as better for this problem. The algorithm that is used in this research work has different parameters that can be adjusted to improve detection. The adjustment of these parameters is done based on the performance metrics computed on the validation set. The validation results allow us to select the best models. The model selection is described in Section III.G and later computes the final performance metrics on the test set.

The evaluation of the algorithms is done in two ways: per individual feature and for the whole profile. First, each feature gives a label for the profile from the point of view of this feature. Then the results are combined using the majority voting to get the final decision for the profile. Having a result per feature allows us to evaluate features individually and learn which ones contribute the most to the detection of the anomalous behavior produced by malware. The analysis of the performance of individual features might also help us better understand how malware communicates.

The analysis of the results is also done by analyzing individual features and then analyzing the result of the majority voting. The analysis of the results shows that most experiments achieved a very low false positive rate (FPR) for individual features in the experiments. FPR does not exceed 0.01 in three out of four experiments for all features. When majority voting is used to produce the final label, the FPR was 0.0 in all experiments. The highest achieved true positive rate (TPR) was 0.44. Although it may seem low, this is the final result among all tested profiles with the FPR 0.0. It means that the algorithm will detect one out of three anomalous profiles with 99.9% approximately success, while at the same time it will not detect a normal profile as an anomalous.

The major highlights and the gap of this study are as follows:

1) Analyze the state-of-the-art methods for detecting malicious behaviors with special attention to anomaly detection techniques.

2) Propose and implement anomaly detection method to detect changes in computer network user behavior analysis. Infected computers change their behavior, this testing method would help to detect them securely.

3) Experimentally evaluate the proposed solution on datasets from cognitive threat analytics (CTA), developed by Cisco Systems, Inc., is a cloud-based software-as-a-service product designed to detect infections on client machines.

4) Analyze results of the implemented system and propose further improvements and applications of the solution in network security.

5) Finally, the proposed method which is novel in that it analyze features individually across probability distribution of time-window. Results are promising and show that the high-level analysis may provide a good improvement over the current ADTs and the proposed algorithm achieved a low FPR and reasonably high TPR.

In this paper, the remaining sections are explained as follows: Section 2 introduces related work and state-of-the-art in the area of anomaly detection for network security. Section 3 creates a new anomaly detection methodology and facing the

malware infections of computers in networks. Section 4, describe the functional requirements of the dataset used for training and testing in the domain are investigated by using machine learning algorithms and get the final results. Section 5, experimental results of an anomaly detection method is designed and a good dataset generated for training and testing in the hypothesis by running experiments. Section 6, analysis the final results. Section 7, concludes this study along with possible future directions defined in Section 8.

II. Related Work and State-of-the-Art

In this section, related work and state-of-the-art is discussed in the area of anomaly detection for network security. There is much research that has been done in the field of anomaly detection in general and its application to the network security domain in particular. There are multiple survey papers of algorithms for anomaly detection. Chandola et al. [5] reviewed different types of anomalies, the different fields where anomaly detection is used, challenges of anomaly detection and algorithms that could be used for anomaly detection. The paper [6], mentioned that one of the main challenges of applying anomaly detection in the field of network security is that the nature of anomalies keeps changing with time and intruders try to adapt to evade detection.

A central premise of anomaly detection for security was defined by Patcha et al. [7] as that intrusive activity is a subset of anomalous activity. He mentioned that activities in a network could be split into four categories:

- Intrusive but not anomalous—the source of false negatives.
- Not intrusive but anomalous—the source of false positive.
- Not intrusive and not anomalous—true negatives.
- Intrusive and anomalous—true positives.

Among the systems that use flows for anomaly detection is Minnesota intrusion detection system (MINDS) [8] proposal. The system extracts the following features: source and destination IP addresses, source and destination ports, protocol, flags, number of bytes and number of packets. MINDS compute the anomaly score for each IP flow individually. As an anomaly detection algorithm, the creators of MINDS used local outlier factor [9]. One of the main differences between MINDS and it is proposed that the search for anomalies closer to the actions of the user, and not to the network. Anomalies are studied time, from several aggregations of the type of flows.

The methodology used in MINDS was used by Ertoz et al. [10] to develop an agent-based system to detect anomalies in networks by using multiple correlated anomaly detection techniques. Hubballi, N. et al. [11] used NetFlow data and built a trust model to reduce the number of false positive alarms. They combined the output of each agent to build a trust model. Each agent used not only past observations but also anomaly assessments obtained by other agents.

The algorithms that can be used for anomaly detection are varied and include any algorithm that can differentiate between distributions of data. This is the case of one class SVM that has been used for anomaly detection by Zhang et al. [12], used one class SVM to detect anomalies. They evaluated their approach on the dataset Knowledge discovery data mining KDDCUP99 which was created in 1999. The algorithm showed very promising results compared to other methods. Authors mentioned the problem of obtaining a good dataset with labels to evaluate the anomaly detection methods.

Xu et al. [13] also used NetFlow to analyze the traffic. Their system created a cluster for each internet protocol IP in the current time-window. Clustering was based on the source IP (srcIP). For each cluster, the system computed the normalized entropy of source port (scrPort), destination port (destPort) and destination IP (destIP) and used it as a feature vector to represent clusters. Then the system applied behavior classification scheme to classify each sample in its behavioral class.

All the features in this paper use the state field from IP flows to specify if the connection was established or not established. Mahoney et al. [14] also inspected TCP flags but based on individual packets. The proposed NETAD algorithm built nine models to identify anomalies in nine subsets of packets. Packets were split into subsets based on TCP flag in the packet and on the port. The algorithm achieved 66 detections out of 185 with only 20 false alarms.

One of the examples of creating normal traffic profiles is fire-sight tool [15] from Cisco. This tool allowed a user to specify a sliding time-window length and traffic profile would be created during this window. After the profile was created, the tool allowed detecting abnormal network traffic by comparing it to the profile. To detect abnormal traffic user should define correlation rules which would be triggered ones the traffic deviates from the normal profile.

Another example of profiling a user was presented by G. Pannell et al. [16]. The user profiles was created using multiple characteristics such as the number of running applications, key-stroke analysis, websites viewed, application performance and the number of windows. Each characteristic was modeled separately, and then the evaluation results were combined using a weighting algorithm to produce the final decision. The results showed that combining results was producing a lower FPR than individual characteristics.

The features used for detecting anomalies in this paper were a subset of features created by Benevento, F., et al. [17], and Wagner, Claudia et al [18], both authors of different papers defined a lot of different features to identify users in the social networks even if they would connect from a different place. Subset of features is selected which describes outgoing traffic from a computer of a user since it is wanted to detect the infection of the computer.

III. Methodology

In this section, the complete methodology is described step by step. The assumption of the approach is that after an infection the behavior of a host is changing, and the proposed algorithm have to be able to detect these changes.

To create a new anomaly detection method it is necessary to define what should be detected and why. The definition of anomaly depends on the goal of the system, the data available and the conditions of execution. In this paper, it is wanted to detect when a computer is infected by malware while it's still acting normally in the network. The focus is the malware infections of computers in networks. The data available are network packets, but it is decided to use NetFlow to process all the information quickly and to preserve the privacy of users. In consequence, the method is also evaluating if the use of NetFlow may be enough for a good anomaly detection method. The constraints of the method are that detections should be reported as soon as possible and that the number of false positives should be minimized. The proposed method analyzes anomalies in the behavior of the computers from a high-level perspective. This perspective is the actions of the user as they are reflected in flows in the network. Every time a user interacts with a computer, packets are sent via the network. These packets are grouped into flows according to their protocol. These flows are further grouped in this idea in specific new features, such as the number of flows sent to each destination port. These features are a higher level perspective of the actions in the network. To obtain a detection as soon as possible the traffic of each computer is separated in time-windows of five minutes. These time-windows allow the method to run quickly, to capture enough traffic to model behavior, but not to be too big to process. The decision taken by the anomaly detection method is per time-window.

A. NetFlows

NetFlow [19] is a data structure developed by Cisco Systems that allow to capture and aggregate information about network traffic each packet which is forwarded through a router or a switch is examined for a set of IP packet attributes. Another way to generate NetFlow is to use a monitoring software such as ARGUS tool to generate them directly from network traffic. Usually, packets are identified by the following attributes.

- IP source address
- IP destination address
- Source port
- Destination port
- Layer 3 protocol type
- Class of Service
- Router or switch interface

After the examination, all packets are grouped based on the attributes described in the list. Constructing anomaly detection methods using NetFlow data has been a subject of research in many works, such as [20-22]. NetFlow data are easy to generate. It can be generated flows from traffic captures or obtain it directly from a router. Also, NetFlow data preserves the anonymity of the users, because it does not contain the content of packets.

The flows in experiments were generated from the packet capture ".PCAP file", using the "Argus tool" [23]. One of the reasons of using the Argus is the option to generate bi-directional flows. Bi-directional flows contain information about packets sent in both directions. Argus can generate additional fields to the ones that used for flow creation.

The flag field in flows contains two parts separated by an underscore. These are the TCP flags used in the packets in the flow. In the state field generated by Argus, the letters to the left of the underscore character are the TCP flags used in the packets going from the source to the destination. The letters to the right of the underscore character are the TCP flags used in the packets going from the destination to the source.

B. Established and not Established Connections

It is determined if a flow between two computers is established or not by the flag field. An established TCP connection is the one which completed a three-way handshake [24]. For example, these are the flags of an established connection flag: SRPA_FSPA. The Argus state field summarizes the TCP flags used in the packets. In flow state there are the following TCP flags:

- S—synchronization bit (SYN)
- R—reset bit (RES)
- P—push bit
- A—acknowledgment bit (ACK)
- F—final bit (FIN)

These packets could have been sent in any order. An example state of a not established TCP connection is *S.* It means that the source IP address initiated a connection with SYN flag and did not get any response. For the UDP protocol Argus uses flags such as CON and REQ which are set by Argus. CON flag is set in case of an established UDP connection. REQ flag is set if a client tried to establish a connection but a server did not send anything in response.

C. Profiling to Identify Network user behaviors

A profiling is a high-level representation of user behaviors in a network. To create a profile, it is collected network traffic over a predefined time-window. Currently, the creation of the profile only includes the IP protocol version 4 (IPv4) not (IPv6), and the TCP and UDP protocols. Other protocols are not included such as ICMP because they are by far the minority of the packets.

Each feature is constructed in the following way: First, the purpose of the feature is decided; for example, to capture the variations in the destination ports, according to the flows used by the computer being analyzed when the connection is successful. Second, a subset of all the flows in the current time-window is selected according to the previous criteria. In the example, only the established flows are selected. Third, the subset of flows is separated into two groups, one for the TCP protocol and one for the UDP protocol. This separation is done because the purpose of applications using the TCP and UDP protocols is very different and should not be mixed in a single feature. Fourth, the desired field is extracted for all the flows. So far there are two groups of data, one has all the destination ports for established TCP connections, and the other has all the

destination ports for established UDP connections. Fifth, the extracted data is used to create a histogram of the number of flows per destination port. After these steps there are two features, both having a list of values that represents a probability distribution in a time-window. The first feature Client Destination would be called and Port Number of Flows TCP are Established and the second feature Client Destination Port Number of Flows UDP Established.

It is represented features in a profile as vectors of real numbers $f_k = (x1, x2, ..., x65535)$, where x_i is a value for i-th port and 65, 535 is the maximum amount of ports available. A profile contains the following set of features:

- Client Destination Port Total Bytes UDP Established–distribution of a total number of bytes over ports for established UDP connections.
- Client Destination Port Number Of Flows TCP Established–distribution of a total number of flows over ports for established TCP connections.
- Client Destination Port Number Of Flows UDP Not Established–distribution of a total number of flows over ports for not established UDP connections.
- Client Destination Port Total Packets TCP Established–distribution of a total number of packets over ports for established TCP connections.
- Client Destination Port Number Of Flows UDP Established–distribution of a total number of flows over ports for established UDP connections.
- Client Destination Port Total Packets TCP Not Established–distribution of a total number of packets over ports for not established TCP connections.
- Client Destination Port Total Bytes UDP Not Established–distribution of a total number of bytes over ports for not established UDP connections.
- Client Destination Port Total Bytes TCP Established–distribution of a total number of bytes over ports for established TCP connections.
- Client Destination Port Total Packets UDP Not Established–distribution of a total number of packets over ports for not established UDP connections.
- Client Destination Port Number Of Flows TCP Not Established–distribution of a total number of flows over ports for not established TCP connections.
- Client Destination Port Total Bytes TCP Not Established–distribution of a total number of bytes over ports for not established TCP connections.
- Client Destination Port Total Packets UDP Established–distribution of a total number of packets over ports for established UDP connections.

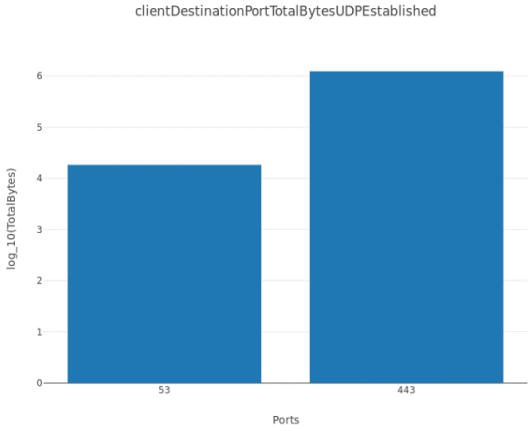
Those features are referenced as "profile features". The profile features describe the behavior of a user from different perspectives.

Fig. 4(a) shows a feature "Client Destination Port Total Bytes UDP Established" of a normal profile and Fig. 4(b) shows the same feature of an anomalous profile. A logarithmic scale is used for the *y*-axis to allow a large range to be displayed without small values being compressed down into the bottom of the graph.

D. Host behavior using Profiles

In the previous section, the complete content of a unique profiling was described, including its twelve features. These profiles are the basic unit of analysis of the anomaly detection method, and together they are part of the complete behaviors of the hosting user. This section first describes how the profiles are used to build the behaviors of a user, and then it describes which is the behavioral analysis method used by anomaly detection techniques.

The behavior of a user is defined by all the actions and decisions taken by the user in a certain period. Their actions are transformed into packets and flows, which are then captured in the already described features of a profiling. This allows each profile to describe the behavior of a user from twelve different points of view, each capturing a different perspective. As time goes by and the user generates more network traffic, and many profiles are generated.



(a) Shows an Example of Histogram for the Feature 'Client Destination Port Total Bytes UDP Established' for a Normal Profile. It can be seen that the Amount of Ports is Small.

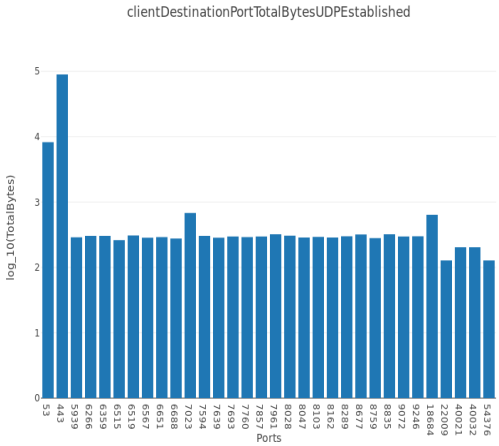


Fig. 4. (b) Shows an Example Histogram for the Same Feature but for an Anomalous Profile (an Attack was being Done). It can be seen a Large Number of Ports used.

The behavior of the user is then defined by all these profiles and their features. However, each feature describes the same data differently and therefore it does not make much sense to compare each feature with each other. Instead, it is proposed to compare each feature in the profile, with the same feature in the rest of the profiles for this user. The idea is that anomalies will arise when the same feature is analyzed in the concatenation of profiles. Fig. 5 shows the idea of searching for anomalies on the same feature on successive profiling.

Each feature in the profile corresponds to some measurement of data per port. Interpreted as a histogram, each feature is defined in the space of 65, 535 dimensions. Working with such a large space has two main limitations. First, data in the histogram are sparse, since most of the ports are never used. Second, the algorithms that analyze this data will have to deal with an increased, and probably unnecessary, complexity. Therefore, a reduction in the dimension of the space of each feature is necessary.

a) Dimensionality Reduction of the Feature Space: After normal profiles are collected, features are aggregated by type of data it measures over all profiles, and these are arranged into matrices Pi (i is a feature number). The matrix Pi is a sparse matrix, and it has dimensionality n × 65, 535, where n is the number of profiles these have been collected and [0 − 65535] is the range of ports. Such number of dimensions might cause a problem with scalability and with the performance of anomaly detection algorithms. To improve the scalability and performance the number of dimensions of the data are needed to be reduced.

A common dimensionality reduction approach is a principal component analysis (PCA). It is a well-known dimensionality reduction technique [25]. PCA was successfully used in [26] to reduce the number of dimensions in features derived from web logs.

PCA derives a reduced set of the most significant uncorrelated features (principal components) that are linear combinations of the original set of features [27]. The new principal components are vectors in the direction of the largest variance of the dataset.

Given m features, PCA selects $d < m$ principal components which define a new k-dimensional space based on normal profiles. There are two ways of specifying d: one can either set d to a fixed number or specify a percentage of variance to preserve, and d will be computed based on this percentage. In that case, the PCA algorithm was configured to preserve 99.9% of variance. It has been observed that dimensions to $d = 8$ could be reduced in some cases and due to a high percentage of preserved variance the much information had not been lost.

However, after some experimentation, it is realized that there had been a problem with using PCA in the approach. The problem was that an infected computer used ports which were not commonly used by a normal computer. Because ports were not used by the normal computer, the matrix *Pi* always had 0 values in the columns corresponding to these ports. PCA was not using these columns when learning how to transform *Pi* to *P'i* in a new basis.

When the transformation learned was applied on the normal profiles to an anomalous profile in which an anomaly was reflected in columns which were not used in creation of the new basis, the information about the anomaly has lost.

Because of this problem it has been decided against using PCA in the said approach. Due to a problem of unseen ports, a different method of dimensionality reduction is applied.

b) Anomaly Detection Methodology: Each profile feature is represented by a vector of real numbers. It is proposed to use the Euclidean distance to compare a profile feature along with other profiles. Fig. 6 below shows the training process of the proposed approach. During the training, the normal profiles are used which are collected at different time intervals. The collected profiles cover the time when a user is active and when the user is idle. Providing a model with different types of the behavior of the user, allows the model to better generalize the network user behaviors.

The dimensionality reduction does not require training, and it is the same for each profile feature. Therefore it is applied during profile creation. The profile features are grouped based on the information they capture. After features have been grouped the profile feature is centered and scaled so all components of individual profile features will have 0 mean and unit variance. The original mean value and standard deviation of components are saved for future because they are required to center and scale new profiles. "Standard-Scaler" is used from python library "sklearn-Preprocessing" for this task.

Then profile features are used as an input to the anomaly detection technique algorithm to train models. The ADTs algorithm is applied to each group of features and for each group it learns a model. After the training phase there are the following models:

- Twelve pairs of parameters ([μ, σ]) for scaling and centering of features.
- Twelve models to classify profile by individual features.

Fig. 7 below depicts the inference phase of the proposed algorithm. The algorithm uses the models created during the training phase. The scaling and centering is applied to individual profile features of a testing profile. The scaled features are used as an input to the trained models. Each model produces one of two labels: 1 if the model considers the testing profile normal according to this feature or −1 if it considers the testing profile anomaly according to this feature.

Each output of the models contributes equally to the final decision. If six out of twelve models classify the profile as normal, the final label will be 1. Otherwise, the profile is labeled with −1.

E. Anomaly Detection Algorithms

The previous sections described how the flows are processed and how the features are created to obtain a suitable set of data to work on. This section describes all the anomaly detection algorithms selected and tested in order to find the best one. Among all the possible options, the following algorithms were selected:

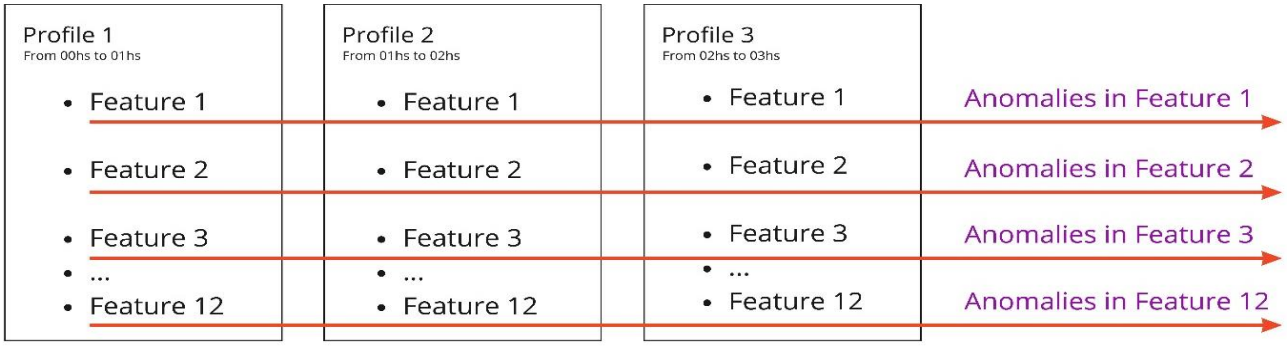


Fig. 5. Design of the Anomaly Detection Techniques. Instead of Comparing Each Feature with Each other, the Method Searches for Anomalies on the Same Feature on Successive Profiling.

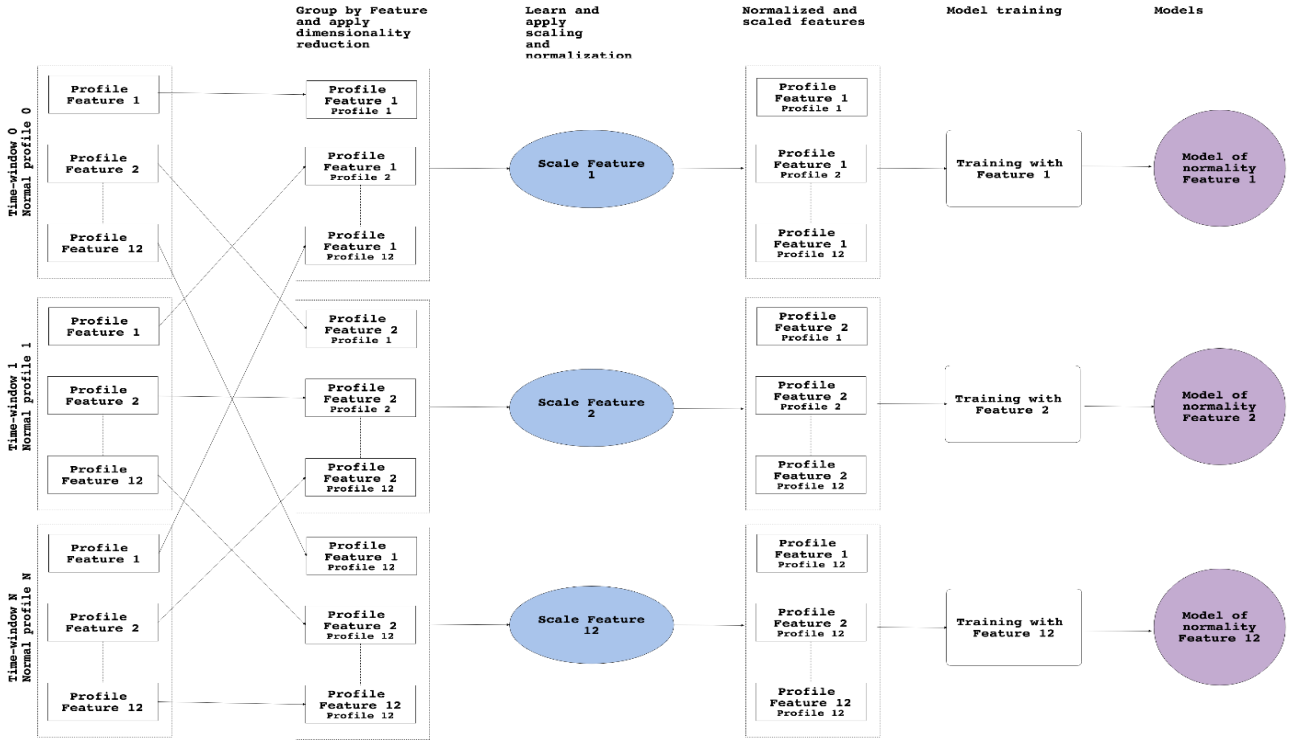


Fig. 6. The Diagram Shows the Training Process of the Proposed Approach. the Training uses *N* Collected Normal Profiles. there are Twelve Pairs of Parameters ($[\mu, \Sigma]$) for Scaling and Centering of Features and Twelve Models after the Training.

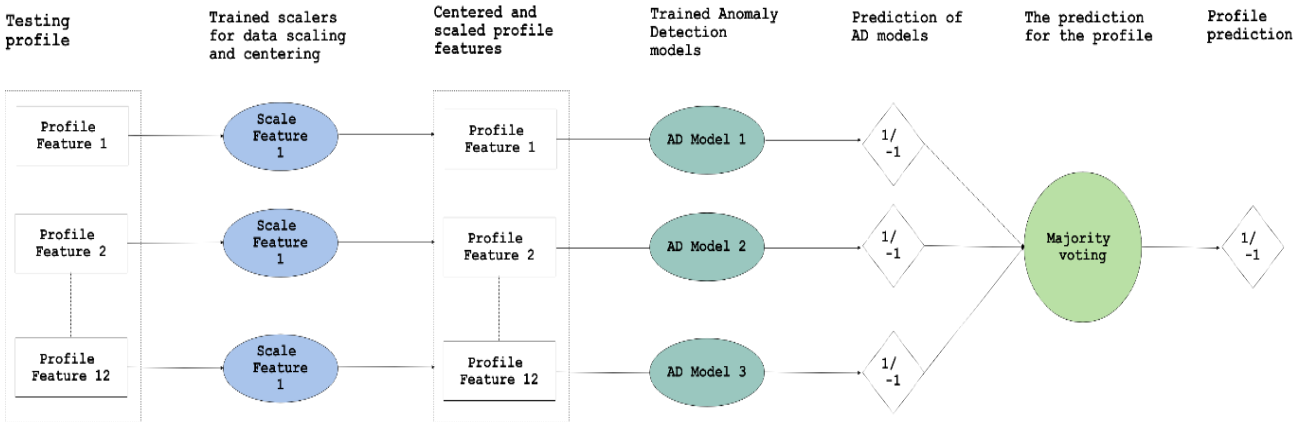


Fig. 7. The Diagram Shows the Inference Process of the Proposed Approach. there is One Profile which is Classified. Each Profile Feature is Scaled and Centered to the Same Scale as in Training Profile Features. Trained Models are used to Give Predictions for Each Feature and then Apply the Majority Voting to Get the Final Prediction for the Profile.

1) Local Outlier Factor (LOF) uses collected normal profiles to compute an anomaly score of a new profile.

2) One Class SVM [28]—outputs a boundary around normal data.

3) Isolation Forest [29]—method based on random forest, outputs a model of normal data.

The following subsections describe the details of how each of these algorithms works:

a) Local Outlier Factor (LOF) Algorithm: The LOF algorithm assigns an anomaly score to each data point based on the idea of density. The LOF measures how density around a point differs from the density of its neighbors. It detects outliers in data on regions with different densities.

Fig. 8 shows two clusters *C1, C2* and two additional *O* notations O_1 and O_2 reflects the facts that the complexity is linear to the number of hostnames. It can be seen that the *C2* cluster is much denser than the *C1*.

According to Hawkins' [30], both points (O_1 and O_2) are outliers. However, it can be shown that there does not exist any distance-based detector that can mark O_2 and not mark all points in the *C1* cluster.

This example shows that distance-based methods have a problem if there are regions with different densities in the data. The LOF algorithm presented in solves this problem by assigning a value to each object which represents its anomaly score.

This example shows that the distance based methods have a problem if there are regions with different densities in data. The LOF algorithm presented in solves this problem by assigning a value to each object which shows the degree of it being an anomaly.

To use the LOF algorithm, authors in [31], define several notions: *E*-neighborhood and *k*-distance.

Let *p* be an object from a database D, let E be a distance value, let k be a natural number and let d be a distance metric on D. Then:

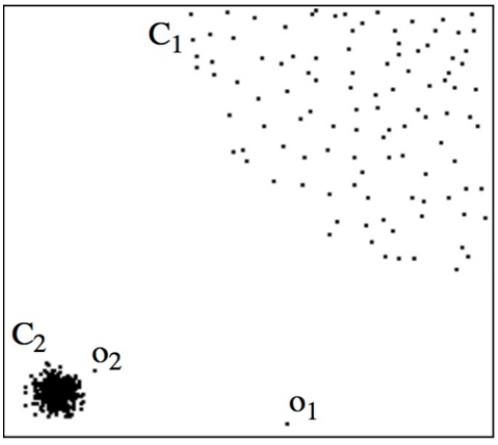


Fig. 8. Shows an Example Situation when there are Clusters with different Density. It is a Demonstration of the Advantage that LOF have over other Distance based Algorithms. This Example is Taken from LOF: Identifying Density-based Local Outliers [9].

Definition 1,

(E-neighborhood)

The E-neighborhood are the objects x with $d(p, x) \leq \in$: $N_{\in}(p) = \{x \in D | d(p, x) \leq \in\}$

Definition 2,

(k-distance)

The k-distance of p is the distance d(p, o) between p and an object o ∈ D, such that it holds at least for k objects o' ∈ D it holds that $d(p, o') \leq d(p, o)$ and for at most k − 1 objects o' ∈ D it holds that $d(p, o') \leq d(p, o)$

Definition 3:

(reachability distance of object p with regard to object o)

Let k ∈ N. The reachability distance of object p with respect to object o is defined as

reach-dist = max{k-distance(o), d(p, o)}

In other words, all objects that belong to the *k*-neighborhood of an object *p* are considered to be equally distant from the object *p*.

The next equation is the equation of local reachability density. It is an inverse of the average reachability of the object *p* from its neighbors.

$$lrd(p) = \frac{1}{\frac{\sum_{o \in N_k(p)} reach-dist_k(p,o)}{|N_k(p)|}} \quad (1)$$

The LOF is computed by comparing with local reachability distances of the neighbors:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd(o)}{lrd(p)}}{|N_k(p)|} \quad (2)$$

For any object which is inside a cluster, the LOF will be around 1. It does not depend on the density of a cluster, and it will be the same for objects inside cluster *C1* and objects inside *C2* [32], as is depicted in above (Fig. 8).

The main drawback of the LOF algorithm is its time complexity. Computation of the LOF has the complexity $O(n^2)$, because it requires computing pairwise distances between all data point.

b) One Class SVM Algorithm: One class SVM is an algorithm that identifies regions of space by their support vectors, of which there are far fewer than data points. The one class SVM algorithm solves the following optimization problem to compute the support vectors:

$$\min_{w \in F, \xi \in R, \rho \in R} \frac{1}{2} w^2 + \frac{1}{mv} \sum_{i=1}^{m} \xi_i - \rho \quad (3)$$

subject to

$$w . \Phi_{(x_i)} \geq \rho - \xi_i \; \forall i = 1, \ldots, m$$

$$\xi_i \geq 0 \; \forall i = 1, \ldots, m$$

Where $\Phi : R^n \; 1 \rightarrow F$ is a nonlinear mapping from data space R^n to feature space F, ξi is a slack variable one for every data

sample, ρ is a distance from the hyper-plane to the origin in feature space, and v is the expected fraction of data samples outside the estimated support. The one class SVM algorithm depends on the choice of two parameters: v and σ. The parameter v controls the sensitivity of the model. More precisely it controls the ratio of outliers in the data. The parameter σ controls the number of support vectors. The lower value of σ leads to "remembering" the training dataset and the model over-fits the larger value leads to oversimplifying dependencies in the dataset, in other words, it leads to high bias.

The other important decision one has to make when training a one class SVM model is a choice of kernel. The general advice is to use a redial basis function kernel (RBF kernel) [33] because usually, it performs better on different datasets. It has been experimented with different kernels, and the RBF kernel was showing the best results. RBF kernel adds one more parameter which can be adjusted to change the performance of the one class SVM algorithm. The parameter γ controls how far the influence of a single training example reaches. The smaller value of γ means that a single example influences other examples far away and the larger value means its influence is shorter.

c) Isolation Forest: Isolation Forest is a model-based approach to detect anomalies. In the context of isolation forest "isolation" means "separating an instance from the rest of the instances". The algorithm constructs trees which isolate every single instance of data. Because anomalies are different, they will be isolated faster by the algorithm, which means they will be closer to the root of a tree. To achieve this, isolation forest takes advantage of two properties of anomalies:

- They are in the minority.
- Attribute-values of anomalies significantly differ from normal samples an example of isolation can be seen in Fig. 9(a), (b).

The authors of the paper has define the path length of a point x as the number of edges and the point of traverses until it terminates in the end node.

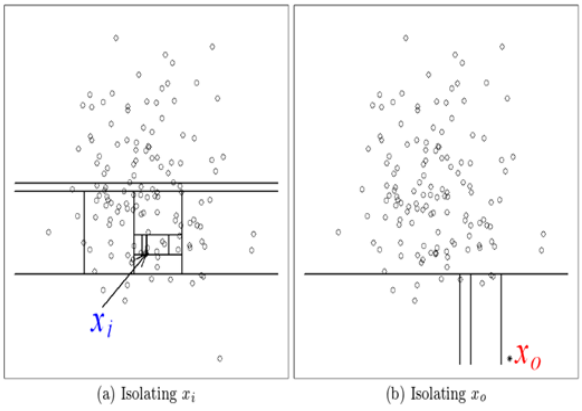


Fig. 9. (a). Shows the Isolation of a Normal Data Sample. (b) Shows the Isolation of an Anomalous Data Sample. Image is Taken from Isolation Forest [29].

There are two stages of anomaly detection with isolation forest. The first stage is a training stage. In this stage, the trees are constructed recursively until all instances are isolated or the specified tree height is reached. The theory behind growing the trees up to some height is that the utmost interest has been shown in points which have a shorter than average path distance. There is no need to grow the trees until each point is isolated. The second stage is the evaluation. The anomaly score of a sample is based on the average path length from root to its termination node.

F. Training and Validation

Cross-validation [34], is going to be used which is a common technique to estimate a test error of a model. Dataset is split and hold out a small subset of data to test the model performance and to make sure that the model does not just memorize the dataset. It is ensured that the model does not have a high FPR and also that it detects anomalies. The model is also prevented from overfitting. It means that a model is trained too much and it is fitted too close to the train set. The main sign of overfitting is that a model has the very low error on the train set, but a much higher error on a validation set. The data is split into three sets: train, validation, and test. The train set consists of normal profiles only. The validation and test sets consist of normal and anomalous profiles. All anomalous profiles are taken randomly to select half of them and to insert them into validation set. The other half is inserted into the test set. The split is the following:

- Train set—70% of normal data.
- Validation set—15% of normal data and 50% of anomalous profiles.
- Test set—15% of normal data and the other 50% of anomalous profiles.

As it has been mentioned before the data are scaled and normalized to have 0 mean and unit variance. To avoid information leak from the training data to test data, it has been learnt a mean value and variance for scaling and centering only on train data, and then scaling is applied to validation and test data. The information leak might lead to test error underestimating the actual error.

G. Model Selection

In Section III.E, it has been described the algorithms that has been tried for anomaly detection. LOF and one class SVM have both hyper-parameters that can be tuned to improve the performance. When search has been made for the optimal set of parameters for our model, a grid of search is conducted over a range of possible parameter values. If a hyper-parameter is restricted to some range of values we use that range. If a hyper-parameter is not restricted selection of some reasonable size and search inside it have been done.

The evaluation of a network anomaly detection algorithm is a very important step in showing the advantages and efficiency of the proposed method. The main challenge is to acquire labeled data to measure the performance of the model.

To select a model the following criteria is used:

1) Find a set, A, of models with the lowest False Positive Rate FPR.

2) Select a model slowest with the highest True Positive Rate TPR from the set A.

3) Find a set of models B with the FPR less than 0.01.

4) Select a model $S_{threshold}$ with the highest TPR from the set B.

5) Select the final model with the highest FPR by comparing FPR of S_{lowest} & $S_{threshold}$.

The FPR metric is very important in anomaly detection. If a model has FPR around 0.05 in a small network with 10 computers and 5 minutes profiles, if will generate 144 false reports daily. Since each report should be checked manually by a system administrator, it would consume many resources.

It has been tried to find model parameters which will have a good trade-off between FPR and TPR. The goal is to maximize TPR while at the same time keeping the FPR below 0.01 or 1%.

IV. Machine Learning Datasets

The characteristics of the dataset used for training and testing a machine learning algorithm [35] are very important for the results obtained. In fact, the dataset is so important that it completely define if the algorithm works or not, it defines its performance and its generalization power. It does not make much sense to talk about if an algorithm is good or bad without discussing the dataset used.

The dataset used in this research work was created from scratch to fulfill the requirements. In particular, it was very important to have a dataset of real malicious activities [36,37] at the same time that the normal user is also using the computer. This was achieved through a large process of configurations and infections. The setup to create the datasets consisted of a Windows 10 virtual machine running on Virtual-Box. The traffic was captured by Virtual-Box in a ".pcap file". After the capture was finished, the pcap file was processed with the Argus tool to obtain bidirectional flows.

To work with an anomaly detection algorithm, it's necessary to have a dataset which contains two main parts: normal user activity, and a mix of normal activity and malicious activity. The normal user activity is used for creating a model of normal traffic. Later on, the models are evaluated using the mix of normal traffic and malicious traffic [38].

To generate normal activity, multiple accounts in different services such as Facebook, Gmail, Dropbox, and Twitter are created [39]. All these accounts were used to generate real normal traffic, where the user creates and writes in new documents, chat with friends and synchronizes data in the cloud [40]. There is also normal activity such as visiting websites, searching for information and downloading files, including executable files. In all type of datasets the normal activities lasted several hours, up to several days.

After the normal activity was done, the computer was infected with some malware while the user keeps doing normal actions. This mixed traffic was kept for several hours also. During all the experiments all the activities done by the normal user were logged and stored. This log was later used for labeling the profiles with normal and anomalous labels [41].

A. Ground-Truth Labels Process

In this section, labeling process has been described. For the evaluation of the performance, it is needed to have ground-truth labels. The captures of network traffic [42] are split into two parts: before and after an infection. When profiles are created from the flows it is known that if it belongs to pre or post-infection part.

The normal label (label = 1) is assigned to all profiles which are created before the infection of the virtual machine run on web based adaptive data-driven networks (DDN) management and cooperative network communities [43]. A virtual machine before each capture is created which ensures that it is clean and does not contain any malicious software. The anomaly label (label = −1) is put to all profiles which are created after the infection of the virtual machine. The reason of assigning the anomaly labels this way is the assumption that everything after an infection worth detecting and that malware produces changes in behavior. It is known that not all the attacks are anomalies, but this assumption helps us better evaluate the algorithm.

V. Experimental Results

With the help of anomaly detection techniques, method designed and a good dataset generated it was possible to train and test the hypothesis by running experiments. There were five experiments in total, each one verifying a different algorithm with a different dataset. The goal of these experiments was to verifying if the hypothesis was true. Else if hypothesis was not true then it is possible to detect the infected users and to have a small number of false positives by focusing on the high-level behaviors in time. The side effect verification was to see if the method was capable of generalizing to malware which it has not seen during the training.

All the experiments in point were evaluated in two ways. First, labeling to each feature individually was used to trained models. Then second way is to use a voting mechanism to decide if the analyzed profile was anomalous or not. The voting is described in Section III.D.b.

A summary of the experiments follows:

- The first experiment uses one-class SVM trained on the normal part of the first dataset and validated on the mixed part of the first dataset.
- The second experiment uses one-class SVM trained on the normal part of the second dataset and verification on the mixed part of the second dataset (and the mixed part of the third dataset).
- The third experiment uses LOF trained on the normal part of the first dataset and validated on the mixed part of the first dataset.
- The fourth experiment uses LOF trained on the normal part of the second dataset and validated on the mixed part of the second dataset.

- The fifth experiment uses Isolation Forest trained on the normal part of the first dataset and validated on the mixed part of the first dataset.

All the experiments were run in docker container using Ubuntu 16.04, Python 3.6 and the following versions of libraries: “pandas: 0.20.3”, “numpy: 1.13.1” and “scikit-learn: 0.19.0”.

A. *The First Experiment*

In the first experiment, one class SVM algorithm is used, as it was described in Section III.E.b. this algorithm was trained with the normal part of the first dataset in Section IV, and it was validated on the mixed part of the same first dataset. The SVM algorithm used the Euclidean distance and the RBF kernel function. To train the SVM a grid search over the following parameters is used:

- *Gamma* parameter for RBF kernel: values in range $[10^{-9},10^{3}]$ with step 10^{2}.

Nu parameter for lower bound of fraction of support vectors: values in range [0.01, 0.99] with step 0.01, depicted in Fig. 10.

The training of the algorithm was performed by splitting the normal part of the first dataset into three sets: train, validation and test. This was done using the train_test_split function from the python library sklearn [44-46]. To make the data split repeatable, we specify that the random seed is 42. The mixed part of the first dataset (that contains both normal and malicious traffic) was split into two sets: validation and test. Both were generated by randomly selecting half of the profiles in that dataset.

With the sets of data defined a model for each of the twelve features in a profile (see in Section III.C) Is trained. After that all models have been trained, the best model is selected by using the validation set. The best model was selected for each feature (see in Section III.G).

The first analysis of these results is done according to each feature. Fig. 11 and Table I below shows that one class SVM achieved low FPR for each feature. It does not exceed 0.01 in any case, and for eight of the features, it is 0.0. The algorithm also has 1.0 TPR for five features. All of these five features are TCP features. It means that the malware actively uses TCP protocol to communicate and this is very anomalous compared with the normal user.

The interesting observation is that the feature Client_DestinationPort_NumberOfFlows_TCP_Established has 0.0 TPR.

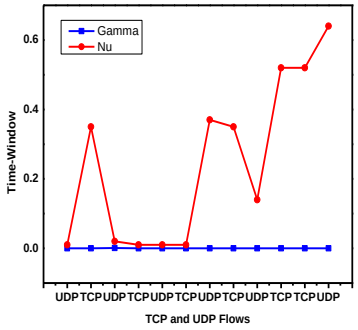


Fig. 10. Shows the Winning Parameters for One Class SVM Model in the First Experiment.

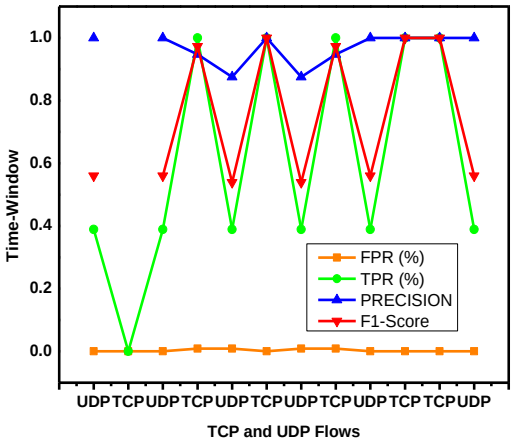


Fig. 11. Shows the Validation Results for Each of the Twelve Features, Present in below Table I.

TABLE. I. VALIDATION RESULTS FOR THE FIRST EXPERIMENT. ONE CLASS SVM ALGORITHM TRAINED ON THE NORMAL PART OF FIRST DATASET AND TESTED ON THE MIXED PART OF THE FIRST DATASET. THE VALIDATION SET CONSISTS OF SOME NORMAL PROFILES AND SOME MALICIOUS PROFILES

Twelve Features Name	**FPR%**	**TPR%**	**Precision**	**F1-Score**
1. Client_DestinationPort_TotalBytes_UDP_Established	0.0	0.389	1.0	0.560
2. Client_DestinationPort_NumberOfFlows_TCP_Established	0.0	0.0	---	---
3. Client_DestinationPort_NumberOfFlows_UDP_NotEstablished	0.0	0.389	1.0	0.560
4. Client_DestinationPort_TotalPackets_TCP_Established	0.009	**1.0**	0.947	0.973
5. Client_DestinationPort_NumberOfFlows_UDP_Established	0.009	0.389	0.875	0.539
6.Client_DestinationPort_TotalPackets_TCP_NotEstablished	0.0	**1.0**	1.0	1.0
7. Client_DestinationPort_TotalBytes_UDP_NotEstablished	0.009	0.389	0.875	0.539
8. Client_DestinationPort_TotalBytes_TCP_Established	0.009	**1.0**	0.947	0.973
9. Client_DestinationPort_TotalPackets_UDP_NotEstablished	0.0	0.389	1.0	60 0.560
10. Client_DestinationPort_NumberOfFlows_TCP_NotEstablished	0.0	**1.0**	1.0	1.0
11. Client_DestinationPort_TotalBytes_TCP_NotEstablished	0.0	**1.0**	1.0	1.0
12. Client_DestinationPort_TotalPackets_UDP_Established	0.0	0.389	1.0	0.560

After running the validation using grid search on the parameters, the best model is selected by using the methodology explained in Section III.G. The winning parameters for these one class SVM models were discussed below:

After the winner parameters were selected during the validation phase, it can be tested the model for its generalization power using the test set Fig. 12 below shows the results on the test set. The results are very similar to the validation set results, which means that there is a good generalization. This test set contains the same malware that was used in the validation set. After testing the one class SVM using our first method of evaluation, the evaluation of the results is done by applying a majority voting mechanism on the output generated for each feature to decide if the profile was anomalous or not. As in the previous testing, it has been tested the majority voting model on the test set from the first dataset. Our approach to a majority voting is described in Section III.D.b. Therefore, the results are shown below in Table II.

Table II shows that the majority votes among the models produces a good result on the test set. The TPR of 0.444 or 44% may seem low, but this is the final result for all the profiles in time, with a 0 FPR and a 100% precision. These results mean that on average the one anomaly will be detected out of three with probability 99.9%. It also means that if it is used the five-minute time-windows to create profiles, it can raise the alarm after first 15 minutes after a malware becomes active. Considering that the evaluation is per profile, it is believe that these results are very good in this area.

B. *The Second Experiment*

In the second experiment, the one class SVM is used, as it was described in (section III.E.b). this algorithm was trained with the normal part of the second dataset in (section IV.), and it was validated on the mixed part of that same dataset. The SVM algorithm used the Euclidean distance and the RBF kernel function.

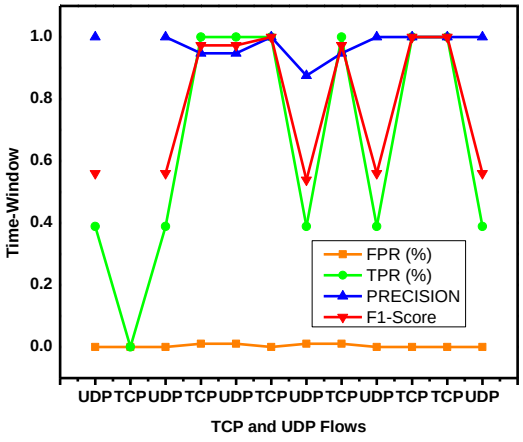


Fig. 12. Shows the Results on the Test Set Experiment, the Results are very Similar to the Validation Set Results, Dipict in (Fig. 11 above) which means that there is Good Generalization.

TABLE. II. TEST RESULTS FOR THE FIRST EXPERIMENT USING MAJORITY VOTING. THE MAJORITY VOTING TO THE TWELVE RESULTS ON EACH PROFILE IS APPLIED TO GET THE FINAL DECISION ABOUT A PROFILE. THE TEST SET FOR THE FIRST DATASET WAS USED FOR THE TESTING. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH THE MALWARE

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.444	1.0	0.615

To train the SVM model a grid search over the following parameters is used:

- Gamma parameter for RBF kernel: values in range $[10^{-9}, 10^{3}]$ with step 10^{2}.
- Nu parameter for lower bound of fraction of support vectors: values in range [0.01, 0.99] with step 0.01.

The training of the algorithm was performed by splitting the normal part of the dataset into three sets: train, validation and test. This was done using the train_test_split function from the python library sklearn. To make the data split repeatable, it is specified that the random seed is 42. The mixed part of the second dataset (that contains both normal and malicious traffic) was split into two halves: the first half was added to the validation set and the second half was added to the test set. Both were generated by randomly selecting half of the profiles in that dataset.

With the sets of data defined a model is trained for each of the twelve features in a profile (see in Section III.C). After having trained all models, the best model is selected by using the validation set. The best model was selected for each feature (see in Section III.G). After the validation is finished the winner model is selected.

The first analysis of the results is done with regard to each feature, Fig. 13, shows that the models achieved low FPR for each feature. It does not exceed 0.01 in any case, and for seven of the features, it is 0.0. However, TPR is lower than in the first experiment. This might be caused by the nature in which this malware communicates. The mixed capture of the second dataset was generated using [47]. Dark-VNC virtual network computing is used to silently control the computer of a victim, and it does not generate much additional traffic.

This is an example of how the anomaly detection technique might help better understand the communication details of malware. By analyzing results, the analyst could very fast see which protocols are used by malware, if it generates many connections and sends much information. The winning parameters for the best one class SVM models were discussed below:

After the winner parameters were selected during the validation phase, it can be tested the model for its generalization power using the test set. See Fig. 14 and Table III below shows the results on the test set. This test set contains the same malware that was used in the validation set.

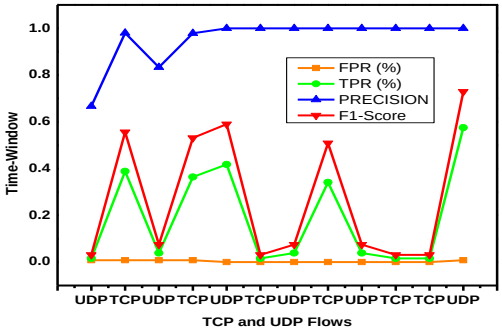


Fig. 13. Shows the Results that Model Achieved Low FPR for Each Feature.

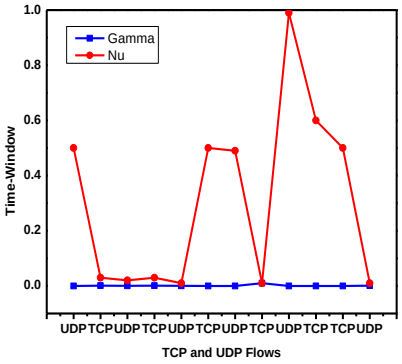


Fig. 14. Shows the Results on the Test Set, this Test Set Contains the same Malware that was used in the Validation Set, Present in Table III.

TABLE. III. THE WINNING PARAMETERS FOR ONE CLASS SVM MODELS IN THE SECOND EXPERIMENT. THESE PARAMETERS WERE OBTAINED USING THE VALIDATION SET OF THE SECOND DATASET

Features Name	***Gamma***	***Nu***
Client_DestinationPort_TotalBytes_UDP_Established	10^{-9}	0.50
Client_DestinationPort_NumberOfFlows_TCP_Established	10^{-3}	0.03
Client_DestinationPort_NumberOfFlows_UDP_NotEstablished	10^{-4}	0.02
Client_DestinationPort_TotalPackets_TCP_Established	10^{-3}	0.03
Client_DestinationPort_NumberOfFlows_UDP_Established	10^{-4}	0.01
Client_DestinationPort_TotalPackets_TCP_NotEstablished	10^{-9}	0.50
Client_DestinationPort_TotalBytes_UDP_NotEstablished	10^{-8}	0.49
Client_DestinationPort_TotalBytes_TCP_Established	10^{-2}	0.01
Client_DestinationPort_TotalPackets_UDP_NotEstablished	10^{-9}	0.99
Client_DestinationPort_NumberOfFlows_TCP_NotEstablished	10^{-7}	0.60
Client_DestinationPort_TotalBytes_TCP_NotEstablished	10^{-9}	0.50
Client_DestinationPort_TotalPackets_UDP_Established	10^{-3}	0.01

It is wanted to test the generalization power of the models even further, and the mixed capture is used from the third dataset in Section IV to test the performance of the models. Since the mixed capture from the third dataset contains traffic from a different type of malware and it was not used during the selection process of models, this evaluation could be a good estimate for the real error. See Fig. 15 below contains the results obtained for the mixed capture from the third dataset using the models trained on the second dataset:

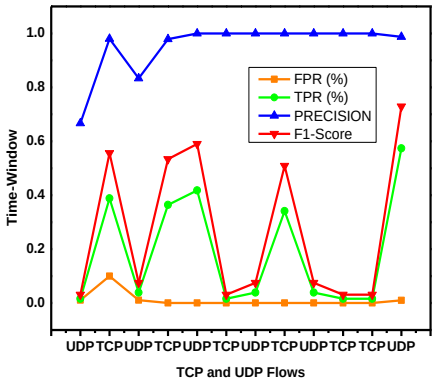


Fig. 15. Shows the Results Obtained for the Mixed Capture from the Third Dataset using the Models Trained on the Second Dataset.

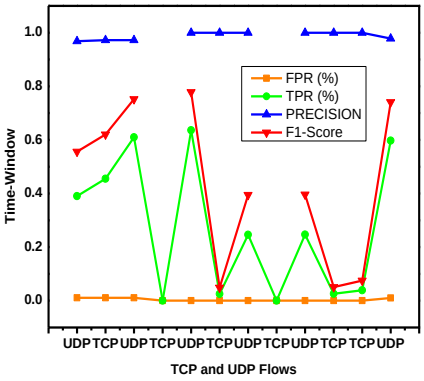


Fig. 16. Shows the Results of different Features Contributing to the Detection of different Types of Infected Malware.

These results are first analyzed per each feature. Fig. 16 below shows that different features contribute to the detection of different types of malware.

For example, on the one hand, the feature Client_DestinationPort_NumberOfFlows_UDP_NotEstablished contributes a lot to the detection of Simba malware and on the other hand the same feature does not contribute to the detection.

After testing the one class SVM using the first method of evaluation, it is now evaluated the results of applying a majority voting mechanism on the output generated for each feature to decide if the profile was anomalous or not. As in the previous testing, the majority voting model is tested on the test set from the second dataset. The approach to a majority voting is described in Section III.D.b and the results are shown in above Table II.

Table IV below shows that the majority voting among the models produces FPR of 0.0. However, the TPR is very low, which could be explained by the difficulty of detecting this particular malware. In the future it is wanted to experiment with approaches other than majority voting, for example, neural network will be trained to summarize the outputs of the twelve models into one final result.

Majority of voting to classify the profiles was generated during the infection with the Simba malware in the third dataset. The results are shown in below Table V.

TABLE. IV. THE SECOND EXPERIMENT. THE MAJORITY VOTING IS APPLIED TO THE TWELVE RESULTS TO GET THE FINAL DECISION ABOUT A PROFILE. THE RESUTS ARE SHOWN ON THE TEST SET FOR THE SECOND DATASET AND THE FIRST MALICIOUS PART. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH THE FIRST MALWARE

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.031	1.0	0.060

TABLE. V. THE SECOND EXPERIMENT THAT IS APPLIED MAJORITY VOTING TO THE TWELVE RESULTS TO GET THE FINAL DECISION ABOUT A PROFILE. THE RESULTS ARE SHOWN ON THE TEST SET FOR THE SECOND DATASET AND THE SECOND MALICIOUS PART. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH THE SECOND MALWARE

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.233	1.0	0.378

The TPR of 0.233 or 23% is lower than in the first experiment. However, this result means that the algorithm will detect one anomalous profile out of five with probability 99.9%. If we use five minutes time-windows to create profiles, the alarm will be raised during the first 25 minutes after a malware becomes active.

C. The Third Experiment

In the third experiment, the use of the LOF algorithm has been done, as it was described in Section III.E.a. This algorithm was trained with the normal part of the first dataset see in Section IV and it was validated on the mixed part of that same first dataset. The LOF algorithm used the Euclidean distance to compute density estimation.

To train the LOF, a grid search over the following parameters is used:

- *k* parameter for number of neighbors: values in range [1 −10] with a step 1.
- *Contam* parameter for contamination rate (the ratio of anomalies): values in range [0.01, 0.1] with a step approximately 0.002.

The training of the algorithm was performed by splitting the normal part of the dataset into three sets: train, validation and test. This was done using the train_test_split function from the python library sklearn. To make the data split repeatable, it is specified that the random seed is 42. The mixed part of the first dataset (that contains both normal and malicious traffic) was split into two halves: the first half was added to the validation set and the second half was added to the test set. Both were generated by randomly selecting half of the profiles in that dataset. With these sets of data defined the model has been trained a model for each of the twelve features in a profile (see in Section III.C); after that all trained models, select the best model by using the validation set. The best model was selected for each feature (see in Section III.G).

After the validation is finished the winner model is selected. The first analysis of these results is done according to each feature. Fig. 17 below shows that LOF achieved low FPR for each feature. It exceeds 0.01 only for one profile and only by 0.001, and for other eleven of the features, it is 0.0. The algorithm also has 1.0 TPR for three features. All of these three features are TCP features. It means that the malware actively uses TCP protocol to communicate and this is very anomalous compared with the normal user.

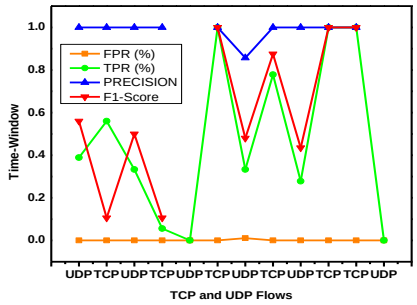


Fig. 17. Shows the Results of LOF Algorithm Achieved Low FPR for each Feature.

After running the validation using grid search on the parameters, the best model is selected by using the methodology explained in Section III.G. The winning parameters for these LOF models are shown in Fig. 18 and Table VI.

After the winner parameters were selected during the validation phase, it can be tested the model for its generalization power using the test set given in Fig. 19 shows the results on the test set. The results for TPR are similar to the results obtained on the validation set, but the FPR is slightly higher for nine profile features. This test set contains the same malware that was used in the validation set.

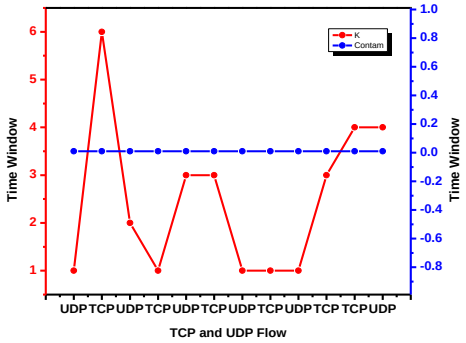


Fig. 18. Shows the Winning Parameters of LOF Model in the Third Experiment, Present in Table VI.

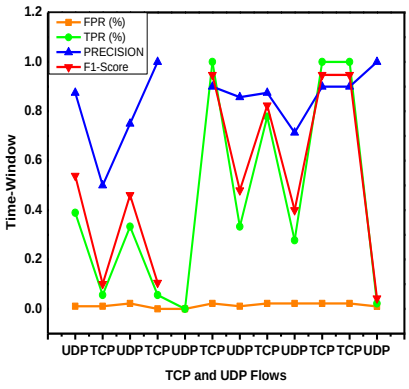


Fig. 19. Shows Results on the Test Set for the First Dataset, the Test Set Contains Normal Profiles and Profiles Created During the Infection with the Malware.

TABLE. VI. THE WINNING PARAMETERS FOR LOF MODELS IN THE THIRD EXPERIMENT. THESE RESULTS WERE OBTAINED USING THE VALIDATION SET OF THE FIRST DATASET

Features Name	*k*	*Contam*
Client_DestinationPort_TotalBytes_UDP_Established	1	0.01
Client_DestinationPort_NumberOfFlows_TCP_Established	6	0.01
Client_DestinationPort_NumberOfFlows_UDP_NotEstablished	2	0.01
Client_DestinationPort_TotalPackets_TCP_Established	1	0.01
Client_DestinationPort_NumberOfFlows_UDP_Established	3	0.01
Client_DestinationPort_TotalPackets_TCP_NotEstablished	3	0.01
Client_DestinationPort_TotalBytes_UDP_NotEstablished	1	0.01
Client_DestinationPort_TotalBytes_TCP_Established	1	0.01
Client_DestinationPort_TotalPackets_UDP_NotEstablished	1	0.01
Client_DestinationPort_NumberOfFlows_TCP_NotEstablished	3	0.01
Client_DestinationPort_TotalBytes_TCP_NotEstablished	4	0.01
Client_DestinationPort_TotalPackets_UDP_Established	4	0.01

TABLE. VII. THE THIRD EXPERIMENT WHICH IS APPLIED MAJORITY VOTING TO THE TWELVE RESULTS TO GET THE FINAL DECISION ABOUT A PROFILE. IS SHOWN IN RESULTS ON THE TEST SET FOR THE FIRST DATASET. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH THE MALWARE

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.333	1.0	0.496

After testing the LOF using the first method of evaluation, it is now evaluated the results of applying a majority voting mechanism on the output generated for each feature to decide if the profile was anomalous or not. As in the previous testing, the majority voting model is tested on the test set from the first dataset. The approach to a majority voting is described in Section III.D.b. The results are shown in below Table VII.

The above Table II shows that the majority voting among the models produces a good result on the test set. The TPR of 0.333 or 33.3% may seem low, but this is the final result for all the profiles in time, with a 0 FPR and a 100% precision. These results mean that on average it will be detected one anomaly out of three with probability 99.9%. It also means that if the five-minute time-windows is used to create profiles, it can raise the alarm during the first 15 minutes after a malware becomes active. Considering that the evaluation is per profile, it is believed that these results are very good in the area.

D. The Fourth Experiment

The first analysis of the results is done with regard to each feature. Fig. 20 below shows that the models achieved low FPR for each feature. It does not exceed 0.01 in any case, and for seven of the features, it is 0.0. However, TPR is lower than in the first experiment. This might be caused by the nature in which this malware communicates. The mixed capture of the second dataset was generated using Dark-VNC malware.

Dark virtual network computing (Dark-VNC) is used to silently control the computer of a victim, and it does not generate much additional traffic.

After running the validation using grid search on the parameters, which are selected the best model using the methodology explained in Section III.G. The winning parameters for these LOF models are shown in Fig. 21 and Table VIII.

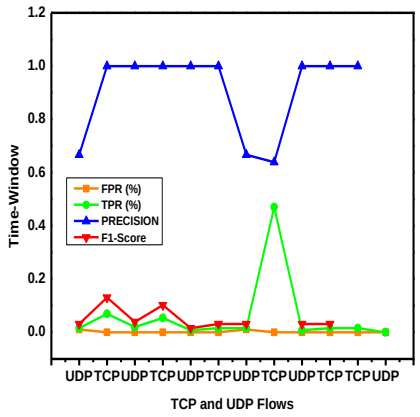


Fig. 20. Shows that the Models Achieved Low FPR for each Feature it doesn't Exceed 0.01 Score in any Case, and for Seven of the Feature, it Takes 0.0 Score.

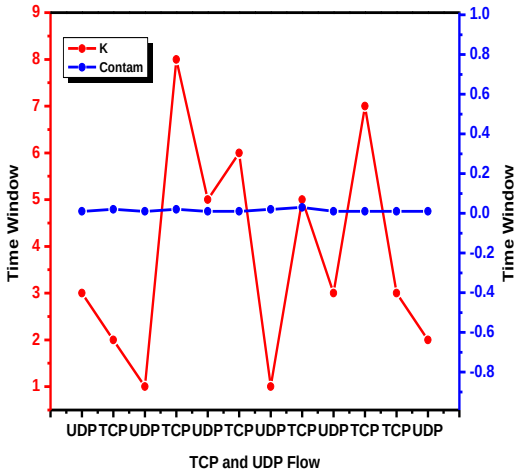


Fig. 21. Results Shows the Wining Parameters for LOF Model in Fourth Experiment, Present in Table VIII.

TABLE. VIII. THE WINNING PARAMETERS FOR LOF MODELS IN THE FOURTH EXPERIMENT THESE RESULTS WERE OBTAINED USING THE VALIDATION SET OF THE SECOND DATASET

Feature Name	*K*	*Contam*
Client_DestinationPort_TotalBytes_UDP_Established	3	0.01
Client_DestinationPort_NumberOfFlows_TCP_Established	2	0.02
Client_DestinationPort_NumberOfFlows_UDP_NotEstablished	1	0.01
Client_DestinationPort_TotalPackets_TCP_Established	8	0.02
Client_DestinationPort_NumberOfFlows_UDP_Established	5	0.01
Client_DestinationPort_TotalPackets_TCP_NotEstablished	6	0.01
Client_DestinationPort_TotalBytes_UDP_NotEstablished	1	0.02
Client_DestinationPort_TotalBytes_TCP_Established	5	0.03
Client_DestinationPort_TotalPackets_UDP_NotEstablished	3	0.01
Client_DestinationPort_NumberOfFlows_TCP_NotEstablished	7	0.01
Client_DestinationPort_TotalBytes_TCP_NotEstablished	3	0.01
Client_DestinationPort_TotalPackets_UDP_Established	2	0.01

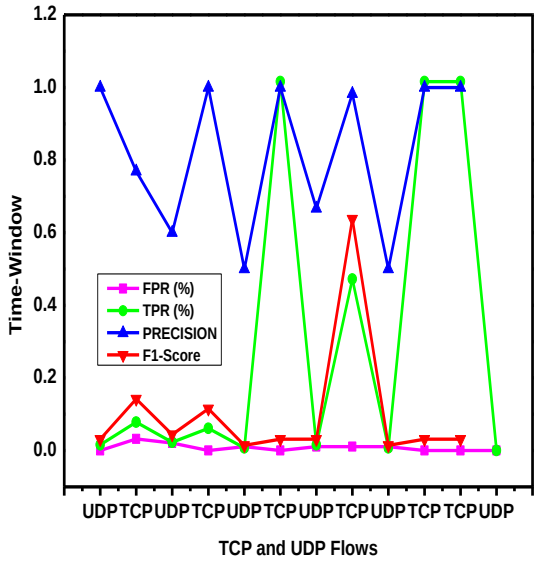


Fig. 22. The Results Shows that the TPR is Similar to the Results Obtained on the Validation Set, but the FPR is Slightly Higher for Five Profile Features.

After the winner parameters were selected during the validation phase, it can be tested the model for its generalization power using the test set. Fig. 22 shows the results on the test set. The results for TPR are similar to the results obtained on the validation set, but the FPR is slightly higher for five profile features. This test set contains the same malware that was used in the validation set.

It is wanted to test the generalization power of the models even further, and it is used the mixed capture from the third dataset (Section IV) to test the performance of the models. Since this mixed capture contains traffic from a different type of malware and it was not used in the during the selection process of models, this evaluation could be a good estimate for the real error. Fig. 23 contains the results obtained for the mixed capture from the third dataset using the models trained on the second dataset:

After testing the LOF using the first method of evaluation, it is now evaluated the results of applying a majority voting mechanism on the output generated for each feature to decide if the profile was anomalous or not. As in the previous testing, It is also apply majority voting to classify the profiles which were generated during the infection with the Simba malware in the third dataset. The results are shown in Table IX, as in the second experiment results of the majority, voting are not satisfactory, and it is going to address this problem in the future work.

E. The Fifth Experiment

In this experiment, it has been tried to use isolation forest on our data. The results were surprising because it was not able to train a model with any true positive detection. The isolation forest is shown to out-perform many algorithms including LOF and one class SVM [48, 49].

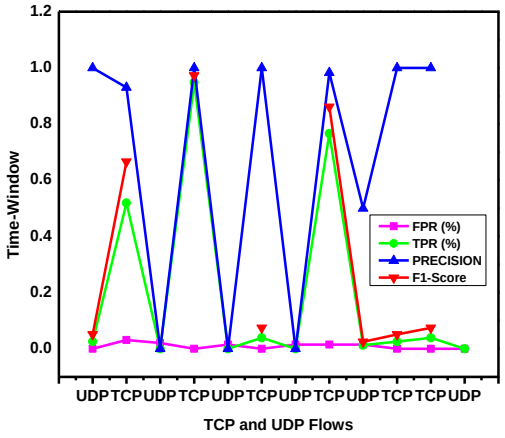


Fig. 23. Contains the Results Obtained for the Mixed Capture from the Third Dataset using the Models Trained on the Second Dataset.

TABLE. IX. THE FOURTH EXPERIMENT WHICH IS APPLIED MAJORITY VOTING TO THE TWELVE RESULTS TO GET THE FINAL DECISION ABOUT A PROFILE. RESULTS ARE SHOWN ON THE TEST SET FOR THE THIRD DATASET. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH SIMBA MALWARE

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.0	---	---

TABLE. X. THE FIFTH EXPERIMENT WHICH IS APPLYED MAJORITY VOTING TO THE TWELVE RESULTS TO GET THE FINAL DECISION ABOUT A PROFILE. RESUTLS ARE SHOWN ON THE TEST SET FOR THE SECOND DATASET. THE TEST SET CONTAINS NORMAL PROFILES AND PROFILES CREATED DURING THE INFECTION WITH DARKVNC MALWARE.

Feature Name	FPR%	TPR%	Precision	F1-Score
Majority voting	0.0	0.0	---	---

It could be caused by the nature of anomalies in the dataset. As it is described in Section III.C, each feature of a profile is represented by a vector. When it is detected anomalies among one feature of a profile, it means that it is detected anomalous vectors.

Anomalies in the datasets are reflected in vector components which are irrelevant during the training. They are irrelevant because normal data have only 0 values in these components and these components do not contribute to model training, the final results are shown in Table X.

When it is run interference on a testing profile, the model does not use these components to isolate the profile faster.

As a result, the profile is labeled as normal by the model. Another reason for the poor results could be a mistake in the way it has been trained the isolation forest model. It will be investigated more closely this issue in the future.

VI. ANALYSIS OF RESULTS

The experiments proposed in our analysis try to find how the anomaly detection algorithms may work in a realistic setup where a normal user is infected at the same time that they continue to work. In this sense, this is new computer network testing work in the security area that publishes results using a mixed dataset of real normal actions and real malware actions.

Our experiments were designed, so they were trained with real users and tested with real malware. They were also designed to detect if a profile is anomalous and not the IP address of a user. Finally, the time-window of the profiles is five minutes, which also may affect the algorithm if changed.

Using the one class SVM, it was possible to obtain a good TPR of 44% with 0% FPR. Since these results are per five-minute profile, it means that the algorithm will, out of three anomalous profiles, detect one with 100% probability (or 2 out of 5). It also means that there will be a detection at most every 15 minutes. Moreover, these results are based on a majority voting mechanism, which is not considered to be the best way of improving the results.

In particular, there are some individual features that may have better results under specific circumstances. In the first experiment, six out of twelve features reach 100% TPR with 0% FPR. In the case of the Local Outlier Factor algorithm, the results are very similar, with an average of 33% TPR detection and 0% FPR using majority voting. This means that LOF can also detect one profile correctly out of three anomalous profiles with 100% probability. The detection time is similar to one-class SVM: one anomalous profile every 15 minutes. If the detection of profiles is not done with majority voting, then LOF can reach 77.8% TPR with 2% FPR by using the feature called Client_DestinationPort_TotalBytes_TCPEstablished.

LOF also had very good results on the third experiment that used the first dataset. In this case the algorithm can have a 100% TPR, but only at the expense of a 2% FPR. The good part was that these results were obtained with three different features:

client_DestinationPort_NumberOfFlows_TCP_NotEstablished

client_DestinationPort_TotalBytes_TCP_NotEstablished,

client_DestinationPort_TotalPackets_TCP_NotEstablished.

The LOF algorithm also had good results on the fourth experiment, which used the second dataset. In this case, the algorithm achieved 47.2% TPR with a 1.1% FPR.

These results were obtained also with the feature Client_DestinationPort_TotalBytes_TCP_Established.

In the fourth experiment, with LOF on the third dataset, the algorithm obtained a very good 94.8% TPR with 0% FPR with the feature client_DestinationPort_TotalPackets_TCP_Establis.

VII. Conclusions

The detection of attacks and malware using anomaly detection techniques is a very well-known topic in the area of artificial intelligence and machine leaning. This study proposes a new perspective on the problem by analyzing the behavioral features of users in the network and by applying a high-level detection on features in time. By using a completely novel dataset and known anomaly detection methods, promising results can be obtained.

To the best of our knowledge, this research work presented the new anomaly detection method where users were profiled using their network traffic to create behavioral features, and these features were analyzed from different perspectives. The presented anomaly detection method was based on high-level view of the global network traffic by generating behavioral profiles of the activity of the users inside fixed time-windows. The new profiles of the users were compared to the past profiles to classify them as anomalous or normal. Our approach is different from other anomaly detection algorithms because the model user behavior by combining detailed features that describe all actions of the user from different perspectives.

It is classified profiles by comparing each feature with the same feature in other past profiles. The decision on whether there was an anomaly or not was taken for each feature and then the final label of the profile was decided by majority of voting. The anomalies along with each feature were found using well known algorithm local outlier factor (LOF) and one class Support Vector Machine (one class-SVM).

To evaluate our approach it is needed the data from real normal users and the data from real network traffic infection. The produced datasets were unique because they contain real malware activities at the same time that the real normal user was using the computer. The datasets were made open to the public and feature research.

The datasets were used to test and evaluate how our approach would detect different types of malware. The multiple experiments show that our approach could help in reducing the number of false positive alarms while at the same time being effective in detecting true anomalies. In multiple experiments, it is possible to detect one out of three anomalous profiles with 99% success, and it had 0% false alarms.

Even though the results are satisfactory, there is much research to be done. One of the problems which want to be worked in the future is to solve how to combine twelve different results in order to get the final decision. It would like to be to experiment with training another model which would accept the output of the twelve models described in this work and give the final decision. If it is provided enough data during the training, it may be possible that such a model could help find some non-oblivious relationship between data.

Also, our approach lacks the very important process of updating the model of normal behavior in order to adopt to new network traffic. Since the behavior of user might change with time by time, it is needed to find a way how to keep our models up-to-date. For this, it is needed to create large datasets which would cover an extended period in different situations.

VIII. Future Directions

A very promising research direction it may be worked on the usage of anomaly detection methods in to the Internet of Things (IOT) devices and scheduling based on mobile edge computing [50,51] such as: IP cameras, thermostats, printers mobile users, mobile devices and multiple base stations etc. The number of attacks on IOT devices is growing as well as the amount of malware designed to target these IOT devices. Our approach could be useful in protecting IOT devices because the traffic from these devices is far more stable than a human computer, and therefore it changes less diversity, and the results from an anomaly detection method may be easier to obtain.

ACKNOWLEDGMENT

This paper is supported by National Natural Science Foundation (NSFC) of China under grant numbers 61572095, 61877007 and 61802097. Conflicts of Interest: The authors declare no conflict of interest.

REFERENCES

[1] Xiao, F., Lin, Z., Sun, Y. and Ma, Y., 2019. Malware Detection Based on Deep Learning of Behavior Graphs. Mathematical Problems in Engineering, 2019.

[2] Pandey, S.K., 2019. Design and performance analysis of various feature selection methods for anomaly-based techniques in intrusion detection system. Security and Privacy, 2(1), p.e56.

[3] Koning, R., Buraglio, N., de Laat, C. and Grosso, P., 2018. CoreFlow: Enriching Bro security events using network traffic monitoring data. Future Generation Computer Systems, 79, pp.235-242.

[4] Wang, W., Zhang, X., Gombault, S. and Knapskog, S.J., 2009, December. Attribute normalization in network intrusion detection. In 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks (pp. 448-453). IEEE.

[5] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41, no. 3 (2009): 15.

[6] Tahir, M., Li, M., Ayoub, N. and Aamir, M., 2019. Efficacy Improvement of Anomaly Detection by using Intelligence Sharing Scheme. Applied Sciences, 9(3), p.364.

[7] Patcha, A. and Park, J.M., 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer networks, 51(12), pp.3448-3470.

[8] Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J. and Dokas, P., 2004. Minds-minnesota intrusion detection system. Next generation data mining, pp.199-218.

[9] Breunig, M.M., Kriegel, H.P., Ng, R.T. and Sander, J., 2000, May. LOF: identifying density-based local outliers. In ACM sigmod record (Vol. 29, No. 2, pp. 93-104). ACM.

[10] Ertoz, L., Lazarevic, A., Eilertson, E., Tan, P.N., Dokas, P., Kumar, V. and Srivastava, J., 2003, July. Protecting against cyber threats in networked information systems. In Battlespace Digitization and Network-Centric Systems III (Vol. 5101, pp. 51-57). International Society for Optics and Photonics.

[11] Hubballi, N. and Suryanarayanan, V., 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. Computer Communications, 49, pp.1-17.

[12] Zhang, M., Xu, B. and Gong, J., 2015, December. An anomaly detection model based on one-class svm to detect network intrusions. In 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN) (pp. 102-107). IEEE.

[13] Xu, K., Zhang, Z.L. and Bhattacharyya, S., 2005. Reducing Unwanted Traffic in a Backbone Network. SRUTI, 5, pp.9-15.

[14] Mahoney, M.V., 2003, March. Network traffic anomaly detection based on packet bytes. In Proceedings of the 2003 ACM symposium on Applied computing (pp. 346-350). ACM.

[15] Siraj, S., Gupta, A. and Badgujar, R., 2012. Network simulation tools survey. International Journal of Advanced Research in Computer and Communication Engineering, 1(4), pp.199-206.

[16] Pannell, G. and Ashman, H., 2010. Anomaly detection over user profiles for intrusion detection.

[17] Benevenuto, F., Rodrigues, T., Cha, M. and Almeida, V., 2009, November. Characterizing user behavior in online social networks. In Proceedings of the 9th ACM SIGCOMM conference on Internet measurement (pp. 49-62). ACM.

[18] Wagner, C., Mitter, S., Körner, C. and Strohmaier, M., 2012, April. When Social Bots Attack: Modeling Susceptibility of Users in Online Social Networks. In # MSM (pp. 41-48).

[19] NetFlow, C.I., 2006. Introduction to cisco ios netflow a technical overview. White Paper, Last updated: February. (accessed on 19 May 2019).

[20] Botros, S.M., Diep, T.A. and Izenson, M.D., Visa International Service Association, 2013. Synthesis of anomalous data to create artificial feature sets and use of same in computer network intrusion detection systems. U.S. Patent 8,527,776.

[21] Veres, G. and Loop, S., Exinda Networks Pty Ltd, 2019. Method and system for triggering augmented data collection on a network based on traffic patterns. U.S. Patent Application 10/193,808.

[22] Chandrasekaran, B., Srinivas, A. and Zafer, M., NYANSA Inc, 2019. System and method for using real-time packet data to detect and manage network issues. U.S. Patent Application 10/230,609.

[23] Moustafa, N., Hu, J. and Slay, J., 2019. A holistic review of Network Anomaly Detection Systems: A comprehensive survey. Journal of Network and Computer Applications, 128, pp.33-55.

[24] Postel, J., 1981. Transmission control protocol (No. RFC 793).

[25] Pearson, K., 1901. Principal components analysis. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 6(2), p.559.

[26] Sipola, T., Juvonen, A. and Lehtonen, J., 2012. Dimensionality reduction framework for detecting anomalies from network logs. Engineering Intelligent Systems, 20(1/2).

[27] Huang, T., Sethu, H. and Kandasamy, N., 2016. A new approach to dimensionality reduction for anomaly detection in data traffic. IEEE Transactions on Network and Service Management, 13(3), pp.651-665.

[28] Dreiseitl, S., Osl, M., Scheibböck, C. and Binder, M., 2010. Outlier detection with one-class SVMs: an application to melanoma prognosis. In AMIA Annual Symposium Proceedings (Vol. 2010, p. 172). American Medical Informatics Association.

[29] Liu, F.T., Ting, K.M. and Zhou, Z.H., 2008, December. Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

[30] Hawkins, D.M., 1980. Identification of outliers (Vol. 11). London: Chapman and Hall.

[31] Breunig, M.M., Kriegel, H.P., Ng, R.T. and Sander, J., 1999, September. Optics-of: Identifying local outliers. In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 262-270). Springer, Berlin, Heidelberg.

[32] Han, J., Pei, J. and Kamber, M., 2011. Data mining: concepts and techniques. Elsevier.

[33] Singla, M.H.S.C.S. and Shen, Y., Kernel Selection and Dimensionality Reduction in SVM Classification of Autism Spectrum Disorders.

[34] James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

[35] Haria, S., 2019. The growth of the hide and seek botnet. Network Security, 2019(3), pp.14-17.

[36] Xiong, H., Malhotra, P., Stefan, D., Wu, C. and Yao, D., 2009, December. User-assisted host-based detection of outbound malware traffic. In International Conference on Information and Communications Security (pp. 293-307). Springer, Berlin, Heidelberg.

[37] Cabaj, K., Gawkowski, P., Grochowski, K. and Osojca, D., 2015. Network activity analysis of CryptoWall ransomware. Przeglad Elektrotechniczny, 91(11), pp.201-204.

[38] Ken, F.Y. and Harang, R.E., 2017, October. Machine learning in malware traffic classifications. In MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM) (pp. 6-10). IEEE.

[39] Awad, Y., Nassar, M. and Safa, H., 2018, May. Modeling Malware as a Language. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[40] Smith, Z.M., 2016. Building an Adaptive Cyber Strategy. Air Command and Staff College, Air University Maxwell Air Force base united States.

[41] Fu, Y., 2017. using botnet technologies to counteract network traffic analysis.

[42] Tahir, M., Li, M., Ayoub, N., Shehzaib, U. and Wagan, A., 2018. A Novel DDoS Floods Detection and Testing Approaches for Network Traffic based on Linux Techniques. Int. J. Adv. Comput. Sci. Appl, 9, pp.341-357.

[43] Tahir, M., Li, M., Shaikh, A.A. and Aamir, M., 2017. The Novelty of A-Web based Adaptive Data-Driven Networks (DDN) Management &

Cooperative Communities on the Internet Technology. Int. J. Adv. Comput. Sci. Appl, 8, pp.16-24.

[44] Bergstra, J., Yamins, D. and Cox, D.D., 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In Proceedings of the 12th Python in science conference (pp. 13-20).

[45] Komer, B., Bergstra, J. and Eliasmith, C., 2014. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In ICML workshop on AutoML (pp. 2825-2830).

[46] Raschka, S., 2015. Python machine learning. Packt Publishing Ltd.

[47] Damshenas, Mohsen, et al. "A survey on malware propagation, analysis, and detection." International Journal of Cyber-Security and Digital Forensics, vol. 2, no. 4, 2013, p. 10+. Academic OneFile, (Accessed 19 May 2019).

[48] Liu, F.T., Ting, K.M. and Zhou, Z.H., 2010, September. On detecting clustered anomalies using SCiForest. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 274-290). Springer, Berlin, Heidelberg.

[49] Emmott, A., Das, S., Dietterich, T., Fern, A. and Wong, W.K., 2015. A meta-analysis of the anomaly detection problem. arXiv preprint arXiv:1503.01158.

[50] Naeem, M.R., Khan, M.U., Shaikh, M.T., Altaf, M., Rana, S.M. and Iqbal, M.M., 2016. Smart Network Communication Using Secure And Smart Internet of things and Fog Computing. Science International, 28(4).

[51] Zheng, X., Li, M., Tahir, M., Chen, Y. and Alam, M., 2019. Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing. IEEE Access.