A real time OCSVM Intrusion Detection module with low overhead for SCADA systems

Leandros A. Maglaras
Department of Computing, University of Surrey
Guilford, UK
Email: l.maglaras@surrey.ac.uk

Jianmin Jiang
Department of Computing, University of Surrey
Guilford, UK
Email: jianmin.jiang@surrey.ac.uk

***Abstract*—In this paper we present a intrusion detection module capable of detecting malicious network traffic in a SCADA (Supervisory Control and Data Acquisition) system. Malicious data in a SCADA system disrupt its correct functioning and tamper with its normal operation. OCSVM (One-Class Support Vector Machine) is an intrusion detection mechanism that does not need any labeled data for training or any information about the kind of anomaly is expecting for the detection process. This feature makes it ideal for processing SCADA environment data and automate SCADA performance monitoring. The OCSVM module developed is trained by network traces off line and detect anomalies in the system real time.**

In order to decrease the overhead induced by communicated alarms we propose a new detection mechanism that is based on the combination of OCSVM with a recursive k-means clustering procedure. The proposed intrusion detection module $\mathcal{K}-\mathcal{OCSVM}$ is capable to distinguish severe 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. The most severe alarms are then communicated with the use of IDMEF files to an IDSIDS (Intrusion Detection System) system that is developed under CockpitCI project. Alarm messages carry information about the source of the incident, the time of the intrusion and a classification of the alarm.

***Keywords*—SCADA systems; OCSVM; intrusion detection**

I. Introduction

Cyber-physical systems are becoming vital to modernizing the national critical infrastructure systems. Cyber attacks usually target valuable infrastructures assets, taking advantage of architectural/technical vulnerabilities or even weaknesses in the defense systems. While there is the case in some situations, most weaknesses in CIs arise from the fact that most CIs are adopting off-the-shelf technologies from the IT world, without a significant change in terms of the operator mindset, still based on the ”airgap” security principle that suggests that an apparently isolated and obscure system is implicitly secure. Once you open the system to off-the-shelf solutions, you also increase its exposure to cyber-attacks. Several techniques and algorithms have been reported by researchers for intrusion detection. One big family of intrusion detection algorithms is rule based algorithms. 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 neighbour (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 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 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 online 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 [1]–[4]. IDS can provide active detection and automated responses during intrusions [5]. 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 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 [6]–[8]

A. 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 of One class classifiers [9].

The article makes the following contributions:

- OCSVM is tested for intrusion detection in SCADA system
- A new one class classifier $\mathcal{K}-\mathcal{OCSVM}$ is proposed.
- The proposed classifier combines 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 OCSVM and the proposed method with different parameters is conducted.

The rest of this article is organized as follows: Section II describes the OCSVM method. Section III describes the proposed $\mathcal{K}-\mathcal{OCSVM}$ method; In Section IV the use of OCSVM in SCADA systems is presented; Section V presents the features extracted from the network traces for the testing and training of the model Section VI describes how our model is integrated in the IDS system, section VII presents the evaluation of OCSVM and $\mathcal{K}-\mathcal{OCSVM}$ methods and results. Section VIII concludes the article.

II. 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.

Scholkopf et al. [10] 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

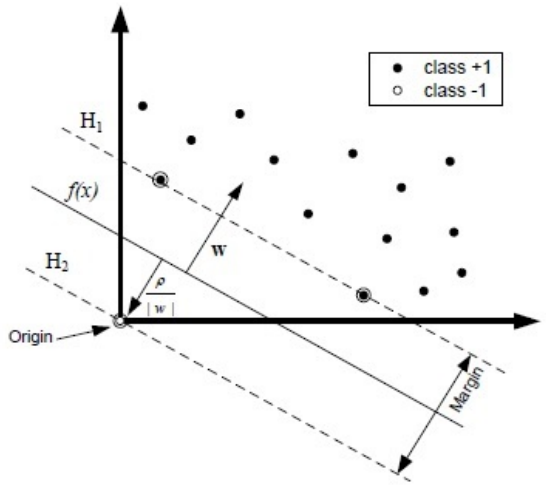

Fig. 1. OCSVM maps input data into a high dimensional feature space

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

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 $\sum_i a_i \leq 1$.

Where 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 [11], which includes a small amount of samples of the outlier class, will be also applied and evaluated in the near future.

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, 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.

Unnthorsson et al. [12] 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.

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

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 [13].

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. 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.

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

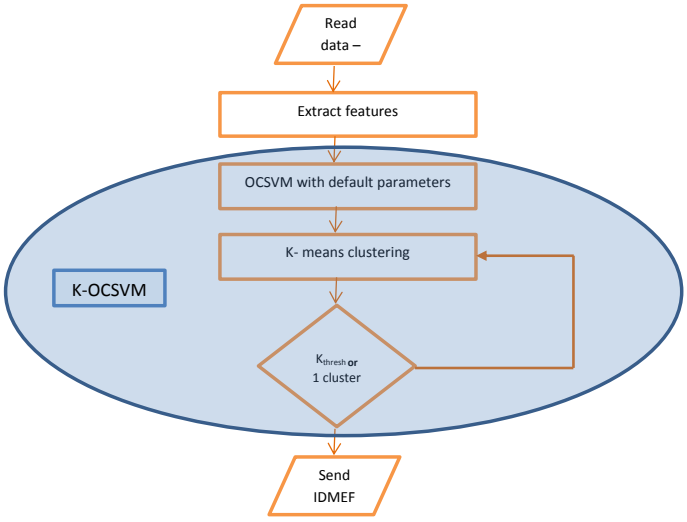

Fig. 2. $\mathcal{K}-\mathcal{OCSVM}$ module

(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. For bigger value (3 or more) the number of alerts rises till the method degrades to the initial OCSVM. The optimal value for the given parameter k_{thres} is a matter for future investigation.

IV. OCSVM FOR SCADA SYSTEM

Cyber-attacks against SCADA systems [14] are considered extremely dangerous for Critical Infrastructure (CI) operation and must be addressed in a specific way [15], [16]. Presently one of the most adopted attacks to a SCADA system is based on fake commands sent from the SCADA to the RTUs. OCSVM [17]–[19] 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 .

V. Attribute Extraction

Feature extraction is essential in a classification problem. In order to train the OCSVM module properly we used a network trace file from a SCADA system. Based on the analysis of data that is presented in the upcoming subsections we selected some initial features that are used as attributes for our OCSVM model. We also found that additional features of the system must be combined with these initial ones in order to better represent the current state of the network operation.

A. Analysis of data

The dataset consists of about 1570 rows each one representing a send packet. There are several sources(12) and destinations(17). The packets use different protocols (13) each performing a different task in the network. Modbus/TCP protocol is a commonly available means of connecting industrial electronic devices (PLCs).

No.	Time	Source	Destination	Protocol	Length	Info
1	0	AsustekC_b2:	Broadcast	ARP	60	Who has 192.168.1.4? Tell 192.168.1.2
2	0.000017	AsustekC_b2:	Broadcast	ARP	60	Who has 192.168.1.4? Tell 192.168.1.2
3	0.497982	Cisco_70:37:1	Spanning-tree-(	STP	64	RST. Root = 32768/0/08:d0:9f:70:37:12 Cost = 0 Port = 0x8002
4	0.498211	Cisco_70:37:1	Spanning-tree-(	STP	64	RST. Root = 32768/0/08:d0:9f:70:37:12 Cost = 0 Port = 0x8003
5	2.059351	192.168.1.2	192.168.1.3	FTP	60	Request: FREE
6	2.059358	192.168.1.2	192.168.1.3	FTP	60	[TCP Retransmission] Request: FREE
7	2.07154	192.168.1.3	192.168.1.2	FTP	98	Response: 200 free space on SD card: size = 14464000
8	2.071547	192.168.1.3	192.168.1.2	FTP	98	[TCP Retransmission] Response: 200 free space on SD card: size = 14464000
9	2.075051	192.168.1.7	192.168.1.254	DNS	90	Standard query A geoip.ubuntu.com.192.168.1.254
10	2.221634	fe80::d5c8:42	ff02::1:3	LLMNR	84	Standard query A wpad

Fig. 3. Snapshot of SCADA dataset.

The addresses that create the biggest traffic (packets created) in the system are mainly two (192.168.1.2, 192.168.1.3) which represent probably PCs or PLCs. The distribution of sources and destinations is shown in Figure 4. The basic protocols used are: DNS, FTP, MDNS, MODBUS, TCP, UDP.

Figure 4 presents the source/destination distribution of the packets in the network. It is evident that most of the traffic is produced by only a small portion of the network (two or three sources) that send their packets to a limited set of nodes. This communication mainly constist of control packets exchanged between industrial based computers and Programmable Logic Controllers (PLCs). Thorough knowledge of the type of the

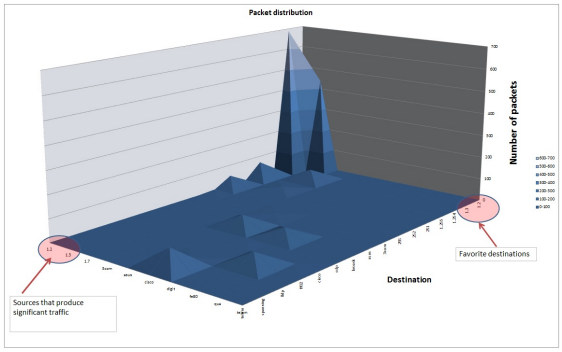

Fig. 4. Source - destination distribution

source that produces the traffic and a reputation metric that represents the history of the source in terms of vulnerability would add additional intelligence to the detection module, making it capable of distinguishing between traffic bursts and actual attacks.

Protocols used for the communication among nodes of the network may also include information about a possible abnormal behavior of the system. Protocols are used for specific tasks in a network. Different attacks use different protocols. A proper filtering of the network data or use of protocol as an attribute for the OCSVM module, may be useful for the detection intrusion mechanism.

It is evident from Figure 5 that the accumulative packet size over time is almost a straight line with some sudden rises only is the three instances when big files are circulated in the system. These files are FTP files of size 590 bytes which is ten times larger than the usual send packets in the system. This smooth packet growth over time represents a normal behavior of the system when no malicious data is detected and the slop of the line can vary from time to time, depending on the load of the system. When malicious data are broadcasted over the network the accumulative packet size may rise, but since in many situations intruders send 0 lenght packet sizes (ack/nack packets) this feature by itself is not enough to detect these situations.

A burst in traffic injected in the system is another characteristic of an intrusion. Infected nodes may broadcast messages flooding the system with messages that are of no use and block the normal operation of the network.

In Figure 6 we observe that when in the upper graph there exist horizontal lines then we have a burst of traffic in the system. In the lower graph this burst is more easily observed. Traffic rate can be easily extracted from network trace files.

B. Attributes extracted

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 preprocessing was required before pattern classification models could be built. Pre-processing consistes of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes and second step implemented scaling The

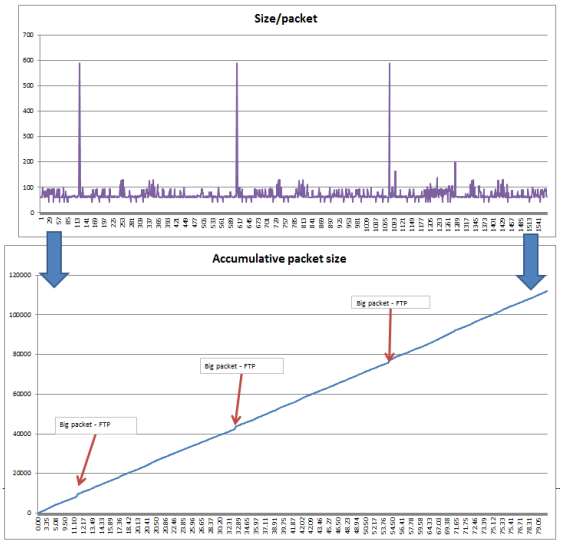

Fig. 5. Accumulative packet size

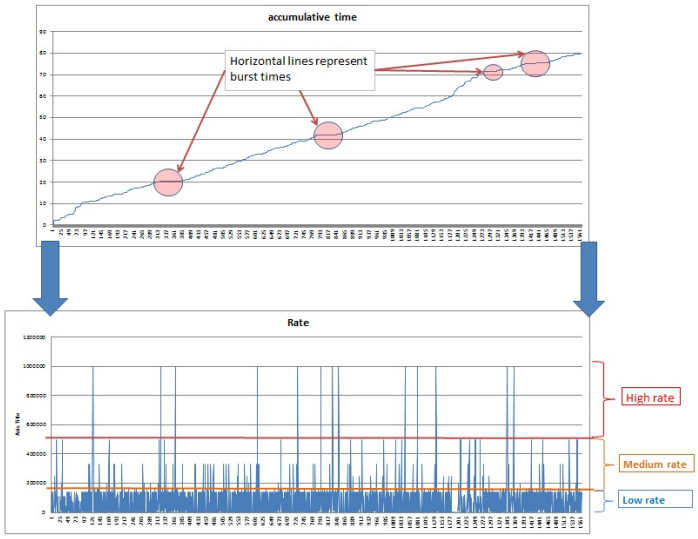

Fig. 6. Events over time - rate of traffic

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:

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

VI. Integration of OCSVM module

The above described OCSVM detection approach could be incorporated to the performance management scheme of the SCADA system. The performance management system contains sub systems and is summarized as follows:

- Firstly, the off line monitoring data is used to train the OCSVM and generate the model : Detector training
- Once tested, the model is transferred to performance monitoring and management system and detect the anomalies real time: Anomaly detection
- Once anomalies are detected they are classified into different classes according to their severities: Classification
- IDMEF files are send from the OCSVM module to the main correlator.

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 [20] files.

A. Communication messages

The IDMEF defines experimental standard for exchanging intrusion detection related events. As a standard, it can be used as a vendor or product independent enabling intercommunication between different agents such as NIDS or Honeypots. According to the RFC 4765 the data model of this format address several problems associated with representing intrusion detection alert data:

- As alert information is inherently heterogeneous, with some alerts with a very incomplete definition and others with very detailed information. The data model by using an object-oriented model provides extensibility via aggregation and sub classing.
- The intrusion detection environments are different. There are for example NIDS and HIDS analyzers. The data model defines support classes that accommodate the differences in the data sources among agents.
- The data model must allow for conversion to formats used by tools other than intrusion detection analyzers, for the purpose of further processing the alert information.
- The data model should accommodate the existing differences in operating environments and networks.

- Certain sensors deliver more or less information about certain types of attacks. The object oriented approach allows flexibility while the subclassing rule provides the integrity of the model.

A typical IDMEF file produced by our system is shown in Figure 7. The IDMEF message contains information about the source of the intrusion, the time of the intrusion detection, the module that detected the problem and a classification of the detected attack. The source node that the intrusion is detected is very important feature in an IDS system. Once the infected node is spotted the infection can be limited by the isolation of this node from the rest network. Fast and accurate detection of the source node of a contamination is crucial for the correct function of an IDS.

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message version="1.0" xmlns:idmef="http://iana.org/idmef">
  - <idmef:Alert>
      - <idmef:Analyzer>
          - <idmef:Node category="unknown">
              <idmef:location>IT Network</idmef:location>
              <idmef:name>OCSVM</idmef:name>
            </idmef:Node>
        </idmef:Analyzer>
        <idmef:CreateTime ntpstamp="0x1123">2014-02-13</idmef:CreateTime>
      - <idmef:Source>
          - <idmef:Node>
              - <idmef:Address>
                  <idmef:address>AsustekC_b2:ce:52</idmef:address>
                </idmef:Address>
            </idmef:Node>
        </idmef:Source>
        <idmef:Classification text="POSSIBLE ALARM"/>
    </idmef:Alert>
</idmef:IDMEF-Message>
```

Fig. 7. Typical IDMEF message produced by OCSVM module

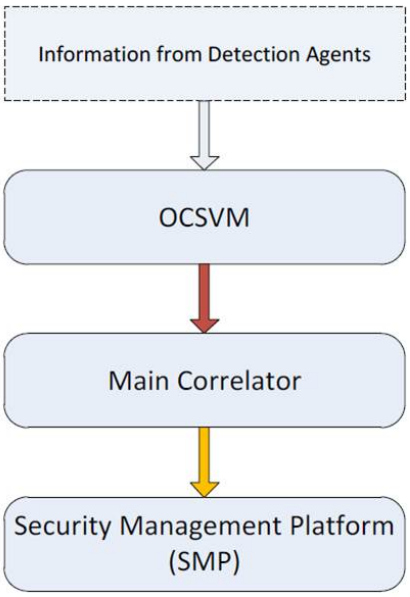

Fig. 8. Interfaces linked with OCSVM detection module

```
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
```

Fig. 9. Format of the transformed Network trace file

B. OCSVM interfaces

Once the detector training phase is complete, OCSVM detection is capable of detecting possible intrusions (abnormal behavior) on the SCADA system. Detection agents will gather new monitoring data (with corresponding attributes) and will feed the data to the OCSVM detection module. The detection module will classify each event whether it is a normal event or a possible intrusion. This information will then be send to the main correlator in order to react accordingly to the detected intrusions as shown in Figure 8

VII. Performance Evaluation

A. Training of OCSVM model

Training of OCSVM module was conducted using the transformed network trace file (Figure 9). 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. The parameter σ is chosen 0.07 and the parameter ν 0.01.

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.

B. Testing of OCSVM model

In order to test our model we use the initial network trace file and we also spit the trace file in two separate files (A,B). The split was random and two datasets were constructed from the initial trace file. The two datasets were then used for training and testing. The dataset A was initially used for training and dataset B for testing and vice versa. The size of dataset A is 1000 rows and of B 570 rows. The results of our OCSVM detection module for each split are shown in Table VII-B. The accuracy of the classification of the data is high for all the tests conducted.

TABLE I. Accuracy of OCSVM module under different split of data.

Split	*Accuracy*
All	98.8796
A	98.42
B	99.12

The outcomes of the classification method for the testing conducted in whole the dataset and in the two spitted sets are

shown in Figure 11. The observed malicious data give small negative values and can only be classified as possible alerts. This is due to the fact that all the testing datasets are part of the initial trace file, which is captured during a normal operation of the system.

Malicious datasets that represent attack scenarios e.g. Man in the Middle (MITM) by ARP (Address Resolution Protocol) poisoning, SYN Flooding and honeypot [21] interaction, are used in the next subsection in order to test the performance of our $\mathcal{K}-\mathcal{OCSVM}$ module.

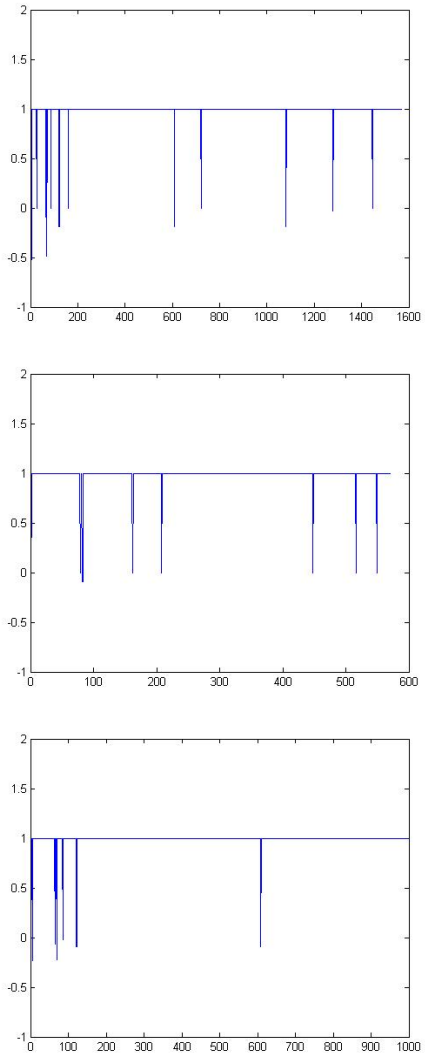

Fig. 10. OCSVM classification outcome for the three training / testing datasets (Upper figure:Original dataset, Middle figure: A/B dataset, Lower figure: B/A dataset

C. Testing of K-OCSVM model

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

Parameter	*Range of Values*	*Default value*
σ	0.1 - 0.0001	0.007
ν	0.002 - 0.05	0.01
$Threshold$	2-3	2

TABLE II. EVALUATION PARAMETERS

1) **Wireless network:** In order to test our model we use another network trace files sniffed from the wireless network. The testing trace file consists of 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 III 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 almost 500 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 III. PERFORMANCE EVALUATION OF $\mathcal{K}-\mathcal{OCSVM}$ AND OCSVM FOR $K_{thers} = 2$.

In figure 11 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 $\mathcal{K}-\mathcal{OCSVM}$ necesary tool for proper intrusion detection.

2) **Testbed scenario:** The second trial is conducted off line with the use of two datasets extracted from the testbed (Figure 12. 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. 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

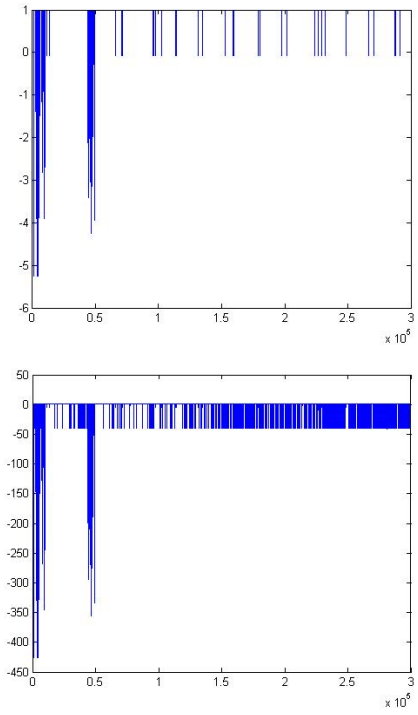

Fig. 11. OCSVM classification outcome for different values of parameters σ, ν Upper diagram : $OCSVM_{0.007,0.001}$, Lower diagram : $OCSVM_{0.01,0.05}$

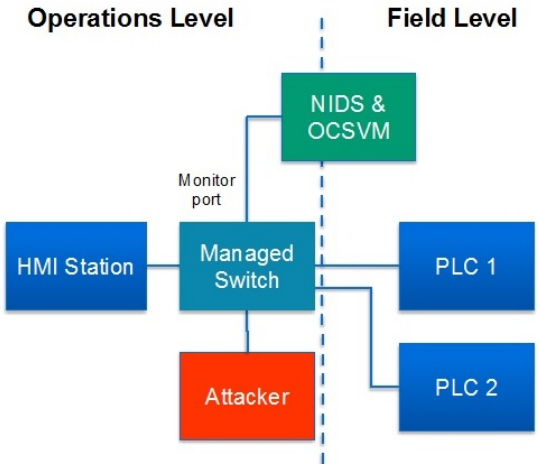

Fig. 12. Architecture of the testbed

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 attackesends 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 halfopen 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 IV 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 concering the DoS attack that creates the biggest flunctuation 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 IV. PERFORMANCE EVALUATION OF $\mathcal{K}-\mathcal{OCSVM}$ AND OCSVM FOR $K_{thers} = 2$.

3) **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	3

TABLE V. AGGREGATED ALARMS PRODUCED BY $\mathcal{K}-\mathcal{OCSVM}$ ARE SIGNIFICANTLY DECREASED COMPARED TO THE INITIAL ALARMS

From table V 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 singificant 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 exceshive number of SYN packets in order to flood the communication channel.

VIII. Conclusion

We have presented a intrusion detection module for SCADA systems that is based in OCSVM technique. The module is trained offline by network traces, after the attributes are extracted from the network dataset. The initial attributes used for training and testing of the module are rate and packet size which represent the traffic in the system. The intrusion detection module is part of an IDS system developed under CoCkpitCI. Output of the detection module is communicated to the system by IDMEF files that contain information about the source, time and severity of the intrusion.

After the execution of the $\mathcal{K}-\mathcal{OCSVM}$ method 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 method is stable and its performance is not influenced by the selection of parameters ν and σ. 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.

As future work we will conduct an in depth performance evaluation on proposed mechanism. Using malicious and attack-free datasets of the SCADA testbed, we are going to evaluate $\mathcal{K}-\mathcal{OCSVM}$'s performance in terms of false positive rate, accuracy and runtime. Using the evaluation outcomes we are planning to enhance the proposed $\mathcal{K}-\mathcal{OCSVM}$ in order to further decrease false alarms and improve overall performance. Additional attributes like reputation of the source and the protocol used for communication may add more precision to our system and it is a matter of future research. Use of different models according to the area or the time period may further improve the performance of the method.

Acknowledgment

The authors wish to acknowledge the financial support of the project CockpitCI, funded under European Framework-7 Programme (contract No. 285647).

References

[1] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class svm," in *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*. IEEE, 2004, pp. 358–364.

[2] K. Heller, K. Svore, A. D. Keromytis, and S. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," in *Workshop on Data Mining for Computer Security (DMSEC), Melbourne, FL, November 19, 2003*, 2003, pp. 2–9.

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

[4] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class svm for anomaly detection," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5. IEEE, 2003, pp. 3077–3081.

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

[6] A. Glazer, L. Michael, and S. Markovitch, "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*, 2013.

[7] X. Song, G. Fan, and M. Rao, "Svm-based data editing for enhanced one-class classification of remotely sensed imagery," *Geoscience and Remote Sensing Letters, IEEE*, vol. 5, no. 2, pp. 189–193, 2008.

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

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

[10] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

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

[12] T. P. Runarsson and M. T. Jonsson, "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*, 2003.

[13] X. Li, L. Wang, and E. Sung, "Adaboost with svm-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, 2008.

[14] R. R. R. Barbosa and A. Pras, "Intrusion detection in scada networks," in *Mechanisms for Autonomous Management of Networks and Services*. Springer, 2010, pp. 163–166.

[15] B. Zhu, A. Joseph, and S. Sastry, "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, 2011, pp. 380–388.

[16] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.

[17] J. Jiang and L. Yasakethu, "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, 2013, pp. 82–88.

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

[19] L. Maglaras and J. Jiang, "Intrusion detection in scada systems using machine learning techniques," in *Proceedings of the 2nd SAI conference*. SAI, 2014.

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

[21] L. Spitzner, "Honeypots: definitions and value of honeypots," *URL: http://www. t racking-hackers. com/papers/honeypots. html*, 2003.