In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy.

A Software Architecture for Classifying Users in e-Payment Systems

Gianluigi Folino and Francesco Sergio Pisani

Institute of High Performance Computing and Networking (ICAR-CNR)
gianluigi.folino,fspisani@icar.cnr.it

Abstract

In modern payment systems, the user is often the weakest link in the security chain. To identify the key vulnerabilities associated with the user behavior and to implement a number of measures useful to protect the payment systems against these kinds of vulnerability is a real hard task. To this aim, we designed an architecture useful to divide the users of a payment system into pre-defined classes according to the type of vulnerability enabled. In this way, it is possible to address actions (information campaigns, alerts, etc.) towards targeted users of a specific group. Unfortunately, the data useful to classify the user typically presents many missing features. To overcome this issue, a tool was developed, based on artificial intelligence and adopting a meta-ensemble model, to operate efficiently with missing data. Each ensemble evolves a function for combining the classifiers, which does not need of any extra phase of training on the original data. The approach is validated on a well-known real dataset of Unix users demonstrating its goodness.

1 Introduction

In the last few years, as a consequence of our interconnected society, the interest in cyber security problems has really been increasing and cyber crime seriously threatens national governments and the economy of many industries [3]. Countermeasures must be undertaken to protect security vulnerabilities and weaknesses of the systems from the potential attacks, in order to minimize the risks. In addition, computer network activities, human actions, etc. generate large amounts of data and this aspect must be seriously taken into account.

Many works present in literature [4][5] remark that most serious threats and vulnerabilities concern the user and its wrong behaviors. In modern payment systems, the user is often the weakest link in the security chain. Indeed, in 2015, according to Kaspersky Lab research[1], 73% of organizations had internal data breach issues that can severely compromise their business. Typical examples are the use of insecure passwords, saving critical data not encrypted on a notebook, etc. In particular for e-payment systems, the users, the operators and even the top managers of these systems are often unaware of the risks associated with vulnerabilities enabled by their behavior and that can be dangerous and costly. Therefore, a system for the protection of payment systems can not be separated from the analysis of the vulnerabilities related to the users so that the appropriate countermeasures can be undertaken.

In light of these considerations, it is a very hard task to identify the key vulnerabilities associated with the user behavior and to implement a number of measures useful to protect the payment systems against these kinds of vulnerabilities.

A scalable solution for operating with the security weaknesses derived from the human factor must consider a number of critical aspects, such as profiling users for better and more focused actions, analyzing large logs in real-time and also it should work efficiently in the case of missing data.

[1] https://blog.kaspersky.it/

Data mining techniques could be used to fight efficiently, to alleviate the effect or to prevent the action of cybercriminals, especially in the presence of large datasets [1]. In particular, classification is used efficiently for many cyber security applications, i.e. classification of the user behavior, risk and attack analysis, intrusion detection systems, etc. However, in this particular domain, datasets often have different number of features and each attribute could have different importance and cost. Furthermore, the entire system must also work if some features are missing. Therefore, a single classification algorithm performing well for all the datasets would be really unlikely, especially in the presence of changes and with constraints of real time and scalability. In the ensemble learning paradigm [6][7], multiple classification models are trained by a predictive algorithm, and then their predictions are combined to classify new tuples. This paradigm presents a number of advantages with regard to using a single model, i.e., it reduces the variance of the error, the bias, and the dependence on a single dataset and works well in the case of unbalanced classes; furthermore, the ensemble can be build in an incremental way and can be easily implemented on a distributed environment.

In order to protect the e-payment systems from the problems concerning the user behavior, in this work, a general architecture was designed. It is useful to monitor the user behavior, divide the users of a payment system into pre-defined classes according to the type of vulnerability enabled and to make possible to address suitable actions (information campaigns, alerts, etc.) towards targeted users of a specific group. Typically, the data useful to classify the user presents many missing features. To overcome this issue, a classification tool was also used, based on artificial intelligence (i.e., an evolutionary-based algorithm) and adopting a meta-ensemble model to operate efficiently with missing data. For each ensemble, the function for combining the classifiers, which does not need of any extra phase of training on the original data, is evolved by a genetic programming (GP) tool. Therefore, in the case of changes in the data, the function can be recomputed in an incremental way, with a moderate computational effort; this aspect together with the advantages of running on parallel/distributed architectures make the algorithm suitable to operate with the real time constraints typical of a cyber security problem. The tool was developed in a previous work [2], but here it is adopted in this novel architecture.

Among the strategies operating with ensemble for coping with missing data, we remember the work in [8]. The authors build all the possible LCP (Local Complete Pattern), i.e., a partition of the original datasets into complete datasets, without any missing features; a different classifier is built on each LCP, and then they are combined to predict the class label, basing on a voting matrix. The experiments compared the proposed approach with two techniques to cope with missing data, i.e., deletion and imputation, on small datasets and show how the approach outperforms the other two techniques. However, the phase of building the LCP could be really expensive. Another correlate work is Learn++.MF [9], which is an ensemble-based algorithm with base classifiers trained on a random subset of the features of the original dataset. The approach generates a large number of classifiers, each trained on a different feature subset. In practice, the instances with missing attributes are classified by the models generated on the subsets of the remaining features. Then, the algorithm uses a majority voting strategy in order to assign the correct class under the condition that at least one classifier must cover the instance. When the number of attributes is high, it is unfeasible to build classifiers with all the possible sets of features; therefore, the subset of the features is iteratively updated to favor the selection of those features that were previously undersampled. However, this limits the real applicability of the approach to datasets with a low number of attributes

The rest of the paper is structured as follows: in Section 2 some related works concerning the classification of user profiles are illustrated; Section 3 introduces the general architecture

of the proposed approach; Section 4 is devoted to some background information concerning the problem of missing data and incomplete datasets and the ensemble of classifiers; in Section 5, the main algorithm used to classify the user is explained; Section 6 shows the experiments conducted to verify the effectiveness of the approach and to compare it with other similar approaches; finally, Section 7 concludes the work.

2 Methods for classifying user profiles

The inspiration of the approach taken in this paper comes from a project on cyber security, in which one of the main tasks consists in dividing the users of an e-payments systems into homogenous groups on the basis of their weakness or vulnerabilities from the cyber security point of view. In this way, the provider of an e-payment system can conduct a different information and prevention campaign for each class of users, with obvious advantages in terms of time and cost savings. In addition, specialized security policies can be conducted towards the users of a specific class.

This technique is usually named segmentation, i.e. the process of classifying customers into homogenous groups (segments), so that each group of customers shares enough characteristics in common to make it viable for a company to design specific offerings or products for it. It is based on a preliminary investigation in order to individuate the variables (segmentation variables) necessary to distinguish one class of customers from others. Typically, the goal is to increase the purchases and/or to improve customer satisfaction.

Different techniques can be employed to perform this task; in order to cope with large datasets, the most used are based on data mining approaches, mainly clustering and classification; anyway, many other techniques can be employed (see [10] for a survey of these techniques).

Another issue to be considered in order to construct the different profiles is the information collection process used to gather raw information about the user, which can be conducted through direct user intervention, or implicitly, through software that monitors user activity. Finally, profiles maintaining the same information over time are considered static, in contrast to dynamic profiles that can be modified or improved over time [11].

In the general case of computer user profiling, the entire audit source can include information from a variety of sources, such as command line calls issued by users, system calls monitoring for unusual application use/events, database/file accesses, and the organization policy management rules and compliance logs. The type of analysis used is primarily the modeling of statistical features, such as the frequency of events, the duration of events, the co-occurrence of multiple events combined through logical operators, and the sequence or transition of events. An interesting approach to computer user modeling is the process of learning about ordinary computer users by observing the way they use the computer. In this case, a computer user behavior is represented as the sequence of commands she/he types during her/his work. This sequence is transformed into a distribution of relevant subsequences of commands in order to find out a profile that defines its behavior. The ABCD (Agent behavior Classification based on Distributions of relevant events) algorithm discussed in [12] is an interesting approach using this technique.

3 Background: Classification, Ensemble and Missing Data

In this section, some background is given; in particular, the main methods to cope with missing data and incomplete datasets are illustrated, together with the interesting paradigma of the

ensemble of classifiers used for the data mining task of classification.

Classification is one of the most important data mining tasks and consists in assigning a class label (among a pre-defined set of a classes) to a set of unclassified cases on the basis of the input data. Typically, the algorithm of classification is trained on a dataset, named the training set, consisting of multiple records each having multiple attributes or features. The final result is an accurate description or model for each class using the features present in the data. Finally, this model is used to classify test data for which the class is not known.

The ensemble paradigma combines multiple (heterogenous or homogenous) models in order to classify new unseen instances. Usually, ensemble model improves accuracy and robustness over single model methods. For example, in hard domains, such cyber security, having real-time requirements, which do not permit re-training the base models, ensemble strategies can use some functions, which can be combined without using the original training set [13] [14] and also works in an incremental way [15].

Different schemas can be considered to generate the classifiers and to combine the ensemble, i.e. the same learning algorithm can be trained on different datasets or/and different algorithms can be trained on the same dataset. In practice, after a number of classifiers are built usually using part of the dataset, the predictions of the different classifiers are combined and a common decision is taken.

When we consider data with missing features, some main patterns are commonly used. We remember: missing completely at random (MCAR), to describe data, in which the complete cases are a random sample of the originally dataset, i.e., the probability of a feature being missing is independent of the value of that or any other feature in the dataset; missing at random (MAR) describe data that are missing for reasons related to completely observed variables in the data set. Finally, the MNAR case considers the probability that an entry will be missing depends on both observed and unobserved values in the data set. Therefore, even if MCAR is more easy to handle, we try to cope with the MAR case, as it is a more realistic model and it is suitable to many real-world applications, i.e., the cyber security scenario.

Data mining and in particular classification algorithms must handle the problem of missing values on useful features. The presence of missing features complicates the classification process, as the effective prediction may depend heavily on the way missing values are treated. Typically, missing values are present in both the training and the testing data, i.e., the same sources of data are not available for all the users. However, in our approach, without any loss of generality, we suppose that the training dataset is complete. Even in the case of the presence of a moderate number of tuples presenting missing data, it can be reported to the previous case, simply by deleting all the incomplete tuples. However, handling missing data by eliminating cases with missing data will bias results, if the remaining cases are not representative of the entire sample. Therefore, different techniques can be used to handle these missing features (see [16] for a detailed list of them). In addition to the above-mentioned strategy to remove any tuple with missing values, we remember the option of using a classification algorithm, which can deal with missing values in the training phase and the strategy of imputing all missing values before training the classification algorithm, i.e., replace the missing value with a meaningful estimate.

Indeed, in our particular problem, we are more interested in handling groups of missing features, and consequently, we focus on constructing a classifier on the incomplete dataset directly.

More formally, in our scenario, we have $D_1, D_2, \ldots, D_k$ datasets; typically each dataset comes from a different source of data, but can be used to predict the same class. Therefore, the corresponding i_{th} tuple of the different datasets can be used to predict the class of the same user. However, a particular tuple of a dataset can be missing, i.e., all the features belonging to

the same source of data of that tuple are missing.

Without any loss of generality, even a problem of missing features of an incomplete dataset can be reported to the previous one, by grouping tuples with the same missing features.

4 An architecture for classification of user profiles in e-payment systems

In our scenario, the classes, in which the users will be divided, are individuated on the basis of their expertise in computer science and in the domain of the e-payments systems. Indeed, most of the vulnerabilities are associated with the behavior and the practices correlated with the knowledge of the computer and/or of the e-payment system. As example, contrarily to the normal belief, a vulnerability study confirmed that software developers are the most vulnerable to attacks [4]. Furthermore, an excess of confidence and the consequent download and installation of a number of applications can cause vulnerabilities; in the same way, misconfigurations of the system due to inconsistent application of security associated with a lack of competency could abilitate other kinds of vulnerabilities. Similar behaviors could be seen in the activities of users of the e-payments system.

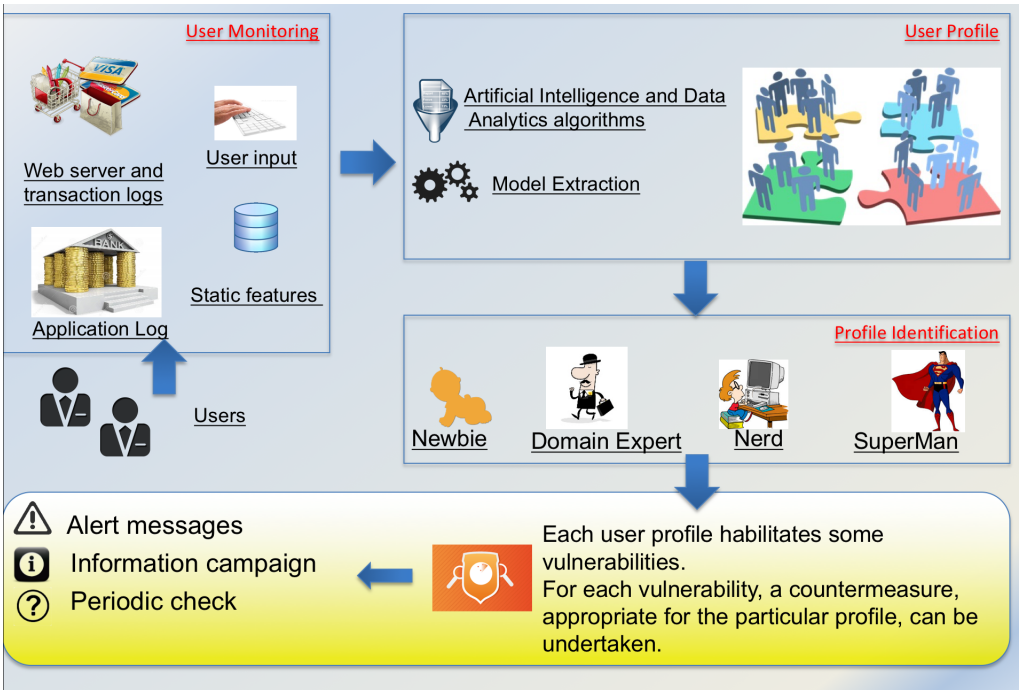

Figure 1: A general architecture to monitor and to classify the users of an e-payment system and to undertake the needed countermeasures.

Given these considerations, Figure 1 shows the general architecture proposed in this work to handle the problem of mitigating the consequences of the user behavior. The information concerning the user is supplied by using different sources of information or monitoring tools (i.e. generally automatic software analyzing the action and the behavior of the users). Going

more into detail, user datasets can include demographic and education information, e.g., name, age, country, education level, computer knowledge, task knowledge, etc. and may also includes information concerning the contest in which the users operate and the roles they have in the systems. In addition to these data, which usually do not change if we consider a reasonable amount of time, the monitoring tool collects operational and behavioral data (e.g. IP addresses from which users connect to the system, operating system and browser used, the duration of the session, etc.), for which changes over time should be also considered. Finally, we also collect user input (i.e., commands entered using the keyboard or via GUI, using the mouse, etc.) This information should be captured in a dynamic way, by logging user actions. Unfortunately, all these kinds of data are not present for each user for clear reasons of privacy and for a number of different motivations (i.e., we have users with different roles and therefore, it is possible to monitor only some types of user, some users do not want to give authorization to disclose some data, etc.). Therefore, for different users, some sources are missing and this problem must be faced efficiently in order to obtain an accurate classification. Using these data, the users of the e-payment system are classified into pre-defined classes according to the type of vulnerability enabled by using the meta-ensemble approach, described in detail in the next section. Finally, suitable actions (information campaigns, alerts, periodic checks, etc.) can be undertaken towards targeted users of a specific group.

5 A Meta-Ensemble Approach for Classifying user profiles

In this section, we illustrate the software architecture and we detail the main steps of the meta-ensemble approach used to classify the user profiles, also in the case of missing data.

The overall software architecture of the meta-ensemble approach, named CAGE-MetaCombiner, is illustrated in Figure 2.

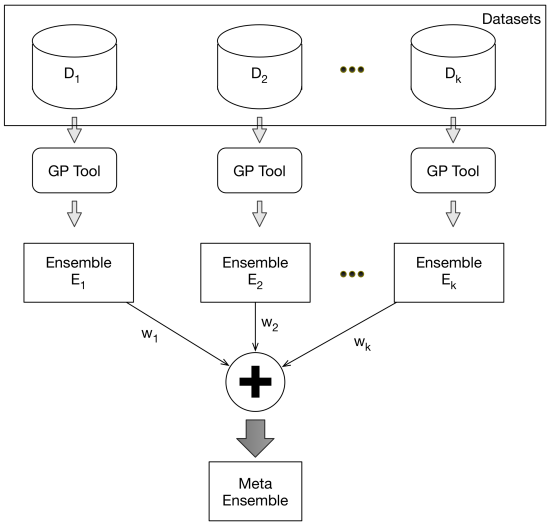

Figure 2: The software architecture of the meta-ensemble architecture.

In practice, an ensemble is built for each dataset by using a distributed GP tool, used

to generate the combiner function.This GP tool is a distributed/parallel GP implementation, named CellulAr GEnetic programming (CAGE) [17], running both on distributed-memory parallel computers and on distributed environments. The learning models (classifiers) composing the ensemble are taken from the well-known WEKA tool.

This tool is used to evolve the combiner functions, which the ensemble will adopt to classify new tuples. Implicitly, the function selects the more suitable classifiers/models to the specific datasets considered.

The different ensembles perform a weighted vote in order to decide the correct class. It is worth remembering that each ensemble evolves a function for combining the classifiers, which does not need any extra phase of training on the original data. The final classification is obtained computing the error using the same formulae as the Adaboost.M2 algorithm used by the boosting algorithm, by computing the error of the entire ensemble instead of a single classifier as in the original boosting algorithm.

Note that the approach is also suitable to work on incomplete datasets (named $D_1, D_2, \ldots, D_k$ in the figure). It is worth noticing that, as described in the background section, it is equivalent whether each dataset comes from a different source of data, or they are obtained from a partition of an incomplete dataset by removing groups of missing features. The only strong assumption is that each corresponding tuple of the different datasets is used to predict the same class, i.e., the class of the user of the scenario shown in section 4. The corresponding tuple can be missing in one or more datasets, but if it is missing in all the datasets, it will be discarded and counted as a wrong prediction in the evaluation phase.

The entire process can be summarized as follows, by considering a dataset partitioned into training, validation and test set:

1. The base classifiers are trained on the training set; then, a weight, proportional to the error on the training set, is associated with each classifier together with the support for each class, i.e. the decision support matrix is built. This phase could be computationally expensive, but it is performed in parallel, as the different algorithms are independent of each other.
2. The combiner function is evolved by using the distributed GP tool, CAGE, on the validation set. No extra computation on the data is necessary, as validation is only used to verify whether the correct class is assigned and consequently to compute the fitness function.
3. The final function is used to combine the base classifiers and classify new data (test set). This phase can be performed in parallel, by partitioning the test set among different nodes and applying the function to each partition.

A complete description of the meta-ensemble algorithm can be found in [18].

6 Experimental Results

In this section, the experiments conducted to analyze the capacity of our approach on handling missing features are described together with the main parameters and the main characteristics of the dataset used.

The software must be validated with real data to prove the validity of the approach. But it is very hard to find useful dataset with personal data and information about the behavioural profiles. The nature of the data limits the availability of public release of this kind of dataset. For this reason, we have chosen the UNIX dataset, described in [19]. It contains data about users working on linux consoles. The users are profiled according to their expertise with console

commands. The high number of features gives us the opportunity to test the performance of the algorithm for the case of missing features.

All the experiments were performed on a Linux cluster with 16 Itanium2 1.4GHz nodes, each with 2 GBytes of main memory and connected by a Myrinet high performance network. No tuning phase has been conducted for the GP algorithm, but the same parameters used in the original paper [20] were used, listed in the following: a probability of crossover of 0.7 and of mutation of 0.1, a maximum depth of 7, and a population of 132 individuals per node. The algorithm was run on 4 nodes, using 1000 generations and the original training set was partitioned among the 4 nodes. All the results were obtained by averaging 30 runs.

Two type of experiments were performed in order to evaluate the ability of the CAGE-MetaCombiner approach in handling missing features. The first type analyzes the behavior of the algorithm when we vary the percentage of tuples with missing data. The second one compares the performance of our software with the EvABCD algorithm [12], described in the introduction.

The dataset is preprocessed in the same way used in [12]. For each user we consider the first 100 and 500 commands used; then the commands subsequences of fixed length (from 3 to 6) are extracted from the list. Each user represents a record in the processed dataset and all the distinct subsequences are used as record attribute and the attribute value is the number of times the subsequence is typed by the user (0 means that the subsequence is never typed). In addition, the dataset is partitioned in four subsets, each one with the same number of features. It is worth remembering that we are interested in handling cases in which entire groups of features are missing and not in coping with random patterns of missing features. To simulate the missing data, for each partition a tuple can be removed according to a probability threshold, i.e., this parameter controls the percentage of tuples, which have missing attributes. For instance, if this parameter is set to 10%, the entire partition of the features belonging to this tuple has a probability of 0.1 to be missing. If all the partitions of a tuple are missing, this tuple will be considered as an error of classification. We choose the values for the threshold in the range 0-80%, with an interval of 10%, with 0% means no missing data.

Table 1: Classification accuracy (Percent) for Cage-MetaCombiner with the Unix dataset using different subsequence lengths and with different percentages of missing data.

Commands	Sequence	Percentage of Missing					
		0%	10%	20%	30%	40%	80%
100	3	85.73 ± 1.22	84.00 ± 1.33	83.60 ± 1.95	83.57 ± 2.07	83.20 ± 2.82	81.69 ± 3.00
	4	83.86 ± 0.81	83.73 ± 1.24	83.20 ± 1.74	83.10 ± 1.73	82.89 ± 1.51	82.11 ± 3.01
	5	83.96 ± 0.74	83.93 ± 0.81	83.73 ± 1.46	83.33 ± 1.62	83.21 ± 2.02	82.51 ± 2.66
	6	85.06 ± 0.53	84.23 ± 0.99	84.26 ± 1.29	83.60 ± 1.32	83.09 ± 2.01	82.38 ± 2.43
500	3	84.39 ± 7.36	84.08 ± 7.59	84.26 ± 6.93	83.80 ± 6.84	83.27 ± 5.38	82.01 ± 5.00
	4	83.86 ± 0.85	83.57 ± 2.15	82.93 ± 1.59	82.10 ± 2.12	81.91 ± 2.70	81.04 ± 3.28
	5	83.76 ± 1.04	83.73 ± 1.22	83.63 ± 1.51	83.30 ± 1.86	83.34 ± 2.08	82.11 ± 2.53
	6	83.76 ± 1.01	83.20 ± 1.41	83.17 ± 2.12	83.14 ± 2.36	83.10 ± 2.11	82.44 ± 2.68

In Table 1, we show the Cage-MetaCombiner performance on the Unix dataset using different subsequence lengths and different percentage of missing data. The classification rate for the 100 commands experiment is slightly better than for the case with 500 commands probably because the increase in the number of commands could lead to an increment in the number of features and consequently the training phase becomes harder than the previous case. Owing to the high number of features, the results are not much affected by the missing rate, that is the algorithm has good performance with all the percentages of missing data.

In Table 2 the results of comparison between Cage-MetaCombiner and EvABCD are re-

Table 2: Comparison of the Cage-MetaCombiner vs the EvABCD algorithm for the Unix dataset (classification accuracy).

Commands	Sequence	Cage-Combiner	EvABCD
100	3	85.73	64.90
	4	83.86	64.50
	5	83.96	67.90
	6	85.06	64.30
500	3	84.39	59.50
	4	83.86	59.20
	5	83.76	66.70
	6	83.76	70.80

ported [12]. The EvABCD algorithm is a technique for classification of the behavior profiles of users. As Cage-MetaCombiner, it learns different behaviors from training data. For a different profile, it builds one or more prototypes computing the frequency of all the sequences of commands with a defined length. Moreover, it updates the models in an incremental way. Our algorithm performs better than EvABCD in most cases. By using Cage-MetaCombiner, the results for a different number of commands extracted do not influence the accuracy much, while the EvABCD algorithm improves its accuracy when using 500 commands.

7 Conclusions and Future work

A general architecture for profiling users for better and more focused actions is proposed. The proposed methodology also works whether some logs, concerning the behavior of the users, are missing. In particular, a meta-ensemble-based framework for classifying datasets was designed and implemented. It is based on data mining and artificial intelligence (inspired to the evolutionary theory) techniques and permits to handle missing features, as each ensemble is specialized to operate with a different group of features. Experimental results, conducted on a real dataset, demonstrate the proposed system has a better accuracy in comparison with a correlated approach. In particular, the accuracy does not degrade significantly when the percentage of missing data increases. Future works aim to test the framework on a real environment.

Acknowledgment

This work has been partially supported by MIUR-PON under project PON03PE_00032_2 within the framework of the Technological District on Cyber Security.

References

[1] G. Folino, P. Sabatino, Ensemble based collaborative and distributed intrusion detection systems: A survey, J. Netw. Comput. Appl. 66 (C) (2016) 1–16. doi:10.1016/j.jnca.2016.03.011. URL http://dx.doi.org/10.1016/j.jnca.2016.03.011

[2] G. Folino, F. S. Pisani, P. Sabatino, A distributed intrusion detection framework based on evolved specialized ensembles of classifiers, in: Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part I, Vol. 9597 of Lecture Notes in Computer Science, Springer, 2016, pp. 315–331.

[3] CERT Australia, Cyber crime and security survey report, Tech. rep. (2012).

[4] V. Subrahmanian, M. Ovelgonne, T. Dumitras, B. A. Prakash, The Global Cyber-Vulnerability Report, 1st Edition, Springer, 2015.

[5] M. van Zadelhoff, The biggest cybersecurity threats are inside your company, Digital article - Harvard Business Review (2016).

[6] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[7] Y. Freund, R. Shapire, Experiments with a new boosting algorithm, in: Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Morgan Kaufmann, 1996, pp. 148–156.

[8] Y. Wang, Y. Gao, R. Shen, F. Yang, Selective ensemble approach for classification of datasets with incomplete values, in: Foundations of Intelligent Systems, Springer, 2012, pp. 281–286.

[9] R. Polikar, J. DePasquale, H. S. Mohammed, G. Brown, L. I. Kuncheva, Learn++. mf: A random subspace approach for the missing feature problem, Pattern Recognition 43 (11) (2010) 3817–3832.

[10] D. Godoy, A. Amandi, User profiling in personal information agents: A survey, Knowl. Eng. Rev. 20 (4) (2005) 329–361.

[11] S. Gauch, M. Speretta, A. Chandramouli, A. Micarelli, User profiles for personalized information access, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Vol. 4321 of Lecture Notes in Computer Science, Springer, 2007, pp. 54–89.

[12] J. A. Iglesias, P. P. Angelov, A. Ledezma, A. Sanchis, Creating evolving user behavior profiles automatically, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 854–867.

[13] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[14] G. Folino, F. S. Pisani, Combining ensemble of classifiers by using genetic programming for cyber security applications, in: Applications of Evolutionary Computation - 18th European Conference, EvoApplications 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings, Vol. 9028 of Lecture Notes in Computer Science, Springer, 2015, pp. 54–66.

[15] G. Folino, F. S. Pisani, P. Sabatino, An incremental ensemble evolved by using genetic programming to efficiently detect drifts in cyber security datasets, in: Genetic and Evolutionary Computation Conference, GECCO 2016, Denver, CO, USA, July 20-24, 2016, Companion Material Proceedings, 2016, pp. 1103–1110.

[16] R. J. A. Little, D. B. Rubin, Statistical Analysis with Missing Data, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[17] G. Folino, C. Pizzuti, G. Spezzano, Cage: A tool for parallel genetic programming applications, in: J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, W. B. Langdon (Eds.), Proceedings of EuroGP'2001, Vol. 2038 of LNCS, Springer-Verlag, Lake Como, Italy, 2001, pp. 64–73.

[18] G. Folino, F. S. Pisani, Evolving meta-ensemble of classifiers for handling incomplete and unbalanced datasets in the cyber security domain, Appl. Soft Comput. 47 (2016) 179–190.

[19] S. Greenberg, Using unix: Collected traces of 168 users, in: Research Report 88/333/45., Department of Computer Science, University of Calgary, Calgary, Canada., 1988.

[20] G. Folino, C. Pizzuti, G. Spezzano, A scalable cellular implementation of parallel genetic programming, IEEE Transactions on Evolutionary Computation 7 (1) (2003) 37–53.