Extracting of temporal patterns from data for hierarchical classifiers construction

Marcin Szpyrka*†, Adam Szczur†, Jan G. Bazan† and Łukasz Dydo†
*Department of Applied Computer Science,
AGH University of Science and Technology,
Mickiewicza 30, 30-059 Kraków, Poland
Email: mszpyrka@agh.edu.pl
†Interdisciplinary Centre for Computational Modelling,
University of Rzeszów
Pigonia 1, 35-310 Rzeszów, Poland

***Abstract*—A method of automatic extracting of temporal patterns from learning data for constructing hierarchical behavioral patterns based classifiers is considered in the paper. The presented approach can be used to complete the knowledge provided by experts or to discover the knowledge automatically if no expert knowledge is accessible. Formal description of temporal patterns is provided and an algorithm for automatic patterns extraction and evaluation is described. A system for packet-based network traffic anomaly detection is used to illustrate the considered ideas.**

***Keywords*—network anomaly detection, hierarchical classifiers, feature extraction, temporal patterns, LTL logic**

I. Introduction

Classification refers to the task of predicting a class label for a given unlabeled object. There are numerous approaches to constructing classifiers to be found in literature [1], [2], [3], [4]. A permanent growth of volume of gathered data and complexity of analyzed concepts necessitates new methods of data mining, process mining and classifiers constructing to meet the challenge of nowadays applications. As an example of such a challenge the telecommunication network traffic anomaly detection problem can be considered. The growing number of IP networks threats and the growing volume of transmitted data require new methods of network traffic data analysis. Nowadays, network anomaly detection is a very broad and heavily explored subject but the problem of finding a generic method for a wide range of network anomalies is still unsolved.

The most popular approaches are signature-based [5]. Signatures describe illegal patterns in network traffic and require expert knowledge about the given network threat given a priori. Such solutions do not cope with slightly modified or so-called 0-days attacks [6]. Another group of anomaly detection methods is based on entropy analysis [7], [8], [9]. These methods focus on analysis of traffic features distributions. Especially, parametrized entropy-based approaches (Tsallis entropy, Renyi etropy) have been a hot research recently. It is worth to mention that formal methods e.g. Petri nets have been successfully adapted for identification of cyber threats. For example colored Petri nets were also utilized for detection of DoS attacks in Wide Area Networks [10]. In this case colored Petri nets were adapted to model router network connections in the area of the United States. Another approach based on colored Petri nets and ontology can be found in [11] and [12]. This approach uses classifiers in the form of colored Petri nets modules and is able to cope with 0-days attacks.

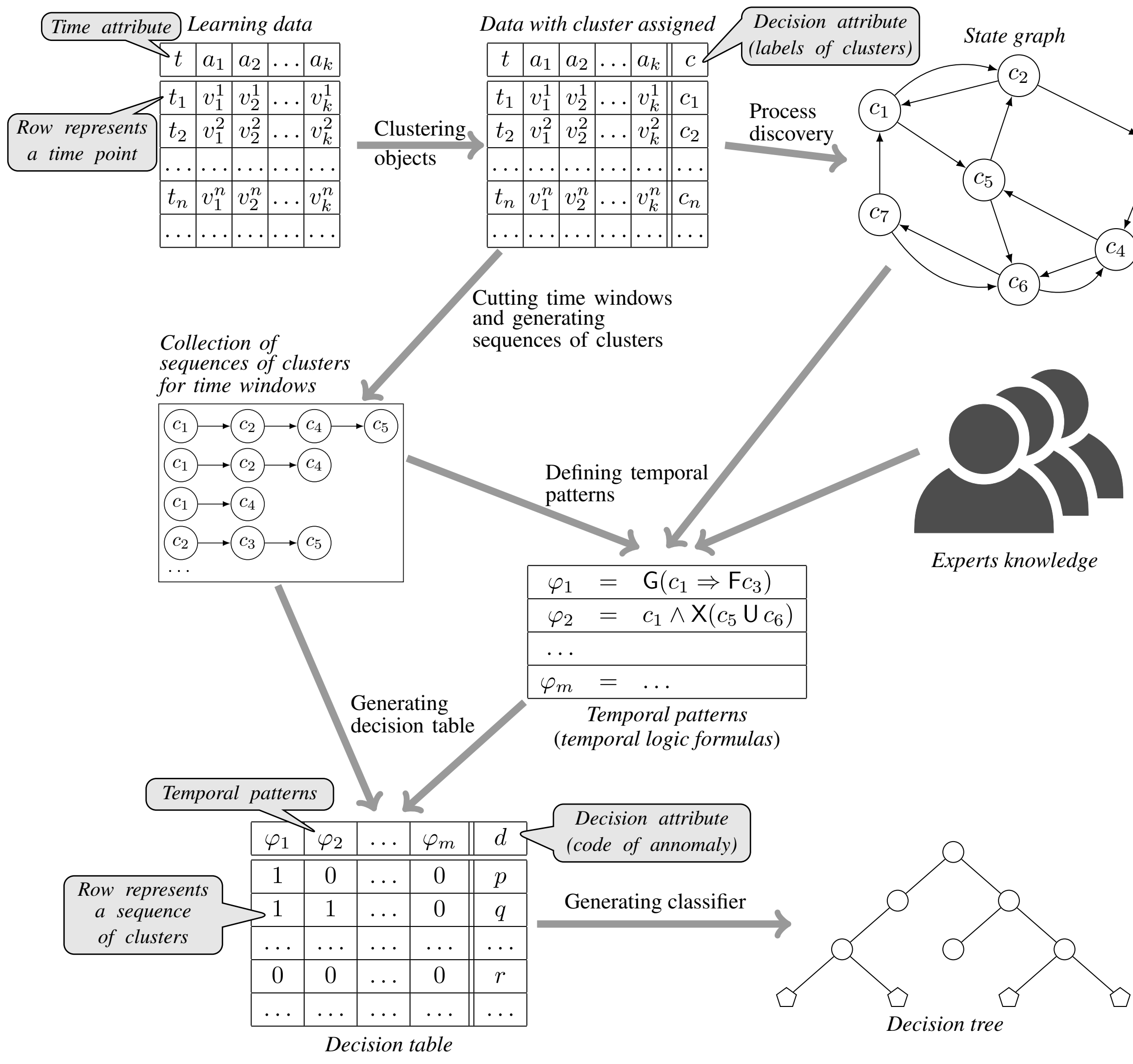
The approach considered in this paper is based on hierarchical behavioral patterns based classifiers initially described in [13]. The decision provided by such a classifier is based on a presence and/or absence of some temporal patterns in the considered data. The temporal patterns describe legitimate or illegitimate network traffic and has been successfully used to recognize selected network attacks [13]. The temporal patterns used in the previous approach have been defined by experts. In this paper we presents preliminary results of the method of extraction of temporal patterns automatically.

The paper is organized as follows. Section II presents an overview of the considered approach to classifiers construction. Syntax and semantics of temporal patterns are described in Section III. Temporal patterns extraction method is presented in section IV. A short summary is given in the final section.

II. Hierarchical behavioral patterns based classifiers – overview

Hierarchical behavioral patterns based classifiers (HBPB classifiers for short) have been proposed in [13]. They constitute a new approach for constructing classifiers from huge volume of temporal data. The novelty of the introduced method lies in a multi-stage approach to constructing hierarchical classifiers that combines process mining, feature extraction based on temporal patterns and constructing classifiers based on a decision tree.

The general scheme of constructing of HBPB classifiers is given in Fig. 1. Let $A = \{a_1, \ldots, a_k\}$ denote a set of attributes selected to describe important features of the system under consideration. The *learning data* used to construct a classifier take the form of tuples with values of these attributes in a sequence of time points $t_1, t_2, \ldots$. The learning data are stored in a table and each row of the table can be treated as an object. These objects (called here *time points*) are grouped with a *metric* that describes the similarity (distance) between time points. We use the well-known in literature k-means method



Time attribute
Learning data
Row represents a time point
Clustering objects
Data with cluster assigned
Decision attribute (labels of clusters)
State graph
Process discovery
Cutting time windows and generating sequences of clusters
Collection of sequences of clusters for time windows
Defining temporal patterns
Experts knowledge
$\varphi_1 = \mathsf{G}(c_1 \Rightarrow \mathsf{F} c_3)$
$\varphi_2 = c_1 \wedge \mathsf{X}(c_5 \,\mathsf{U}\, c_6)$
$\varphi_m = \dots$
Temporal patterns (temporal logic formulas)
Generating decision table
Temporal patterns
Decision attribute (code of annomaly)
Row represents a sequence of clusters
Generating classifier
Decision table
Decision tree

Fig. 1. Scheme of constructing of hierarchical classifiers

of clustering. The result of clustering are data with additional column of membership to clusters.

These data are considered from two points of view. In the first stage the whole data are used to construct the so-called *state graph*. Groups of time points (clusters) obtained from clustering process are represented as nodes in the state graph. If two consecutive time points belong to two different groups an arc is included into the graph that connects corresponding nodes. Multiple arcs going from node c_i to c_j are represented by a single arc. This stage allows us to cope with huge amount of time points and makes the approach scalable. In the consecutive stages, we use the clustered learning data sliced into pieces called *time windows*.

Temporal patterns take the form of LTL logic formulas [14], [15] (see Sec. III for more details). They represent some temporal dependencies between nodes of the state graph. The temporal patterns can be defined by experts using the generated state graph or can be extracted from the learning data automatically. The paper focuses on the latter approach e.i. extracting temporal patterns using the time windows generated from learning data.

Let $\Phi = \{\varphi_1, \dots, \varphi_m\}$ denote the set of temporal patterns. They are treated as input attributes for the classification problem. For a given time window a path is generated and for each behavior pattern it is checked whether the corresponding LTL formula holds for the path or not. Thus for a given path a sequence of m Boolean values is evaluated. Due to the fact that for learning data the values of the decision attribute are known, we use this information to create values of the decision attribute for each time window. Each time window provides a row for *high level learning data* described by conditional attributes $\varphi_1, \dots, \varphi_m$ and the decision attribute d. Finally, we remove duplicate rows from the high level data, to ensure scalability of our approach. The result are the reduced data, which are used to build the classifier based on a decision tree [13].

III. Temporal patterns

The decision provided by a HBPB classifier is based on a presence and/or absence of some temporal patterns in the considered time window. The *Linear-time Temporal Logic* (LTL) [14], [15] is used to describe the temporal patters. The logic provides modalities referring to time. Aside from the propositional logic operators, the temporal operators are: G (globally), F (finally), X (next), U (until).

Anomaly detection with an HBPB classifier is based on the analysis of sequences of clusters. Linear-time temporal logic seems to be the most convenient form of temporal patterns encoding. The four temporal operators are sufficient to describe presence or absence of some clusters in the analysed sequence, order of clusters etc. In addition to this, in case of finite sequences of clusters, it is easy to implement algorithms that check satisfiability of such formulas.

Let $C = \{c_1, \ldots, c_n\}$ denote the set of clusters i.e. nodes in the state graph. Let $\mathcal{T}$ denote the set of all temporal patterns. Temporal patterns are LTL formulae over the set C formed according to the following grammar.

$$\mathit{true}, \mathit{false} \in \mathcal{T} \tag{1a}$$

$$\forall_{c_i \in C}\ c_i \in \mathcal{T} \tag{1b}$$

$$\text{if } \varphi, \psi \in \mathcal{T} \text{ then } (\varphi),\ \neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,$$

$$\varphi \Rightarrow \psi,\ \varphi \Leftrightarrow \psi,\ \mathsf{X}\varphi,\ \mathsf{G}\varphi,\ \mathsf{F}\varphi,\ \varphi \,\mathsf{U}\, \psi \in \mathcal{T} \tag{1c}$$

The precedence order on operators is as follows: $\neg$; $\mathsf{X}, \mathsf{G}, \mathsf{F}$; $\wedge$; $\vee$; $\Leftrightarrow$; $\Rightarrow$; U. Parenthesis can be used to change order of a formula evaluation.

It should be emphasized that the length of any time window is finite. For a sequence of clusters $\pi = c^1 \ldots c^k$, $\pi[i]$ will denote the i-th element, $\pi[i\ldots]$ will denote the suffix of π starting from the i-th element and $\overline{\overline{\pi}}$ will denote the length of the sequence. Temporal patterns stand for properties of sequences of clusters. Any sequence can either satisfy a temporal pattern or not. Any sequence of clusters satisfies the formula *true*, and none satisfies *false*. Let $\varphi, \psi \in \mathcal{T}$ and $c_i \in C$. The satisfaction relation is defined as follows:

- $\pi \models c_i$ iff $\pi[1] = c_i$
- $\pi \models \neg\varphi$ iff $\pi \not\models \varphi$
- $\pi \models \varphi \wedge \psi$ iff $\pi \models \varphi$ and $\pi \models \psi$
- $\pi \models \varphi \vee \psi$ iff $\pi \models \varphi$ or $\pi \models \psi$
- $\pi \models \varphi \Rightarrow \psi$ iff $\pi \not\models \varphi$ or $\pi \models \psi$
- $\pi \models \varphi \Leftrightarrow \psi$ iff $\pi \models \varphi$ and $\pi \models \psi$ or $\pi \not\models \varphi$ and $\pi \not\models \psi$
- $\pi \models \mathsf{X}\varphi$ iff $\overline{\overline{\pi}} > 1$ and $\pi[2\ldots] \models \varphi$
- $\pi \models \mathsf{G}\varphi$ iff $\forall_{i \in 1\ldots\overline{\overline{\pi}}}\ \pi[i\ldots] \models \varphi$
- $\pi \models \mathsf{F}\varphi$ iff $\exists_{i \in 1\ldots\overline{\overline{\pi}}}\ \pi[i\ldots] \models \varphi$
- $\pi \models \varphi \,\mathsf{U}\, \psi$ iff $\exists_{i \in 1\ldots\overline{\overline{\pi}}}\ \pi[i\ldots] \models \psi$ and $\pi[j\ldots] \models \varphi$ for all $0 \le j < i$

It should be underlined that due to the clustering algorithm for any sequence π the following property holds:

$$\forall_{i \in 1\ldots\overline{\overline{\pi}}-1} \pi[i] \neq \pi[i+1] \tag{2}$$

```
$T' \leftarrow [\,]$                                   ▷ [ ] – empty list
$L' \leftarrow [\,]$
for all $x \in X$ do
  for all $b \in Bind(x)$ do
    for all $\pi \in S$ do
      if $\pi \models \varphi(x,b)$ then
        if $\varphi(x,b) \notin T$ then
          $T' \leftarrow T' + [\varphi(x,b)]$          ▷ list concatenation
          $L' \leftarrow L' + [1]$
        else
          $L'[\varphi(x,b)] \leftarrow L'[\varphi(x,b)] + 1$
        end if
      end if
    end for
  end for
end for
```

Fig. 2. Temporal patterns extracting algorithm – searching for patterns

If a time window with length k is used then for any sequence π, $1 \le \overline{\overline{\pi}} \le k$. This influence the interpretation of LTL formulas. Let us consider a few examples of temporal patterns with a single temporal operator. Let a sequence of clusters π be given:

- $\pi \models \mathsf{F}c_i$ iff π contains cluster c_i;
- $\pi \models \mathsf{F}(c_i \vee c_j \vee c_k)$ – iff π contains at least one cluster from the formula;
- $\pi \models \mathsf{G}c_i$ iff $\overline{\overline{\pi}} = 1$ and $\pi[1] = c_i$.
- $\pi \models \mathsf{G}(c_i \vee c_j \vee c_k)$ – iff π contains only clusters from the formula;
- $\pi \models \mathsf{X}c_i$ iff $\pi[2] = c_i$;
- $\pi \models c_i \,\mathsf{U}\, c_j$ iff $\pi[1] = c_j$ or $\pi[1] = c_i$ and $\pi[2] = c_j$.

IV. Temporal patterns extraction

The multi-stage approach to construction HBPB classifiers has been chosen to deal with a huge volume of learning data. Examples of such data include medical and financial data, data from vehicles monitoring, or data from telecommunication networks, e.g. information about packages flow. The clustering stage reduces the volume of data significantly. On the other hand, this stage provides some difficulties with defining temporal patterns that constitute the high level attributes for the construction of classifiers.

It seems that the simplest way to extract the patters is the analysis of the states graph by experts and expressing their knowledge in the form of suitable LTL logic formulas. The value of patterns defined by experts is undisputed, but in practice it hardly possible to define a suitable set of patterns at once. The preliminary set of features is usually updated many times to provide an acceptable level the classifier quality. To reduced the process of features extraction, we propose an automatic method that searches the set S of sequences of clusters generated for learning data and provides some rating of temporal patterns satisfied by the sequences.

The set $\mathcal{T}$ of all possible temporal patterns is infinite, thus it is not possible to check all of them. The temporal

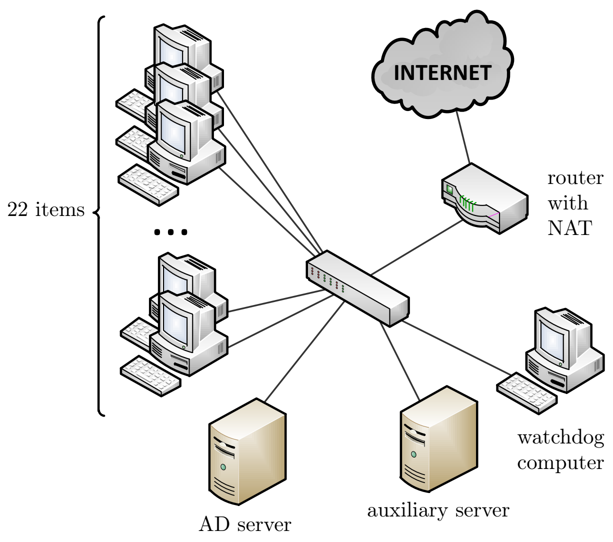


Fig. 3. Network topology

TABLE I. DATA ATTRIBUTES

Attribute name	Description
id	packet identifier
srcIP	source IP
srcPort	source port
destIP	destination IP
destPort	destination port
protocol	protocol
length	packet length (bytes)
time	packet transmission time
relTime	time from starting monitoring
info	short information about packet (from Wireshark)
srcMAC	source MAC address
dstMAC	destination MAC address
deltaTime	time difference between current and previous packet
ipFlags	IP flags
ttl	packet Time To Live
tcpFlags	TCP flags
icmpType	type of ICMP traffic
udpLength	UDP packet length

TABLE II. RELATIONSHIP BETWEEN CLUSTERS AND TYPES OF NETWORK ATTACKS

Clusters	Types of network attacks				
$c1$	0	1	2	3	4
$c4$	0	1	2	3	
$c7$	0	1	2	3	4
$c8$	0	1	2	3	4
$c10$	0				4
$c14$	0		2		4
$c16$	0	1	2	3	4
$c17$	0				4

patterns extracting algorithm starts with a set X of temporal patterns templates (see Fig. 2). A *temporal pattern template* is an LTL formula defined by (1), but containing some variables $v_1, \ldots, v_m$ instead of names of clusters. For a given template $x \in X$, let $Var(x)$ denote the set of variables that occur in x. A *binding* of a template x is a substitution b that replaces each variable of $Var(x)$ with a cluster $c \in C$. The set of all possible bindings of x will be denoted by $Bind(x)$. If $x \in X$ and $b \in Bind(x)$, then $\varphi(x, b)$ denotes the temporal pattern that is the result of replacing x variables with clusters' names defined by b.

The extraction algorithm presented in Fig. 2 for a given set X of temporal pattern templates checks for each formula $\varphi(x, b)$ how many times the formula is satisfied by the sequences from set S. The result is the list of temporal patterns that are satisfied by at least one sequence. We print the list ordered by values of list L' for convenience.

As a proof of concept for the approach considered in the paper a system for packet-based network traffic anomaly detection is used [13]. For the purpose of this work, a part of the university network was selected to capture data for analysis. The network topology is given in Fig. 3. The experiment environment consists of 22 work stations with Windows 7 operating system, Active Directory server, a watchdog computer (Windows 7) and auxiliary server with Windows Server 2008 R2 operating system. The NAT router has been used to separate the network from the whole university network and to provide an access to the Internet. The router is equipped with a mirror port used to send copies of all packets to the watchdog computer. The auxiliary server provides FTP (port 21), RDP (port 3389) and MySQL (port 3306) services. The Wireshark 1.10.7 software was used to monitor the network traffic. It provides the possibility of real time observing of sending and receiving packets for the given interface and to backup them to *pcapng* files. The typical network traffic generated by students lessons was captured as the legitimate traffic. Further to that, each day we generated four different network traffic anomalies including *network scan*, *IP-spoofed scanning* and *brute force*. It should be underlined that we used the network topology shown in Fig. 3, so IP-spoofed scanning was possible. The result of network traffic capturing was three 24-hours data sets in the form of *pcapng* files. Captured data were converted into *csv* files. Received time points were described with attributes presented in Table I. Finally, we received the state graph with 20 clusters. For more details on clustering method see [13].

Most of the clusters concern the legitimate traffic only. Thus, sequences that contain only such clusters do not point out any network attack. There are eight clusters that may point out some anomalies. They are shown in Table II, where 0 denotes the legitimate traffic.

The set S of sequences of clusters generated from learning data contains 6456 different elements. Based on the states graph we have chosen the following temporal patterns templates:

$$\mathsf{F}v_1$$
$$\mathsf{G}v_1$$
$$\mathsf{F}(v_1 \wedge \mathsf{X}v_2)$$

The templates describe temporal patterns that may point out the presence of some clusters in the considered sequence or point out a move from one cluster to another.

It follows that there are 800 temporal formulas to check. 704 of them hold for at least one sequence and only these sequences are considered in the following stages. Next, we remove from the set of temporal patterns these ones that concern cluster denoting legitimate traffic only. In other words, only temporal patters with at least one cluster from Table II may be useful for the classification process. In the considered example, 467 temporal patterns fulfill this requirement.

The set of 467 temporal formulas can be used by experts to select best temporal patterns according to their knowledge. The

TABLE III. RESULTS OF TEMPORAL PATTERNS EXTRACTION

Temporal pattern	Number of seq.	Temporal pattern	Number of seq.
$\mathsf{F}c1$	419	$\mathsf{F}(c8 \wedge \mathsf{X}c10)$	53
$\mathsf{F}c4$	4005	$\mathsf{F}(c8 \wedge \mathsf{X}c16)$	13
$\mathsf{F}c7$	4672	$\mathsf{F}(c8 \wedge \mathsf{X}c17)$	23
$\mathsf{F}c8$	1035	$\mathsf{F}(c10 \wedge \mathsf{X}c1)$	32
$\mathsf{F}c10$	1908	$\mathsf{F}(c10 \wedge \mathsf{X}c4)$	132
$\mathsf{F}c14$	927	$\mathsf{F}(c10 \wedge \mathsf{X}c7)$	515
$\mathsf{F}c16$	1665	$\mathsf{F}(c10 \wedge \mathsf{X}c8)$	138
$\mathsf{F}c17$	1094	$\mathsf{F}(c10 \wedge \mathsf{X}c14)$	54
$\mathsf{G}c7$	4	$\mathsf{F}(c10 \wedge \mathsf{X}c16)$	61
$\mathsf{F}(c1 \wedge \mathsf{X}c4)$	28	$\mathsf{F}(c10 \wedge \mathsf{X}c17)$	99
$\mathsf{F}(c1 \wedge \mathsf{X}c7)$	221	$\mathsf{F}(c14 \wedge \mathsf{X}c4)$	25
$\mathsf{F}(c1 \wedge \mathsf{X}c8)$	82	$\mathsf{F}(c14 \wedge \mathsf{X}c7)$	284
$\mathsf{F}(c1 \wedge \mathsf{X}c10)$	8	$\mathsf{F}(c14 \wedge \mathsf{X}c8)$	116
$\mathsf{F}(c4 \wedge \mathsf{X}c1)$	32	$\mathsf{F}(c14 \wedge \mathsf{X}c10)$	52
$\mathsf{F}(c4 \wedge \mathsf{X}c7)$	350	$\mathsf{F}(c14 \wedge \mathsf{X}c17)$	147
$\mathsf{F}(c4 \wedge \mathsf{X}c8)$	34	$\mathsf{F}(c16 \wedge \mathsf{X}c1)$	58
$\mathsf{F}(c4 \wedge \mathsf{X}c10)$	107	$\mathsf{F}(c16 \wedge \mathsf{X}c4)$	211
$\mathsf{F}(c4 \wedge \mathsf{X}c14)$	13	$\mathsf{F}(c16 \wedge \mathsf{X}c7)$	618
$\mathsf{F}(c4 \wedge \mathsf{X}c16)$	265	$\mathsf{F}(c16 \wedge \mathsf{X}c8)$	101
$\mathsf{F}(c4 \wedge \mathsf{X}c17)$	38	$\mathsf{F}(c16 \wedge \mathsf{X}c10)$	86
$\mathsf{F}(c7 \wedge \mathsf{X}c1)$	112	$\mathsf{F}(c16 \wedge \mathsf{X}c14)$	19
$\mathsf{F}(c7 \wedge \mathsf{X}c4)$	785	$\mathsf{F}(c16 \wedge \mathsf{X}c17)$	72
$\mathsf{F}(c7 \wedge \mathsf{X}c8)$	292	$\mathsf{F}(c17 \wedge \mathsf{X}c1)$	12
$\mathsf{F}(c7 \wedge \mathsf{X}c10)$	747	$\mathsf{F}(c17 \wedge \mathsf{X}c4)$	57
$\mathsf{F}(c7 \wedge \mathsf{X}c14)$	321	$\mathsf{F}(c17 \wedge \mathsf{X}c7)$	336
$\mathsf{F}(c7 \wedge \mathsf{X}c16)$	348	$\mathsf{F}(c17 \wedge \mathsf{X}c8)$	36
$\mathsf{F}(c7 \wedge \mathsf{X}c17)$	345	$\mathsf{F}(c17 \wedge \mathsf{X}c10)$	147
$\mathsf{F}(c8 \wedge \mathsf{X}c1)$	45	$\mathsf{F}(c17 \wedge \mathsf{X}c14)$	126
$\mathsf{F}(c8 \wedge \mathsf{X}c4)$	72	$\mathsf{F}(c8 \wedge \mathsf{X}c7)$	278

process of selecting temporal patterns from a set of predefined ones is significantly simpler than constructing such formulas from scratch. Moreover, an expert has a possibility to modify the formulas e.g. joining similar formulas into one.

In the automatic approach, to reduce further the size of the set of temporal patterns, we decided to keep only these temporal patterns that concern only clusters from Table II. Then any two temporal patterns were checked whether the same sequences of clusters satisfy both of them. In case of conformity rate more than 95%, only one of such formulas was chosen. The final set of 58 temporal patterns used for the classifier construction is shown in Table III.

To evaluate our classifier we use a well-known in literature train&test method and samples of data (25530 records as learning sample and 20097 records as test sample). The classifier we use is based on the so-called decision tree of the local discretization (see, e.g., [16], [17], [18]). It is a binary tree, created by multiple binary partitions (cuts) of the set of objects into two groups with the value of a selected attribute. We made three experiments using three measures of cut quality: pair indiscernibility, entropy and Gini index. Results for all measures are similar. The overall accuracy is 79,7%, while the accuracy of recognizing the legitimate traffic is 92,6%. In spite of the acceptable accuracy of anomaly traffic detection, the considered set of temporal patterns failed to distinguish one type of attack from another. This aspect needs further research and experiments with more elaborated temporal patterns.

V. CONCLUSION

Hierarchical classifiers considered in the paper combine process mining, extraction of attributes on the basis of temporal patterns and constructing classifiers based on decision trees methods. The paper focuses on the extraction of attributes stage. Some results on automatic extraction of temporal patterns used as input attributes for HBPB classifiers have been presented in the paper. The temporal patterns take form of LTL logic formula and describe some temporal properties of the system under consideration. To illustrate the presented approach a system for network traffic anomaly detection has been constructed. Preliminary results presented in the paper confirm usability of the method.

ACKNOWLEDGEMENT

This work was partially supported by the Polish National Science Centre grant DEC-2013/09/B/ ST6/01568 and by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland.

REFERENCES

[1] J. G. Bazan, "Hierarchical classifiers for complex spatio-temporal concepts," *Transactions on Rough Sets*, vol. 5390, no. IX, pp. 474–750, 2008.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2008.

[3] G. J. Nalepa and A. Ligęza, "The hekate methodology. hybrid engineering of intelligent systems," *Applied Mathematics and Computer Science*, vol. 20, no. 1, pp. 35–53, 2010.

[4] M. Zaki and W. Meira Jr., *Data Mining and Analysis. Fundamental Concepts and Algorithms*. New York, NY, USA: Cambridge University Press, 2014.

[5] Z. Li, A. Das, and J. Zhou, "USAID: Unifying signature-based and anomaly-based intrusion detection," in *Advances in Knowledge Discovery and Data Mining*, ser. LNCS, T. Ho, D. Cheung, and H. Liu, Eds. Springer Berlin Heidelberg, 2005, vol. 3518, pp. 702–712.

[6] B. Jasiul, M. Szpyrka, and J. Śliwa, "Detection and modeling of cyber attacks with Petri nets," *Entropy*, vol. 16, pp. 6602–6623, 2014.

[7] P. Bereziński, M. Szpyrka, B. Jasiul, and M. Mazur, "Network anomaly detection using parameterized entropy," in *Proceedings of the 13th International Conference on Computer Information Systems and Industrial Management Applications CISIM 2014*, ser. LNCS. Springer-Verlag, 2014.

[8] P. Bereziński, B. Jasiul, and Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, pp. 2367–2408, 2015.

[9] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, and D. Sornette, "Accurate network anomaly classification with generalized entropy metrics," *Computer Networks*, vol. 55, no. 15, pp. 3485–3502, 2011.

[10] L. Healy, "A model to study cyber attack mechanics and denial-of-service exploits over the internet's router infrastructure using colored Petri nets," http://commons.emich.edu/theses/218, Tech. Rep., 2009, masters Theses and Doctoral Dissertations.

[11] B. Jasiul, M. Szpyrka, and J. Śliwa, "Malware behavior modelling with colored petri nets," in *Computer Information Systems and Industrial Management Proceedings of the 13th IFIP TC8 International Conference CISIM 2014*, ser. LNCS. Springer-Verlag, 2014, vol. 8838, pp. 667–679.

[12] B. Jasiul, J. Śliwa, K. Gleba, and M. Szpyrka, "Identification of malware activities with rules," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, Warsaw, Poland, 2014.

[13] J. Bazan, M. Szpyrka, A. Szczur, Ł. Dydo, and H. Wojtowicz, "Classifiers for behavioral patterns identification induced from huge temporal data," *Fundamenta Informaticae*, 2015, (in print).

[14] C. Baier and J.-P. Katoen, *Principles of Model Checking*. London, UK: The MIT Press, 2008.

[15] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, Massachusetts: The MIT Press, 1999.

[16] J. Bazan, S. Bazan-Socha, S. Buregwa-Czuma, P. W. Pardel, and B. Sokolowska, "Predicting the presence of serious coronary artery disease based on 24 hour holter ecg monitoring," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS 2012), September 9-12, Wroclaw, Poland*, 2012, pp. 279–286.

[17] J. G. Bazan, H. S. Nguyen, S. H. Nguyen, P. Synak, and J. Wróblewski, in *Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems*, ser. Studies in Fuzziness and Soft Computing, L. Polkowski, T. Y. Lin, and S. Tsumoto, Eds. Heidelberg, Germany: Springer-Verlag/Physica-Verlag, 2000, vol. 56, pp. 49–88.

[18] H. S. Nguyen, "Approximate boolean reasoning: Foundations and applications in data mining," *LNCS Transactions on Rough Sets V*, vol. 4100, pp. 334–506, 2006.