Unknown Malware Detection Using Network Traffic Classification

Dmitri Bekerman, Bracha Shapira, Lior Rokach, Ariel Bar
Department of Information Systems Engineering
Ben-Gurion University of the Negev
Beer Sheva, Israel
Email: {bekerdmi@post.bgu.ac.il, bshapira@bgu.ac.il, liorrk@post.bgu.ac.il, arielba@bgu.ac.il}

***Abstract*— We present an end-to-end supervised based system for detecting malware by analyzing network traffic. The proposed method extracts 972 behavioral features across different protocols and network layers, and refers to different observation resolutions (transaction, session, flow and conversation windows). A feature selection method is then used to identify the most meaningful features and to reduce the data dimensionality to a tractable size. Finally, various supervised methods are evaluated to indicate whether traffic in the network is malicious, to attribute it to known malware “families” and to discover new threats. A comparative experimental study using real network traffic from various environments indicates that the proposed system outperforms existing state-of-the-art rule-based systems, such as Snort and Suricata. In particular, our chronological evaluation shows that many unknown malware incidents could have been detected at least a month before their static rules were introduced to either the Snort or Suricata systems.**

***Index Terms*— Network security, Network intrusion detection systems, Malware detection, Machine learning.**

I. Introduction

Modern malware software utilizes sophisticated ways to hide itself, not only from the most modern anti-malware software, but also from the most experienced IT engineers. Some malware [1] has remained under the radar for years, stealing confidential data, disrupting enterprise systems, and damaging dedicated equipment.

Many malware programs use the Internet in order to communicate with the initiator of the attack in order to receive new tasks, software updates, or to leak collected data. Yet, when such malware tries to communicate with its Command and Control (C&C) center, it most likely uses common and known network protocol to pass through firewalls. In some cases, popular web resources might be involved in malicious activities as a proxy or as part of the communication algorithm with the C&C center [2].

Malware programs are capable of hiding themselves in systems or disabling their activity when they discover attempts to detect them. Therefore, it is required to use passive systems (i.e., trusted monitoring) that can detect malicious activities on targeted machines without accessing them.

Some previous studies focused on analyzing network traffic usually by either focusing on a specific network layer or protocol [3-5], or on certain malware or malware families [6]. The behavior of various malware might be reflected in different layers or protocols, rendering these “partial” perspectives inadequate. The leading existing solutions solve tasks assigned to them with high accuracy, but find it difficult to adapt to the constant evolution of existing malware types, as well as new types of malware. Moreover—as often occurs in reality—when attackers understand that their camouflage technique is discovered, they develop a new technique that can bypass existing anti-malware mechanisms. Hereby, techniques that handle specific or known malware become irrelevant shortly after their publication.

In this study we detect malicious communication such as interaction with C&C servers in order to enable alerts about the intrusion. Our solution is based on cross-layers and cross-protocols traffic classification, using supervised learning methods. We offer a solution that can detect previously unknown malware, based on previously learned ones. Our solution is dynamically adaptive, always remaining one step ahead of attackers. These traits enable us to discover malicious activities earlier than other platforms, sometimes doing so at least one month prior to state of the art rule-based systems.

To achieve this goal, we adapted methods and techniques used for network classification. These methods are used to classify network protocols or applications running behind these protocols that are related to one or more network layers. Motivated by these techniques, we developed an approach that crosses all network layers (except the physical layer). The proposed method analyzes DNS, HTTP, and SSL protocols, and combines different network classification methods in different resolutions of network activities in order to better differentiate malicious activities against benign network traffic.

An additional strength of our approach is its ability to take into account the fact that targeted machines can be behind NAT (Network Address Translation). This setting makes it impossible to determine the ID of the targeted machine but only its sub-network. In addition, we analyzed traffic behavioral patterns rather than their payload to support handling of encrypted malicious communication with C&C.

In the empirical study conducted to evaluate our approach, we used traffic that included malware generated in different sandboxes, as well as traffic from two real enterprise networks. We trained our model on data from several network

978-1-4673-7876-5/15/$31.00 ©2015 IEEE

environments and applied it on previously unseen network traffic to evaluate our model's domain-independence. In addition, our test data contained multiple types of malware that were not included in the data used in the training of the model. The extremely high accuracy obtained by our model attests to its robustness.

We compare the results of our model to some of today's most popular rule-based Network Intrusion Detection Systems (NIDS): Snort [7] and Suricata [8]. These systems utilize a comprehensive set of rules provided by leading companies in the field such as VRT [9] and ETPro [10]. Using our new method, we were able to detect malicious activities at least one month before a deterministic detection rule was deployed in these systems. Our method can thereby be used to strengthen the detection rate of existing rule-based systems, especially by reducing the time between the first day of attack and the day of detection rule deployment.

II. Related Work

Network behavioral modeling is a popular approach for malware detection and malware family classification [12]. However, most of the existing studies (e.g., [6], [13]) focus on specific types of malware, such as Bots, or on a specific type of attack such as DoS or anomalies detection in specific protocols or network layers. Our work combines features from different layers and protocols, extracted in various resolutions, and are able to detect a variety of known and new malware.

Many studies have focused solely on abusing the DNS protocol, such as Domain Generation Algorithms [6] [14], and Fast-Flux DNS [15]. Unfortunately, malware has become sophisticated and uses legitimate sites for its malicious purposes [2], including cloud storage and web applications that may take the role of a proxy between malware and attacker. For such scenarios, solutions that are based solely on data obtained from DNS may fail to detect malicious activities.

Other approaches for network behavioral modeling of malware detection summarized network activity information from the application layer [3], [4] or analyzed structural similarities among malicious HTTP traffic traces [5]. Alternatively, real-time solutions were designed to detect known and new attacks in network traffic using attributes from IP, TCP, UDP and ICMP headers [16], [13]. They focused solely on packets rather than on including the network flow as well. Thus, they cannot handle cases where legitimate network resources are involved in a malicious communication path. Saeed and Ali [12] examined network flows. They classified known malicious families. However, they did not attempt to distinguish between malicious and benign traffic as we are in the current research.

One major shortcoming of the approaches that aimed at specific use cases was that they could not detect previously unseen malware. Our solution combines several network protocols at different network layers and different resolution of network activities, and involves machine learning methods. A major advantage of our approach is the ability to detect unknown new malware or malware families that were not previously investigated.

We present a new set of engineered network features that represent various aspects, layers and resolutions of the network traffic. Our approach is strengthened by Beigi et al. [17], who focused their study on the analysis of the relative importance of network traffic-based features generated by bots, and chose the most useful subset of features that would produce the best classification accuracy. They concluded that a "multidimensional" set of features combined from different network flow layers improves classification accuracy.

For the task of malicious network behavioral modeling, we adapted network traffic classification techniques [18], [19]. These techniques are usually applied in classifying network protocols or applications running behind the protocols that are dedicated to one or a couple of network layers. These techniques have the potential to solve difficult network management problems without involving users or hosts (passive way) on the corporate sub-network or ISP level. Today, known approaches [18], [19] classify traffic by recognizing statistical patterns in externally observed attributes about the traffic without deep inspection of the packet payload that can be encrypted or obfuscated.

Network traffic classification can be performed in two different ways:

- Packet level methods examine each packet's characteristics and application signatures.
- Flow level methods are based on the aggregation of packets to flows and extraction of characteristics and statistical analysis from the flow.

Network traffic classification can be based on different major attributes:

- Port based attributes are based on the target TCP or UDP port numbers that are assigned by the Internet Assigned Numbers Authority (IANA).
- Payload based attributes are based on signatures of the traffic at the application layer level.
- Statistical based attributes relate to traffic statistical characteristics (e.g. flow duration, idle time, packets' inter-arrival time and length). These attributes are unique for certain classes of applications and enable distinguishing different source applications from one another.

We adapt network classification techniques in the above proposed solutions to classify between malicious and benign network traffic, instead of classifying the application behind it. Our solution combines several network protocols at different network layers and different resolution of network activities, and involves machine learning methods to detect unknown new attacks. Specifically, we present a new set of engineered network features that represent various new aspects of the network traffic.

III. System Overview

In the interest of achieving our goals, we developed an intrusion detection prototype system. While we were designing our system, we were faced with a major challenge: the source network could be NATed (Network Address Translation), which implies the need to analyze such a sub-network as one resource with many activities. For example: when a few resources of a sub-network simultaneously open connections to the same remote server, it is erroneously reflected as many open connections to the server from one source. This complicates the analysis when the collected data is from a sub-network behind the NAT, due to the fact that the system would alert the whole subnetwork rather than a specific machine.

The main assumption at the base of our model is that sub-networks use well configured firewalls that block connection to/from unknown protocols, as well as ports and application layer protocols that are not in the scope of the organizational policy. Therefore, we focus our work on the most commonly used application layer protocols (DNS, HHTP and SSL).

The data that can pass through firewalls is recorded and relayed further to the global network. Our system analyzes the collected data in order to detect malicious activities and issue alerts when such activities are detected.

A. Network features

The uniqueness of our solution lies in the fact that we observe data stream analysis in four resolutions, based on Internet and Transport and Application layers, with features generated accordingly (as presented in Figure 2). Specifically, we model the following levels of traffic observations as follows:

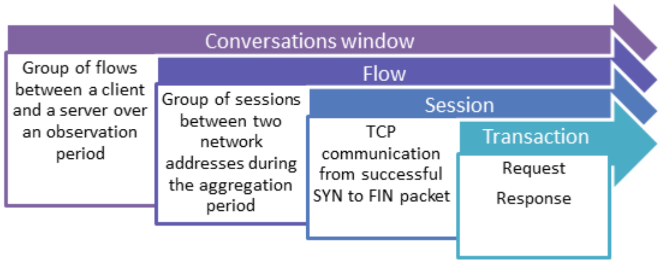
Conversations window
Group of flows between a client and a server over an observation period
Flow
Group of sessions between two network addresses during the aggregation period
Session
TCP communication from successful SYN to FIN packet
Transaction
Request
Response

Fig. 1. Data stream observation resolutions.

- Transaction – Representative of an interaction between a client and a server. It is a two-way communication: the client sends a request to the server, and the server processes the request and sends a response back to the client. We handle the following types of transactions:
  - An HTTP transaction consists of sending one request and response message between a client and a server.
  - A DNS transaction is equivalent to one session with two packets, one for the request and another for the response with the same transaction ID.
  - An SSL transaction is the aggregation of all App-data packets sent from a client to a server and vice versa after a successful handshake step and until the session ends.
- Session – A unique 4-tuple consisting of source and destination IP addresses and port numbers.
  - A TCP session begins with a successful handshake, and ends with either a timeout, or a packet with the RST or FIN flag from any of the devices.
  - A UDP session consists of all packets sent from a client to a server and from a server to a client until a defined communication idle time is reached.
- Flow – A group of sessions between two network addresses (IP pair) during the aggregation period. The aggregation period can be specified by an algorithm as the accurate period of time from the start of the first session in the flow, until the maximum idle time between two sessions. A new flow starts if the time between the end of a session (the last packet) and the start of a new session (first packet) is more than the defined idle time. The new session is then part of the new flow.
- Conversation Windows – A group of flows between a client and a server over an observation period. A conversation can be defined between two network addresses (IP pair) or a group of network resources (e.g., between two autonomous systems).

Each of the above mentioned observation levels have a number of unique properties defining its behavior in both directions of the two-way traffic. We have extracted 927 features that may model their behavior, the full list of which can be found in [20].

Table I presents a few examples of features drawn from different protocols and layers of the network traffic, while the full list is given in [20]. For each of the cumulative features we calculate the following statistics as additional features: minimum, first quartile, median, third quartile, maximum, average, standard deviation, variance and entropy.

TABLE I. EXAMPLE OF FEATURES

Level	Protocol	Feature
Transaction	HTTP	Hostname Referrer Cookie User Agent Content type
	SSL	Server name SSL version Certificate date expired
	DNS	Query name Alexa 1M rank Number of canonical names Response flags Time-to-live
Session	TCP	Destination port Packet size Number of packets with the PUSH bit set Number of out-of-order packets
Flow	TCP	Quantity of keep-alive packets in flow Packet inter-arrival time Number of port reusing packets
	IP	Destination IP IP Geo-location IP Autonomous System number
Conv. Win.	UDP	Ratio between sent and received packets
	DNS	Number of non-existent domain responses
		Number of sessions in flow Total amount of data transmitted

B. Features extraction

To enable the extraction of the above described features we have developed a dedicated feature component that processes the raw network traffic, extracts the features and provides the features as an input to the Machine Learning analyzer. Figure 3 presents a data flow diagram of the feature extractor implementation. It was implemented on top of Wireshark [21] library to extract data from the captured network traffic in tcpdump format [22]. The output of the system is feature vectors in the format of CSV files (Comma-Separated Values) that are then sent to the classification algorithms.

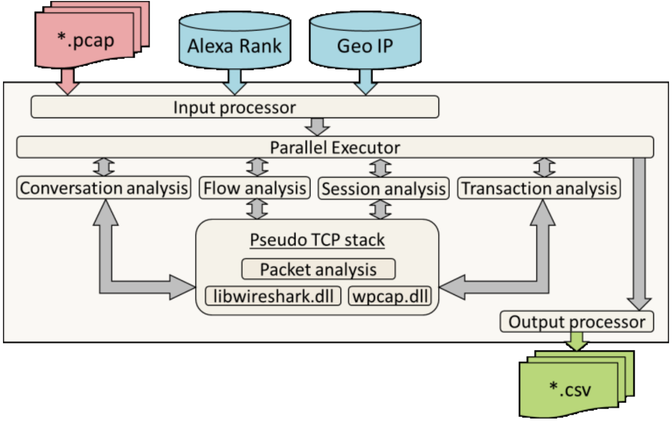

Fig. 2. Feature extraction data flow.

The input file of a feature extractor is *.pcap files and publicly available external databases such as Alexa Rank [23] and GeoIP. All data is passed to the *Input Processor* that serializes input files to corresponding objects. Thereafter, objects are passed to the *Parallel Executor* computation engine which extracts features from each of the four hierarchical network levels by reconstructing a TCP stack on top of Wireshark libraries. The features are then submitted to the output processor. Upon completion of processing an entire pcap file, the *Output Processor* generates a feature vector for each observation level and for their join. Finally, a CSV file containing feature vectors is passed to the classification algorithms.

C. Machine learning

Based on results from previous studies [3-6], [12-16] related to malware detection and network classification, we decided to choose and test three different classification algorithms including Naïve Bayes, a basic and simple model, as well as decision tree (J48) and Random Forest [24]. For feature selection we use the CFS (Correlation Feature Selection) algorithm [25]. All machine learning algorithms that we used were implemented using the Weka library [26].

Since the data is imbalanced, we had to use the True Positive Rate (TPR) and False Positive Rate (FPR) and the Area under the Curve (AUC) metrics to evaluate the performance of our detection results. Due to space limitations we mostly present results in terms of the AUC which is known to be a reliable measure for imbalanced tasks [27].

- TPR – Measurement of the proportion of actual positives (i.e., malicious network activities), which are correctly identified as such.
- FPR – Measurement of the proportion of actual negatives which are incorrectly identified as positives, i.e., the percentage of benign network activities incorrectly identified as malicious.
- AUC – The receiver operating characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of trade-offs between the TPR and FPR. Each point in the curve corresponds to a particular cut-off, having as an x-value the false positive value (specificity) and as a y-value the true positive value (sensitivity). In terms of classifier comparison, the best curve is the leftmost one, the ideal one coinciding with the y-axis. Thus, the area under the curve (AUC) can be used as a performance metric for comparing ROC curves. The AUC range is 0–1. The area under the diagonal 0.5 represents a random classifier. On the other hand, a value of 1 represents an optimal classifier.

IV. Dataset

A. Dataset sources

In order to evaluate the performance and the robustness of the proposed detection method, we used network traffic captures that included malware as well as normal (benign) network traffic that was collected by the Verint [28] and Emerging Threats [29] security companies and by us at our lab. Some of the captures were recorded in sandbox environments, others in real networks.

The following is information about our dataset that consisted of network captured tcpdump *.pcap files from different sources:

1) The Sandbox malicious captures included:

a) 2,585 records obtained from the Verint [28] sandbox.

b) 7,991 records obtained from an academic sandbox.

c) 4,167 records obtained from the Virus Total [30].

d) 23,600 records obtained from the Emerging Threats [29].

e) 12,377 malicious records collected from the web and open source community[1].

2) Benign corporate traffic was captured for 10 days in a students' lab at Ben-Gurion University.

3) Corporate traffic gathered by Verint [28] from a real network including malicious and benign traffic.

B. Labeling methods

The labeling of the data was carried out using two labeling methods (for the many experiments that we conducted): the first utilized well known NIDSs in order to enable comparison of our solution to open source systems that are available to everyone. The second was Verint's blacklist labeling.

All network traffic that was not marked as malicious was considered benign. All data except the proprietary real network traffic was labeled using both labeling methods due to the fact that data provided from the real network was sanitized and could not be labeled by commercial NDISs.

Following are details about each of the labeling methods:

1) For the labeling of the NIDSs we used Snort and Suricata SIDs, which uniquely identify the rules based on deep-packet inspection rules. We used only SIDs related to the "A Network Trojan was Detected" category:

a) Snort uses the VRT [9] rules set

[1] www.contagiodump.blogspot.com www.mlwr.com www.malware-traffic-analysis.net www.virusshare.com

b) Suricata uses the ETPro [10] rules set

2) The proprietary labels provided by Verint unite all malicious activities to 52 unique families. Those labels were based on the domains, URLs and destination IP blacklist.

Table II presents a brief comparison between the two types of labeling we used. While the corporate traffic gathered by Verint was labeled only with their labels, all other datasets were labeled with both types of labels.

We undersampled the majority class in the training data. This is known as a simple yet efficient method for mitigating the imbalance challenge [31]. Specifically, we randomly chose 150k unlabeled instances from over 50M instances and used them as benign.

TABLE II. COMPARISON OF THE TWO LABELING METHODS

	Verint labels	**Snort and Suricata labels**
	Created by cyber security experts	VRT & ETPro SIDs
Based on	Domain names and IP addresses	Deep packet inspection static rules
Existing labels	52	≈ 6500 + 9500
Seen labels	19	325 + 1180
Multi-labeling	Non generic family selected	The oldest rule selected
Example	Name: Virut Value: *.2traff.cn	SID: 2803111 Severity: ETPRO TROJAN Descr: Win32.KSpyPro.A.Checkin

C. Dataset analysis

The 18,000 malicious instances labeled with Verint labels that relate to 19 malware families are summarized in Table III.

TABLE III. MALWARE FAMILIES INSTANCES DISTRIBUTION

Name	**Count of instances**	**Percentage**
General Malware	5287	29.42%
Virut	39	0.22%
Sality	1005	5.60%
Zeroaccess	47	0.26%
Nuqel	25	0.14%
Conficker	486	2.71%
Backdoor.Paproxy	2	0.01%
APT1	337	1.88%
Shiz	377	2.10%
Mebroot	23	0.13%
Xpaj	223	1.24%
Weelsof	1145	6.38%
PowerLoader	6	0.03%
Pony Loader	5	0.03%
Pincav	125	0.70%
Oficla	165	0.92%
Matsnu	96	0.54%
Krbanker	18	0.10%
Gypthoy	2	0.01%
Darkcomet	3	0.02%
CryptoLocker	14	0.08%
Carberp	1729	9.64%
Expiro	5	0.03%
Gamarue	1	0.01%
Scar	14	0.08%
Emerging threats	6772	37.74%

For VRT [9] and ETPro [10], labeling only SIDs related to the "A Network Trojan was detected" category was used. All SIDs were associated with their creation date using the public changelog.[2] This date is a de-facto day of malware discovery. If some session contains packets marked with two or more SIDs, the oldest one is selected as the representative SID. In this way, all malicious sessions were marked not only with their rule, but also with their discovery date. To reduce resolution granularity from the day of the rule creation, we aggregated it into weeks.

To reduce the data volume to a tractable size, the number of samples was reduced using stratified sampling by malware family. Thus, the proportion between SIDs was preserved and the total number of malicious sessions was reduced from 328k to 50k. Table IV provides information about the labeled data while Figure 3 presents the count of new SIDs introduced each week and its aggregation. As can be observed, the volume and variety of new SIDs is intensified more than linear over time.

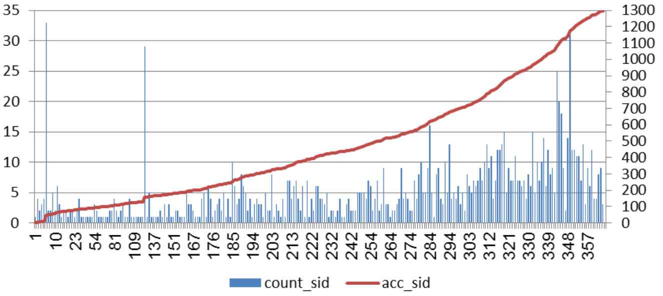

Fig. 3. VRT and ETPro sum of SIDs related to weeks during the period of 15.12.2006 – 10.12.2014.

TABLE IV. VRT AND ETPRO LABELED INSTANCES INFORMATION

	VRT (Snort)	**ETPro (Suricata)**	**Combined VRT & ETPro**
Unique rules	325	1180	1300
Existing Malicious instances	95274	281578	328749
Used Malicious Instances	12524	46046	50000
Period	15.12.2006 – 10.12.2014		
Number of weeks	110	256	260

V. EVALUATION

To evaluate our system and the effectiveness of the extracted features, we conducted experiments to test several aspects. Specifically, we first evaluated the system's abilities related to the malware families: we attributed malicious traffic to known families and detected new families. Then, we evaluated the effect of the environment on the performance; specifically, we tested several sandbox environments as well as real traffic data. In addition, we also conducted chronological experiments in which we demonstrated our system's ability to detect new malware. Specifically, in many cases our system was able to identify novel threats approximately a month prior to the appearance of the corresponding detection rules in state

[2] https://rules.emergingthreats.net/changelogs
https://support.sourcefire.com/notices/seus

of the art NIDSs. Finally, we evaluated the robustness of features across environments.

A. Malware families

1) Family classification

The goal of this experiment was twofold. Our primary goal was to evaluate the capabilities of our method by differentiating benign traffic from malicious traffic. Secondly, we wanted to evaluate the method's ability to further classify the malicious traffic to known labeled families.

Although this task has already been addressed by previous studies [12], our experiments demonstrate that our approach can achieve comparable results. We used Verint labeling families as our training data and added benign network traffic to be classified as an additional family class. We evaluated our solution only for families that had at least 10 instances to enable training. We applied the CFS algorithm for feature selection that identified the 12 network features (out of 927) presented in Table V as most effective for this task. It is apparent that even this small set of features spans across layers, protocols and observation resolution.

TABLE V. 12 Network Features Used For Family Classification

Level	Protocol	Feature
Session	TCP	Number of packets with RST flag
Flow	TCP	Number of packets sent by client with ACK flag Number of destination ports and sessions ratio
	HTTP	Median of inter-arrival time
	DNS	Query name Alexa 1M Rank Count of DNS response addresses records Count of DNS response answer records Count of DNS response authoritative records
Conv. Win.	TCP	Number of duplicate ACKs Number of Keep-alive packets
	DNS	Number of sessions and good DNS responses ratio
		Number of flows

We used 10-fold cross validation to split the data for train and test sets and applied the Random Forest, Naïve Bayes and J48 learning algorithms. As seen in Figure 4 we were able to distinguish between benign and malicious traffic with near-perfect accuracy, as well as to classify malware to their families with very high accuracy.

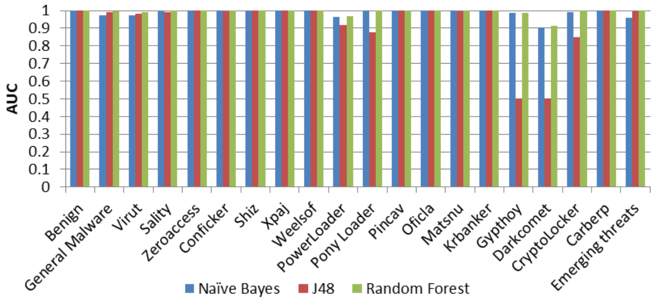

Fig. 4. Cross family classification accuracy.

2) Detection of unknown malicious families

The goal of this experiment was to show that our method can identify previously unknown malware families. We trained our model on certain families and tested the accuracy of classification between benign and malicious traffic on other families. For this task, the general malware families, General Malware and Emerging Threats are excluded from the dataset, as they do not represent any specific behavior. Rather, as observed from the family classification experiment, their behavior distributes among all families. We trained the model in a leave-one out manner, each time leaving one malicious family out, and tested on the one left out family. The suggested method was able to detect all new malware families with high accuracy. As observed in Figure 5, the Random Forest algorithm (which boosts the J48 algorithm) is able to detect all new families with very high accuracy (most of the families detected with an AUC of 0.98 except Conficker that detected with an AUC of 0.77), while the Naïve Bayes—a very simple algorithm—fails for some complex cases (e.g., APT1, Xpaj), but is effective enough in most cases.

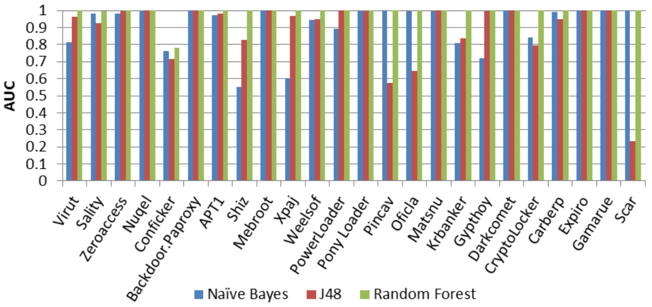

Fig. 5. Detecting unknown malicious family accuracy.

B. Effect of environments

1) Network environment robustness

The goal of this experiment was to show that our method is robust to the differences in the network environments. All together we obtained data from five very different environments (i.e., network traffic obtained from a variety of sources). It is important to notice that learning from one environment and classifying another environment was not addressed in any previous studies related to malware detection, although it is a real and practical challenge. Again, we implemented the leave one out splitting method. We trained the model on all but one environment and tested it on the one left-out environment. As seen in Figure 6, the results are very accurate in classifying sandbox environments trained on other sandbox environments. Results are lower (AUC of 0.7) for the real network environment that was trained on sandbox environments. Thus, we can conclude that the method is more robust between similar environments.

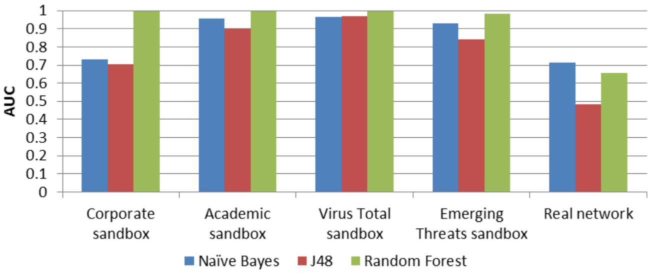

Fig. 6. Network classification from unknown environment.

We believe that in order to improve the robustness between non-similar environments, it is possible to apply transfer learning methods [32] that can also measure the similarity between domains. It is also possible to train the system for separately only for non-similar environments.

2) *Real network traffic experiment*

All previous experiments were applied to a dataset that included both sandbox and real network data. However, the results of the network environment robustness experiment show that the real network environment behaves differently (results were much lower); this phenomenon might stem from the fact that the real environment may include more noise than a synthetic sandbox environment. Therefore, the goal of this experiment is to test the effectiveness of the method in classifying benign and malicious network traffic in a real environment. Since the real network traffic was recorded continuously for several days, the system can use earlier data to train for later in time events. Thus, in this experiment we applied a time-split. The train and test sets were split by their recording time as a percentage of all data records.

In the real network dataset, 2,693 malicious instances were observed to be related to four different families as can be seen in Table VI. To produce the classification model, and preserve balance, only 10k benign instances were randomly chosen for this experiment from the same real network.

TABLE VI. REAL NETWORK MALICIOUS FAMILY DISTRIBUTION

Name	Count of instances	Percentage
General Malware	1987	73.78%
Sality	234	8.69%
Conficker	421	15.63%
Emerging threats	51	1.89%

The results of this experiment are presented in Figure 7. The results reveal that from the split point 60/40 (60% of data in the train test and the remaining 40% in test set), the AUC remains consistently above the level of 0.8. From the split point 80/20, the stable level is raised to 0.9. This result confirms that a moderate amount of data in the train set is enough for high accuracy detection of malicious network flows in a real network.

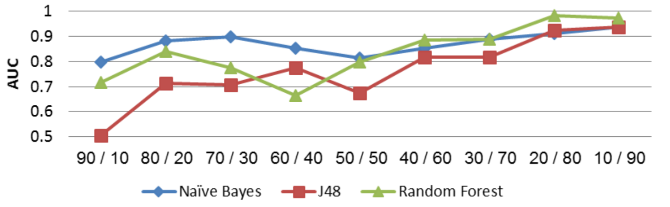

Fig. 7. Real network recording time split results.

C. *Chronological experiments*

Since our labeling includes, for each malicious network instance, the relevant SID that relates to the Snort and Suricata rules, as well as the rules' deployment date in these systems, we were able to evaluate the extent to which our method can handle malicious activities before they had been discovered. This implies that this solution might handle future threats. Here, we used the time split method, and split the data into train and test sets based on the deployment dates of SIDs.

Thus, we train the system on traffic that includes malicious activities that were handled before the split date, and test detection accuracy on traffic from the "future" (i.e., traffic instances for which SIDs are from later weeks). At each split point, the CFS algorithm was used to select the best features that divide benign and malicious traffic for the specific training set. This simulates a regular scenario where the detection model was developed and trained on the split date.

1) *Four weeks foresight*

In this section we would like to evaluate the ability of our method to detect malware that has recently been introduced. Specifically, in this section the test sets consist of sliding windows of 4 weeks ahead of the latest training set point. For example, for week 100, the training set consists of the data labeled with SIDs deployed between weeks 1-100 and the test set consists of data between the weeks 101-104.

The results of this experiment, presented in Figure 8, show that the Random Forest classifier succeeds in detecting most of the malware (except in 2 cases) 4 weeks before a relevant rule was deployed. Naïve Bayes and the J48 fail (AUC<0.5) in some cases, which can be explained by their overfitting property.

The 2 cases with the sharp drop in the AUC between weeks 211 to 214 and between weeks 240 to 243 are the result of malware that behaves very differently from previously trained malware. When the malware's behavior is finally included in the train set, the model stabilizes.

The Random Forest was stable at all periods of time and produced good AUC results. We received an average AUC over all periods of 0.966. Such models can produce a high true positive with a reasonable false positive and can be used in production for different tasks.

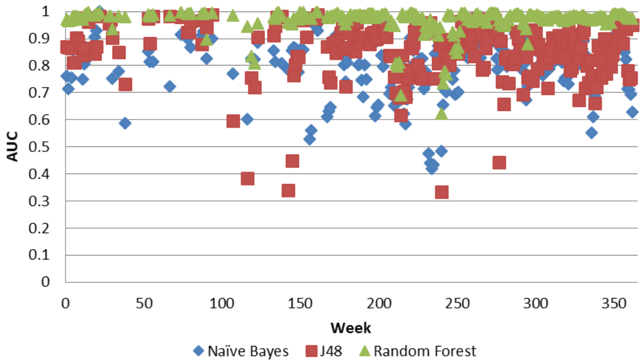

Fig. 8. Chronological experiment by looking four weeks ahead, results.

2) *Global foresight*

Given the encouraging results for the 4 weeks foresight, we wanted to evaluate how early in time our methods could detect undiscovered malware activities. Similarly to the 4 weeks foresight experiment, our train set included all data before the weekly split points, but the train set in this experiment included all future data (not only 4 weeks ahead).

The results presented in Figure 9 show, for each week, the detection performance of all malware that was discovered afterward. We can see that the detection rate deteriorates as we look further in time, because malware gradually evolves over time.

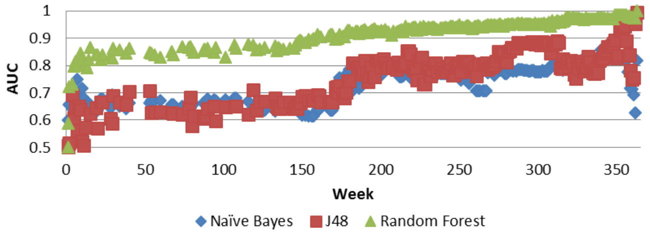

Fig. 9. Chronological experiment by looking at all future datasets, results.

The interesting point is week 159, which is 3.5 years before the newest malware in the dataset. The AUC at this point, and in all the subsequent ones, is at least 0.9, which indicates satisfactory model quality. This implies that our model is able to handle future threats even 3.5 years before they occur. For the week 309—which was 1 year before—and from then on, the results are better, in that the AUC is higher than 0.95.

D. Robustness of features

Analysis of the previous experiments described in this section has shown that only 204 features of the available 927 were ever selected by the CFS algorithm. On average, only 27 features were selected during each week for the respected model as is presented in Figure 10. Furthermore, some of the features were constantly selected for most of the models during all periods (weeks). From Figure 11, it is clearly shown that the significant part of these features were included in models for more than a period of 100 weeks (2 years). This means that a very small set of features can serve the model for a long period of time.

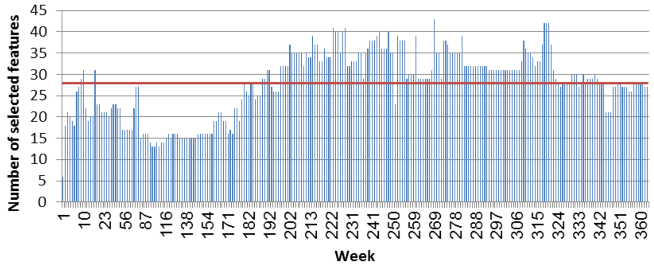

Fig. 10. Number of features used in each week.

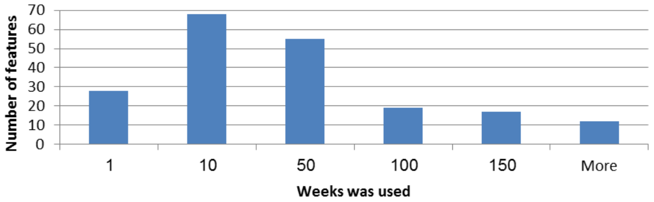

Fig. 11. Atrtributes usage histogram.

As an extension to this phenomena, in this experiment we also examined whether there was a constant set of features that were effective cross environments, and not only over time. Thus, we extracted for each environment (Verint, Academic, Virus Total and Emerging Threats sandboxes), a specific set of features. In addition, as an alternative, for each environment we prepared a model that was based on features selected from all other environments. We refer to this set as the global set. For each environment, the chronological experiment was conducted twice; once with the specific features and once with the global selected features. We compared the results of the AUC measure between the two experiments.

Comparison results are presented in Figure 12. We measure the robustness of our selected features as the difference between the AUC of a model created with global selected features and the AUC of models created with private selected features. In most cases, as seen from Figure 12, (except for the Naïve Bayes) the difference is equal or higher than 0, thus global selected features are better than those obtained using the specific ones, or the difference is not significant. The only anomaly in these results is with the Naive Bayes model which in general produced less accurate results compared to the decision tree models, and thus is more resilient to the features' changes. It therefore can be concluded that it is possible to extract features globally and obtain comparable results.

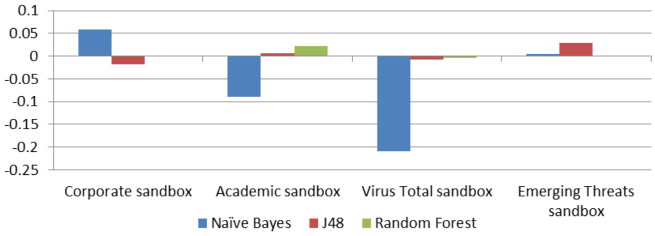

Fig. 12. Difference in average AUC.

VI. Conclusions and Future Work

In this study, we presented a network classification method that was used to classify malicious and benign traffic and attribute malicious activity to a malware family both for known and new malware. We show that different observation resolution, cross layers and protocols features improve classification performance. The accuracy of the proposed was not affected by network environments, neither sandbox nor real networks. The predictive performance results showed significant improvement over modern rule based network intrusion detection systems (e.g., Snort and Suricata); unknown malware was detected at least a month before their static rule was deployed. All this was possible with a small amount of network behavior features, which are suitable over time and for different network environments.

The practical significance of this research is the opportunity to develop highly accurate and good performing NIDS which apply machine learning algorithms, and can detect most modern malicious software as well as new and unknown malware. Since the proposed method analyzes only traffic behavior and not its content, it is effective even for encrypted traffic and for malware that uses legitimate network resources, such as C&C or proxy toward C&C. Since no payload analysis is used for this approach, users' privacy is preserved, thus such NIDS can be integrated in enterprise network systems.

For future work, we intend to extend the research toward transfer learning techniques to improve detection from untrained network environments, evaluate the proposed methods and models on mobile network traffic, test the proposed methods for malware family clustering, and finally adjust the method for online detection for high bandwidth networks.

ACKNOWLEDGMENT

We would like to thank Verint cyber research team for sharing their knowledge, providing the malware labeling database and capturing real network traffic for our experiments. Also, thanks to Chris Montgomery from Emerging Threats for providing the ETPro ruleset and sandbox captures that allowed us to compare our model to the Snort and Suricata systems. This research was sponsored by the Captain Cyber consortium funded by the Chief Scientist of the Israeli Ministry of Economy under the Magnet Program.

REFERENCES

[1] Matrosov, Aleksandr; Rodionov, Eugene; Harley, David; Malcho, Juraj;, "Stuxnet Under the Microscope," ESET LLC, September 2010.

[2] Symantec, "Malware Targeting Windows 8 Uses Google Docs": http://www.symantec.com/connect/blogs/malware-targeting-windows-8-uses-google-docs.

[3] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. Van Steen, F. C. Freiling and N. Pohlmann, "Sandnet: Network Traffic Analysis of Malicious Software," in Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 2011.

[4] N. Stakhanova, M. Couture and A. A. Ghorbani, "Exploring network-based malware classification," in 6th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, Puerto Rico, 2011.

[5] R. Perdisci, W. Lee and N. Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces," in 7th USENIX conference on Networked systems design and implementation (NSDI), San Jose, CA, USA, 2010.

[6] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel and G. Vigna, "Your Botnet is My Botnet: Analysis of a Botnet Takeover," in 16th ACM Conference on Computer and Communications Security (CCS), Chicago, Illinois, USA, 2009.

[7] "Snort": https://www.snort.org/.

[8] "Suricata": http://suricata-ids.org/.

[9] "VRT rulset": https://www.snort.org/downloads.

[10] "ETPro ruleset": http://www.emergingthreats.net/products/etpro-ruleset.

[11] A. S. Raihana, A. F. Mohd, M. N. A. Zul, M. Z. Mohd, S. R. Siti and Y. Robiah, "Revealing the Criterion on Botnet Detection Technique," International Journal of Computer Science (IJCSI), vol. 10, no. 2, pp. 208-215, March 2013.

[12] S. Nari and A. A. Ghorbani, "Automated Malware Classification based on Network Behavior," in IEEE International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA, 2013.

[13] P. Sangkatsanee, N. Wattanapongsakorn and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," Computer Communications, vol. 34, no. 18, pp. 2227-2235, 1 December 2011.

[14] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee and D. Dagon, "From Throw-Away Traffic to Bots Detecting the Rise of DGA-Based Malware," in 21st USENIX Security Symposium, Bellevue, WA, USA, 2012.

[15] T. Holz, C. Gorecki, K. Rieck and F. C. Freiling, "Measuring and Detecting Fast-Flux Service Networks," 15th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 2008.

[16] M. Amini, R. Jalili and H. R. Shahriari, "RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks," Computers & Security, vol. 25, pp. 459 - 468, 2006.

[17] E. B. Beigi, H. Hadian, N. Stakhanova and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in IEEE Conference on Communications and Network Security (CNS), San Francisco, CA, USA, 2014.

[18] T. T. Nguyen and G. Armitage, "A survey of Techniques for Internet Traffic Classification using Machine Learning," IEEE Communication Syrveys & Tutorials, vol. 10, no. 4, pp. 56-76, January 2008.

[19] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, F. Stenio and D. Sadok, "A survey on Internet Traffic Identification," IEEE Communication Survey & Tutorial, vol. 11, no. 3, pp. 37-52, August 2009.

[20] "Full set of features": www.ise.bgu.ac.il/dima/Network_Traffic_Features_Set.pdf.

[21] "Wireshark": www.wireshark.org.

[22] "TCPdump & LibPcap": http://www.tcpdump.org/.

[23] "Alexa rank": http://www.alexa.com/topsites.

[24] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[25] M. A. Hall, "Correlation-based feature selection for machine learning," The University of Waikato, Hamilton, NewZealand, 1999.

[26] "Weka: http://www.cs.waikato.ac.nz/ml/weka/index.html.

[27] C. X. Ling, J. Huang and H. Zhang, "AUC: A Better Measure than Accuracy in Comparing Learning Algorithms," in 16th Conference of the Canadian Society for Computational Studies of Intelligence, Halifax, Canada, 2003.

[28] "Verint," [Online]. Available: http://www.verint.com.

[29] "Emerging Threats": http://www.emergingthreats.net.

[30] "VirusTotal": http://www.virustotal.com.

[31] L. Xu-Ying, W. Jianxin and Z. Zhi-Hua, "Exploratory Undersampling for Class-Imbalance Learning," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 2, pp. 539-550, 2009.

[32] A. Argyriou, A. Maurer and M. Pontil, "An Algorithm for Transfer Learning in a Heterogeneous Environment," in European Conference, ECML PKDD, Antwerp, Belgium, 2008.