Ensemble of Feature Chains for Anomaly Detection

Lena Tenenboim-Chekina, Lior Rokach, Brach Shapira
Department of Information Systems Eng. and Telekom Innovation Laboratories at
Ben-Gurion University of the Negev, Israel

Abstract. Along with recent technological advances more and more new threats and advanced cyber-attacks appear unexpectedly. Developing methods which allow for identification and defense against such unknown threats is of great importance. In this paper we propose new ensemble method (which improves over the known cross-feature analysis, CFA, technique) allowing solving anomaly detection problem in *semi-supervised* settings using well established *supervised learning algorithms.* Theoretical correctness of the proposed method is demonstrated. Empirical evaluation results on Android malware datasets demonstrate effectiveness of the proposed approach and its superiority against the original CFA detection method.

Keywords: ensemble methods, machine learning, anomaly detection, probabilistic methods, network monitoring, Android, malware

1 Introduction

Anomaly detection refers to the problem of findings patterns in data that do not conform to expected behavior [1]. Numerous anomaly detection techniques have been developed over the years and implemented in various domains, such as fault-detection, healthcare applications, and intrusion detection systems. Different types of anomaly detection methods exist depending on application domain, problem and data types, system location etc. The prime differentiation between various techniques is according to the type of utilized detection algorithm: supervised, semi-supervised or unsupervised.

Anomaly detection methods based on *supervised* learning algorithms can be used when the training data includes instances labeled as either `normal' or `abnormal'. Then, a learning algorithm can be applied to distinguish between these types of data, and hence discover anomalies. The major shortcoming of this approach is that it requires examples of anomalous data, which often do not exist or are very scarce (leading to imbalance class distributions, a known machine learning problem). Furthermore, it requires manual labeling of instances. Moreover, it mainly allows detection of known attack patterns. In the case of *semi-supervised* problem settings, only `normal' instances are available for training, and thus, only the normal behavior can be learned and modeled. Instances that deviate from the learned 'normal' models can then be considered as anomalous. *Unsupervised* anomaly detection techniques detect anomalies in an unlabeled dataset under the assumption that the typical normal in-

stances will be much more common than abnormal ones and are looking for instances with less fit to the rest of the data.

In this paper we propose a new technique allowing solving *semi-supervised* anomaly detection problem using *supervised* learning methods for which numerous well established and quick algorithms exist. The idea of this technique was inspired by two existing methods: cross-feature analysis [2] and Classifier Chains [3]. Similarly to cross-feature analysis (CFA) the new technique estimates the probability of a feature getting a certain value, given the values of other features. The estimated probabilities of all the features are then combined into the entry vector's probability following the chaining approach, initially suggested in [3] for solving multi-label classification problems. The proposed approach is theoretically justified and hence is expected to improve the accuracy performance over the original CFA mrthod.

We evaluate the proposed method experimentally on 15 datasets representing network behavior of five real and ten self-developed mobile malware applications and there benign versions. Specifically, we aim to detect mobile malware applications of a new recently appeared self-updating type [4]. The initial version of such malware applications which is hosted on official marketplace sites is absolutely benign and does not contain any malware by itself. Instead after the application is downloaded and installed on end user device the update procedure is initiated and the package containing actual malicious payload is downloaded from the attacker's server insensibly to the user. The update action can be scheduled for any specific or random time in the future, or even be initiated remotely by sending a command message to the devices, using, for instance, Google's push notification service. This new technique allows malware applications to stay undiscovered on the market despite recently deployed scanning service [5] designed to flag malicious applications before they can be downloaded by end users. Additionally, such a self-updating capability makes it possible for malware developers to simultaneously penetrate new threats into numerous devices. The new threats can even exploit system vulnerabilities which were unknown at the time of the development of the initial application version. Developing methods which allow for identification and defense against this new emerging malware type is of great importance. Malware activities of this type and most others regularly affect the application's network behavior and can be detected by monitoring network behavior patterns. Thus, we focus on monitoring applications network behavior and aim to detect its unexplained changes any time they occur. Evaluation results demonstrate that deviations from an application's normal behavior can be detected quickly and accurately. In addition, the proposed ensemble algorithm allows for better detection and lower false positive rates.

The rest of the paper is organized as follows. Section 2 describes the existing methods for anomaly detection. Section 3 presents our new method. Section 4 presents the conducted experiments and their results. Lastly, Section 5 concludes the paper and outlines future research.

2 Related Work

Our task relates to the family of semi-supervised anomaly detection methods which assume that the training data consists of "normal" instances only (or mainly from normal instances while abnormal instances are negligible). These types of problems can be solved, for example, with one-class support vector machines (SVMs), the local outlier factor (LOF) method or clustering based techniques [1, 6]. In the literature there are several attempts to use probabilistic methods for anomaly detection. In particular, Bayesian networks [7] and cross feature analysis [2, 6]. Generally speaking all these methods are based on the notion of likelihood. The idea is to evaluate the likelihood of getting a current behavior given the historical behavior of the application. Formally, it can be defined as follows

$$P(event\ x\ is\ normal) = P(f_1, f_2, \dots, f_L|T),$$

where $\{f_1, f_2, \dots, f_L\}$ is the features vector, L is the total number of features and T is a training set of normal events. If the estimated likelihood is relatively low then we define the current behavior as abnormal and we suspect that it might be due to malicious activity. In order to estimate the likelihood we utilize probabilistic supervised learning methods. Given a training set, these methods can induce a model that estimates the probability of a feature getting a certain value, given the values of all other features. We examine two different ways to estimate the likelihood using probabilistic supervised learning methods: original cross-feature analysis [2] (CFA) and its improved versions referred to as Feature Chains (FC) and Ensemble of Feature Chains (EFC).

2.1 Cross-Feature Analysis

The cross-feature analysis approach was initially presented by Huang *et al.* [2] and then further analyzed by Noto *et al.* [6]. Both of these works have found this approach successful and useful for anomalies detection. Differently from Huang *et al.* [2] which consider discrete features only and from Noto *et al.* [6] who mainly focus on methods for measuring and combining the contributions of each feature predictor, we developed an improved version of cross-feature analysis technique which can handle both numeric and nominal features and is suitable for running on mobile devices. As well, we precisely implemented the original CFA version for comparison purposes. In the following the general idea of cross-feature analysis and the original CFA technique are presented followed by the description of the proposed improvements.

The main assumption underling the cross-feature analysis approach is that in normal behavior patterns, strong correlations between features exist and can be used to detect deviations caused by abnormal activities. The basic idea of a cross-feature analysis method is to explore the correlation between one feature and all the other features. Formally, cross-feature analysis approach tries to solve the classification problem $C_i: \{f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_L\} \rightarrow \{f_i\}$, where $\{f_1, f_2, \dots, f_L\}$ is the features vector and L is the total number of features. Such a classifier is learned for each feature i,

where i = 1, ... L. Thus, an ensemble of learners for each one of the features represents the model through which each features vector will be tested for "normality". The procedure utilized for online analysis of each individual instance is described below.

When a features vector representing a normal event is tested against C_i, there is a higher probability for the predicted value to match (for discrete features) or be very similar (for numeric features) to the observed value. However, in the case of a vector representing abnormal behavior, the probability of such a match or similarity is much lower. Thus, by applying all the features models to a tested vector and combining their results, a decision about vector normality can be derived. The more different the predictions are from the true values of the corresponding features, the more likely that the observed vector comes from a different distribution other than the training set (i.e., represents an anomaly event). A threshold distinguishing between normal and anomalous vectors can be can be computed by calculating the lower bound of output values from normal events.

For each predictor C_i the probability of the corresponding feature value of a vector x to come from a normal event is computed. For numeric features this probability, noted $P(f_i(x)\ is\ normal)$, is calculated as the following:

$$P(f_i(x)\ is\ normal)\ =\ 1 - log_{10}(\frac{C_i(x)}{f_i(x)}) \quad (1)$$

where, $C_i(x)$ is the predicted value and $f_i(x)$ is the actual observed value. Note that if the result of the logarithm function above is greater than one, it is converted to one. Thus, the calculated probability is always in the range [0, 1]. For the nominal features the estimated probability for the true class is utilized. In [2] two options for combining predictions of all features into the final decision are examined: *Average Match Count* and *Average Probability*. The second option which computes the average of probabilities over all classifiers, as follows

$$Average\ Probability = \frac{\sum_{i=1}^{L} P(f_i|f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_L)}{L} \quad (2)$$

was found by the authors as providing better performance results than *Average Match Count*. Thus, we evaluate this approach in our experiments. Events with the *Average Probability* below the threshold learned on normal data are classified as anomaly.

3 Feature Chains

In this section we describe the proposed improvement over the original CFA method which is referred to as *Feature Chains*. First we describe a single feature chain model. Then we show how to build an ensemble of feature chains.

3.1 A Single Feature Chain Model

This new method for likelihood estimation of an observed features vector $\{f_1, f_2, \dots, f_L\}$ was inspired by a successful algorithm for multi-label classification called *Classifier Chains* [19]. Classifier Chains method was recently proposed for

solving multi-label classification problem using binary classifiers in a way that overcomes the label independence assumption. According to the Classifier Chains algorithm, a single binary classifier is associated with each one of the predefined labels in the dataset and all these classifiers are linked in an ordered chain. The feature space of each classifier in the chain is extended with the 0/1 label associations of all previous classifiers. Following this idea we suggest to perform the chaining on the input features (as opposed to the labels chain that was performed in the original Classifier Chains algorithm), for estimating likelihood of the observed features vector in the following way:

$$P(f_1, f_2, \dots, f_L) = \prod_{i=1}^{L} P(f_i | f_1, f_2, \dots, f_{i-1}) \quad (3)$$

Note that Equation (3) is justified by the following equivalence which can be derived by applying Bayes rule ($P(A|B) = \frac{P(A,B)}{P(B)}$) on the features conditional probabilities, calculated at the right side term of the equation (3):

$$P(f_1) * P(f_2|f_1) * P(f_3|f_1, f_2) * \dots * P(f_L|f_1, f_2, \dots, f_{L-1}) =$$

$$= P(f_1) * \frac{P(f_2, f_1)}{P(f_1)} * \frac{P(f_3, f_2, f_1)}{P(f_2, f_1)} * \dots * \frac{P(f_L, f_{L-1}, \dots, f_1)}{P(f_{L-1}, \dots, f_1)} \quad (4)$$

By reducing the corresponding denominator and numerator in Equation (4) we will be left by only the last term $P(f_L, f_{L-1}, \dots, f_1)$ which is equivalent to the left side of Equation (3). Differently, from the equation (2) used in the original version of Cross-Feature Analysis method the Equation (3) used by Feature Chains approach has a theoretical justification. Thus, it potentially can be used for the likelihood estimation of $P(f_1, f_2, \dots, f_L)$ and we expect for a practical improvement in the anomaly detection accuracy.

Similarly to the Cross Features Analysis method, every conditional probability term in Equation (3) is estimated using any probabilistic supervised learning algorithms that can provide the conditional probability of the target feature given the input features. If the examined feature is nominal then classification methods (such as SVM or classification trees) should be used and if the target feature is numeric, then regression methods should be used. Certain methods such as Neural Networks and Classification and Regression tree (e.g., CART) can be used for both nominal and numeric target features.

In the case of numeric features, the distance between actual and predicted values is used as a proxy to the estimated probability for getting the actual value. Various scaling methods can be used to convert the distance to a probability. For example, one of the options is the *log distance* approach as proposed by [2] and presented in Equation (1). Another approach followed by [6] is to calculate the distance as the difference in actual and predicted values divided by the range of that feature's value (i.e., the maximum distance is 1.0). Yet, another option is to divide the difference in value by the mean of the observed values for that feature.

One concern about the proposed feature chain algorithm is the low dimensionality of the data used at the beginning of chaining process. It is true that using only a few attributes in the beginning is oversimplification. But on the other hand the probability can be estimated more accurately due to the low number of parameters to estimate. In

addition we try to overcome this oversimplification by using ensemble of Feature Chains as explained below, thus allowing averaging over numerous different chains.

3.2 Ensemble of Feature Chains

It should be noted of course that the order of features in the chain may have an effect on the model's accuracy. As with any learning algorithm, some models may overestimate the probability value and others underestimate the probability value. A convenient option for solving this issue is using an ensemble of Feature Chain models where each of the models is learned on a different chain of randomly ordered features. This approach was proved successful in the case of Classifier Chains method [3]. Additionally, it is known that ensemble methods are able to improve the prediction performance over a single classifier [8]. Note that Feature Chains method can occasionally be referred to as ensemble method because it involves multiple models. However, none of these models is capable for predicting the likelihood of the entry instance and therefore we use the term ensemble strictly in the sense of combining the final (i.e. instance-related) predictions of multiple models. Below the EFC learning process is described.

An Ensemble of Feature Chains trains *m* Feature Chain models $C_1, C_2, \ldots, C_m$. Each of the C_k is trained with a random features ordering in the chain. Hence predictions of each C_k model depend on underlying features order and are likely to be diverse in border-line cases. For combining the predictions of all the models several approaches exist. In this paper we examine a simple and popular majority voting approach, according to which binary decisions (0 – for normal and 1 – for anomalous instance) of all distinct FC models are summarized and divided by total number of models *m*, so that the output, referred to as *anomaly votes score* is normalized into the range of [0, 1]. A threshold is used to derive the final decision such that an instance is marked as anomaly if its *anomaly votes score* is above the defined threshold *t*.

4 Experimental Studies

This section presents the evaluation of the proposed detection methods. First, the data aggregated from several real and self-developed malware applications utilized in this experiment is described. Then, the system evaluation processes is described and the observed results are presented.

4.1 Evaluated Malware

For the evaluation of the proposed methods we experimented with five real and ten self-written Trojan malware. Each malware has two versions: the original benign application requesting network access permission for various purposes (such as displaying advertisements, high scores update, information or data sharing, etc.) and the repackaged version of the original application with injected malware code utilizing network communication for malicious purposes.

For the experiments with the real malware, five infected applications and their benign versions were used. The infected applications and the corresponding versions of the benign application were obtained from a repository collected by crawling the official and various alternative Android markets for over a year and a half. We used two applications injected with PJApps Trojan - Fling and CrazyFish; two applications injected with Geinimi Trojan - Squibble Lite and ShotGun; and one sample of DroidKungFu-B malware found within the OpenSudoku game. The PJApps Trojan sends sensitive information containing the IMEI, Device ID, Line Number, Subscriber ID, and SIM serial number to a web server, and retrieves commands from a remote command and control server. Similarly, the Geinimi Trojan transmits information from the device to the server and may be instructed to perform certain actions. The DroidKungFu-B malware targets rooted phones and requests for the root privilege; then, with or without the root privilege, it collects and steals the sensitive phone information, such as IMEI, phone model, etc. All the infected applications are mobile games which exploit network communication for certain purposes, such as online advertisements or score updates.

Malware applications with the advanced self-updating capabilities have just started to appear and there are not yet enough known real malware samples of this type. Thus, for the purposes of this paper, we have created the malware packages using two different types of self-updating behavior (type 1, entry application update and type 2, injection of compiled malicious component) and infected several open-source applications with these packages.

The utilized open-source applications are: APG, K-9 Mail, Open WordSearch, Rattlesnake Free and Ringdroid. The APG, the Android Privacy Guard application, provides OpenPGP functionalities, such as encryption and signing of emails. It uses network connections for public and secret keys management. K-9 Mail is an open source email client for Android. Open WordSearch is a game application that uses network connections to synchronize global high scores. Rattlesnake Free is also a game application utilizing network connections for online advertisements. Ringdroid is an application for recording and editing sounds and creating ringtones directly on the Android phone. It uses network connections to share ringtones and other sounds created by users. Each one of these applications was infected and evaluated using the created malware of both types. To simulate malicious behavior within the created malware, we choose to implement some simple malicious behavior patterns of known malware, such as stealing a user's contacts list, recent calls details, and user's GPS location which are sent out to a remote server.

An application infected with the malware component of type 1 will present an "update is available" notification to the user when the corresponding command is received by the device. When a user agrees to install the update, it will download a malicious version of the same application from a remote server and replace the benign version with the malicious one. At this stage the user is presented with a list of permissions to be granted to the new application version, which actually could differ from those granted to the original application. Once installed, the new malicious version will wait for an external command. When the command is received, it steals the user's contacts list and sends it to a remote server.

An application infected with malware component of type 2 will silently download a precompiled malicious payload when the corresponding command is received by the device, and then continue to load and execute malicious code without any notification to the user. The malicious payload will first steal the user's contacts list and send it to a remote server and then continue to report the user's location and recent call details to the server every specified time period (set to two minutes for our experiments).

For data aggregation all the malware applications and their benign counterparts were executed on the specially designated devices and their network behavior features were collected. The list of the utilized features is presented at Table 1. Initially, a benign version of each evaluated application was installed and executed on a device for two days. Then, it was injected\replaced by the malicious version, which was executed for at least one hour.

Table 1. The list of utilized features.

No.	Feature	Brief Description
1	avg_sent_bytes	Represent the average amount of data sent or received by an application at the observed time interval (of 1 min.)
2	avg_rcvd_bytes	
3	avg_sent_pct	Represent the average portion of sent and received amount of data at the observed time interval (of 1 min.)
4	avg_rcvd_pct	
5	pct_avg	Represents the portion of average received amount of data at the
6	inner_ sent	Average time intervals between send\receive events occurring within the time interval of less than 30 seconds.
7	inner_ rcvd	
8	outer_ sent	Average time intervals between send\receive events occurring within the time interval above or equal to 30 seconds.
9	outer_ rcvd	

4.2 Experimental Setup

We implemented all the evaluated methods in Java using Weka [9] open source library. The Decision/Regression tree (REPTree Weka's implementation) algorithm was used as base learning algorithm for CFA and FC methods, as it is can handle both nominal and numeric target features. The decision threshold values were learned on a separate set of labeled data examples during the calibration experiments. The values allowing preserving an acceptably low level of false positive alarms (below 20%) were determined as follows: 0.7 for CFA method and 0.001 for FC method.

The ensemble methods are known for their capability to improve the prediction performance over a single classifier in exchange for more computational resources and longer execution times. Thus, for the sake of a fair comparison, we compare the EFC, utilizing the REPTree method as the base learner of each single chain model, with CFA and FC methods set to use Rotation Forest [10] ensemble as their base learner. The versions of CFA and FC methods utilizing the ensemble algorithm as their base learner are denoted CFA-IE and FC-IE correspondingly (IE stands for Internal Ensemble). Rotation Forest is a recently proposed but already well-known successful method for building classifier ensemble using independently trained decision trees. The Rotation Forest was set to use the REPTree algorithm as its base learner.

The majority voting threshold of the EFC method was set to a commonly used intuitive value of 0.5. Influence of the ensemble models number on the performance accuracy was analyzed on the calibration datasets and *m=50* was selected as always providing stable optimal results. Respectively, the number of iterations for the Rotation Forest was set to 50, also.

For learning the "normal" patterns first 30 records (not counting a few bootstrapping records) of each benign application were used. The rest of the normal data and observed traces of malicious versions were used for testing the methods detection performance. To evaluate the detection capabilities of the proposed methods the following standard measures were employed: True Positive Rate (TPR) measure (also known as detection rate), which determines the proportion of correctly detected instances relating to application's malicious behavior and the False Positive Rate (FPR) measure (also known as false alarm rate), which determines the proportion of mistakenly detected anomalies in an actually normal application behavior. Note that sometime significant deviations in normal application's behavior can be caused by changes in user's behavior. Thus a certain level of false alarms might be acceptable especially for applications with diverse network functionality.

4.3 Results

Initially we compare the new methods, FC and EFC, to the original CFA method. Results of these algorithms for all the evaluated benign\malware application pairs are presented in Table 2. The best result for each evaluation measures on a particular application dataset is marked in bold separately for the FC vs. CFA and EFC vs. CFA pair-wised comparisons.

Table 2. Malware Detection Results – New Methods vs. Original CFA.

	Application name	TPR (%)			FPR (%)		
		CFA	FC	EFC	CFA	FC	EFC
Real malware	Fling	66.8	**69.0**	**67.9**	**0**	4.2	3.5
	OpenSudoku	100	100	100	0	0.0	0
	ShotGun	100	100	100	**0**	4.8	4.8
	Squibble	77.5	**95.0**	**97.5**	15.8	15.8	15.8
	Crazy Fish	90.6	**100**	**100**	**0**	7.7	7.7
Self-update 1	APG	**100**	92.3	92.3	0	0.0	0
	K-9 Mail	91.7	**100**	**100**	**0**	2.0	0
	WordSearch	100	100	100	6.3	6.3	6.3
	Rattlesnake	92.3	92.3	92.3	**8.1**	12.2	**6.5**
	Ringdroid	100	100	100	0	0.0	0
Self-update 2	APG	100	100	92.9	**0**	4.3	0
	K-9 Mail	66.7	**83.3**	**91.7**	**0**	2.9	0
	WordSearch	100	100	100	8.3	8.3	8.3
	Rattlesnake	83.3	**100**	**100**	**8**	16.0	8.0
	Ringdroid	92.3	**100**	**100**	0	0.0	0

Additionally, we perform an experiment comparing EFC with CFA and FC methods utilizing ensemble algorithm as their base learner. Results of these algorithms are presented in Table 3. The best result for each evaluation measures on a particular dataset is marked in bold separately for the FC-IE vs. CFA-IE and EFC vs. CFA-IE pair-wised comparisons.

Table 3. Malware Detection Results – Ensemble Methods.

	Application	TPR (%)			FPR (%)		
		CFA-IE	FC-IE	EFC	CFA-IE	FC-IE	EFC
Real malware	Fling	65.8	**68.5**	**67.9**	**0.7**	2.8	3.5
	OpenSudoku	100	100	100	0	0	0
	ShotGun	99.3	**100**	**100**	**0**	7.1	4.8
	Squibble	82.5	**92.5**	**97.5**	**0**	15.8	15.8
	Crazy Fish	94.9	**100**	**100**	**0**	**0**	7.7
Self-update 1	APG	**100**	**100**	92.3	0	0	0
	K-9 Mail	100	100	100	38.8	**0**	**0**
	WordSearch	100	100	100	6.3	6.3	6.3
	Rattlesnake	92.3	**100**	92.3	36.6	**9.8**	**6.5**
	Ringdroid	80.0	**100**	**100**	16.7	**0**	**0**
Self-update 2	APG	92.9	**100**	92.9	0	0	0
	K-9 Mail	25	**83.3**	**91.7**	40	**0**	**0**
	WordSearch	100	100	100	8.3	8.3	8.3
	Rattlesnake	100	100	100	32	**9.3**	**8.0**
	Ringdroid	100	100	100	16.7	**0**	**0**

Lastly, we evaluate detection performance of EFC method with respect to the number of ensemble models. The TPR and FPR results on two of the evaluated applications are presented in Fig. 1. For all other evaluated applications similar results were observed.

Statistical significance of the difference between algorithms' results was determined by Wilcoxon signed-ranks test [11]. The exact confidence level is mentioned specifically for each comparison at the results discussion.

Generally, it can be seen that for almost all malicious applications, the high level of deviation (80-100% of anomalous instances) from the normal network behavior was detected by all the evaluated methods. Additionally, it can be seen that the FPR of all the detection algorithms is below 10% in most cases.

Comparing the performance of the proposed Feature Chain and the original CFA approaches, it can be seen that FC significantly outperforms the CFA in terms of TPR (the difference is statistically significant at 0.05 confidence level). It provides higher detection rate on 7 datasets and lower detection rate on 1 dataset only. However, at the same time it suffers from much higher false alarms rate than CFA method (the difference is also statistically significant at 0.05 confidence level). Yet, the EFC approach successfully overcomes this drawback: aggregation of numerous FC models into a composite ensemble model allows reduction of FPR to statistically indistin-

guishable difference comparing to CFA method, while preserving the very high detection rate on all the datasets.

We continue by comparing the performance of the EFC, FC-IE and CFA-IE methods. As can be seen (in Table 3) both EFC and FC-IE methods provide the highest detection rate on most datasets. There are slight differences between TPR results of these two methods on a few datasets, however this difference is statistically insignificant. On the other hand, it can be seen that CFA-IE method archives lower TPR on 8 and 6 datasets comparing with FC-IE and EFC methods correspondingly. In some of these cases the difference in the achieved detection rates is very meaningful and could lead to much later identification of the malware. At the same time, the CFA-IE method outperforms (in terms of TPR) the FC-IE and EFC methods in 0 and 1 cases, only, correspondingly. The difference between CFA-IE and FC-IE detection rate is statistically significant at 0.01 confidence level. Additionally, considering the FPR of the ensemble algorithms, it can be seen that CFA-IE method has unexpectedly high level (above 20%) of false alarms on several datasets, while both EFC and FC-IE preserve relatively low FPR values.

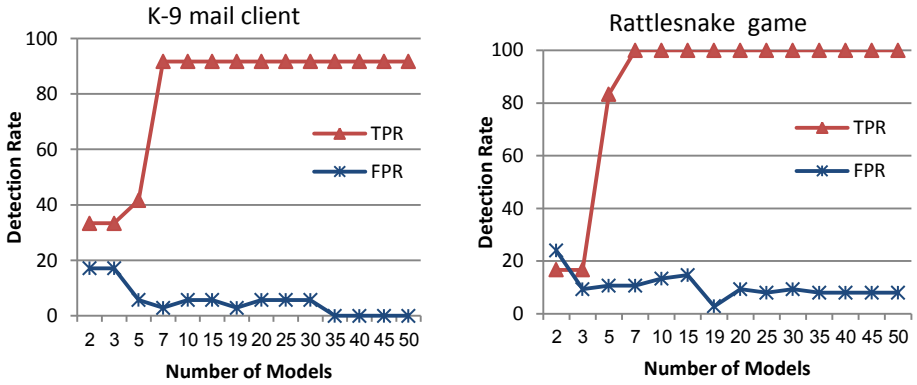

Fig. 1. EFC performance with respect to number of models.

Considering, the EFC performance with respect to the number of ensemble models (as depicted in Fig. 1) it can be seen that high and stable level of True Positive Rate is achieved at relatively low number of models, $m \geq 7$. It can be seen also that larger number of models leads to lower False Positive Rate. However, for achieving a stable low FPR level, a larger number of models, regularly $m \geq 30$, is needed.

Summarizing the above comparison we conclude that the proposed Feature Chains technique allows for significant improvement of the detection performance over the original CFA method. However, it suffers from a higher False Positives Rate. At the same time, the two evaluated ensemble versions of the new Feature Chains methods, EFC and FC-IE, allow for significant reduction in the false alarms rate (the difference is statistically significant at 0.05 confidence level), while preserving the high True Positive Rate. Hence, the results justify using the proposed ensemble methods, FC-IE or EFC, for anomaly detection.

5 Summary and Conclusions

This paper presented a novel probabilistic method for solving semi-supervised anomaly detection problems and its ensemble version. The new method is based on the known cross-feature analysis and classifier chaining methods. It can handle numeric and nominal features and is suitable for running on mobile devices. The presented method can be used for solving various semi-supervised anomaly detection problems. Theoretical correctness of the proposed method was demonstrated.

Empirical evaluation of the proposed methods on the variety of datasets demonstrated effectiveness of the proposed approach for the defined problem: a high TPR along with low FPR could be achieved. The proposed Ensemble of Feature Chains and Feature Chains using internal ensemble proved superior to the original CFA method and its ensemble version.

Among our future research directions are evaluation of the present methods on more datasets from different domains and comparison with other anomaly detection methods.

References

1. V. Chandola, A. Banerjee and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv. 41(3):1–58, 2009.
2. Y.A. Huang, W. Fan, W. Lee and P.S. Yu, “Cross-feature analysis for detecting ad-hoc routing anomalies,” In: IEEE 23rd Int. Conf. on Distributed Computing Systems, 478-487, 2003.
3. J. Read, B. Pfahringer, G, Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," Proc. of 20th European Conference on Machine Learning and Knowledge Discovery in Databases, 2, 254-269, 2009.
4. Symantec blog: http://www.symantec.com/connect/blogs/androiddropdialer-identified-google-play
5. Google mobile blog, android and security: http://googlemobile.blogspot.co.il/2012/02/android-and-security.html
6. K. Noto, C. Brodley and D. Slonim, “Anomaly detection using an ensemble of feature models,” In: Proc. of the 10th IEEE International conf. on Data Mining, 953-958, 2010.
7. N. Ye, M. Xu, S.M. Emran, "Probabilistic networks with undirected links for anomaly detection," In: Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY, 175-179 , 2000
8. L. Rokach and O. Maimon, ”Ensemble Methods for Classifiers,” Data Mining and Knowledge Discovery Handbook. Springer US, 2005.
9. Weka 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/
10. J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso, ”Rotation Forest: A New Classifier Ensemble Method,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10),1619-1630, 2006
11. J. Demsar, ”Statistical comparisons of classifiers over multiple data sets, ” *Journal of Machine Learning Research*, 7,1-30, 2006.