J. Parallel Distrib. Comput. 119 (2018) 18–26

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A scalable distributed machine learning approach for attack detection in edge computing environments

Rafał Kozik [a,*], Michał Choraś [a], Massimo Ficco [b], Francesco Palmieri [c]

[a] UTP University of Science and Technology in Bydgoszcz, Institute of Telecommunication and Computer Science, Bydgoszcz, Poland
[b] Dep. of Industrial and Information Engineering, Università degli Studi della Campania "Luigi Vanvitelli", Aversa (CE), Italy
[c] Department of Computer Science, University of Salerno, Fisciano (SA), Italy

HIGHLIGHTS

- Implementation of a distributed attack detection platform for IoT applications.
- Traffic classification based on models built in the cloud.
- Traffic inspection and classification activities pushed to dedicated edge devices.
- More computationally expensive operations shifted to the cloud.

ARTICLE INFO

Article history:
Received 30 October 2017
Received in revised form 10 January 2018
Accepted 20 March 2018
Available online 6 April 2018

Keywords:
IoT
Edge computing
Extreme learning machines
Distributed machine learning
Attack detection

ABSTRACT

The ever-increasing number of IoT applications and cyber–physical services is introducing significant challenges associated to their cyber-security. Due to the constrained nature of the involved devices, some heavier computational tasks, such as deep traffic inspection and classification, essential for implementing automatic attack detection systems, are moved on specialized "edge" devices, in order to distribute the processing intelligence near to the data sources. These edge devices are mainly capable of effectively running pre-built classification models but have not enough storage and processing capabilities to build and upgrade such models from huge volumes of field training data, imposing a serious barrier to the deployment of such solutions. This work leverages the flexibility of cloud-based architectures, together with the recent advancements in the area of large-scale machine learning for shifting the more computationally-expensive and storage-demanding operations to the cloud in order to benefit of edge computing capabilities only for effectively performing traffic classification based on sophisticated Extreme Learning Machines models that are pre-built over the cloud.

1. Introduction

The emerging IoT technology is rapidly bridging the gap between the traditional information services and the physical environment surrounding us, by strongly integrating more and more sophisticated and miniaturized sensing and remote control devices with Internet-based ubiquitous communications, storage and processing capabilities. This results in unlimited potentialities in many mission critical applications, such as environmental monitoring, healthcare, traffic, emergency and infrastructure management, defense, business and intelligent material science, also drastically improving human interaction with computing facilities. Recent technological trends are fostering the migration of modern large scale IoT solution into the cloud, allowing cyber–physical devices, like controllers, sensors and actuators to be virtualized as cloud resources, so that they can be obtained on demand by multiple different tenants [11]. These cyber–physical devices are essentially demand–response systems (e.g., any kind of sensors or actuators), usually limited in their processing capabilities and often powered by a small battery, consisting of a cyber communication part strictly coupled with a physical control one, that are explicitly conceived to manage sporadic events [16], inherently involving significant fluctuations in communications, processing and storage resources needed. Accordingly, in order to reduce the potentially huge volumes of traffic introduced by these devices towards the cloud, necessary to have their produced data being timely processed, and simultaneously handle in a more scalable way their need for local processing capabilities, there is an emerging trend aiming at distributing some computational and storage power on specialized devices, located nearer to the data sources, according to the so-called "edge computing" paradigm. This enables

* Corresponding author.
E-mail address: rafal.kozik@utp.edu.pl (R. Kozik).

https://doi.org/10.1016/j.jpdc.2018.03.006
0743-7315/

pushing specific applications and service intelligence away from centralized points to the logical extremes (namely the edge) of an infrastructure.

While such an elastic behavior naturally fits in the edge-cloud computing scenario, its side effects can pave the way to multiple security problems and challenges. With the diffusion of service-oriented architectures [13] many vendors started implementing web service-based interfaces at the individual device level, in order to provide access to their fundamental functionalities. Since these solutions commonly do not operate within controlled and security-hardened environments, these interfaces introduce additional security risks that are not usually present in pure virtualized solutions, such as side channel attacks, where a malicious third party gains the ability of compromising a crypto-system functionality by looking at information exposed by the physical component and probing for specific vulnerabilities. In addition, the above interfaces can become the target for Denial of Service (DoS) attacks, exploiting specific vulnerabilities of the underlying communication and transport facilities.

Accordingly, the above security risks and challenges require integrated solutions specifically developed to provide security alert facilities at the edge level, with advanced infrastructure-level attack detection capabilities that leverage the computing power provided by edge systems to perform granular traffic control and spot suspicious communication patterns and trends that can be associated with any kind of cyber-attacks, ranging from exploitation of vulnerable software interfaces, to complex social engineering, or crypto-viruses and worms properly developed to compromise the integrity of specific industrial cyber–physical devices. Such edge-specific detection systems, should provide "Security-as-a-Service" capabilities looking for both physical intrusions (motion detection and tracking, power monitoring etc.) to cyber-related ones (connections monitoring etc.) usually implemented by properly training binary classifiers that perform online inspection of traffic logs provided by network elements and edge-level aggregation devices (e.g. application gateways performing intermediate data collection and processing activities). However, such training activity requires large volumes of attack data and plenty of computational power to successfully build the attack/threat detection model and keep it up-to-date with emerging menaces through continuous learning. This usually imposes a serious barrier to deployment of such solutions at the edge level, due to the limited processing and storage capabilities characterizing edge devices, that are usually able to effectively handle traffic classification based on already built models. Therefore, in this work, we propose a distributed detection scheme, based on Extreme Learning Machine (ELM) classifiers, that uses HPC cluster resources available over the cloud for time-consuming and computationally-expensive classifier training tasks. We proved the correctness and usefulness of the proposed methods through extensive experiments on a cyber-security related real-world dataset. Compared with other binary classification schemes, ELMs obtain better generalization performance together with much faster learning/training speed. Instead, the design and properties of the ELM classifier allow for efficient computations and analysis of the collected data at the edge level, resulting into an extremely scalable distributed attack detection architecture where traffic features are only extracted and matched at the wire speed on edge devices. We prove the correctness and usefulness of the proposed methods by performing extensive experiments on publicly-available cyber-security related benchmark datasets.

2. Related work

As stated in [17], securing cyber–physical solutions within the IoT poses additional requirements, due to the fact that communication channels between the cyber environment and physical assets are not typical and need special attention in terms of protection practices. Nowadays, legacy Cyber–Physical Systems evolve into industrial IoT technologies, that also demand specific approach to security [2]. From the cyber security perspective, the most important objectives (and challenges) include the design of protection systems capable to ensure early detection in case of DoS attacks, as well as resilience to attack propagation (guaranteeing the system integrity), together with security and trustworthiness of all the third-party components involved. Many industrial areas have also their own legal requirements related to the accountability and logging for security purposes, what makes difficult providing the universal and multi-sector security framework for different IoT solutions. On the other hand, with the growing popularity of cloud-edge computing models and with the exponentially increasing volumes of data involved (Big Data concept [23,27]), many large scale solutions, such as those used in power distribution or healthcare, can be efficiently and effectively managed only with the assistance of Cloud services [25]. Clearly this also implies the same trends and dynamics in managing their security.

Critical security issues in IoT solutions from different application domains have been analyzed in [20] whereas the most common security menaces in each of the three tiers (physical, network, and application) characterizing the architecture of cyber–physical systems, have been identified, classified, and carefully analyzed in [8] by also presenting some high-level practices (device identification and authentication, access-control etc.) for mitigating such threats. A methodological framework for the identification of security risks in these environments, starting from the identification of their specific security requirements and determining the security policies needed to meet them has been presented in [7]. Specific security design challenges in cloud-based cyber–physical systems have been presented in [22] pointing out on the physical implications of cyber attacks. A context-aware cloud middleware restricting access to resources based on contextual parameters, such as location and time in order to reduce the risks associated to tricking sensor values has been proposed in [11], while a security-enforcement framework for power monitoring in cyber–physical-based smart grid infrastructures has been presented in [21]. A specific protection solution, operating at the application layer, against HX-DoS (HTTP and XML Denial of Service) attacks can be found in [3]. In addition, the fundamental attack detection and identification problems in these architectures have been discussed in [19] by designing a distributed attack detection model and identifying known limitation in monitoring activities, whereas new research issues in IoT-specific intrusion detection systems have been analyzed by [15]. Moreover, the definition of various attacks related to wireless mesh networks and their effective detection have been discussed in [1] and [14]. Finally, the use of machine learning in network attack detection [5,6,18] is considered a quite promising approach in several areas, in order to cope with the scale and dynamism of modern systems, and the IoT seems to be a really challenging arena for such kind of approaches. Our proposal opens a new direction in attack detection by combining modern machine learning technology with distributed computing capabilities provided by the edge-cloud computing model, resulting in a modern framework that seems to be able to face all the known security challenges in modern cyber–physical systems.

3. ELM-based attack detection system

The architecture of the proposed attack detection solution for edge/cloud-assisted IoT environments is sketched in Fig. 1. Its

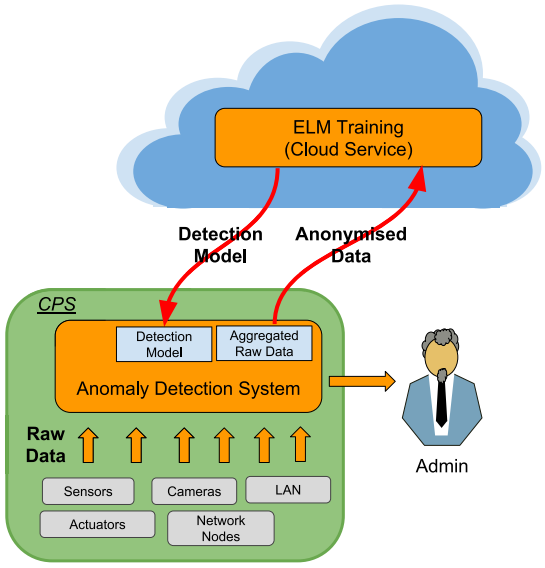

Fig. 1. High-level overview of the proposed cloud service for ELM classifier training.

structure leverages the nature of the ELM machine learning technology and the native elastic and distributed nature of modern edge/cloud infrastructures. The way how data is processed by ELM classifier can be broken-down into two distinct stages. First, the original feature vectors are transposed into a random and multi-dimensional feature space. Afterwards, the classifier is trained on that artificial data to produce the threat and normal activity model.

In the distributed edge/cloud computing reference environment these two stages can be easily separated based on the different entities involved: edge-level security devices and cloud service units (e.g. runtime/storage servers). We assume that the edge devices have not enough resources available to store and analyze the collected data and produce the detection model. Therefore, such data can be passed to multiple service units over the cloud for analysis. However, the data is not sent to the cloud in the original form (where it will be collected and permanently stored) as it may contain sensitive information. Instead, it is first obfuscated through random projection and hence represented in a different form. The cloud computing and storage facilities are used for training the ELM classifier on such anonymized data and build the detection model. Afterwards, such model will be used by classifiers running on the edge devices in order to detect cyber threats and anomalies in the monitored environment.

3.1. Network traffic data acquisition

The data sent by the involved network entities to our Detection System are structured as NetFlow collection entries. The NetFlow is a standardized format for describing bi-directional communication characterized by the following attributes:

- StartTime — Start time of the recorded NetFlow,
- Dur — Duration,
- Proto — IP protocol (e.g. UTP, TCP),
- SrcAddr — Source address,
- Sport — Source port,
- Dir — Direction of the recorded communication,
- DstAddr — Destination Address,
- Dport — Destination Port,
- State — Protocol state,
- sTos — Source type of service,
- dTos — Destination type of service,
- TotPkts — Total number of packets that have been exchanged between source and destination,
- TotBytes — Total bytes exchanged,
- SrcBytes — Number of bytes sent by source.

3.2. Feature vectors extraction

The single NetFlow entries usually does not provide enough evidence to decide whenever a particular device is under attack, infected or if a particular request has malicious symptoms referring to an exploit. Therefore, it is quite common [9,10] that NetFlows are aggregated within specific time windows so that more contextual data can be extracted and malicious behavior recorded (e.g. port scanning, packet flooding effects, etc.). In such approaches, various statistics are extracted for each time window.

In our approach the statistics collection can be considered as computationally inexpensive (so that it can run on edge devices) and has cumulative nature, reflecting their local data aggregation and processing behavior. In principle, the proposed feature extraction method aggregates the NetFlows in the aforementioned windows corresponding to specific time ranges. For each time window, we group the NetFlows by the window number and IP source addresses. For each group (time window, IP source address) we calculate the following statistics:

- number of flows
- sum of transferred bytes
- average sum of bytes per NetFlow
- average communication time with the unique IPs addresses
- number of unique IP addresses
- number of unique destination ports
- number of unique protocols (e.g. TCP, UDP) used by specific IP source addresses.

The advantage of this approach is that it easily allows the identification of possibly compromised hosts/IP addresses by issuing simple queries.

Malicious behaviors recorded over a specific time period can be observed on different time-spans. For example, some time-intensive malicious activities can be observed in a short time-range and usually are characterized by a significant volume of data that is of a specific type (e.g. a number of opened connections, the set of distinct port numbers referred to, etc.). On the other hand, more sophisticated attacks will be stretched over the time and may exhibit relevant information in time windows of a varying length.

To address the above-mentioned issues, we have adopted a sort of multi-scale analysis. As it is shown in Fig. 2, the multidimensional NetFlows can be seen as time series. Moreover, the NetFlows are separated in time windows and for each time window traffic statistics are calculated. For simplicity reasons, in the presented example we are using two scales (one and two minutes wide time windows), but this approach can be easily extended to any number of scaling factors, in order to perform multi-time scale resolution analysis. In the presented example the final feature vector is obtained as a concatenation of the vectors which are calculated for both of the time windows.

3.3. Extreme learning machine-based classification

ELM (Extreme Learning Machines) are characterized by a three layer feedforward structure whose first layer is the input one, the second, also known as the hidden layer, has the purpose of projecting the first layer to a higher dimensionality by relying on a very large number of non-linear sigmoid neurons, and finally, the third and last layer, consisting of neurons with linear input–output characteristic, plays the role of output. They can be successfully

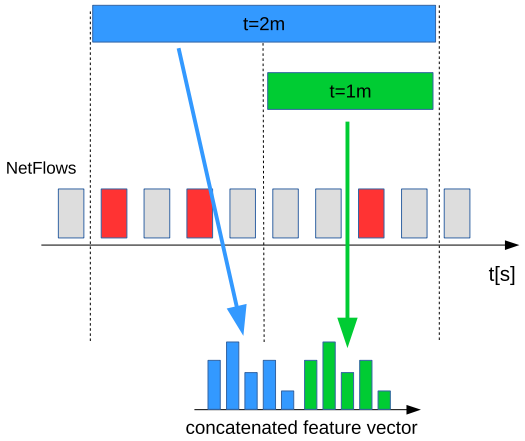

Fig. 2. Overview of the technique used for multi-scale NetFlow time series analysis.

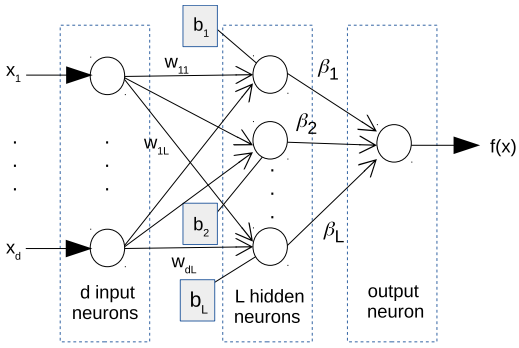

Fig. 3. Extreme learning machine.

used in classification [12], by generalizing the concept of single-hidden-layer neural networks (see Fig. 3) that are able of learning without iterative processing. These models exhibit very satisfactory generalization performance and are able to learn thousands of times faster than networks trained using backpropagation. They are also able of effectively overcoming most of the common problems affecting the traditional gradient-based learning algorithms, such as those related to local minima, stopping criteria, overfitting, etc.

The neurons in the hidden layer may not be necessary neurons in a classical sense and they do not require tuning during the learning phase. That is, the hidden neuron parameters can be randomly generated, making them fully independent of the training data. The output weights can be determined analytically by using a least squares regression method such as the Moore–Penrose generalized inverse. More precisely a response $f(x)$ for a signal x is calculated using the following equation:

$$f(x) = h(x)\beta \tag{1}$$

where β indicates the output layer weights an $h(x)$ is the response of the hidden layer to the input signal x. The hidden layer performs random and non-linear projection based on h. More precisely, each neuron in the hidden layer is fully connected with d input signals through the set of randomly initialized weights w. Each neuron has also a bias b which is again randomly selected. Therefore, Eq. (1), for L neurons in the hidden layer characterized by a response h, and some activation function G, is represented as:

$$f(x) = \sum_{i=1}^{L} G(w_i, b_i, x)\beta_i. \tag{2}$$

Because w and b are initialized randomly, the only parameter that can be tuned during the training process is β. It can be estimated by minimizing the approximation error in the squared error sense.

Let $X, T = \{(x_j, t_j)\}_{j=1}^{N}$, where $x_j \in \mathbb{R}^d$ and $t_j \in \{-1, +1\}$, be the training dataset. One can express the activation matrix as:

$$H_{ij} = G(w_i, b_i, x_j) \tag{3}$$

From this perspective, the associated optimization problem can be formulated as:

$$\begin{aligned} &\min_{\beta \in \mathbb{R}^L} \quad \|H\beta - T\|^2 \\ &\text{where} \quad H_{ij} = G(w_i, b_i, x_j), i = 1, \ldots, L, j = 1, \ldots, N. \end{aligned} \tag{4}$$

The optimal solution in terms of mean squared error can be achieved using the Moore–Penrose (MP) pseudo-inverse (indicated as $H^\dagger$) of matrix H, such that β can be calculated by using the formula:

$$\beta = H^\dagger T. \tag{5}$$

The $H^\dagger$ Moore–Penrose inverse matrix can be built by using the traditional orthogonal projection method, so that:

$$H^\dagger = (H^T H)^{-1} H^T. \tag{6}$$

In the formula (6), the H matrix containing the responses of hidden neurons has L columns and N rows and can be represented as:

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \ldots & h_L(x_1) \\ \vdots & \vdots & \vdots \\ h_1(x_N) & \ldots & h_L(x_N) \end{bmatrix}. \tag{7}$$

In our approach, we consider a binary classification problem, aiming at discriminating an attack from normal activities, thus the classification of a new data sample x can be achieved by using the following equation:

$$C(x) = sign(\sum_{i=1}^{L} G(w_i, b_i, x)\beta_i). \tag{8}$$

3.4. Data anonymization

The hidden layer of neurons used by ELM is used to map the original feature vectors to the new feature space. The mapping can be known only to the entity collecting the data and thus anonymization can be easily achieved on edge devices. (See Fig. 4.)

Usually, the mapping function takes 3 parameters: x — the original feature vector, a — the random projection matrix, and b — the random bias vector. The simplest case would be a hashing function $G(x, a, b) = a \cdot x + b$. However, the variety of potential mapping functions is quite big. In the literature there are different proposals of hidden layer architectures and techniques for feature mapping. For instance, in [4] the author has used Jaccard index, while authors in [24] have used self-organizing maps (SOMs). Techniques adapting fuzzy logic are quite common [26]. However, the commonly used mapping functions have been presented in Table 1.

We have chosen for this purpose a sigmoid activation function. This type of mapping uses randomly generated vectors (bias) and a randomly generated projection matrix. Each training sample x (before being distributed to the cluster) is transformed with the projection matrix a producing an intermediate result. Afterwards, the bias is added. The final vector is obtained by passing the result from the previous operation to the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$.

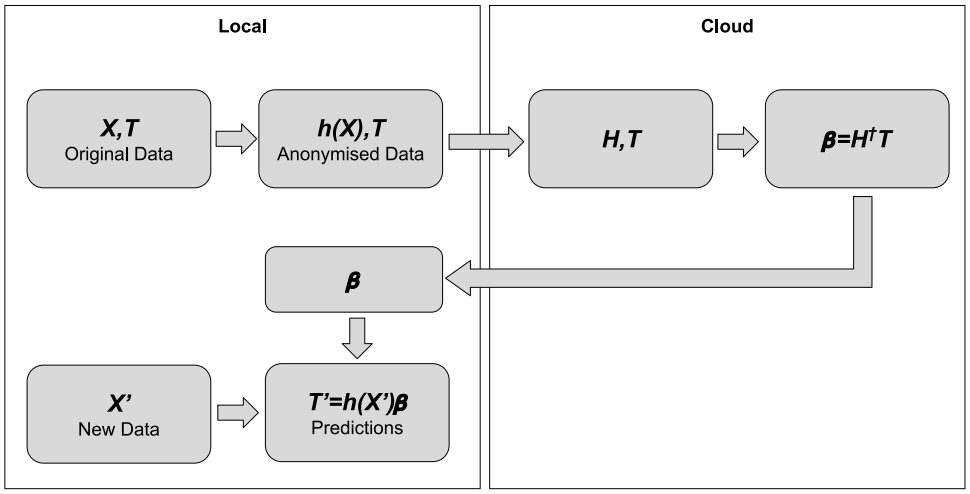

Fig. 4. The overview of an idea for cloud-assisted ELM classifier training.

Table 1
Type of activation functions used in the hidden layer.

$G(x, a, b) = cos(a \cdot x + b)$	Cosine
$G(x, a, b) = \frac{1}{1+exp(-(a \cdot x+b))}$	Sigmoid
$G(x, a, b) = \frac{1-exp(-(a \cdot x+b))}{1+exp(-(a \cdot x+b))}$	Hyperbolic tangent
$G(x, a, b) = exp(-b\|\|x - a\|\|)$	Gaussian

4. Distributing ELM training in the cloud

All the model construction activities that require storing and processing a huge volume of feature vectors can be performed throughout the virtually unlimited storage and computing facilities available over the cloud, so that the resulting classification model can run on multiple edge detection devices in order to achieve full distribution at both the training and the traffic classification level with maximum scalability. Distribution of the training burden on HPC resources available over the cloud is performed by relying on the Apache Spark distributed computing facilities.

4.1. The apache spark framework

The proposed distributed Extreme Learning Machines classifier adopts the Apache Spark scalable data processing framework. Such framework provides an engine that is able to effectively process big data workloads. It uses the data abstraction known as resilient distributed dataset (RDD) to manage the data distributed on an HPC cluster and to implement fault-tolerance over the cloud. There are several key architectural concepts and elements necessary to describe and understand the distribution of heavy computation activities, with the Apache Spark framework:

- node (or host) — is a machine belonging to an HPC cluster,
- driver (an application) — is a module that contains the application code and communicates with the cluster manager,
- master node (a cluster manager) — is the element communicating with the driver and responsible for allocating resources across applications,
- worker node — a machine that can execute application code and holds an executor,
- context (SparkContext) — it is the main entry for Spark functionalities, which provides API for data manipulations (e.g. variables broadcasting, data creations, etc.),
- executor — is a process on a worker node that runs tasks,
- task — is a unit of work sent to the executor.

Moreover, Apache Spark has its own scalable machine learning library called MLlib handling distributed matrices and hence potentially capable of distributing our ELM training activities on a large scale. However, although distributed matrices in MLlib have a multiplication method, this only takes a local matrix as a parameter. This implies that with MLlib we cannot multiply a large matrix with another one as we may face out-of-memory issues. Therefore, in this section, some algorithmic solutions are proposed to solve this problem.

4.2. Making the distributed ELM classifier cost-sensitive

In some machine learning applications, the training samples must be weighted. One of the reasons behind this may be data imbalance problem. In such cases, the number of training samples belonging to some classes is larger in contrast to other classes.

The problem of data imbalance has recently been deeply studied in the area of machine learning and data mining. In many cases, this problem impacts the machine learning algorithms and in terms of result deteriorates the effectiveness of the classifier. Typically, classifiers in such cases will achieve higher predictive accuracy over the majority class, but poorer predictive accuracy over the minority class.

This is caused by fact that the classifier will tend to bias towards the majority class. Therefore, the challenge here is to retain the classification effectiveness even if the proportion of class labels is not equal. The imbalance of labels in our case is significant, as we may expect that in common cases of cyber-security incidents only a few machines in the network will be infected and produce malicious traffic, while the majority will behave normally. Furthermore, attacks are not extremely frequent, at least respect to the amount of normal transaction characterizing the traffic observable within a certain time range. In other words, most data contains clean traffic, while only a few data samples contain malware-related observations.

In general, solutions solving this problem can be categorized as data-related and algorithm-related. The methods belonging to the data-related category use data over-sampling and under-sampling techniques, while the algorithm-related ones introduce a modification to training procedures. This group can be further classified into categories using cost-sensitive classification (e.g. assigning a higher cost to majority class) or methods that use different performance metrics (e.g. the Kappa metric). Recently, cost-sensitive learning has been reported to be the most effective solution for class-imbalance in the large-scale setting.

This can be transposed in our specific ELM-based environment where with weighted ELM we can assign different weights coefficients to the training errors. We can consider the training process (estimation of the $\hat{\beta}$ parameters) as a balanced and L2-regularized ridge regression of the form:

$$\hat{\beta} = \min_{\beta \in \mathbb{R}^L} \left\{ \frac{\lambda}{2} \|\beta\|_2^2 + \frac{1}{2} \sum_{i=0}^{N} C_i \|e_i\|_2^2 \right\} \tag{9}$$

where β are the model parameter to be optimized, λ is the regularization term, C_i is the weight of the ith training sample, and e_i is the training error for ith sample. By using the previously introduced matrix notations, the expression (9) can be rewritten as:

$$\hat{\beta} = \min_{\beta \in \mathbb{R}^L} \left\{ \frac{1}{2} (H\beta - T)^T C (H\beta - T) + \frac{\lambda}{2} \|\beta\|^2 \right\}. \tag{10}$$

Finally, the closed form solution to $\hat{\beta}$, can be obtained using following formula:

$$\hat{\beta} = \left(\lambda I + H^T C H\right)^{-1} H^T C T. \tag{11}$$

4.3. Distributing linear algebra calculations

The formula (11) for calculating $\hat{\beta}$ requires the construction of inverse matrices, as mentioned before, the SVD decomposition is commonly used for this task, instead of a direct approach. In our case, we will have a tall and thin H matrix with a high number of rows and relatively low number of columns (in contrast to rows). This lets us leverage the distributed environment capabilities provided by Apache Spark over the cloud.

The whole procedure to calculate β follows the Algorithm 1. It requires some input parameters as: the cost matrix C indicating the weight of each sample (usually the minority class gets higher values to reduce the bias towards majority class), the λ factor (which controls the model complexity), the vector T containing the target labels, and the matrix H with responses of the hidden layer.

Algorithm 1 Distributed ELM training

Input: Cost matrix C, regularization factor λ, hidden layer response matrix H, and T vector of target labels.

Output: The output layer weights $\hat{\beta}$

1. Let Z be diagonal matrix of square roots of elements of C.
2. Let $A = ZH$ and $B = ZT$ (calculated using Map-Reduce).
3. Calculate (using Map-Reduce) the matrix $G = A^T A$.
4. From G matrix calculate right singular vectors V, singular values D, and left singular vectors U.
5. Calculate $\hat{\beta} = V(D^2 + \lambda)^{-1} D U^T B$

First, we scale the values of the matrix H and vector T according to costs stored in the C matrix. This operation can be easily distributed by using the Map-Reduce programming model.

The calculation of the $A^T A$ matrix can be seen as a sum of outer-products of rows. This can be easily distributed as well by using Apache Spark implementation of Map-Reduce aggregation pattern based on multi-level aggregation trees.

It may look expensive to compute SVD decomposition of a matrix with hidden neuron responses captured for a large volume of NetFlows, but in fact, we can calculate the matrix with right singular vectors along with singular values by using eigenvalues and eigenvectors of the $A^T A$ matrix. In our method the size of this matrix depends on the number of hidden neurons used for data projection. When this number is reasonably small, the eigenvalues and the eigenvectors can be calculated locally at a driver. Otherwise, Apache Spark provides the implementation which utilizes ARPACK, a software designed to solve large scale eigenvalue problems.

5. Performance evaluation

5.1. The methodology

In order to evaluate the performance of the proposed distributed attack detection framework, we have used the CTU dataset to mimic IoT data sources within the same experimental setup described in [10]. This dataset includes different traffic scenarios, also referable to various types of attacks, including DoS ones introduced by several kind of botnets. Each of these scenarios contains collected traffic in form of bidirectional NetFlow files (generated with Argus), including the labels constituting the ground truth. The data were collected within the context of a realistic testbed at the CTU University, Czech Republic, in 2011.

In order to obtain results comparable to those presented in [10], the same evaluation procedure must be preserved. Therefore the scenarios available in the CTU dataset have been used in the same manner as proposed in the aforementioned work. Therefore the system is trained on one set of scenarios and evaluated on different ones.

For the evaluation purposes we have selected the following scenarios that are relevant for IoT-based cyber–physical systems, namely:

- scanning – the attackers scan the servers for few hours and connect to several remote desktop services,
- C&C (Command and Control) communications — the compromised hosts contact different hosts for C&C purposes and receive some encrypted data,
- infected host — the infected machines become part of a specific malicious activity (e.g. spamming campaign).

6. Evaluation experiments and results

The quantitative results of our experiments have been presented in Table 2. As mentioned before we have evaluated our proposed system on three cyber security related scenarios. For each of the scenarios, we have compared three different configurations of the training algorithm. The configurations are denoted as $ELM(c, w)$, where c indicates the weight of the majority class (which is intended to reduce the bias of the ELM classifier) and w which indicates the width of the time windows used for feature extraction. The first two ELM configurations use single (one-minute length) time windows, while the third one uses the concatenations of time windows (one minute plus three minutes). In all of the experiments, we have analyzed the time-based error metrics proposed in [10]. These are calculated by using the following formulas:

- True Positives Ratio (or Sensitivity):

$$TPR = \frac{TP}{FN + TP} \tag{12}$$

- False Positives Ratio:

$$FPR = \frac{FP}{TN + FP} \tag{13}$$

- True Negative Ratio (or Specificity):

$$TNR = \frac{TN}{FN + FP} \tag{14}$$

Table 2
Effectiveness of proposed method.

Algorithm	Scanning scenario								
	TPR	FPR	TNR	FNR	P	ACC	ERR	FM	MC
ELM(0.3,1)	1.00	0.02	0.08	0.00	0.92	0.86	0.02	0.96	0.82
ELM(0.2,1)	1.00	0.02	0.08	0.00	0.92	0.86	0.02	0.96	0.82
ELM(0.2,1+3)	1.00	0.01	0.07	0.00	0.98	0.99	0.01	0.99	0.91
Algorithm	C&C Communication								
	TPR	FPR	TNR	FNR	P	ACC	ERR	FM	MC
ELM(0.3,1)	0.21	0.01	0.99	0.79	0.92	0.76	0.24	0.35	0.32
ELM(0.2,1)	0.23	0.03	0.97	0.77	0.74	0.74	0.26	0.35	0.30
ELM(0.2,1+3)	0.21	0.03	0.97	0.79	0.76	0.74	0.26	0.33	0.28
Algorithm	Infected hosts								
	TPR	FPR	TNR	FNR	P	ACC	ERR	FM	MC
ELM(0.3,1)	0.62	0.01	0.99	0.38	0.97	0.90	0.09	0.76	0.66
ELM(0.2,1)	0.77	0.04	0.96	0.23	0.86	0.91	0.09	0.82	0.74
ELM(0.2,1+3)	0.83	0.01	0.99	0.17	0.95	0.95	0.05	0.88	0.83

- False Negative Ratio:

$$FNR = \frac{FN}{FN + TP} \tag{15}$$

- Precision:

$$P = \frac{TP}{TP + FP} \tag{16}$$

- F-measure:

$$FM = 2 * \frac{P * TPR}{P + TPR} \tag{17}$$

- Accuracy:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

- Error Rate:

$$ERR = \frac{FP + FN}{TP + TN + FP + FN} \tag{19}$$

- Mattew's correlation:

$$MC = \left(\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \right) \tag{20}$$

In the above equations, the TP (True Positive) counter is incremented during the comparison time window whenever an infected IP address is detected as malicious at least once. The TN (True Negative) counter is incremented whenever a normal IP address is detected as normal during the whole time window. The FP (False Positive) counter is incremented whenever a Normal IP address is detected as infected at least once. Finally, the FN (False Negative) counter is incremented whenever an infected IP address is detected as normal during the whole time window.

In general, it may be noticed that scanning activity, being more noisy, can be more easily detectable than the other ones, characterized by less perceivable traffic patterns. Also, increasing the weight of the minority class allows us to increase the number of TP (in the case of two first configuration for the third and fourth scenarios). However, in these cases, also the number of false positives increases.

Moreover, experiments showed that concatenating time windows of different lengths allows us to significantly increase the effectiveness in case of the third scenario.

It must be also noted that all of the compared algorithms give a relatively low value of *TPR* in the second scenario. This may be caused by the fact that in this scenario the infected machine has used only a few and quite specific channels for C&C communications, which have not been reflected in training data.

In almost all the experiments, we can observe satisfactory precision and accuracy values together with quite low error rates. This is clearly reflected in the F-measure, where precision and sensitivity are evaluated in a joint way, also if TNs are not considered. Accordingly, a better indication comes from the Mattews coefficient, that correlates the observed and predicted binary classifications by simultaneously considering true and false positives and negatives, for which we can observe values expressing a good level of agreement between prediction and observation.

6.1. Effectiveness of the parallelization process

In this section we provide some details on the performance of the parallelization process. We analyzed the effects on the overall processing time associated to the model construction with varying runtime resources and training set size. The results reporting the computation time necessary for training the classifier respectively with about 2 millions and 10 millions samples datasets, with an increasing number of processing nodes have been presented in Fig. 5. It can be noticed that the time required to train the ELM classifier significantly decreases when additional computing resources are added to the cluster.

Moreover, in Fig. 6 we can observe how the computational time of the ELM training process scales when the number of training samples increases, in presence of 4 and 8 parallel processing nodes. Also here it can be easily noticed how adding additional computing resources can decrease the ELM training times. Such improvement is particularly visible for a larger training datasets.

7. Conclusions

In this paper, we have proposed an attack detection approach based on distributed Extreme Learning Machine technology that leverages HPC cluster resources for time-consuming and computationally-expensive classifier training. The design and properties of ELM classifier allow for efficient computations and analysis of the collected data in the edge computing environment.

The proposed approach analyzes and classifies online the aggregated traffic that is captured by means of NetFlows on specific edge nodes, located in proximity of the data sources to be protected. We presented the architecture of the proposed system and several implementation details. In particular, we have shown how the ELM classifier training process can be decoupled and shifted to the cloud in order to benefit of edge computing capabilities only for effectively performing traffic classification based on more sophisticated pre-built models.

We have also proved the effectiveness of the proposed detection scheme by using a variety of experiments on a real-world attacks dataset.

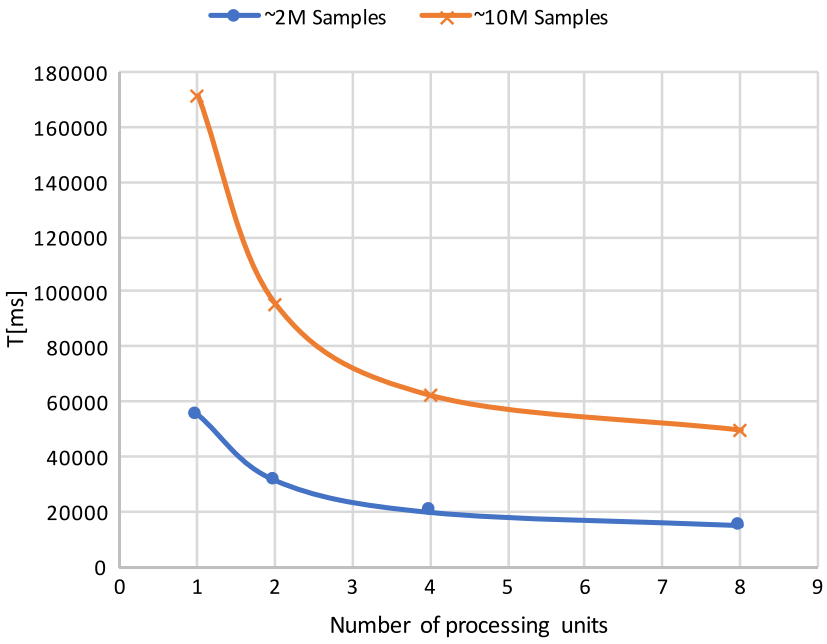

Fig. 5. ELM training time with respect to number of processing units and size of the training dataset.

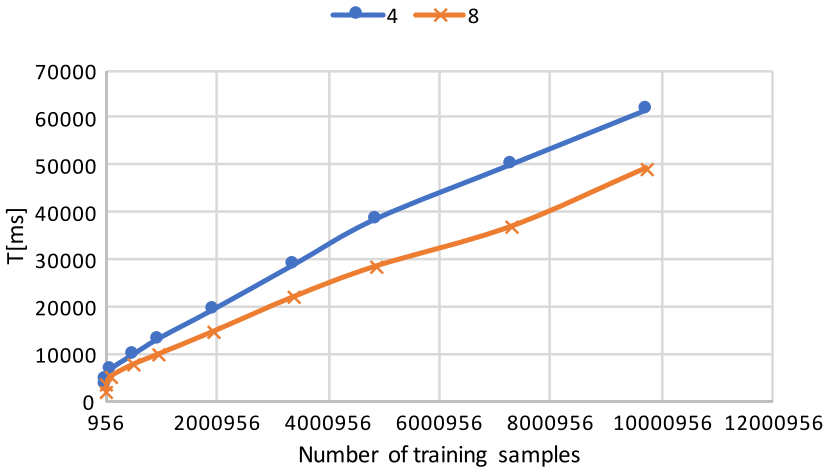

Fig. 6. ELM training time with respect to number of training samples.

References

[1] F.B. Abreu, A. Morais, A. Cavalli, B. Wehbi, E.M. de Oca, W. Mallouli, An effective attack detection approach in wireless mesh networks, Int. J. Space-Based Situated Comput. 5 (2) (2015) 89–99.

[2] S. Ahmad-Reza, C. Wachsmann, M. Waidner, Security and privacy challenges in industrial internet of things, in: Design Automation Conference, DAC, 2015 52nd ACM/EDAC/IEEE, IEEE, 2015, pp. 545–554.

[3] A. Chonka, J. Abawajy, Detecting and mitigating HX-DoS attacks against cloud web services, in: Network-Based Information Systems, NBiS, 2012 15th International Conference on, IEEE, 2012, pp. 429–434.

[4] W.M. Czarnecki, Weighted tanimoto extreme learning machine with case study in drug discovery, IEEE Comput. Intell. Mag. 10 (3) (2015) 19–29.

[5] G. D'Angelo, F. Palmieri, M. Ficco, S. Rampone, An uncertainty-managing batch relevance-based approach to network anomaly detection, Appl. Soft Comput. 36 (2015) 408–418.

[6] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, Network anomaly detection with the restricted Boltzmann machine, Neurocomputing 122 (2013) 13–23.

[7] K.K. Fletcher, X. Liu, Security requirements analysis, specification, prioritization and policy development in cyber-physical systems, in: Secure Software Integration & Reliability Improvement Companion, SSIRI-C, 2011 5th International Conference on, IEEE, 2011, pp. 106–113.

[8] Y. Gao, Y. Peng, F. Xie, W. Zhao, D. Wang, X. Han, T. Lu, Z. Li, Analysis of security threats and vulnerability for cyber-physical systems, in: Computer Science and Network Technology, ICCSNT, 2013 3rd International Conference on, IEEE, 2013, pp. 50–55.

[9] S. Garcia, Identifying, Modeling and Detecting Botnet Behaviors in the Network, (Ph.D. thesis), Instituto Superior de Ingenierá de Software Tandil Departamento de Computacioń y Sistemas, 2014.

[10] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Comput. Secur. 45 (2014) 100–123.

[11] S. Hiray, R. Ingle, Context-aware middleware in cyber physical cloud (CAM-CPC), in: Cloud & Ubiquitous Computing & Emerging Technologies, CUBE, 2013 International Conference on, IEEE, 2013, pp. 42–47.

[12] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), vol. 2, 2004, pp. 985–990.

[13] F. Jammes, H. Smit, J.L.M. Lastra, I.M. Delamer, Orchestration of service-oriented manufacturing processes, in: Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on, vol. 1, IEEE, 2005, p. 8.

[14] X. Li, Y. Shen, J. Zhang, The verifiable secure schemes for resisting attacks in cloud deduplication services, Int. J. Grid Util. Comput. 7 (3) (2016) 184–189.

[15] R. Mitchell, I.-R. Chen, A survey of intrusion detection techniques for cyber-physical systems, ACM Comput. Surv. 46 (4) (2014) 55.

[16] A. Mohan, D. Mashima, Towards secure demand-response systems on the cloud, in: Distributed Computing in Sensor Systems, DCOSS, 2014 IEEE International Conference on, IEEE, 2014, pp. 361–366.
[17] C. Neuman, Challenges in security for cyber-physical systems, in: DHS Workshop on Future Directions in Cyber-Physical Systems Security, 2009, pp. 545–554.
[18] F. Palmieri, U. Fiore, Network anomaly detection through nonlinear analysis, Comput. Secur. 29 (7) (2010) 737–755.
[19] F. Pasqualetti, F. Dörfler, F. Bullo, Attack detection and identification in cyber-physical systems, IEEE Trans. Automat. Control 58 (11) (2013) 2715–2729.
[20] J. Puttonen, S.O. Afolaranmi, L.G. Moctezuma, A. Lobov, J.L.M. Lastra, Security in cloud-based cyber-physical systems, in: P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC, 2015 10th International Conference on, IEEE, 2015, pp. 671–676.
[21] M.A. Rahman, E. Al-Shaer, P. Bera, A noninvasive threat analyzer for advanced metering infrastructure in smart grid, IEEE Trans. Smart Grid 4 (1) (2013) 273–287.
[22] Y.B. Reddy, Cloud-based cyber physical systems: design challenges and security needs, in: Mobile Ad-Hoc and Sensor Networks (MSN), 2014 10th International Conference on, IEEE, 2014, pp. 315–322.
[23] A. Riyadh, S. Sritrusta, P. Dadet, F. Nobuo, Classification extension based on IoT-big data analytic for smart environment monitoring and analytic in real-time system, Int. J. Space-Based Situated Comput. 7 (2) (2017) 82–93.
[24] K.H. Tan, T. Logenthiran, W.L. Woo, Forecasting of wind energy generation using self-organizing maps and extreme learning machines, in: 2016 IEEE Region 10 Conference, TENCON, 2016, pp. 451–454.
[25] Yin Zhang, et al., Health-CPS: Healthcare cyber-physical system assisted by cloud and big data, IEEE Syst. J. 2015 (2015) 545–554.
[26] Z. Yong, E.M. Joo, S. Sundaram, Meta-cognitive fuzzy extreme learning machine, in: 2014 13th International Conference on Control Automation Robotics Vision, ICARCV, 2014, pp. 613–618.
[27] M. Yuriyama, T. Kushida, Integrated cloud computing environment with it resources and sensor devices, Int. J. Space-Based Situated Comput. 1 (2/3) (2011) 163–173.

Rafał Kozik Ph.D. Eng. is an assistant professor in the Department of Telecommunication of University of Technology and Life Sciences in Bydgoszcz (UTP). In 2013 he received his Ph.D. in telecommunications from University of Science and Technology (UTP) in Bydgoszcz. Since 2009 he has been involved in number of international and national research projects related to cyber security, critical infrastructures protection, software quality, and data privacy (e.g. FP7 INTERSECTION, FP7 INSPIRE, FP7 CAMINO, FP7 CIPRNet, SOPAS, SECOR, H2020 Q-Rapids). He is an author of over 70 reviewed scientific publications.

Michał Choraś holds the professor position at University of Science and Technology in Bydgoszcz (UTP) where he is the Head of Teleinformatics Systems Division. His interests include information management and pattern recognition in several domains, such as cyber security, image processing, biometrics and safety (critical infrastructures protection). He has been involved in several EU FP7 projects (e.g. CIPRNet, INTERSECTION, INSPIRE, TACTICS) and he coordinated FP7 project CAMINO on cyber crime and cyber terrorism. He is an author of over 170 reviewed scientific publications.

Massimo Ficco is Assistant Professor at the Universitá degli Studi della Campania "Luigi Vanvitelli". His research interests include security, cloud computing and pervasive systems. He has a Ph.D. in computer engineering from the University of Napoli "Parthenope", Italy. He serves as the editor-in-chief of an international journal, participates to the editorial board of several journals, and has been conference chair and member of international conference committees.

Francesco Palmieri is an associate professor at the University of Salerno, Italy. He is habilitated as a full professor in both Computer Science and Computer Engineering. He received from the same university an Italian "Laurea" degree and a Ph.D. in computer science. His major research interests concern high performance networking protocols and architectures, routing algorithms and network security. Previously he has been an assistant professor at the Second University of Naples, and the Director of the telecommunication and networking division of the Federico II University, in Naples, Italy. At the start of his career, he also worked for several international companies on networking related projects. He has been closely involved with the development of the Internet in Italy as a senior member of the Technical–Scientific Advisory Committee and of the CSIRT of the Italian NREN GARR. He has published a large number of papers in leading technical journals, books and conferences and currently serves as the editor-in-chief of an international journal and is part of the editorial board or associate editor of several other well reputed ones.