An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach

[1]*Farhoud Hosseinpour*, [2]*Payam Vahdani Amoli*, [3]*Juha Plosila*, [4] *Timo Hämäläinen, and*
[5]*Hannu Tenhunen*
[*1, 3 and 5] *Department of Information Technology, University of Turku, Finland.*
{farhos;juplos;hatenhu}@utu.fi
[2 and 4]*Faculty of Information Technology, University of Jyväskylä, 40100, Jyväskylä, Finland.*
[2] *pavahdan@student.jyu.fi*
[5] *timo.t.hamalainen@jyu.fi*

Abstract

The Internet of Things (IoT) is widely used in advanced logistic systems. Safety and security of such systems are utmost important to guarantee the quality of their services. However, such systems are vulnerable to cyber-attacks. Development of lightweight anomaly based intrusion detection systems (IDS) is one of the key measures to tackle this problem. In this paper, we present a new distributed and lightweight IDS based on an Artificial Immune System (AIS). The IDS is distributed in a three-layered IoT structure including the cloud, fog and edge layers. In the cloud layer, the IDS clusters primary network traffic and trains its detectors. In the fog layer, we take advantage of a smart data concept to analyze the intrusion alerts. In the edge layer, we deploy our detectors in edge devices. Smart data is a very promising approach for enabling lightweight and efficient intrusion detection, providing a path for detection of silent attacks such as botnet attacks in IoT-based systems.

Keywords*: Intrusion Detection Systems, Smart Data, Fog Computing, Internet of Things*

1. Introduction

International commerce is developing all the time, which puts pressure on the whole supply chain. Logistics is one of the main factors in a supply chain, and assuring the safety of logistic systems has become one of the critical challenges at the national, European and global levels. Especially, food safety and security has become one of the major concerns for manufacturers and end users. China is utilizing a system for its national food industry that enables a citizen to track the whole supply chain of food and drinks. The European Commission also envisions an integrated approach to food safety aiming to ensure a high level of food safety within the European Union through coherent farm-to-table measures and adequate monitoring [1]. Moreover, food and beverages industries are keen to prove the quality and authenticity of their products from farmland to a dining table. In Figure 1 a holistic view of the supply chain is shown. The safety of the transportation needs to be guaranteed throughout the entire supply chain. Thus, having a system that is capable of protecting the supply chain is of utmost concern - particularly in international markets.

Along with the ever-increasing demand for technological solutions for supply chain management systems, Internet of Things (IoT) has been identified as a promising technology to ease the management and monitoring of the system. On the other hand, such technologies increase cyber security concerns for supply chain and logistic systems. Furthermore, the usage of commercially available off-the-shelf (COTS) components or just-in-time (JIT) manufacturing processes increases the security threats as most of them originate from unsecured foreign facilities. A single failure in a logistics system may result in significant consequences for the shipping materials or human being. A failure can be caused by adversaries who try to compromise the functionality of a system by disrupting software, hardware, physical environment or its connectivity. The attack model for cyber-physical systems comprises short and long term attacks. In short term attacks, adversary immediately tries to disrupt the system and cause a failure. The second type is more sophisticated and difficult to detect as the adversary tries not to leave any footprint by disturbing the system's functionality, before fully

[1] Corresponding Author: Email: farhos@utu.fi

intruding to several components to launch a distributed attack. An intrusion detection system (IDS) is required to tackle cyber threats in logistic systems.

In IoT applications such as cyber-physical systems, safety and security of monitoring a physical environment is a critical issue that is underestimated in recent and current studies. First, unlike in typical computer systems, in IoT systems the physical environment can be affected through IoT actuators. Second, attackers can affect a cyber-physical system by manipulating the physical environment. Moreover, as we are dealing with resource constrained devices in IoT, lightweight approaches need to be undertaken to ensure the quality of service and feasibility of such security measures.

The remainder of the paper is organized as follows. In Section **2** we briefly review the IoT and fog computing technologies. In Section 3 we discuss IDS systems. In Section 4 we present a new promising approach for lightweight IDS called Smart Data. Section **5** presents the proposed Artificial Immune System (AIS) based IDS. In Section 6, the experimental results of the proposed architecture are presented and, finally, we end with some concluding remarks in Section **7**.

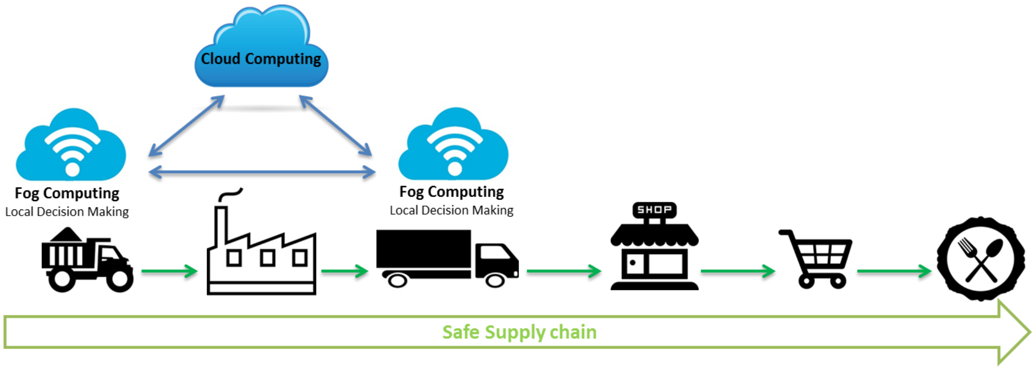

Figure 1. Food supply chain

2. Internet of Things and Fog Computing

The last decade has witnessed the wide deployment of IoT technology in various application domains, and its pervasive role will continue to strengthen in the future. IoT is a concept that realizes communication and control among a very large set of different devices [2]. By connecting the devices, such as sensors, communication devices and data processing units, IoT allows distributed, autonomous decision making and intelligent data processing and analysis [3]. The cloud computing is being recognized as a success factor for IoT, providing ubiquity, reliability, high-performance and scalability [4]. However, because of its geographically centralized nature and communication implications, cloud computing based IoT fails in applications that require very low and predictable latency and which are geographically distributed, fast mobile, or large-scale distributed control systems [5]. Fog computing provides a promising technology to tackle the low-latency and geographical distribution required by IoT devices.

Fog computing is a paradigm extending cloud computing and its services to the edge of the network as shown in Figure 2. Fog is distinguished from the cloud in its proximity to end-users/nodes, dense geographical distribution, support for mobility, heterogeneity, interoperability and pre-processing. Fog computing does not replace cloud computing. On the contrary, it complements cloud computing and aims to provide a computing and storage platform physically closer to the end nodes, provisioning a new breed of applications and services with an efficient interplay with the cloud layer [5]. The expected benefit is a better quality of service for applications that require low latency. Lower latency is obtained by performing data analysis already at the fog layer. This data analysis is lightweight, and therefore more advanced analyses and processing will be done at the cloud layer. Naturally, some applications do not require real-time computation, or they need high processing power, and therefore they are performed at the cloud layer. Fog devices, located at the fog layer, are heterogeneous in nature, ranging from end-user devices and access points to edge routers and switches allowing their use in a wide variety of environments.

The fog layer embodies software modules in the form of fog services and embedded operating systems. At the fog layer, it is also possible to analyze gathered data obtained from the sensor layer and thus make decisions locally. Local decision making is a key to reduce latency, and thus to provide quick responses to unusual behaviors, for example, in the case of security and safety breaches. Such local processing requires lightweight, energy efficient algorithms and applications that can be performed nearby the source of data.

In a systematic view, fog computing is composed of distributed and heterogeneous resources that are deployed based on a hierarchal model. In this model, the fog nodes constitute a virtualized and hierarchical topology and provide a distributed computing platform. Each physical node is composed of computing and storage components and has interfaces for communication with neighboring fog nodes at the same, one step higher, or one step lower level of the hierarchy. Figure 3 illustrates a hierarchical architecture of physical fog computing nodes at different levels [4].

The fog computing platform shown in Figure 2 gathers data from different sensors via a wireless communication module using wireless communication protocols such as Wi-Fi, 6LoWPAN and Bluetooth Low Power depending on the used application. The gateway uses the data in the analysis to anticipate and spot any abnormal behavior in the system. In the case of food transport, the intelligent gateway gathers the food quality data from several sensor nodes. The raw food quality information can be transformed into information (structured data) by developing fog computing services. Naturally, the priority of the safety services is higher that of the other services to ensure prompt action for all threats.

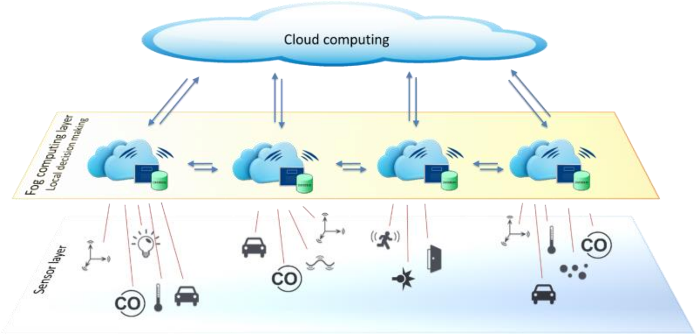

Figure 2: Fog computing platform

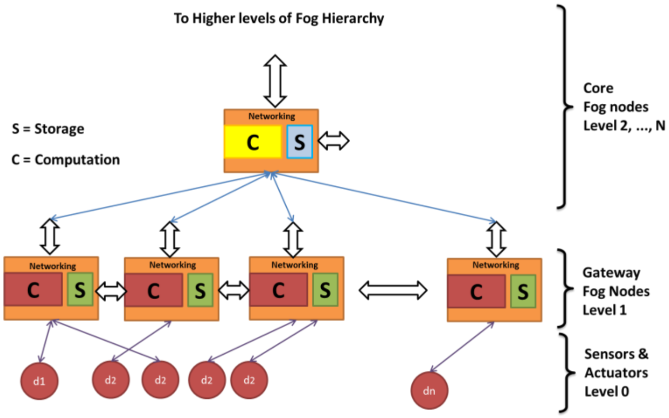

Figure 3. Hierarchical architecture of fog computing.

IoT is not only applied to consumer electronics such as wearables and sport gadgets but also, in the industry in various applications including building/home automation, smart cities, smart manufacturing and smart logistics. In logistics, safety is essential given the mission-critical deployments of logistic systems. ISO 60601 defines safety as “the avoidance of hazards to the physical environment due to the operation of a device under normal or single fault condition.”

3. Intrusion Detection Systems

Detection of silent attacks requires constant monitoring and behavioral analysis of the system's components and communication. Therefore, precise and swift safety monitoring and intrusion detection are of utmost importance in IoT-based logistic systems. An Intrusion Detection System (IDS) with local decision making will prevent failures caused by adversaries and decide proper alert to prevent intrusion or to mitigate the impact of an intrusion. Anomaly-based IDS have been broadly studied as defensive techniques to address the detection of unknown or zero-day attacks. Unlike misuse-based or signature-based types of IDS, which take advantage of the predetermined signature of known attacks, an anomaly-based IDS deals with the detection of new types of attacks that are unknown to the system [6]. This process is done by detecting variation in the systems' behavior from a previously defined normal system profile.

The artificial immune system (AIS) comprises promising techniques in the form of biologically inspired computing that are applied for solving various problems in the field of information security including IDS. The AIS is inspired by the human immune system (HIS), which has the ability to distinguish internal cells and molecules of the body from foreign pathogens, so called self and non-self respectively, and to protect the body against diseases [7]. Like HIS protection against foreign pathogens in the human body, the AIS suggests a multi-layered protection structure for protecting computer networks against attacks [8]. This protection is accomplished through the Innate or Adaptive mechanisms. The innate immune response is immediate and the first line of defense for the HIS and provides a non-specific protection. Therefore, it has no prior knowledge of specific outsiders. The adaptive immune response, on the other hand, is antigen-specific and trained using a pre-defined profile of specific attacks [9]. This is done through specific autoimmune cells called antibodies which act as agents of HIS in the body. The adaptive immune system also includes a "memory" that makes future responses against a specific antigen more efficient [10]. Likewise, the AIS is also proposed as a distributed agent-based system [11]–[13] that is composed of a set of detectors generated as a result of a self-training phase in its main engine. We presented an architectural design of a distributed AIS system in [14]. The AIS comprises three main parts:

1) A training engine that has the responsibility of learning from an initial learning data set and training its detectors. Such processes are complex and require powerful processing units to enable a real-time monitoring of the system. Hence, a cloud computing at the center of the network is the best option to set up the training engine. Training the detectors is an intensive task that is done at the initialization phase of the AIS at the center of the network. Hence, it does not need much communication with the nodes at the edge of the network at this phase. The training process is done through a negative selection algorithm which we presented in our previous work [15]. In the analyzing phase, in turn, the AIS requires more communication between the infected nodes and the main engine.
2) An analyzer engine has the responsibility of analyzing anomalies detected by the detectors to come up with an intrusion alert or to reject the false positive signals. To increase the detection precision, an optimized detector that is called a memory cell detector is generated. Based on our previous works [15], [16], we have utilized genetic algorithms to generate the most optimized memory cell detectors based on the profile of the reported attack and also the triggered detectors. Since the analyzer engine requires more communication with the edge devices, we deploy the analyzer engine on the fog layer
3) A set of detector sensors that are accommodated in each node executing the monitoring task. The detectors are distributed in the network providing an intelligent and collaborative monitoring of the network and the computing nodes. Because each type of attack could be carried out in various forms, to increase the precision of the detection, at the learning phase, a number of different detectors are generated for each attack, and the best detectors, which have more affinity with the targeted attack profile, are selected. So, each type of attack could be detected by a number of detectors. Once an anomaly occurred in a node, a number of detectors will be triggered by the anomaly. If the number of the triggered detectors is more than a threshold, then the anomaly will be reported to the analyzer engine for further analyses. In this case, based on the result provided by the analyzer engine, a more accurate intrusion alert will be given.

Like other anomaly-based detection techniques, the AIS also takes advantage of monitoring variations of the system's behavior as an adaptive immune response, according to a pre-defined normal activity profile. This is done through a learning phase in which a data set containing these profiles is utilized for this purpose. Hence, the efficiency of anomaly detection in the AIS depends heavily on the learning data set. Substantial studies have been conducted so far for improvement and utilization of AIS-based IDS, the majority of which have utilized a pre-defined and offline data set as the learning data for training the IDS. This will reduce the efficiency of the IDS and limit the knowledge base to that particular learning data set. Moreover, it is extremely difficult to create a data set of self-samples with all variations. To cope with this problem, we proposed an online self-training method for our AIS based IDS in [16] using unsupervised machine learning methods, which act as an innate immune response. The innate immune system provides an online and dynamic categorization of network flows into self and non-self flows, which is then used by the adaptive immune system to generate attack-specific detectors.

In integrating the AIS technology in IoT and fog computing systems, a challenging task is building a lightweight smart agent system that is computing and energy efficient and also requires less communication to save the network bandwidth. To this end, we take advantage of the smart data approach, which we presented in [17].

4. Smart Data

Smart data is an active and intelligent data structure using a fog computing system, which facilitates the management of Big Data in IoT-based applications. Such a data cell is initially very simple and light-weight, but it evolves (grows) when traveling through the hierarchical fog computing system towards the cloud, merging with other cells or vice-versa, if the data moves from the cloud towards the actuators.

Figure 4 illustrates the general structure of a smart data cell. The smart data is composed of three main parts: payload data, metadata, and virtual machine. The payload contains the main data collected from the sensors. It undergoes a series of processing or pre-processing steps and is thereby converted into more meaningful information. The metadata part of smart data contains key information such as the source of data (sensors), the destination of data, the physical entity which data belongs to, timestamps, current status and logs as well as rules for accessing, fusing or diffusing, and processing data, for example. The virtual machine part, in turn, acts as a platform which enables and manages the execution of the rules specified in the metadata part. The VM at the very beginning stage contains only basic application codes. Then, it evolves by adding other code modules of the application when they are needed. Each code module provides specific functionalities and services to the smart data. The modular structure of the VM component makes smart data extendable, allowing it to manage the overhead of carrying the code by removing unnecessary code modules and adding the required modules only when they are needed. To enable this, we consider a remote code repository node which contains all necessary code modules as plugins. Whenever a smart data cell requires a specific code module, it communicates with the code repository node and requests for the required code module. To minimize the communication involved in downloading the plugins, the most recently downloaded plugins are also cached in the physical fog nodes for some period of time. So, if the requested code module does not exist in the local fog node, it will be downloaded from the remote code repository node. We have presented a detailed design and specification of our smart data in [17].

Indeed, the smart data acts as a software agent that is able to travel through a fog computing and IoT network, monitor, gather data, and process/pre-process them. The main objective of encapsulating a set of data already at the sensor level, instead of constantly sending discrete data, is to reduce the communication overheads in a very resource constrained environment as well as to reduce the data velocity in the Big Data context. To enable lightweight intrusion detection through the AIS system in a fog computing based IoT environment, we utilized the smart data as a package of suspected anomaly which is needed to be processed and determine if a real intrusion happens.

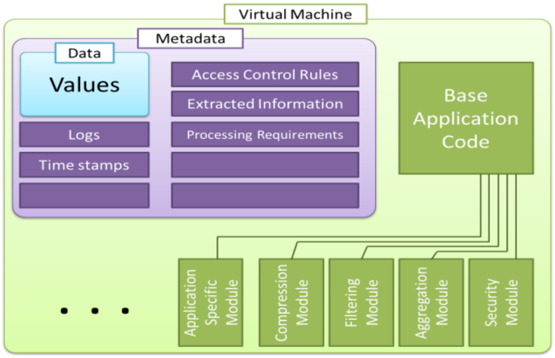

Figure 4. Structure of Smart Data

5. Proposed Intrusion Detection System

As discussed earlier, the training phase in the AIS main engine involves more intensive processes without a significant need for communicating with the distributed nodes at the edge of the network. Hence, the cloud computing at the center of the network is the proper computing platform to run such computations. In contrast, detection by the trained detectors requires less computing and a higher amount of communication compared to processes in the main engine. According to this strategy, we have developed our IDS architecture based on the three-layered structure of fog-based IoT systems (Figure 5). Based on this architecture, the cloud computing accommodates the IDS main engine which is composed of two sub-engines called a clustering engine and a training engine. The clustering engine, using unsupervised clustering methods, divides the primary network traffic into self (normal) and non-self (intrusion) packets which are used as the online training data set for our AIS based IDS. The training engine, in turn, trains a set of detectors based on the learning data obtained from the clustering engine by using a negative selection algorithm. These detectors are called primary detectors. The primary detectors, after the training phase, are stored in a detector repository database at the cloud and also distributed to the devices at the edge of the network.

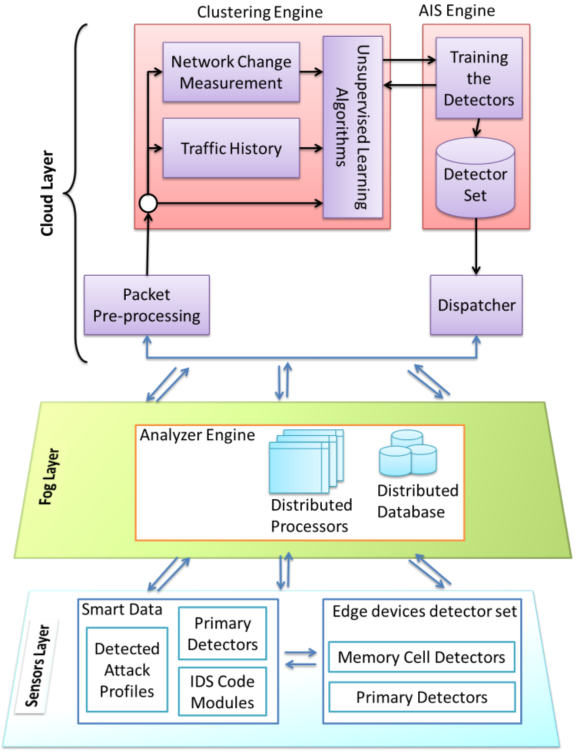

Figure 5. Proposed IDS architecture.

The primary detectors at the edge of the network act as sensors for our IDS which monitor the behaviour of the edge devices. If an anomaly is detected by any of these sensors, they initiate a process for investigating the anomaly by producing a smart data cell. Such smart data contains the information about the suspected connection in its payload and the triggered primary detector in its metadata. In order to increase the precision and avoid false alarms, the smart data is sent to the fog layer for investigation only if the number of triggered detectors exceeds a threshold. Based on our previous work [16], we set this threshold to three. The smart data in the fog layer will fetch the required code modules to build an optimized detector called a memory cell detector for detecting that particular type of attacks. The memory cell detector will be sent to the detector repository in the cloud and from there it will be distributed to all the other devices at the edge.

In this case, the connections that do not trigger a sufficient number of detector sensors will be omitted. So, the IDS will not be able to monitor the trend of the system and detect the long-term or silent attacks. The silent attacks, unlike the short-term attacks, are launched over a longer period of time and from distributed nodes, while keeping the system's functionality as normal as possible to make it difficult and more sophisticated to detect. To cope with this problem, we take advantage of the smart data concept. Smart data has the ability to store and encapsulate the sensory data (profile of a suspected connection detected by any of detector sensors) over a time. We introduce the time dimension to enable the IDS to detect the long-term attacks. If the number of triggered detector sensors is more than a predefined threshold, the smart data cell will be sent to the fog computing platform for further analysis. Otherwise, if the number of triggered detector sensors is less than the threshold, the information of a suspected attack will be stored in a smart data cell. In this case, after a particular time interval, the smart data cell will be sent to the fog computing. In the upper level, the smart data cell will be aggregated with other smart data cells coming from other devices. So, if the similar anomaly had occurred in another device, the aggregated smart data will include the profile of suspected connections collected from a larger amount of devices over time. In a similar way, smart data will be aggregated at the higher levels of the fog hierarchy (Figure 3) with other smart data collected from a larger geographical area. In this case, the smart data become mature which means it contains the information of suspected connections from a larger amount of distributed devices over a longer period. Hence, the silent attacks will become more visible. Once the number of similar attacks that are aggregated and collected by smart data becomes larger than a threshold, the smart data will fetch the code module for analyzing the attack. In this phase, if the attack pattern is similar in all suspected anomalies then the system will alert an intrusion.

In the following subsections the detailed functionalities of each component are discussed:

5.1. Clustering Engine

To detect unseen intrusions without using any previous knowledge (training by labeled traffic or signature), we introduce a clustering engine as an innate immune response. The clustering engine employs the DBSCAN clustering technique to classify the real network traffic into clusters and count them as self, while behaviors outside of the clusters will be deemed as noise or non-self. For this purpose, the engine continuously compares the number of network flows in different network resolutions (subnets of /0, /8, /16, /24), with a threshold which is dynamically computed by our suggested network measurement formula in Table 2. Since high-speed networks have a higher amount of traffic, there is a notable probability of missing the sign of network attacks. To overcome this issue, the system will also control the behavior of the network in small resolutions to minimize the possibility of fading the attacks in the regular traffic.

To obtain a precise threshold, the system requires determining the past behavior of the network. It is probable that small attacks to be fade with the existence of large attacks, thus we have applied standardization on the number of network flows by using logarithm (Log) to increase the probability of detecting small attacks during the existence of large attacks. To determine changes in the network traffic, the system will calculate the "standard deviation" of the number of network flows in different windows from last minute of the traffic. As shown in Table 1 the previous 60 seconds of traffic is broken into four 15 seconds windows. For instance, δ_1 is the standard deviation of the number of network flows in the first window which is from the last 65 seconds to the last 50 seconds of the previous network traffic. As it has been seen in so many datasets, it takes 2 to 3 seconds from starting time of the network attacks (such as DOS/DDOS attacks) till its own peak. To reduce the impact of initial traffic of the attacks on the threshold, the network measurement formula considers a 5 seconds gap between every one minute of traffic to calculate the threshold for the current traffic. The sum of the highest standard deviations (from δ_1 to δ_4) and the heaviest traffic from the last minute of traffic can determine the highest traffic which could be accepted as normal. The following equation represents the network measurement formula which calculates the network traffic threshold "T_{nt}":

$$T_{nt} \;=\; \left(Max\,\{\delta_i | i = 1 \cdots 4\} \;+ Max\,\left\{X_j \log X_j \,\middle|\, j = 1 \cdots 60\right\}\right) \times \gamma$$

Where δ_i is standard deviation of number of network flows in i_{th} window and X_j is number of network fellow in jth second of last minute's traffic. And γ is a coefficient value which can be set to determine the final threshold.

Table1. Elements of network measurement formula

Last Minute Traffic				***Gap***	***Current Traffic***
Window 1	Window 2	Window 3	Window 4		T_{nt}
δ_1	δ_2	δ_3	δ_4		
65-50	50-35	35-20	20-5	5-0	
$X_j \log X_j$					

The DBSCAN algorithm requires two parameters: the maximum radius of the neighborhood (β) and the minimum number of samples required to form a cluster (α). The real network contains traffic from different classes of users such as normal users, busy users, and servers. In general, the number of busy users and servers is smaller than α thus they may not form a cluster in DBSCAN. Since the proposed model in our previous work [16] considers all of the network behavior (in the clean traffic windows) as normal, this will increase the acceptable distance β for DBSCAN by a high value of Δ to include all of the points inside the nearest cluster. Clustering the data with a high value of acceptable distance increases the false negative rate (FNR) in certain cases. To overcome this issue, we will propose a new method which compares the previous behavior of outliers to distinguish normal high traffic users from intrusions.

Similar to the proposed model in our previous work [16], whenever the volume of network flows passes the threshold, the cluster engine uses the DBSCAN to cluster the in-bounded and out-bounded network flows for each machine to find the attacker/s.

During the training phase, DBSCAN will obtain the most accurate α and β from the most recent clean network traffic. In our proposed model, the network traffic can be considered "clean" if it occurs before the threshold (T_{nt}) raise the alarm. Technically, the normal users will form into clusters while the density of busy users or servers may not reach the required level. Nevertheless, since training phase uses the clean network traffic the proposed model will consider outliers as busy machines with normal profiles.

Afterward, to find the anomalous outliers which caused the high volume of network traffic, the clustering engine clusters the suspicious network traffic window. The outliers' IP addresses from detection phase will be compared to their previous profile. If the distance of current behaviors and the previously seen behavior does not exceed the acceptable distance β, the clustering engine will mark it as normal high traffic machine. Otherwise, if the new behaviors of outliers IP exceed the distance it will consider the behavior of that machine as abnormal. It is important to note that if the outlier IP addresses do not have any profile from the training phase, the clustering engine will mark it as abnormal.

5.2. Training Engine

The training engine has the responsibility of training the primary detectors of the IDS. As discussed earlier, because training of the detectors does not require much communication with the edge of the network, this component is located in cloud computing at the center of the network. The training engine first transforms the network flow information into binary strings with a total 112-bit length as a flow profile (Table 2). Then, utilizing a negative selection algorithm, it trains and creates the primary detectors. The negative selection algorithm first creates some random detectors (immature detectors) and then trains them with samples of marked flows from the cluster engine. If any immature detector matches with any self-sample of the data set, then the system will drop it and create another in its place. After checking all of the immature detectors with all self-samples, the remaining immature detectors undergo the next step of the negative selection algorithm and become mature detectors. Each mature detector will be checked with all non-self samples of labeled flows. If a mature detector fails to match with some non-self samples, the system will discard this detector; otherwise, this detector will be added to the final detector set. This process will continue until all non-self packets are matched with at least three mature detectors.

The negative section algorithm utilizes the r-Contiguous matching bit role proposed in [18] to check the matching between two strings. In this method, two strings are matched if they have at least r contiguous identical bits. Finally, the output of the negative selection algorithm is a set of primary detectors, which are archived and synchronized in a detector set repository in the cloud and then distributed to the edge devices. These detectors are analogous to primary immune response in the HIS.

Table 2. Depiction of fields in flows profile strings

Name of the Field	Minimum and Maximum Value	Binary Strings Length (bits)
Destination IP Address	0.0.0.0 - 255.255.255.255	32
Source IP Address	0.0.0.0 - 255.255.255.255	32
Destination Port No	0 – 65535	16
Duration	0 – 65535	12
Protocol	0 – 65535	4
Source Port No	0 – 65535	16

5.3. Analyzer Engine

The analyzer engine has the responsibility of analyzing detected anomalies and giving intrusion alert. It employs the proposed genetic algorithm in [19] to evolve the highly fit detectors activated when an anomaly has been encountered. The analyzer engine requires more communication with the edge devices so we deploy the analyzer engine in the distributed fog computing at the edge of the network. In order to save the deployment cost of the analyzer engine in the fog computing, we take the advantage of the modular structure of the smart data cells. Indeed, we deploy the analyzer engine in code repositories in fog computing. If a smart data which is sent to the fog computing needs the processes of this engine, it fetches the required code module from the nearest code repository. The smart data contains 1) the suspected flow reported from the hosts, 2) profile of the activated detectors, and 3) their affinity with reported flow. It utilizes the fetched code modules to analyze the anomaly and generates an optimized detector, called memory cell detector. A memory cell detector is a high-affinity and attack-specific detector with a higher detection ability and analogous to secondary immune response in the HIS [10]. The following operations are carried out in this case:

A selection operation is undertaken on activated detectors to select the detectors with the highest affinity for cloning and formation of primary population for genetic algorithm. Those detectors having a fitness value greater than or equal to cloning threshold undergo cloning. The cloning threshold is set as follows.

$$\text{Cloning Threshold} = \frac{\sum_{i=0}^{n} \text{Fitness of detectors}}{n}$$

Where “n” is the total number of activated detectors.

Winner detectors that consist of the cloned detectors and remaining activated detectors are subjected to the genetic operators of Mutation, Crossover, and Reproduction, which facilitates the evolution of these detectors. This process is repeated and continued for a few generations until a detector with a fitness value higher than all the winner detectors is generated. The optimized detector from the genetic algorithm is treated as a memory cell.

5.4. Host Side Detectors

The detectors are distributed to the edge devices. Figure 6 illustrates the architecture of edge devices that comprises of sensors, actuators, connection platforms, and processing units. The IDS detectors in the edge devices act as sensors for our IDS. All inbound and outbound network flows are checked using these sensors. In each device, we consider two detectors as follows.

Primary Detectors: comprise a set of trained detectors that have the ability to discriminate between self and non-self flows. These detectors are non-specific and responsible for the primary immune response for anomalies that occur for the first time. If a flow matches a detector with an effective affinity, that detector is considered an activated detector and the flow is suspected as an intrusion. To improve the accuracy of detection and reduce the false-positive errors in IDS, we have defined an intrusion threshold (T_i). If the number of activated detectors by a suspected flow is more than T_i, the flow is detected as an intrusion.

Memory cell detectors: composed of a set of optimized detectors generated by the analyzer engine. As the secondary response of the AIS, memory cells have more accurate intrusion detection abilities. Hence, any flow that activates any of these detectors is treated as an intrusion.

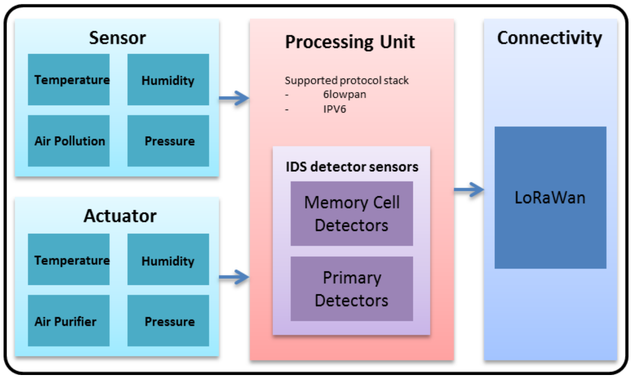

Figure 6. Architecture of Edge Devices

6. Experimental Result

To evaluate the efficiency of two popular clustering algorithms, we utilized KDD-Cup 99 data set, which is extracted from DARPA-98 traffic network. In addition, we have tested our model on SSH Brute Force from ISCX dataset [20]. Since today most of the servers with SSH protocol limit the number of user attempt, we have changed the SSH Brute Force attack in ISCX to a distributed model, which a various number of bots have participated in it. Figure 7 show the network's behavior during the attack. As shown in Figure 7 (A) the ratio of outbound flows to the threshold is below one because the number of attackers is high. However, in Figure 7 (B) the threshold for inbounded traffic raise alarm since all of the traffic goes to the limited number of machines.

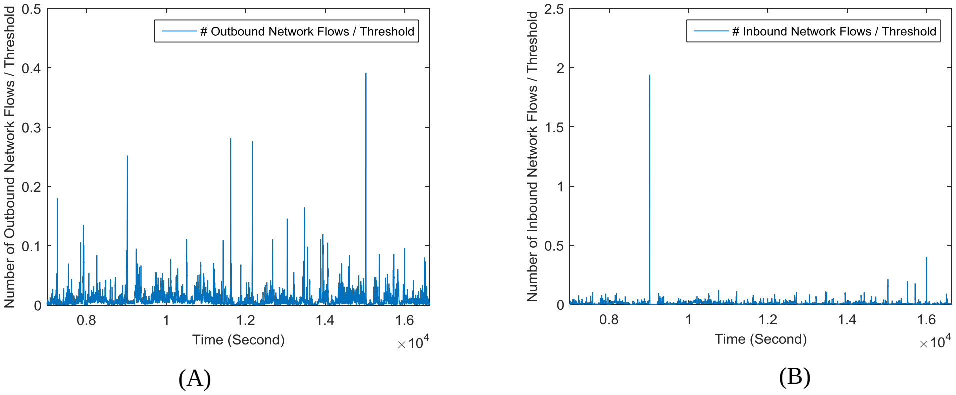

Figure 7. Network's Behaviour during Distributed SSH Brute Force Attack

Figure 8 shows the self-training phase during distributed SSH Brute Force attack. As mentioned before the clustering engine marks the IP addresses of the machines which were located inside the clusters as normal. However, the IP addresses of outliers will be profiled as busy users or servers. As shown in Figure 9 during the comparison phase all of the outliers will be compared to their previous history. If the distance does not exceed the threshold, the cluster engine will mark them as normal devices (with high traffic). Otherwise, if the device exceeds its traffic abnormally, the clustering engine will mark it as the abnormal device. Figure 10 shows the final decision of clustering engine.

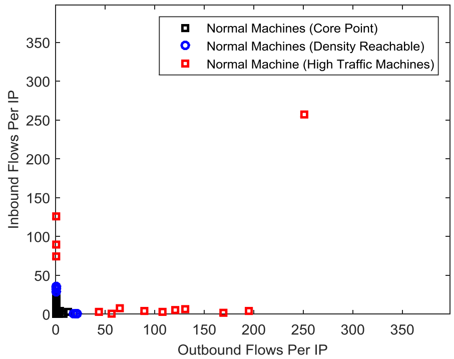

Figure 8. Self-Training Phase During Distributed SSH Brute Force Attack.

Table 3 shows the comparison of average performances of the new proposed model and our previous work. To evaluate the performance of "different behavioral classes" feature in the new proposed model we have added traffic from busy users and servers during the occurrence of an intrusion. Since the proposed model compares the behavior of outliers with their previous history, the overall performance was higher than our previous proposed model [16].

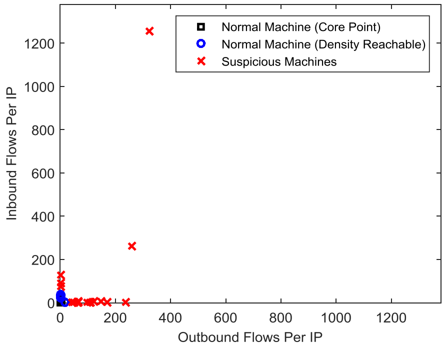

Figure 9. Comparison Phase During Distributed SSH Brute Force Attack.

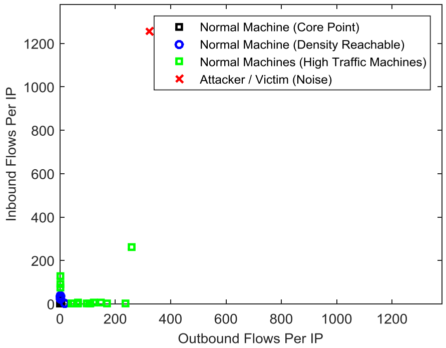

Figure 10. Detection Phase During Distributed SSH Brute Force Attack.

Table 3. Performance Evaluation.

	New proposed model	**Previous Proposed model** [16]
False positive rate	3.51%	4.53%
True negative rate	96.49	95.47%
Accuracy	98.35%	96.23%
Recall	100%	95.37%
Precision	97.83%	91.21%

To examine the effectiveness of the AIS engine and thus the proposed model, in our test, we set the fitness value of "rc" for R-Contiguous matching bit algorithm to 13 and the threshold "Ti" to 3. Furthermore, with experimenting the genetic algorithm for the formation of memory cells, in different circumstances, the values for the probability of genetic operations of Crossover, Mutation, and Reproduction have been set to 30%, 40% and 30% respectively. The system is examined in both centralized and distributed mode. Figure 11 corresponds the self-improvement rate of AIS based IDS in the central and distributed forms. According to this chart, the self-improvement rate in distributed mode is better than centralized mode and it reaches to its steady maximum amount after only 6 cycles, while this happens after 10 cycles in centralized mode. This is due to the dynamic distribution and synchronization of recently created memory cells to each device.

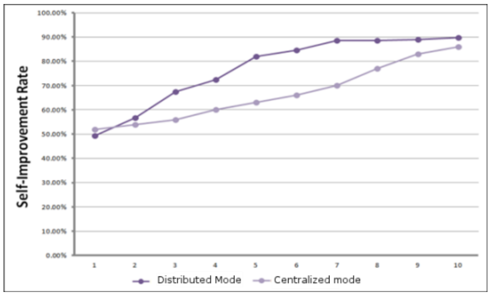

Figure 11. Comparison of self-improvement rate in distributed and centralized mode.

7. Conclusion

Development of lightweight intrusion detection systems is critical for the safety and security of advanced IoT-based logistics systems. In this paper, we presented a new lightweight architecture for an AIS based IDS for IoT systems. This paper extends our previous work [16], into a three-layered structure of IoT systems including the cloud, fog and edge layers. We utilized our proposed smart data approach to develop a lightweight and efficient analyzing engine in fog computing platform for our IDS. Smart data is a very promising framework for enabling lightweight and efficient intrusion detection providing also a path for detection of silent attacks such as botnet attacks in IoT-based systems. We also presented a new approach for clustering the primary network connections which is more efficient method than the one used in our previous work. Our future work will mainly focus on detection of potential botnet attacks using the smart data technology.

Acknowledgement

This work was supported by Turku University Foundation and EIT Digital.

8. References

[1] "Food Safety: overview," *European Commission*, 2016. [Online]. Available: http://ec.europa.eu/food/index_en.htm.

[2] Y. J. Guo-Zhen TAN, Hao Wang, "IoT-based Distributed Situation Awareness for Traffic Emergent Events," *Int. J. Adv. Comput. Technol.*, vol. 5, no. 7, pp. 1050–1059, 2013.

[3] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet : The Internet of Things Architecture , Possible Applications and Key Challenges," in *10th International Conference on*

Frontiers of Information Technology Future, 2012.

[4] A. R. Biswas and R. Giaffreda, “IoT and cloud convergence: Opportunities and challenges,” *2014 IEEE World Forum Internet Things*, pp. 375–376, Mar. 2014.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog Computing: A Platform for Internet of Things and Analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*, vol. 546, N. Bessis and C. Dobre, Eds. Cham: Springer International Publishing, 2014, pp. 169–186.

[6] S. and G. G. Feixian, “Research of Immunity-based Anomaly Intrusion Detection and Its Application for Security Evaluation of E-government Affair Systems.,” *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 20, pp. 429 – 437, 2012.

[7] L. N. de Castro and J. Timmis, *Artificial {I}mmune {S}ystems: A {N}ew {C}omputational {A}pproach*. London. UK.: Springer-Verlag, 2002.

[8] M. Tan, H. Yu, Z. Zhao, Z. Liu, and F. Liu, “An artificial immunity-based proactive defense system,” in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, 2007, pp. 2239–2243.

[9] et al. Xishuang, D., “Multi-word-Agent Autonomy Learning Based on Adaptive Immune Theories,” *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 7, no. 3, pp. 723–745, 2013.

[10] A. A. Ademokun and D. Dunn-Walters, “Immune Responses: Primary and Secondary,” in *eLS*, John Wiley & Sons, Ltd, 2001.

[11] S. Forrest, S. A. Hofmeyr, and A. Somayaji, “Computer Immunology,” *Commun. ACM*, vol. 40, no. 10, pp. 88–96, Oct. 1997.

[12] F. Hosseinpour, K. A. Bakar, A. H. Hardoroudi, and N. Kazazi, “Survey on Artificial Immune System As a Bio-inspired Technique for Anomaly Based Intrusion Detection Systems,” in *Proceedings of the 2010 International Conference on Intelligent Networking and Collaborative Systems*, 2010, pp. 323–324.

[13] S. A. Hofmeyr and S. A. Forrest, “Architecture for an Artificial Immune System,” *Evol. Comput.*, vol. 8, no. 4, pp. 443–473, Dec. 2000.

[14] F. Hosseinpour, K. A. Bakar, A. Hatami Hardoroudi, and A. Farhang Dareshur, “Design of a new distributed model for Intrusion Detection System based on Artificial Immune System,” in *Advanced Information Management and Service (IMS), 2010 6th International Conference on*, 2010, pp. 378–383.

[15] F. Hosseinpour, A. Meulenberg, S. Ramadass, P. Vahdani Amoli, and Z. Moghaddasi, “Distributed Agent Based Model for Intrusion Detection System Based on Artificial Immune System,” *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 7, pp. 206–214, 2013.

[16] F. Hosseinpour, P. V. Amoli, F. Farahnakian, and J. Plosila, “Artificial Immune System Based Intrusion Detection : Innate Immunity using an Unsupervised Learning Approach,” *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 8, no. 5, pp. 1–12, 2014.

[17] F. Hosseinpour, P. Juha, and H. Tenhunen, “Smart Data: Reshaping Data Structure in IoT for Tackling the Five Vs of Big Data using Fog Computing.” TUCS Technical Reports 1159, 2016.

[18] T. Stibor, “Foundations of r-contiguous Matching in Negative Selection for Anomaly Detection,” *Nat. Comput.*, vol. 8, no. 3, pp. 613–641, Sep. 2009.

[19] D. Dal, S. Abraham, A. Abraham, S. Sanyal, and M. Sanglikar, “Evolution Induced Secondary Immunity: An Artificial Immune System Based Intrusion Detection System,” in *Computer Information Systems and Industrial Management Applications, 2008. CISIM ’08. 7th*, 2008, pp. 65–70.

[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.