EAI Endorsed Transactions
on Security and Safety
Research Article ICST.ORG

A novel intrusion detection method based on OCSVM and K-means recursive clustering[★]

Leandros A. Maglaras[1,*], Jianmin Jiang[1]

[1]University of Surrey, Department of Computing, Guildford, UK

Abstract

In this paper we present an intrusion detection module capable of detecting malicious network traffic in a SCADA (Supervisory Control and Data Acquisition) system, based on the combination of One-Class Support Vector Machine (OCSVM) with RBF kernel and recursive k-means clustering. Important parameters of OCSVM, such as Gaussian width σ and parameter ν affect the performance of the classifier. Tuning of these parameters is of great importance in order to avoid false positives and over fitting. The combination of OCSVM with recursive k-means clustering leads the proposed intrusion detection module to distinguish real alarms from possible attacks regardless of the values of parameters σ and ν, making it ideal for real-time intrusion detection mechanisms for SCADA systems. Extensive simulations have been conducted with datasets extracted from small and medium sized HTB SCADA testbeds, in order to compare the accuracy, false alarm rate and execution time against the base line OCSVM method.

Keywords: Cyber security, SCADA systems, support vector machine, machine learning

Received on 27 October 2014; accepted on 30 October 2014, published on 30 January 2015

doi:10.4108/sesa.2.3.e5

1. Introduction

Several techniques and algorithms have been reported by researchers for intrusion detection [2, 3]. One big family of intrusion detection algorithms is rule based algorithms [4]. In real applications though, during abnormal situations, the behavior of the system cannot be predicted and does not follow any known pattern or rule. This characteristic makes rule based algorithms incapable of detecting the intrusion.

Generally, anomaly detection can be regarded as binary classification problem and thus many classification algorithms which are utilized for detecting anomalies, such as neural networks, support vector machines, K-nearest neighbor (KNN) and Hidden Markov model can be used. However, strictly speaking, they are not intrusion detection algorithms, as they require knowing what kind of anomaly is expecting, which deviates the fundamental object of intrusion detection. In addition these algorithms may be sensitive to noise in the training samples.

Segmentation and clustering algorithms [5] seem to be better choices because they do not need to know the signatures of the series. The shortages of such algorithms are that they always need parameters to specify a proper number of segmentation or clusters and the detection procedure has to shift from one state to another state. Negative selection algorithms [6] on the other hand,are designed for one-class classification; however, these algorithms can potentially fail with the increasing diversity of normal set and they are not meant to the problem with a small number of self-samples, or general classification problem where probability distribution plays a crucial role. Furthermore, negative selection only works for a standard sequence, which is not suitable for on line detection. Other algorithms, such as time series analysis are also introduced to anomaly detections, and again, they may not be suitable for most of the real application cases.

To minimize the above mention drawbacks an intelligent approach based on OCSVM [One-Class Support Vector Machine] principles are proposed for intrusion detection. OCSVM is a natural extension of the support vector algorithm to the case of unlabeled data, especially for detection of outliers. The OCSVM algorithm maps input data into a high dimensional feature space (via a kernel) and iteratively finds the maximal margin hyperplane which best separates the training data from the origin (Figure 1).

OCSVM principles have shown great potential in the area of anomaly detection [7–9]. IDS can provide active detection and automated responses during intrusions [10]. Commercial IDS products such as NetRanger, RealSecure, and Omniguard Intruder alert work on attack signatures. These signatures needed to be updated by the vendors on a regular basis in order to protect from new types of attacks. Most of the current intrusion

[★]This is an extended version of the paper [1]
[*]Corresponding author. Email: l.maglaras@surrey.ac.uk

detection commercial softwares are based on approaches with statistics embedded feature processing, time series analysis and pattern recognition techniques. Several extensions of OCSVM method have been introduced lately [11, 12].

OCSVM similar to other one-class classifiers e.g. GDE[13], PGA [14], suffer from false positive and over fitting situations. Intrusion detection systems (IDS) fail to deal with all kinds of attacks, while on the other hand, false alarms that are arisen from high sensitive IDS arise high economic risks. These situations are described in subsection 1.1

1.1. motivation

For the OCSVM with an RBF kernel, two parameters σ and ν need to be carefully selected in order to obtain the optimal classification result. A common strategy is to separate the data set into two parts, of which one is considered unknown. The prediction accuracy obtained from the unknown set more precisely reflects the performance on classifying an independent data set. An improved version of this procedure is known as cross-validation. Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

In ν-fold cross-validation [15, 16], the training set is divided into ν subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining ν .. 1 subsets . Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The cross-validation procedure can prevent the over fitting problem.

Using an ensemble of decision mechanisms with different parameters, is another method to have an optimal result. An ensemble of classifiers [17] is a set of classifiers whose individual decisions are combined in some way. more trusted final decision. Ensemble systems of classifiers are widely used for intrusion detection in networks. Classifier ensemble design aims to include mutually complementary individual classifiers which are characterized by high diversity either in terms of classifier structure [18], internal parameters [19] or classifier inputs [20].

Unnthorsson et al. [21] proposed another method to select parameters for the OCSVM. In their method, ν was first set to a user-specified allowable fraction of misclassification of the target class (e.g. 1% or 5%), then the appropriate σ value was selected as the value for the classification accuracy curve of training samples first reaches $1 - \nu$. The obtained ν and σ combination can then be used in the OCSVM classification.

OCSVM similar to other one-class classifiers suffer from false positive and over fitting. The former is a situation that occurs when the classifier fires an alarm in the absence of real anomaly in the system and happens when parameter σ has too large vale. The latter is the situation when a model begins to memorize training data rather than learning to generalize from trend and it shows up when parameter σ is given relatively small value [22].

In this article we propose the combination of OCSVM method with a recursive k-means clustering, separating the real from false alarms in real time and with no pre-selection of parameters σ and ν.

1.2. Contributions

The present article develops a intrusion detection method, namely the *K-means OCSVM ($\mathcal{K}-\mathcal{OCSVM}$)*. Using the well known OCSVM method with default values for parameters σ and ν we distinguish real from false alarms with the use of a recursive k-means clustering method. This is very different from all previous methods that required pre-selection of parameters with the use of cross-validation or other methods that ensemble outcomes of One class classifiers.

The article makes the following contributions:

- A new one class classifier $\mathcal{K}-\mathcal{OCSVM}$ is proposed.
- The proposed classifier combines a OCSVM with RBF kernel with a recursive K-means clustering method.
- $\mathcal{K}-\mathcal{OCSVM}$ separates in real time false from real alarms.
- A performance evaluation of the proposed method against a OCSVM classifier with different parameters is conducted, which attest the stability of the new structure.

The rest of this article is organized as follows: Section 2 describes the OCSVM method. In Section 3 the use of OCSVM in SCADA systems is presented; Section 4 presents the features extracted from the network traces for the testing and training of the model Section 5 describes the $\mathcal{K}-\mathcal{OCSVM}$ method; Section 6 presents the simulation environment and results. In Section 7 discusses possible enhancements of the method. Section 8 concludes the article.

2. OCSVM method

The one-class classification problem is a special case of the conventional two-class classification problem, where only data from one specific class are available and well represented. This class is called the target class. Another class, which is called the outlier class, can be sampled very sparsely, or even not at all. This smaller class contains data that appear when the operation of the system varies from the normal, due to a possible attack. In general cases, the outlier class might be very difficult or expensive to measure. Therefore, in the one class classifier training process, mainly samples from the target class are used and there is no information about its counterpart. The boundary between the two classes has to be estimated from data in the only available target class. Thus, the task is to define a boundary around the target class, such that it encircles as many target examples as possible and minimizes the chance of accepting outliers.

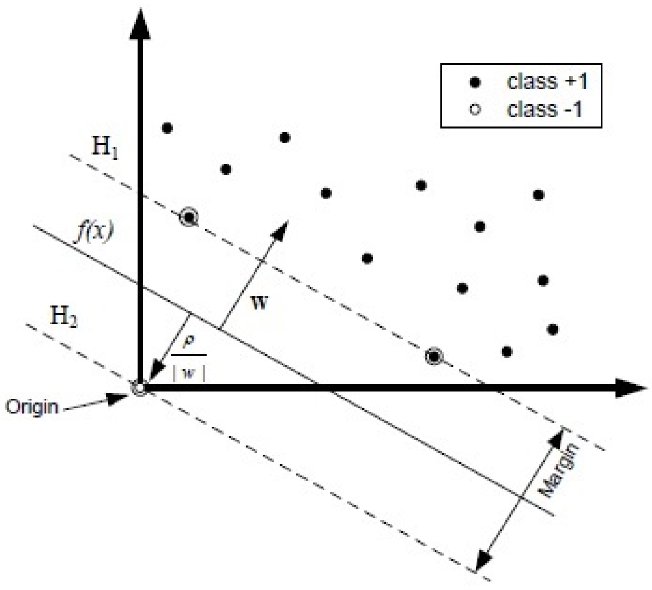

Figure 1. OCSVM maps input data into a high dimensional feature space

Scholkopf et al. [23] developed an OCSVM algorithm to deal with the one-class classification problem. The OCSVM may be viewed as a regular two-class SVM, where all the training data lie in the first class, and the origin is taken as the only member of the second class. The OCSVM algorithm first maps input data into a high dimensional feature space via a kernel function and then iteratively finds the maximal margin hyperplane, which best separates the training data from the origin. Thus, the hyperplane (or linear decision boundary) corresponds to the classification function

$$f(x) =< x, w > +b \tag{1}$$

where w is the normal vector and b is a bias term. The OCSVM solves an optimization problem to find the function f with maximal geometric margin. This classification function can be used to assign a label to a test example x. If $f(x) < 0$, then x is labeled as an anomaly (outlier class), otherwise it is labeled normal (target class).

Using kernel functions, solving the OCSVM optimization problem is equivalent to solving the following dual quadratic programming problem.

$$min \frac{1}{2} \sum_{i,j} a_i a_j K(x_i, x_j) \tag{2}$$

subject to $0 \leq a_i \leq 1/\nu l$, and $\sum_i a_i \leq 1$.

Parameter a_i is a Lagrange multiplier, which can be thought of as a weight for example x, such that vectors associated with non-zero weights are called support vectors and solely determine the optimal hyperplane, ν is parameter that controls the trade-off between maximizing the number of data points contained by the hyperplane and the distance of the hyperplane from the origin, l is the number of points in the training dataset, and $K(x_i, x_j)$ is the kernel function.

Using the kernel function to project input vectors into a feature space, nonlinear decision boundaries are allowed. Generally, four types of kernel are often used: linear, polynomial, sigmoid and Gaussian radial basis function (RBF) kernels. In this paper, we use the RBF kernel, which has been commonly used for the OCSVM.

$$K(x_i, x_j) = exp(-\sigma||x_i - x_j||^2), \quad \sigma > 0 \tag{3}$$

Although the OCSVM requires samples of the target class only as training samples, some studies showed that when negative examples (i.e. samples of outlier classes) are available, they can be used during the training to improve the performance of the OCSVM. In this paper only normal data were used for the training of the method, though a similar to the one proposed by Tax [24], which includes a small amount of samples of the outlier class, will be also applied and evaluated in the near future.

3. OCSVM for SCADA system

Cyber-attacks against SCADA systems [25] are considered extremely dangerous for Critical Infrastructure (CI) operation and must be addressed in a specific way [26]. Presently one of the most adopted attacks to a SCADA system is based on fake commands sent from the SCADA to the RTUs. OCSVM [27, 28] possesses several advantages for processing SCADA environment data and automate SCADA performance monitoring, which can be highlighted as:

- In the case of SCADA performance monitoring, which patterns in data are normal or abnormal may not be obvious to operators. Since OCSVM does not require any signatures of data to build the detection model it is well suited for intrusion detection in SCADA environment.

- Since the detection mechanism does not require any prior information of the expected attack types, OCSVM is capable of detection both known and unknown (novel) attacks.
- In practice training data, taken from SCADA environment, could include noise samples. Most of the classification based intrusion detection methods are very sensitive to noise. However, OCSVM detection approach is robust to noise samples in the training process.
- Algorithm configuration can be controlled by the user to regulate the percentage of anomalies expected.
- Due to the low computation time, OCSVM detectors can operate fast enough for online SCADA performance monitoring.
- Typically monitoring data of SCADA systems consists of several attributes and OCSVM is capable of handling multiple attributed data.

4. Attribute extraction

Feature extraction is essential in a classification problem. In order to train the OCSVM module properly we used network trace files from a secure wireless network in University of Surrey. Based on previous analysis of data [29] we selected some initial features that are used as attributes for our OCSVM model .

Attributes in the network traces datasets have all forms: continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence pre-processing was required before pattern classification models could be built. Pre-processing consists of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes and second step implemented scaling The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation.

Based on the above observations we used the network trace dataset in order to test our OCSVM module. The attributes that we used for this initial training / testing phase, were rate & packet size. The values were scaled to the range [0,1].

The rate (1^{st} attribute) was calculated using the equation 4:

$$Rate_{scaled} = \frac{Time\ difference}{Max\ time\ difference} \tag{4}$$

Time difference is calculated by the difference of time of current packet and the time of previous packet injected in the system.

The packet size (2^{nd} attribute) was scaled using the equation 5:

$$Packet_{scaled} = \frac{packet\ size}{Max\ packet\ size} \tag{5}$$

5. $\mathcal{K}-\mathcal{OCSVM}$

The proposed $\mathcal{K}-\mathcal{OCSVM}$ combines the well known OCSVM classifier with the RBF kernel with a recursive K-means clustering module. Figure 2 illustrates the procedure of intrusion detection of our proposed $\mathcal{K}-\mathcal{OCSVM}$ model.

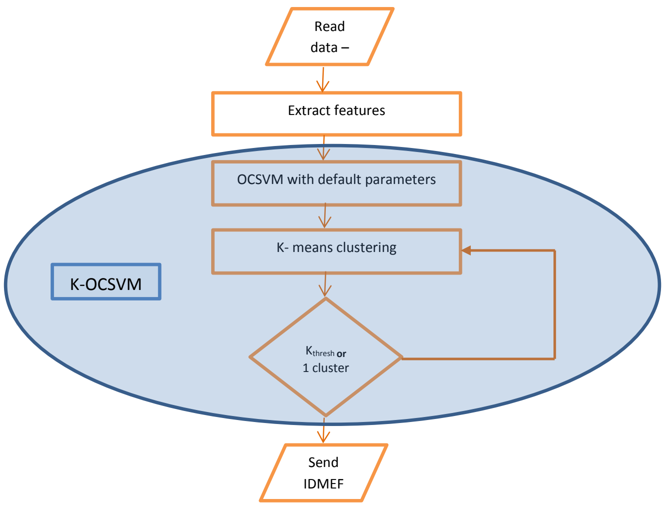

Figure 2. $\mathcal{K}-\mathcal{OCSVM}$ module

The OCSVM classifier runs with default parameters and the outcome consists of all possible outliers. These outliers are clustered using the k-means clustering method with *2* clusters, where the initial means of the clusters are the maximum and the minimum negative values returned by the OCSVM module. From the two clusters that are created from the K-means clustering, the one that is closer to the maximum negative value (severe alerts) is used as input in the next call of the K-means clustering. This procedure is repeated until all outcomes are put in the same cluster or the divided set is big enough compared to the initial one, according to the threshold parameter k_{thres}.

K-means clustering method divides the outcomes according to their values and those outcomes with most negative values are kept. That way, after the completion of this recursive procedure only the most severe alerts are communicated from the $\mathcal{K}-\mathcal{OCSVM}$. The division of the data need no previous knowledge about the values of the outcomes which may vary from *-0.1* to *-160* depending of the assigned values to parameters σ and ν. The method can find the most important/possible outliers for any given values to parameters σ and ν.

One important parameter that affects the performance of $\mathcal{K}-\mathcal{OCSVM}$ is the value of threshold k_{thres}. For given value *2*, the final cluster of severe alerts that the method communicates to other parts of the IDS system is limited to *2* to *4* alarms. For bigger value (*3* or more) the number of alerts also rises till the method degrades to the initial OCSVM. The optimal value for the given parameter k_{thres} is a matter for future investigation.

In order to cooperate with the other modules the OCSVM module needed to be integrated in the *PID* system and communicate with the other modules. Once an intrusion is detected several actions can be taken by the *IDS* (intrusion detection system. These actions include recording of intrusions in log files, sending of alert messages, limit the bandwidth of the intruder or even block all connections from the intruder. In order to better cooperate with the other components/modules that are being produced in the *CockpitCI* project the OCSVM model sends IDMEF [30] files.

6. Performance evaluation

6.1. Training of OCSVM model

The initial training of both the OCSVM and the $\mathcal{K}-\mathcal{OCSVM}$ modules is conducted using several trace files

- a trace file that is sniffed out of a typical wireless network (Figure 3) that consists of *10.000* lines each representing a packet send in the network.
- datasets of a testbed under normal operation
- datasets of a medium sized SCADA system under normal operation

To train the OCSVM, we adopt the *RBF* for the kernel equation. This kernel nonlinearly maps samples into a higher dimensional space so it can handle the case when the relation between class labels and attributes is nonlinear.

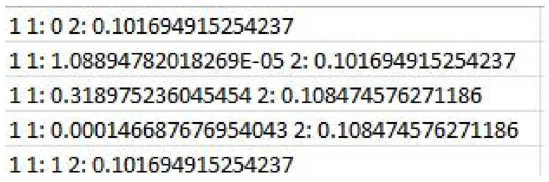
```
1 1: 0 2: 0.101694915254237
1 1: 1.08894782018269E-05 2: 0.101694915254237
1 1: 0.318975236045454 2: 0.108474576271186
1 1: 0.000146687676954043 2: 0.108474576271186
1 1: 1 2: 0.101694915254237
```

Figure 3. Format of the transformed Network trace file

The training model that is extracted after the training of the OCSVM is used for on line detection of malicious data. Since the model is based on features that are related to network traffic, and since the traffic of the system varies from area to area and from time period to time period, possible generation of multiple models could improve the performance of the module.

The network traffic in electric grids varies according to the activity which is not constant during the day. Also in some areas the activity follows different patterns according to the local demand. These characteristics maybe be critical for the proper training of the module and the accurate detection of intruders.

6.2. Testing of OCSVM model

We evaluate the performance of the method using data from the wireless network of the University campus, from a testbed that mimics a small-scale SCADA system and from a Hybrid testbed of a medium sized SCADA system. The parameters used for the evaluation of the performance of $\mathcal{K}-\mathcal{OCSVM}$ are listed in Table 1.

Parameter	*Range of Values*	*Default value*
σ	0.1 - 0.0001	0.007
ν	0.002 - 0.05	0.01
k_{thresh}	2 - 3	2

Table 1. Evaluation Parameters

Wireless network. In order to test our model we use another network trace file sniffed from the wireless network. The testing trace file consists of it 30.000 lines. We compare the performance of our proposed model against OCSVM classifiers having the same values for parameters σ and ν. We name each OCSVM classifier according to the parameters σ and ν : $OCSVM_{0.07,0.01}$ stands for OCSVM classifier with parameters $\sigma =$ *0.07* and $\nu =$ *0.01*.

In Table 2 we show the number of observed anomalies detected from OCSVM and $\mathcal{K}-\mathcal{OCSVM}$ respectively. From this table it is shown how parameters σ, ν affect the performance of OCSVM. Even for a value of ν equal to *0.005*, OCSVM produces over *400* possible attacks, making the method inappropriate for a SCADA system where each false alarm is costly.

Parameter σ	*Parameter* ν	$\mathcal{K}-\mathcal{OCSVM}$	*ocsvm*
0.007	0.002	3	408
0.007	0.01	3	299
0.007	0.005	2	408
0.0001	0.01	3	274
0.1	0.01	2	295

Table 2. Performance evaluation of $\mathcal{K}-\mathcal{OCSVM}$ and OCSVM for $K_{thers} = 2$.

In figure 4 we present the outcome that OCSVM produces for the training network trace under different values of parameters σ and ν. From this figure it is obvious that the outcome is strongly affected by the values of these parameters, making $\mathcal{K}-\mathcal{OCSVM}$ necessary tool for proper intrusion detection.

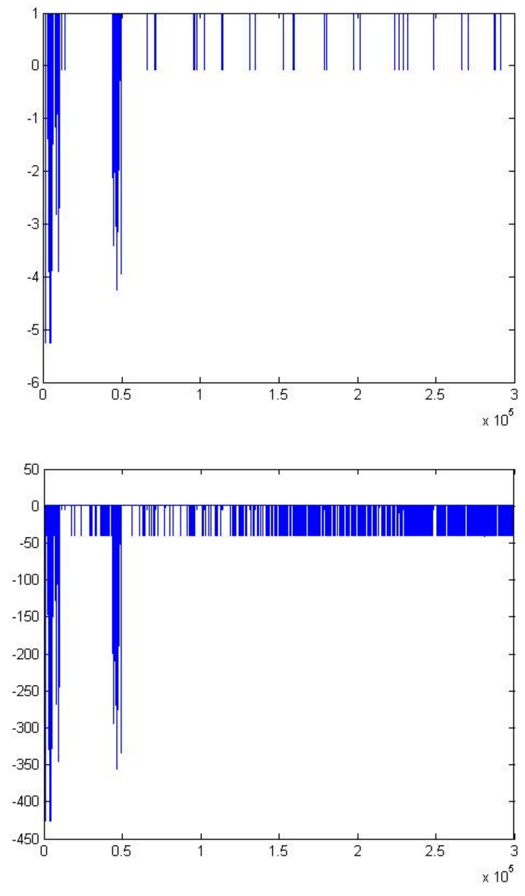

Figure 4. OCSVM classification outcome for different values of parameters σ, ν Upper diagram : $OCSVM_{0.007,0.001}$, Lower diagram : $OCSVM_{0.01,0.05}$

Testbed scenario. The second trial is conducted off line with the use of two datasets extracted from the testbed (Figure 5). The testbed architecture mimics a small-scale SCADA system, comprising the operations and field networks and including a Human-Machine Interface Station (for process monitoring), a managed switch (with port monitoring capabilities, for network traffic capture), and two Programmable Logic Controller Units, for process control. The NIDS and OCSVM modules are co-located on the same host, being able to intercept all the traffic flowing on the network scopes.

During the testing period several attack scenarios are simulated in the testbed. These scenarios include network scan, network flood and MITM attack.

Three kinds of attacks are being evaluated:

- **Network scan attack** In typical network scan attack, the attacker uses TCP/FIN scan to determine if ports are closed to the target machine.

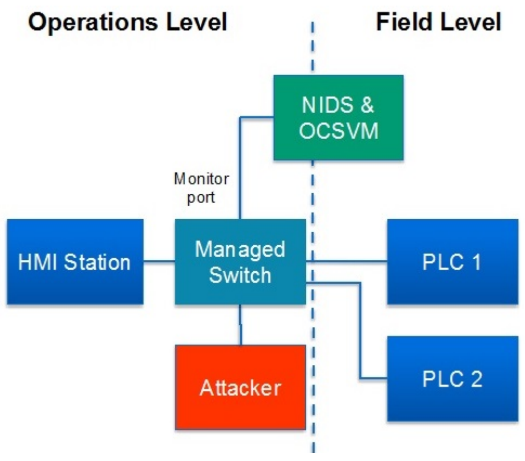

Figure 5. Architecture of the testbed

Closed ports answer with RST packets while open ports discard the FIN message. FIN packets blend with background noise on a link and are hard to be detected.

- **ARP cache spoofing - MITM attack ARP cache spoofing** is a technique where an attacker sends fake ARP messages. The aim is to associate the attacker's MAC address with the IP address of another host, causing any traffic meant for that IP to be sent to attacker instead. The attacker could choose to inspect the packets, modify data before forwarding (**man-in-the-middle attack**) or launch a denial of service attack by causing some of the packets to be dropped.

- **DoS attack** Network flood is the instance where the attacker floods the connection with the PLC by sending SYN packets. In a TCP SYN flooding attack, an attacker sends many SYN messages, with fictitious (spoofed) IP addresses, to a single node (victim). Although the node replies with SYN/ACK messages, these messages are never acknowledged by the client. As a result, many half open connections exist on the victim, consuming its resources. This continues until the victim has consumed all its resources, hence can no longer accept new TCP connection requests.

In Table 3 we show the number of alert messages (IDMEF) sent from OCSVM and $\mathcal{K}-\mathcal{OCSVM}$ respectively. From this table it is shown how parameters σ, ν affect the performance of OCSVM for the testbed scenario. While for the same network trace file OCSVM produces from *10529* to *10704* alert messages according to the values of the parameters, $\mathcal{K}-\mathcal{OCSVM}$ produces the same *120* alert messages. All the reported attacks are concerning the *DoS* attack that creates the biggest fluctuation in the network traffic.

Parameter σ	*Parameter* ν	$\mathcal{K}{-}\mathcal{OCSVM}$	*ocsvm*
0.007	0.002	120	10529
0.007	0.01	120	10703
0.007	0.005	120	10584
0.0001	0.01	120	10602
0.1	0.01	120	10704

Table 3. Performance evaluation of $\mathcal{K}{-}\mathcal{OCSVM}$ and OCSVM for $K_{thers} = 2$.

Testbed scenario with split testing periods. Since the attacks are performed during different time periods we divide the testing dataset in several smaller ones, each containing a different attack. Testing data consists of normal data and attack data and the composition of the data sets are as follows:

- Testing set-A' : *1 - 5000*: Normal data
- Testing set-B' : *5000 - 10000*: Normal data + **Arp spoofing** attack + **Network scan**
- Testing set-C' : *10000 - 25000*: Normal data + **Flooding Dos attack** + **Network scan**
- Testing set-D' : *25000 - 41000*: Normal data + **MITM attack**

Dataset	*Initial alarms*	*Aggregated alarms*
A	129	2
B	658	3
C	9273	120
D	203	3
All	10507	120

Table 4. Aggregated alarms produced by $\mathcal{K}{-}\mathcal{OCSVM}$ are significantly decreased compared to the initial alarms

From table 4 we observe that not only the most important intrusions are detected and reported but also the total overhead on the system is limited. For all time periods the messages communicated reflect actual attacks in the network, except from the testing set-A'. In this time period *HMI* station demonstrated a significant variation in the rate that it injected packets in the system between testing and training of the module. This is due to the limited training of the OCSVM and can be avoided if training dataset consists of data that represent the traffic in the network during under work loads. The increased number of alarms created from $\mathcal{K}{-}\mathcal{OCSVM}$ for the dataset B' is due to the fact in this time period the attacker uses an excessive number of SYN packets in order to flood the communication channel.

Hybrid Testbed scenario. The third trial is conducted off line with the use of large datasets extracted from a Hybrid Testbed (*HTB*) scenario. The Hybrid testbed architecture mimics a medium-scale SCADA system, comprising the operations and field networks and including Human-Machine Interface Stations (for process monitoring), six managed switches (with port monitoring capabilities, for network traffic capture), and several Programmable Logic Controller Units, for process control. The initial dataset consists of over *3* million rows, each representing a packet sent in the system, capturing network of several days. The dataset is split in *65* smaller ones of *50.000* rows. The datasets contain only data from a normal operation of the *HTB*.

Both OCSVM and $\mathcal{K}{-}\mathcal{OCSVM}$ are trained and tested with these datasets, using cross validation. The mean number of alert messages sent by the two modules is shown in Table 5.

Parameter σ	*Parameter* ν	$\mathcal{K}{-}\mathcal{OCSVM}$	*ocsvm*
0.007	0.002	1 - 2	40
0.007	0.01	1 - 2	207
0.007	0.005	1 - 2	105
0.0001	0.01	1 - 2	85
0.1	0.01	1 - 2	271

Table 5. Performance evaluation of $\mathcal{K}{-}\mathcal{OCSVM}$ and OCSVM for $K_{thers} = 2$

Using real datasets of a medium sized HTB SCADA system the performance of the proposed $\mathcal{K}{-}\mathcal{OCSVM}$ method is very stable compared to a simple OCSVM under the same configuration. This behavior is very promising since $\mathcal{K}{-}\mathcal{OCSVM}$ method has a very low false alarm rate (*lower than 0.02 %*) while on the same time the overhead induced by the method is negligible (C.F. Subsection 6.3. We must state here that for the *HTB* we had available only non malicious datasets for the evaluatioin of the proposed method. In the future when datasets containing malicious attacks are available an extensive evaluation of the $\mathcal{K}{-}\mathcal{OCSVM}$ method is going to be conducted in terms of accuracy and false alarm rate.

6.3. Computational cost and time overhead

Complexity of an intrusion detection system can be attributed to hardware, software and operation factors. For simplicity, it is usually estimated as the computing time required to perform classification of the dataset and output the final alarms. In order to evaluate the complexity of the proposed method we calculate the execution time and compare it to a simple OCSVM module. The evaluation is conducted on a PC with Intel core *2* duo *1.7* Mhz CPU, *2*GB main memory, *80*GB hard disk *7200* rpm hard disk and Microsoft windows *7 64*bit. In Figure 6 we represent the time performance of the method compared to a simple OCSVM module for the testbed scenario.

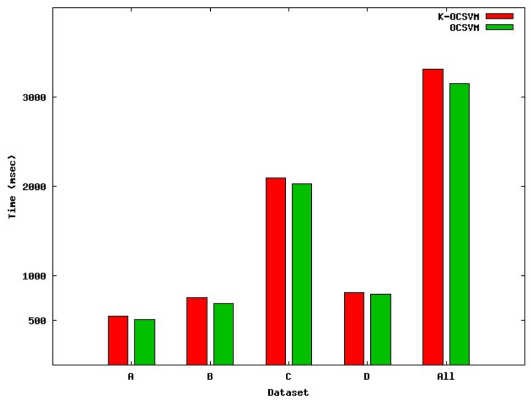

Figure 6. Computational cost for the testbed scenario

According to Figure 6 execution time of the proposed $\mathcal{K}-\mathcal{OCSVM}$ is slightly bigger compared to a simple OCSVM method. The performance gap is around *5%* to *10%* for all the datasets used in the simulation. Based on these observations we conclude that the system, performs a classiffication in a comparable time to that of a simple OCSVM classifier, and it thus can be adopted in soft real-time applications. We have to mention that the performance evaluation which is conducted in this subsection, does not include the time that each detection mechanism needs in order to create and disseminate IDMEF messages. It is evident that the OCSVM classifer, compared to the proposed $\mathcal{K}-\mathcal{OCSVM}$, needs significant additional time in order to send all the detected alarms.

7. Discussion

The proposed $\mathcal{K}-\mathcal{OCSVM}$ can significantly reduce the produced alarms from the OCSVM module that is the heart of the detection mechanism. The profound advantages of low overhead and low false alarm rate come with the cost of lower accuracy and higher computational overhead. In this section we discuss some enhancements of the proposed method that can improve its performance.

7.1. Parameter k_{thres}

As stated is Section 5 $\mathcal{K}-\mathcal{OCSVM}$ method is a recursive clustering of the alarms produced by the OCSVM module. This recursive procedure is used in order to distinguish severe from possible alarms and finally disseminate only those that represent an actual misbehavior of the system. This way the OCSVM module is enhanced in both the decreased overhead that induces to the system from the disseminated alarm files and in the decreased false alarm rate that it has. The recursive method stops when either all initial alarms are put in the same cluster or when the divided set is big enough compared to the initial one, according to the threshold parameter k_{thres}.

In the simulations presented in Subsection 6 the parameter is set to *2*. When raising the value of this parameter the produced final alarms of the proposed $\mathcal{K}-\mathcal{OCSVM}$ method raises. This raise also leads to a raise in the false alarm rate. On the other hand the accuracy of the method raises since less profound attacks are detected. By raising the value of the parameter above one limit the method degrades to the initial OCSVM. The optimum value for parameter $\mathcal{K}-\mathcal{OCSVM}$ varies according to the architecture of the network. For big disperse networks large value of the parameter would lead to the creation of too many alarms from the module, while on the same time in a medium sized network very small value of the parameter would lead to a dangerous decrease of the detection capabilities of the module.

Except from the static configuration of the network, traffic conditions can also affect the performance of the method. In real SCADA systems the network traffic varies between daytime and night, weekdays and weekends. In order to cope with both static and dynamic features of the network an enhanced $\mathcal{K}-\mathcal{OCSVM}$ method that dynamically adapts the parameter k_{thres}, similar to [31, 32] could be effective.

7.2. Multi stage $\mathcal{K}-\mathcal{OCSVM}$

The proposed method uses only the values of the initial alarms in order to filter out those that don't represent an actual attack. That way attacks that cause significant variation in certain features of the OCSVM module are detected, while on the same time other more insidious attacks are passing undetected. A multi stage $\mathcal{K}-\mathcal{OCSVM}$ where both the number of attacks that share common characteristics, like origin, destination, port number e.t.c., and the actual values of the attacks can be developed in order to better detect different kinds of attacks.

Figure 7 represents a possible architecture of a two stage $\mathcal{K}-\mathcal{OCSVM}$ module. The number of the stages can be increased and the alarms produced by each stage can be further aggregated. The fusion of the outputs of the different stages can be done using any of the existing ensemble methods .i.e. majority voting, performance weighting, distribution summation, order statistics e.t.c.

8. Conclusions

We have presents an intrusion detection module for SCADA systems that is based in OCSVM classifier and

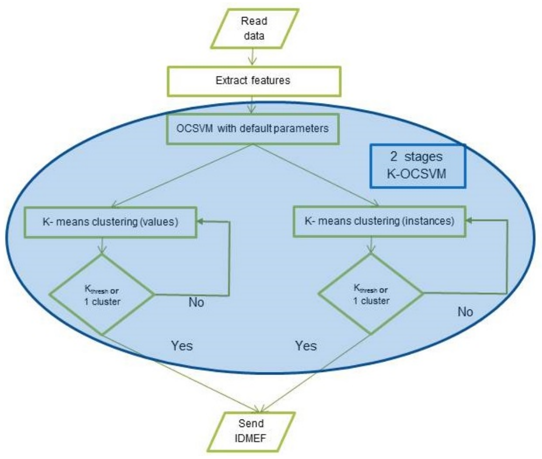
Read data
Extract features
OCSVM with default parameters
2 stages K-OCSVM
K- means clustering (values)
K- means clustering (instances)
K_{thresh} or 1 cluster
No
Yes
Yes
K_{thresh} or 1 cluster
No
Send IDMEF

Figure 7. Architecture of a Multi stage $\mathcal{K}-\mathcal{OCSVM}$

a recursive k-means clustering method. The module is trained off-line by network traces, after the attributes are extracted from the network dataset. The intrusion detection module is part of an distributed IDS system.

The method is tested on three different datasets. For the first testing scenario, traces of a wireless network are used. This test shows that the method is stable and its performance is not influenced by the selection of parameters ν and σ. For the second scenario, testing of the proposed module is conducted with datasets that are sniffed of a testbed that mimics a small-scale SCADA system under different attack scenarios. After the completion of the test, not only the most important intrusions are detected and reported by $\mathcal{K}-\mathcal{OCSVM}$ but also the total overhead on the system is limited. Finally extensive testing of the $\mathcal{K}-\mathcal{OCSVM}$ module with real datasets extracted from a medium sized HTB SCADA system shows that the performance of the proposed $\mathcal{K}-\mathcal{OCSVM}$ method remains very stable under different configurations. After the execution of the $\mathcal{K}-\mathcal{OCSVM}$ method, for all the simulated scenarios, only severe alerts are communicated to the system by IDMEF files that contain information about the source, destination, protocol and time of the intrusion.

The main feature of $\mathcal{K}-\mathcal{OCSVM}$ module is that it can perform anomaly detection in a time-efficient way, with good accuracy and low overhead. Low overhead is an important evaluation metric of a distributed detection module that is scattered in a real-time system, since frequent communication of IDMEF files from detection agents degrade the performance of the SCADA network. Recursive k-means clustering, reassures that small fluctuations on network traffic, which most of the times cause OCSVM to trigger false alarms, are ignored by the proposed detection module. The added computational time of the method compared to a simple OCSVM varies between 5% and 10% which results in a neglective time overhead. This overhead does not include the time that each detection mechanism needs in order to create and disseminate IDMEF messages. By adding the time needed in order to create and send each IDMEF file to the IDS management system the proposed $\mathcal{K}-\mathcal{OCSVM}$ method prevails on the overall performance in terms of time efficiency.

Finally we investigate how parameter k_{thres} affects the performance of the method and proposed a multi-stage $\mathcal{K}-\mathcal{OCSVM}$ method for better accuracy. As future work we will conduct an in depth performance evaluation of the proposed mechanism. Using malicious and attack-free datasets of the HTB SCADA testbed, we are going to evaluate $\mathcal{K}-\mathcal{OCSVM}$'s performance in terms of false positive rate and accuracy. Using the evaluation outcomes we are planning to modify the proposed $\mathcal{K}-\mathcal{OCSVM}$ in order to further decrease false alarms and improve overall performance.

Acknowledgement. Research supported by the project "CockpitCI: Cybersecurity on SCADA: risk prediction, analysis and reaction tools for Critical Infrastructures", funded under European Framework-7 Programme (contract No. 285647)

References

[1] Maglaras, L.A. and Jiang, J. (2014) Ocsvm model combined with k-means recursive clustering for intrusion detection in scada systems. In *Proceedings of the 10th Qshine conference* (EAI).

[2] Khan, L., Awad, M. and Thuraisingham, B. (2007) A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB JournalThe International Journal on Very Large Data Bases* **16**(4): 507–521.

[3] Mukkamala, S., Janoski, G. and Sung, A. (2002) Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on* (IEEE), **2**: 1702–1707.

[4] Roesch, M. *et al.* (1999) Snort: Lightweight intrusion detection for networks. In *LISA*, **99**: 229–238.

[5] Portnoy, L. (2000) Intrusion detection with unlabeled data using clustering .

[6] Kim, J. and Bentley, P.J. (2001) An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of GECCO*: 1330–1337.

[7] Wang, Y., Wong, J. and Miner, A. (2004) Anomaly intrusion detection using one class svm. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC* (IEEE): 358–364.

[8] Ma, J. and Perkins, S. (2003) Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, **3**: 1741–1745 vol.3.

[9] Li, K.L., Huang, H.K., Tian, S.F. and Xu, W. (2003) Improving one-class svm for anomaly detection. In

Machine Learning and Cybernetics, 2003 International Conference on (IEEE), **5**: 3077–3081.

[10] Dasgupta, D. and Gonzalez, F.A. (2001) An intelligent decision support system for intrusion detection and response. In *Information Assurance in Computer Networks* (Springer), 1–14.

[11] Glazer, A., Michael, L. and Markovitch, S. (2013) q-ocsvm: A q-quantile estimator for high-dimensional distributions. In *In Proceedings of The 27th Conference on Neural Information Processing Systems (NIPS-2013),Lake Tahoe, Nevada.*

[12] Song, X., Fan, G. and Rao, M. (2008) Svm-based data editing for enhanced one-class classification of remotely sensed imagery. *Geoscience and Remote Sensing Letters, IEEE* **5**(2): 189–193.

[13] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S. (2002) A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security* (Springer), 77–101.

[14] Knorr, E.M. and Ng, R.T. (1997) A unified notion of outliers: Properties and computation. In *KDD*: 219–222.

[15] Burman, P. (1989) A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika* **76**(3): 503–514.

[16] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J. and Tibshirani, R. (2009) *The elements of statistical learning*, **2** (Springer).

[17] Menahem, E., Rokach, L. and Elovici, Y. (2013) Combining one-class classifiers via meta learning. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (ACM): 2435–2440.

[18] Tsoumakas, G., Katakis, I. and Vlahavas, I. (2004) Effective voting of heterogeneous classifiers. In *Machine Learning: ECML 2004* (Springer), 465–476.

[19] Kim, M.J. and Kang, D.K. (2010) Ensemble with neural networks for bankruptcy prediction. *Expert Systems with Applications* **37**(4): 3373–3379.

[20] Krawczyk, B. and Woźniak, M. (2014) Diversity measures for one-class classifier ensembles. *Neurocomputing* **126**: 36–44.

[21] Runarsson, T.P. and Jonsson, M.T. (2003) Model selection in one-class ν-svms using rbf kernels. In *Proceedings of 16th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management.*

[22] Li, X., Wang, L. and Sung, E. (2008) Adaboost with svm-based component classifiers. *Engineering Applications of Artificial Intelligence* **21**(5): 785–795.

[23] Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J. and Williamson, R.C. (2001) Estimating the support of a high-dimensional distribution. *Neural computation* **13**(7): 1443–1471.

[24] TAX, D. (2001) One-class classification. *PhD thesis,Delft University of Technology, The Netherlands* .

[25] Barbosa, R.R.R. and Pras, A. (2010) Intrusion detection in scada networks. In *Mechanisms for Autonomous Management of Networks and Services* (Springer), 163–166.

[26] Zhu, B., Joseph, A. and Sastry, S. (2011) A taxonomy of cyber attacks on scada systems. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing* (IEEE): 380–388.

[27] Jiang, J. and Yasakethu, L. (2013) Anomaly detection via one class svm for protection of scada systems. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on* (IEEE): 82–88.

[28] Zhang, R., Zhang, S., Lan, Y. and Jiang, J. (2008) Network anomaly detection using one class support vector machine. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, **1**.

[29] Maglaras, L.A. and Jiang, J. (2014) Intrusion detection in scada systems using machine learning techniques. In *Proceedings of the 2nd SAI conference* (SAI).

[30] Debar, H., Curry, D.A. and Feinstein, B.S. (2007) The intrusion detection message exchange format (idmef) .

[31] Campbell, C., Cristianini, N. and Shawe-Taylor, J. (1999) Dynamically adapting kernels in support vector machines. *Advances in neural information processing systems* **11**: 204–210.

[32] Cao, L. and Gu, Q. (2002) Dynamic support vector machines for non-stationary time series forecasting. *Intelligent Data Analysis* **6**(1): 67–83.