Event Correlation on the basis of Activation Patterns

Peter Teufl*, Udo Payer†, Reinhard Fellner*

*Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Austria, 8010 Graz, Inffeldgasse 16a
Email: {Peter.Teufl@iaik.tugraz.at, felrei@sbox.tugraz.at}
†FH Campus02 University of Applied Science,
Austria, 8021 Graz, Koerblergasse 126
Email: {Udo.Payer@campus02.at}

***Abstract*—Intrusion Detection Systems (IDS) deploy various sensors that collect data, process this data and report events. The process of combining these events or superordinate incidences is known as *event correlation*. The key issues of this process are (1) to find a way how to combine events based on different data types (e.g. log entries, connection statistics or protocol identifiers), (2) to build a model representing the relations between the events and (3) to apply subsequent analysis that allow us to extract meaningful information from the trained model. In order to address these key issues, we introduce the concept of *Activation Patterns*. These patterns are generated by applying various techniques from machine learning and artificial intelligence to the raw event data. The presented technique is then integrated into an event correlation system. We describe the system and evaluate it by analyzing a popular intrusion detection data set consisting of a wide range of different features.**

I. Introduction

In the area of network security we often need to deploy intrusion detection systems with a large number of arbitrary sensors. The nature of the deployed sensors depends on the environment and the main focus of the intrusion detection system (e.g. network or host based). These sensors collect and process information and report events. The subsequent process of combining or correlating the events or information flows produced by the sensors is known as *event correlation*. This process should enable us to extract knowledge to detect attacks, gain insight about the relation between events, find clusters of similar co-occurring events, search for related behavior patterns and detect anomalies.
When dealing with the *event correlation* process, the following questions immediately arise: *(1) How can we combine sensor data with different scale or different units? (2) How can we combine discrete information with continuous data? And finally, if we are able to answer the previous questions, (3) how can we represent the correlated information and extract knowledge about the relations between the correlated events?*
In order to find an answer to these questions we introduce the concept of *Activation Patterns* and describe an event correlation system based on this technique. The key concept behind the proposed system is the transformation of the raw sensor information into these *Activation Patterns*. The *Activation Patterns* are the basis for subsequent analysis procedures that help us to get insight on the relations between the various events. The transformation process and the analysis are based on various techniques from machine learning and artificial intelligence.
In the following sections we give an introduction to the topic of event correlation and describe the details behind the concept of *Activation Patterns* and the proposed event correlation system. Finally, we use the introduced system to analyze real intrusion detection data – the KDD data set.

II. Event Correlation and Related Work

To be able to classify event correlation approaches, different classification schemas have been proposed. The classification schema we use in this section is taken from [3]. The graphical representation of this schema can be found in Figure 1:

1) *Models based on Artificial Intelligence* - Models of this group are based on human problem-solving capabilities. Most of all, *expert systems* are used, which attempt to reproduce the performance of a human expert, and which are traditional applications or subfields of artificial intelligence.

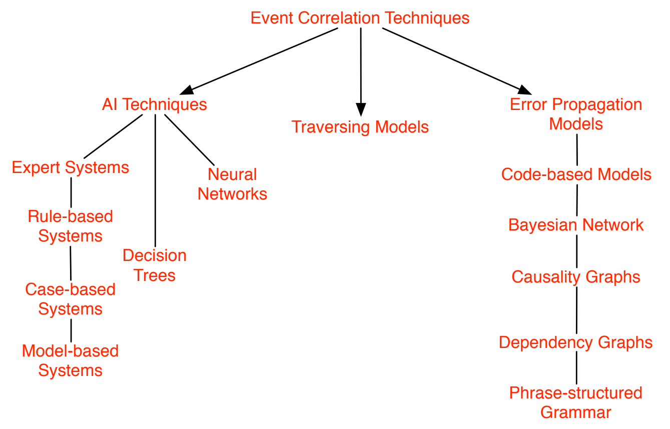

Fig. 1: ECS-techniques according to [3].

2) *Traversing Models* - Traversing is part of the field of graph-theory and is a method to identify a route whereas all nodes and edges of a graph are visited only once. A model which is based on ”traversing” needs a formal presentation of relations (edges) between the nodes, whereas to identify the ”*error behavior*”, the graph has to be processed in reverse order.
3) *Error Propagation Models* - Models of this group need to have predefined impact to other components of the model, caused by a specific error. Based on these predefined and related effects, we can determine the cause of a specific error.

When comparing these three classes, we have to note that models of class 2 and 3 are very expensive in terms of generation-costs, since changes in the event environment needs an adaption of the model. This does not differ from class 1, but models based on artificial intelligence can be trained automatically, which is much ”cheaper” than designing new models. Therefore, models of class 2 and 3 are often used in the field of telecommunication, since the model environment is quite simple and does not change very often.

The most frequently used ECS-approach is the *Rule-based Technique* whereas rules are applied to all incoming events of a system. It is obvious that these rules have to be predefined and therefore it is clear that such systems can only be used efficiently if systems events do not change very often or not at all. Therefore, these systems do not play an important role in the field of adaptive IDSs.

Another very often used approach are *Case-based Techniques*, which utilized the knowledge about past events to fill a *case library*. New incoming events are compared to events stored in this library and if no prior event can be found in the database, the best matching known event is taken as a starting point for a *problem solution process*. Once this process is completed, the result is stored in the *case library* for future use [4] (see also Figure 2).

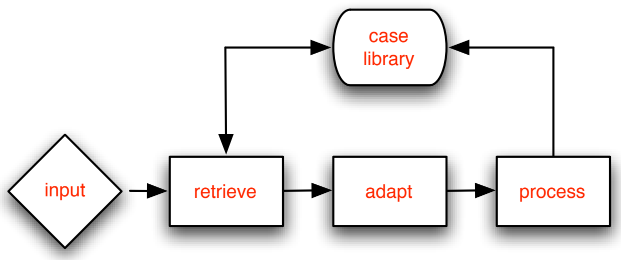

Fig. 2: Case based EC architecture - according to [4].

While only a few Decision-Tree based AI-approaches have been designed [5], several Neural Network based approaches can be found [6], [7]. But the problem with neural networks is to chose the correct level of model-complexity. This means that it is difficult to determine the right number of hidden units or correct regularization parameters. Coming along with the complexity of neural networks is the problem of ”overfitting”[1], which leads to models with bad generalizing.

The proposed event correlation system and the concept of *Activation Patterns* belongs to the broader category of AI Techniques.

III. Methods

Before we can describe the concept of *Activation Patterns* and how these patterns are used in the proposed system, we need to give a short introduction on the employed techniques. For the details on their combination in the proposed event correlation system, we refer the reader to the next section. The processes that transform raw sensor data into *Activation Patterns* and all subsequent analysis procedures are based on various techniques from the areas of machine learning

[1]Designing a statistical model that has too many parameters. This also reduces or destroys the ability of the model to generalize beyond the fitting data

and artificial intelligence. The following three building blocks are required for the generation of *Activation Patterns* and their subsequent analysis:

Associative networks [8] are directed or undirected graphs that store information in the network nodes and use edges (links) to present the relation between these nodes. Typically, the links are weighted according to a weighting scheme. Associative networks play an important role within Information Retrieval (IR) systems such as [9], and [10].

Spreading activation (SA) algorithms [11] can be used to extract information from associative networks. The process activates an arbitrary number of nodes within the associative network and spreads the activation according to the strength of the links to neighboring nodes.

Unsupervised learning algorithms group similar data into clusters without any external information such as class labels. There are algorithms for the clustering of *nominal features* and *distance-based features*. However, for the generation of the *Activation Patterns* and the analysis of these patterns we only require cluster algorithms for *distance-based features* features. Due to advantages for the later described generation process of the *Activation Patterns*, we employ a highly sophisticated Neural Gas [12] based algorithm – Robust Growing Neural Gas (RGNG) [13].

IV. How do we correlate events?

Before we are able to describe the idea behind *Activation Patterns* we need to focus on the structure of the data represented by events and why this is important for subsequent analysis procedures. In the case of intrusion detection systems, events are generated by various sensors depending on the nature of the monitored data. Thereby, the possible types of sensors are endless (see Figure 3): There could be very simple sensors that report the protocol (e.g. TCP, UDP) of all seen connections, or the number of connections to all or a specific system, or watch system logs and report certain or all log entries. Other sensors might employ complex mathematical functions to gain information about the monitored data, or even monitor the output of other event correlation systems. Regarding the distribution of sensors, they could be placed on a single host, on the whole Internet, on different protocol layers, in hardware or software.

Regardless of the nature of the employed sensors we are able to map the features of events thrown by all possible sensors to two basic types:

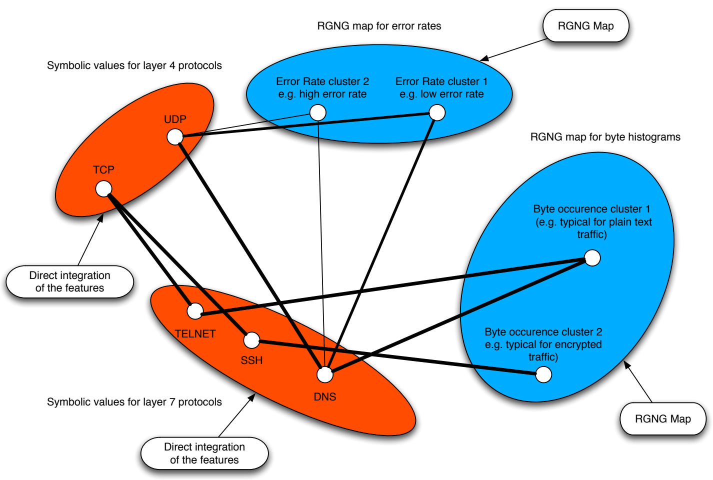
RGNG map for error rates
RGNG Map
Symbolic values for layer 4 protocols
Error Rate cluster 2 e.g. high error rate
Error Rate cluster 1 e.g. low error rate
UDP
TCP
RGNG map for byte histograms
Byte occurence cluster 1 (e.g. typical for plain text traffic)
Direct integration of the features
TELNET
SSH
DNS
Byte occurence cluster 2 e.g. typical for encrypted traffic)
Symbolic values for layer 7 protocols
RGNG Map
Direct integration of the features

Fig. 3: Combination of *nominal* and *distance-based* feature values and relations between these features

- *Distance-based features:* In this case it makes sense to define a distance measure such as the Euclidean distance or the cosine similarity between the values reported by the sensor: Examples are error rates, histograms (e.g. byte occurrence in network traffic) or timing based values.
- *Nominal features:* In this case it does not make sense to define a distance measure. Examples are log entries or protocol identifiers (e.g. TCP, UDP, ICMP).

This distinction plays an important role for *Event Clustering*, where similar behavior patterns (e.g. normal behavior, attack types, etc.) are clustered in an unsupervised way. The following key issues arise when we take a closer look on these two types of features and the unsupervised clustering process:

- *Different value ranges for distance-based features:* In this case the features with values that have a wide range have more influence on the distance between two concepts and therefore, important details from other features might be ignored.
- *Different dimension of output vectors generated by sensors:* The number of output values depends on the nature of the sensor and might range from single values (e.g. duration of a connection) to large high dimensional histograms (e.g. 256 entries for a byte occurrence histogram). This variance is a problem

for the unsupervised algorithm, since larger feature vectors have more influence on the distance to other concepts than feature vectors with a small number of entries.

- *Combination of nominal and distance-based features:* Unsupervised algorithms can either cluster *distance-based features* features or *nominal features*, the combination of both features is not possible. Therefore, the *distance-based features* features need to be transformed into *nominal features* by applying some kind of discretization.
- *Further Analysis: Event Clustering* forms an important part of the analysis process. However, we also want to get an insight on how events are related, we want to execute semantic search queries that allow us to specify certain features and leave others undefined and we want to be able to detect anomalies. This cannot be done by relying on unsupervised clustering algorithms and their generated models alone.

The first two issues can partly be addressed by applying normalization operations to the information gained from different sensors. The third issue could be addressed by applying discretization operations to the *distance-based features* values. However, these schemas are very simple and cannot be used for more complicated analysis procedures such as anomaly detection. Even if we address these three issues the forth issue still remains and leads to the following question: *How are we able to build a model that allows us to apply all of the described analysis procedures?* The following section will introduce the concept of *Activation Patterns* that will help us to find an answer to this question.

V. Activation Patterns

In order to build a model for the described analysis procedures, we transform the raw event data consisting of different *distance-based features* and *nominal features* into *Activation Patterns* by introducing five layers depicted in Figure 4. The general idea is to store the event information in an associative network, where the nodes represent the event values and the links between these nodes represent the relations and their strength between these nodes. By applying SA strategies, we are able to generate *Activation Patterns*, which can be used for the subsequent analysis. For the description of the five layers we assume that we have an arbitrary number of data vectors, where each of these vectors stores information about events that share some kind of relation. For this work the co-occurrence

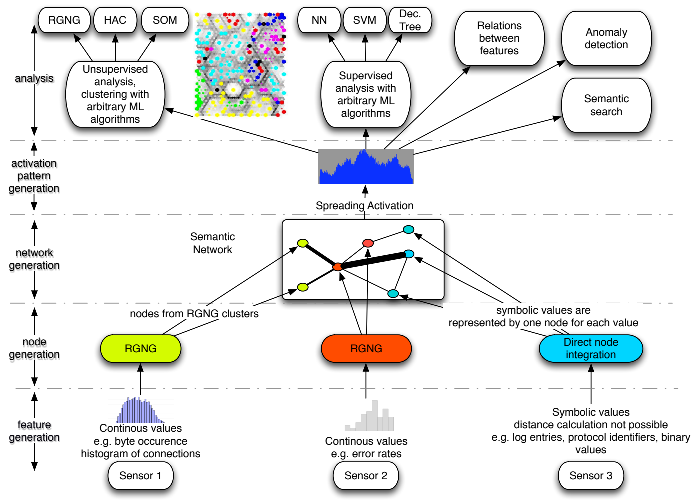
RGNG
HAC
SOM
NN
SVM
Dec. Tree
Relations between features
Anomaly detection
analysis
Unsupervised analysis, clustering with arbitrary ML algorithms
Supervised analysis with arbitrary ML algorithms
Semantic search
activation pattern generation
Spreading Activation
Semantic Network
network generation
nodes from RGNG clusters
symbolic values are represented by one node for each value
node generation
RGNG
RGNG
Direct node integration
feature generation
Continous values e.g. byte occurence histogram of connections
Continous values e.g. error rates
Symbolic values distance calculation not possible e.g. log entries, protocol identifiers, binary values
Sensor 1
Sensor 2
Sensor 3

Fig. 4: The whole event correlation framework

of different events within a given time frame is used as relation. Obviously, the number of events per data vector does not need to be constant.

L1 - Feature extraction: As mentioned before, features of any data set can be separated into the categories *distance-based features* and *nominal features*. These two types of features are handled differently by subsequent processing steps and need to be identified correctly in *L1*. For *distance-based features* we have the option to create groups that represent features with similar meanings and value ranges. This grouping is not a requirement for further analysis, but reduces the computational complexity.

L2 - Node generation: This process layer creates the nodes of the associative network that will be generated in the next layer. The process of mapping feature values to nodes depends on the type of the particular feature. For *nominal features* the possible values are directly mapped to separate nodes. For *distance-based features* we need to apply some kind of discretization operation to map values onto nodes. Although there is a wide range of discretization algorithms available, we employ a different strategy. We apply the Robust Neural Growing Neural Gas (RGNG) algorithm to the *distance-based* values and use the found prototypes as nodes for the associative network.

L3 - Network generation: In this layer the links between nodes are created according to their relations in the following way:

1) The features are analyzed according to the two categories determined in L1. *Nominal features* are directly mapped to nodes according to the mapping from the previous step. For *distance-based features* (single values or groups) we find the prototype of the corresponding RGNG-map that has the smallest distance to the data vector. This prototype is called the Best Matching Unit (BMU). Its corresponding node is found according to the mapping generated in L2.
2) All these nodes are now linked within the associative network. Newly created links between two nodes are initialized with weight 1. The weight of existing links is increased by 1. This linking represents the co-occurence of different values of distinct features. The link weight represents the strength of this relation.

In order to extract information from the network, the next layer utilizes the SA-algorithm. However, before this processing step we need to normalize the link weights within the associative network, so that the maximum weight is equal to 1. We can apply different strategies here that normalize the links locally or globally.

L4 - Activation Pattern Generation: The associative network resulting from L3 represents relations between features. The values of the features are represented as nodes. The information about relations can be extracted by applying the SA-algorithm to the network. For each data vector, we can determine those nodes in the network that represent the values stored in the data vector. By activating these nodes for a given data vector, we can spread the activation over the network according to the links and their associated weights for a predefined number of iterations. After this spreading process, the activation value for each node is extracted and presented in a vector - the *Activation Pattern*. The areas of the associative network that are activated and the strength of the activation gives information about which feature values occurred and how they co-occur.

L5 - Analysis: The *Activation Patterns* are the basis for the following analysis procedures:

- *Unsupervised clustering and search queries:* Due to the transformation of the raw data into *Activation Patterns* we can apply standard distance-based unsupervised clustering algorithms. This allows us to find clusters of similar behavior patterns (e.g. attacks, normal behavior) and to deduce common features within a cluster. By varying the model complexity, we are able to build a hierarchy from a very coarse grained categorization down to a very detailed representation of system behaviors. The distance information between the *Activation Patterns* can be used to implement semantic search algorithms that retrieve similar behavior patterns. These search queries can also be used to specify certain events and find closely related patterns (e.g. which patterns correspond to the protocol FTP and a packet loss rate of 80%).
- *Relations between events:*. The associative network allows us to define arbitrary relations between events (e.g. based on time or location). By activation one or more nodes (corresponding to events) within the associative network, and spreading their activation via the links to the neighbors, we are able to extract details about the relations between events. Based on this information we would also be able to increase/decrease the link strength and thereby change the influence of certain events.
- *Anomaly detection:* The associative network stores the relations between events and how this events co-occur in the training data set. Anomalies – either new events that were not present during the training phase, or events that co-occur in a way that was not seen before – can be detected by analyzing the total energy of activation functions. By applying these procedures we are able to ignore known/normal behavior and concentrate on anomalies.

VI. Case Study: Applying this EC Mechanism to KDD

To explain the correlation mechanism in detail, the proposed approach will be discussed on the base of the KDD data set [14], which was published by the Massachusetts Institute of Technology (MIT) in 1999 in the context of a machine learning competition. Due to space constraints, we will focus our discussion on the unsupervised analysis procedure and leave the other techniques (semantic search queries and anomaly detection) to future work. For the visualization of the high dimensional *Activation Patterns*, we employ Self Organizing Maps (SOM).

The analyzed data set contains the network traffic of a dedicated network, which was exposed to attacks for a duration of nine weeks. During this period, several GB of data were collected by the following Sensors:

- TCPDump: A very well known tool to store and analyze network traffic
- Basic Security Module (BSM): A tool, which is used to monitor the OS-kernel. BSM can be used as a host based IDS.
- System file dumps: Different tools are used to generate dump- and log-files.

The raw data of the KDD data set was gathered in 1999 and was published to the attendees of the competition in a slightly modified form. Data packets—*belonging to the same connection*—have been separated from the rest and have been stored in separate files. Thereafter, a fraction of these data files was offered to the attendees of the competition as a set of training data, while a small set was retained as test data. After the competition (which was won by Dr. Bernhard Pfahringer from the Austrian Research Institute for Artificial Intelligence) the whole data set was published and is down to the present day a very popular data set in the field of network intrusion detection.
One of the reason for its popularity is that most of the test data is based on simple DOS attacks and so it is very easy to build DOS-detection models with a hit rat higher than 95%. But among the huge number of DOS samples, the data set also contains a small number of U2R[2] and R2L[3] attacks. Table I gives an idea about the small number of U2R- and R2L attacks compared to the frequent occurrence of probing- and DOS attacks.

	training	**test**
normal	19.69%	19.48%
probe	0.83%	1.34%
DOS	79.24%	73.90%
U2R	0.01%	0.07%
R2L	0.23%	5.20%

TABLE I: U2R and R2L attacks are chosen rare

Each entry in the KDD-data set describes a connection by using 41 features. These features can be subdivided into four groups:

1) *Connection related features:* Features of this group contains information about a peer-to-peer connections (e.g.: service, protocol. duration, etc.)
2) *Content related features:* Packet content has been separated, classified and extended.
3) *Time based features:* Based on predefined time intervals, similarities between pairs of peer-to-peer connections have been analyzed and interpreted (e.g. the number of connections to the same target within a predefined time interval).
4) *Host based features:* These features contains the long term analysis of network connections and are summarized in some sort of statistical information (e.g. the percentage of connections using malformed flags)

[2]User to root (U2R): unauthorized access to local super user

[3]Remote to local (R2L): unauthorized access from a remote machine

To demonstrate the proper application of the proposed EC-mechanism, the KDD data set was used, since KDD offers a nice mixture of *distance-based features* and *nominal features*.

After applying the five layers of the event correlation system and thereby transforming the raw event data into *Activation Patterns* we visualize the high dimensional data of Table I in Figure 5. As expected, a large part of the data set is normal traffic (magenta), which flocks together in the north-east part of the presented SOM. When giving a closer look at Figure 5, we can see that the rest of the SOM is covered with DOS attacks (green). U2R- (blue) and R2L attacks (red) are forming small and well defined areas, but are mixed up with normal traffic. This means that there are no big differences in feature values. Only probing attacks (yellow) are forming two delimited clusters, which are not mixed with normal traffic.
Thus, when applying unsupervised clustering mechanisms, it is very easy to find clusters—*which are located far from each other*—but which belong to the same category.

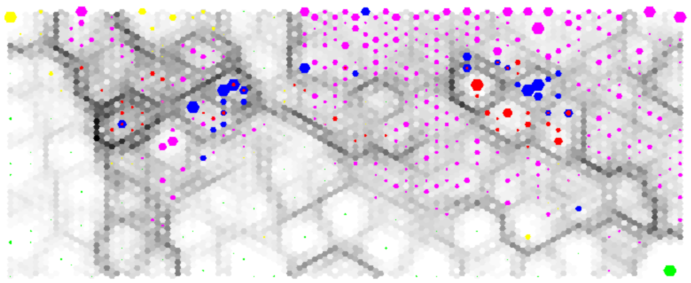
Fig. 5: Visualization of 10% of the KDD data set.

But we are interested in finding clusters—*belonging to the same attack*—without having any tagged information. In doing so, we want to utilize the *joint occurrence* of *not obviously related* features, by the help of associative networks. Thus, the actual problem we want to solve is to *correlate events* (or *features*), which are of different nature, but closely related to each other.

Normally, we do not know the relation between a single instance and the corresponding class. Therefore, it is important to keep the number of classes low, which needs to be allocated manually. To demonstrate the effect of a high number of clusters, the detection rate of an unreduced set of 168 clusters—*generated by RGNG*—is shown in Table II.

Cluster	*normal*	*probe*	*DOS*	*U2R*	*R2L*
168	0.9960	0.9560	0.9485	0	0.7176

TABLE II: Detection rate of a pure RGNG analysis

What we can see in Table II is, that even in the case of a maximal number of clusters (168) no U2R attacks can be detected. This effect is due to the low number of instances (16) and the similarity of U2R- to R2L attacks.

Assuming the case that we do not have information about the relation between a single instance and the corresponding class, the following mechanism can lead to the desired results:

1) Generate a model—*which is not too complex*—by using a standard unsupervised clustering mechanism. The complexity of the requested model should be based on the number of expected attacks.
2) Determine the cluster for all instances of the data set and determine the corresponding class. Find a mapping between all classes and clusters and repeat this step for all remaining clusters.

Since the complexity of RGNG can be preset, the number of detected clusters can be controlled. In doing so, models of different complexity can be generated, which are shown in Table III:

Cluster	*normal*	*probe*	*DOS*	*U2R*	*R2L*
11	0.9836	0.6406	0.3766	0	0
30	0.9839	0.8826	0.7282	0	0.2443
45	0.9953	0.8875	0.8899	0	0.2443
168	0.9960	0.9560	0.9485	0	0.7176

TABLE III: Number of found clusters - controlled by the RGNG-complexity

A useful upper-bound for the complexity of RGNG are 30 clusters, since we know that there are 23 different attacks in the data set plus several types of normal traffic. To get an impression of all process steps in the special case of 30 RGNG clusters, the classification steps are shown in Figure 6, Figure 7 and Figure 8.
First, RGNG was used to determine 30 different clusters, which are enumerated randomly in Figure 6, without any relation to the classification. Thus, the SOM BMU[4]-colors are meaningless and do not represent any class IDs. As described above, the next step is to count the number of different instances within each single cluster. As a result of this counting, the whole cluster is assigned to those class with the highest number of counts. In doing so, eight clusters can be identified, which are shown in Figure 8.

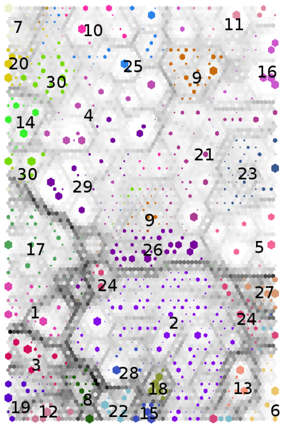

Fig. 6: Real class distribution.

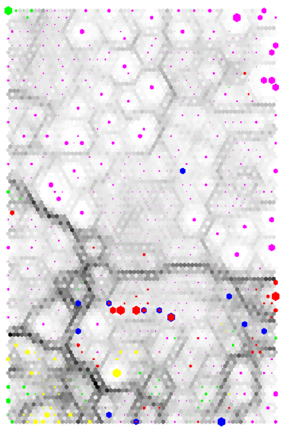

Fig. 7: Real classes mapped to 30 RGNG clusters.

These eight clusters are: 3 (yellow in Figure 8), 7 (green in Figure 8), 8, 12, 18, 19, 27 and 28. All remaining clusters (magenta) are classified as normal traffic. Now, we can identify attacks in the real traffic—*which cannot be classified by the proposed mechanism*—if colored BMUs in Figure 7 are compared to BMUs in Figure 8. Notable—*for instance*—is an area close to 24 (red and blue in 7) which was not identified by RGNG as a unique cluster. Furthermore, cluster 24 (blue in 7) which was obviously similar to

[4]Best Matching Unit (BMU)

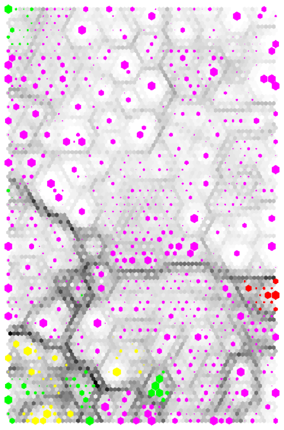
Fig. 8: Classification results if RGNG-complexity is set to 30 clusters.

normal traffic so that it was not classified as a separate cluster. The reason for this is that the distances between different clusters are too small (e.g. 24 vs. 2). This fact is represented by a grey or light grey border in the map. Strongly separated clusters are separated by a dark grey or even black border between clusters.

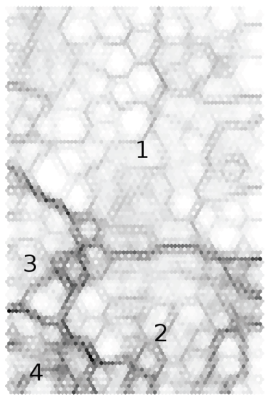

Fig. 9: 4 main areas in the map.

VII. Case Study Results

Beside the topology in the map, we are also interested in the distribution of single attacks. Figure 10 shows the result of 8 detected attacks and the corresponding areas in the map (generated by RGNG-30). The mapping between real attacks and detected clusters is shown in Table IV. This table represents the fact that such as in the case of a dedicated DOS attack (e.g. DOS-19) the set of ”generating attacks” consists of: Neptune (148), Sata (10) and Portsweep (2). This is due to the unsupervised RGNG, which does not know the fine differences between these three attacks. Thus, RGNG groups them together to a new class of DOS attacks.

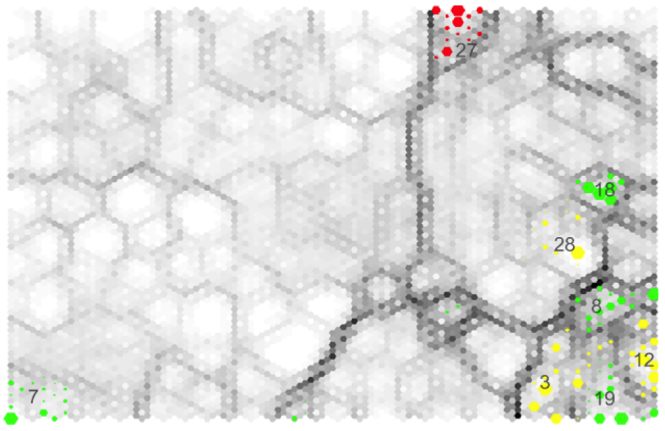

Fig. 10: Detected attacks with a RGNG complexity of 30 clusters

Detected Cluster	*Attack 1*	*Attack 2*	*Attack 3*	*Attack 4*	*Attack 5*
R2L (27)	Warezcl.-32	Norm.-27	-	-	-
Probe (3)	Portsw.-93	NMAP-11	Norm.-2	-	-
DOS (8)	Neptune-24	NMAP-14	Portsw.-8	Norm.-3	Land-2
DOS (7)	Back-208	Norm.-119	-	-	-
DOS (18)	Smurf-28	Norm.-5	-	-	-
Probe (28)	IPsweep-116	Pod-16	NMAP-6	Norm.-1	-
Probe (12)	Satan-138	Portsw.-3	-	-	-
DOS (19)	Neptune-148	Satan-10	Portsw.-2	-	-

TABLE IV: Mapping between detected attack and causing attacks in the data set

When mapping all detected instances to the whole set of attacks, we get detection rates as shown in Table V.

Attacks	*# of Inst,*	*Detected Inst,*	*Detection Rate*
Neptune	179	172	96.0%
Back	220	208	94.5%
Portsweep	104	93	89.4%
IPsweep	124	110	88.7%
Satan	158	138	87.3%
Smurf	39	28	71.8%
Warezclient	102	32	31.3%

TABLE V: Detected attacks in a RGNG-map with a complexity of 30 clusters

Related to these detection results, we can make the following observations:

- There are no clear borders between different clusters. Instances of a specific cluster can also be found in at least one neighboring cluster.

- Sometimes, the level of mixing is very high. In the case of R2L, we have a mixing ratio of 27 : 32.
- Attack detection is highly influenced by the *service*- and *flag* feature. But in the map, we can also find different clusters with identical *service*- and *flag* features. The reason—*why they are different*—is due to strong differences in the remaining features.
- The identified attack Neptune is distributed between two clusters. In doing so, cluster 19 contains the bigger part of all instances, while cluster 8 contains a set of different groups.
- There is a little number of clusters containing different attacks of the same class. Nevertheless, if it is the case (e.g. cluster 12 and 28), then the number of mixed instances is very low.

Motivated by these observations, we now want to give a closer look to the clusters in the map and the corresponding attacks:

In the case of a **"normal data traffic"**, we can identify a large part of clusters, which do not differ too much. This means that the borders between clusters of normal traffic are mostly light grey or grey. The smallest distance between two clusters within area 1 is—*regarding to the shade of grey varies of the border cells*—in the north east corner of the map (see Figure 9). When giving a look to Figure 6, we can identify these clusters as cluster number 11 and 16. When computing the mean value of all feature vectors of these two clusters and computing the difference, we can see that these two clusters only differ in a single feature. While borders between clusters in area 1 are very smooth, the separators in area 2 are more distinctive, which is represented by "turbulent" feature vector differences.

Due to the fact that the R2L-detection rate is very bad (only about 24%), the ability to detect **"Warezclient"** merits a closer examination. In doing so, the following facts have been analyzed:

- *Missing instances*: The detected cluster (27) only contains 30% of all Warezclient instances.
- *Data mixing*: The ratio between Warezclient- and normal traffic instances is almost identical.
- *Borders to normal data traffic*: The question is, if distances to neighboring clusters are sufficient.

As already mention, more than 60% of all Warezclient instances are not in cluster 27, which was used to detect this attack. Therefore, it would be interesting to **"localize the missing instances"**.

In Figure 11 we can see a lot of feature vector-differences[5] between Warezclient instances of cluster 27 and vectors of Warezclient instances in cluster 2. These differences results in a distinctive separation of clusters 2 and 27.

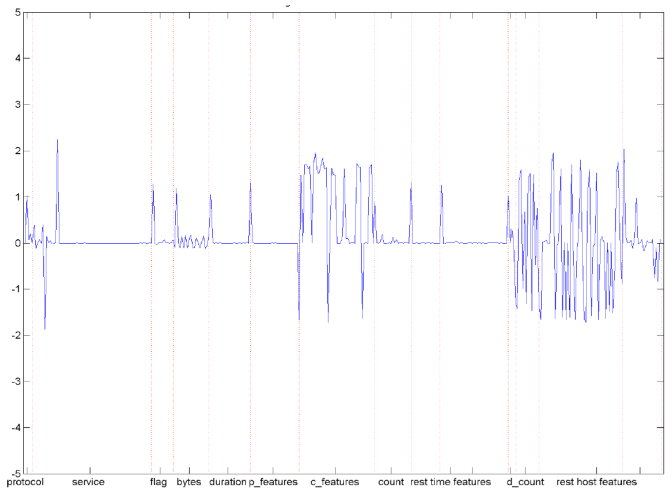

Fig. 11: Differences between Warezclient clusters 27 and 2

A closer look to the original data and a comparison of the original data with the data generating clusters 27 and cluster 2 resulted in the following observations:

- Warezclient uses the FTP-service to move from one infected node to the next one. Thus, the use of FTP is an necessary indicator, but FTP exists in two different ways in the data set (ftp, ftp_data). This leads to the confusion of the RGNG classification mechanism.
- There is a big difference in the number of sent- and received byte, but also in the duration of a connection. In all cases where we can detect *service=FTP* (cluster = 27), the duration of such connections are always longer than connections with *service=FTP_data*. One reason for this is that *service=FTP* also includes the receipt of data, while *service=FTP_data* only sends data.
- When giving a look at Figure 11, we can see a difference between time based traffic features (count, rest_time_features, *where we can see two peaks up*) and host based traffic features (d_count, rest_host_features, *where we can see a lot of differences in both directions*). This behavior—*which*

[5]Differences in feature vectors, which identify different clusters, are represented in a graph to demonstrate similarities or differences.

was observed in a small set of training data—can also be observed in the original data set.

- There are also big differences in content based features (c_features). In this particular case, two features are highly affected: hot and is_hot_login. Both features can be clearly assigned to one of the services *FTP* or *FTP_data*. Very interesting are also variations in the associative network, which can be generated by a bad combining of features or by co-activation of nodes in the associative net.

Thus, based on these observations, we can say that the clustering mechanism did nothing wrong, since due to the properties of the given data, several instances of a single attack was classified. In addition, if normal data looks (almost) identical to an attack, which we want to identify automatically, machine learning techniques reach the end of their ability.

VIII. Conclusions

In this paper, we introduced the concept of *Activation Patterns* in the field of *event correlation*. The basic idea was to use associative networks to describe *relations* between events or features, which cannot be directly processed by classical classification methods. By applying *spreading activation*, we are able to deduce *Activation Patterns* that allow us to apply conventional unsupervised clustering algorithms, to execute semantic search queries, to detect anomalies and to extract information about the relations between events.
Based on the well known KDD data set, we concentrated primarily on the unsupervised clustering analysis and presented the applicability of the proposed approach. In almost all instances, the mechanism performed very well. Only in the case of a single attack, the mechanism did not generate the expected results. Therefore, we investigated into the bad detection rate of "Warezclient". The reason for this was—*on the one hand*—the similarity to normal traffic but also the numerous different types of a single attack *on the other hand*.

IX. Acknowledgements

We especially want to thank P. N. Suganthan for providing the Matlab sources of RGNG [13].

References

[1] S. Klinger, S. Yemini, Y. Y. ad D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Proceedings of the fourth international symposium on Integrated network management IV*, 1995, pp. 266–277.

[2] G. Jakobson and M. Weissman, "Real-time telecommunication network management: extending event correlation with temporal constraints," 1995, pp. 290–301.

[3] M. Steinder and A. Sethi, "The present and future of event correlation: A need for end-to-end service fault localization," http://citeseer.ist.psu.edu/steinder01present.html, 2001.

[4] L. M. Lewis, "A case-based reasoning approach to the resolution of faults in communication networks," in *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*. North-Holland Publishing Co., Amsterdam, 1993, pp. 671–682.

[5] K. Ali, "On the link between error correlation and error reduction in decision tree ensembles," Department of Information and Computer Science University of California, Irvine, CA, Tech. Rep., 1995.

[6] M. G. DONDO, N. JAPKOWICZ, and R. SMITH, "A neural network event correlation approach," in *Proceedings of SPIE, the International Society for Optical Engineering*, 2006.

[7] H. Wietgrefe, K. dieter Tuchs, and G. Carls, "Using neural networks for alarm correlation in cellular phone networks," in *In Proc. International Workshop on Applications of Neural Networks in Telecommunications*, 1997.

[8] M. R. Quillian, "Semantic memory," Cambridge, MA, pp. 227–270, 1968.

[9] C. Fellbaum, "Wordnet: An electronic lexical database (language, speech, and communication)," Hardcover, May 1998. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/026206197X

[10] G. Tsatsaronis, M. Vazirgiannis, and I. Androutsopoulos, "Word sense disambiguation with spreading activation networks generated from thesauri," January 2007. [Online]. Available: http://www.ijcai.org/papers07/Abstracts/IJCAI07-279.html

[11] F. Crestani, "Application of spreading activation techniques in information retrieval," pp. 453–482, 1997.

[12] T. Martinetz and K. Schulten, "A "neural gas" network learns topologies," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: Elsevier, 1991, pp. 397–402.

[13] A. K. Qin and P. N. Suganthan, "Robust growing neural gas algorithm with application in cluster analysis," *Neural Netw.*, vol. 17, no. 8-9, pp. 1135–1148, 2004.

[14] U. of California Irvine, "Kdd cup 1999 data."