Journal of Universal Computer Science, vol. 24, no. 5 (2018), 577-602
submitted: 23/1/18, accepted: 28/4/18, appeared: 28/5/18 © J.UCS

Design of Cognitive Fog Computing for Autonomic Security System in Critical Infrastructure

S.Prabavathy
(Thiagarajar College of Engineering, Tamil Nadu, India
s.praba.pranauv@gmail.com)

K.Sundarakantham and S.Mercy Shalinie
(Thiagarajar College of Engineering, Tamil Nadu, India
kskcse@tce.edu and shalinie@tce.edu)

Abstract: The rapid growth of Internet of Things(IoT) has reached all the facets of life including critical infrastructures. It has become the foundation for most of the critical infrastructures. The increased connectivity and the heterogeneity in IoT have widened the attack surface of critical infrastructures for attackers to exploit. Certain cyber-attacks in critical infrastructures can lead to catastrophe and hence the attack has to be identified as early as possible to stop or reduce its impact by activating suitable responses. Therefore, the critical infrastructures require an intelligent security mechanism which can intelligently interpret the attacks from the IoT traffic and efficiently handle the attack scenario by activating appropriate response at faster rate. In this work, an autonomic security system with intelligent self-protect mechanism has been proposed for critical infrastructures. The autonomic security system can autonomously detect known attacks using Extreme Learning Machine, predict the unknown attacks using Gaussian process regression, and select suitable response for handling the attack using fuzzy logic. This intelligence of self-protect mechanism is incorporated in the distributed fog nodes to handle the attack scenario at faster rate and protect the critical infrastructures with minimal human intervention. The experimental analysis of the proposed autonomic security system proves to be efficient in detecting and defending the cyber-attacks with high accuracy and success rate. The results on network load and response time indicates the effectiveness of fog computing in proposed system.
Key Words: Critical infrastructure, Internet of Things, Autonomic system, Fog computing, Gaussian process regression, Extreme Learning Machine.
Category: C.2.1, C.2.4, I.2.6, I.2.11.

1 Introduction

The advances in emerging Internet of Things (IoT) have influenced its involvement in critical infrastructure[Rinaldi et al. 2001]. IoT interconnects numerous heterogeneous cyber physical devices ranging from simple sensors to high end servers using Internet as core technology [Whitmore et al. 2015]. A variety of cutting-edge technologies such as cloud computing, software defined networking, big data analysis and intelligent sensors. have been developed to utilize the complete power of IoT. However, most of these technologies of IoT are in the developing stage and is subject to increase the technical implications of IoT [Atzori et al. 2010]. Thus, the heterogeneity and nonstandard technologies

of IoT create the potential for increasing threat vectors and new vulnerabilities in critical infrastructures in addition to the known vulnerabilities of critical infrastructures.

The critical infrastructures link the multiple networks which lead to vulnerabilities from multiple domains. The increased communication complexity also increases the attack surface for the potential attackers. Moreover, critical infrastructure involves more number of devices with more interconnections leading to more entry points for adversaries to exploit [Ng et al. 2006]. The complexity and heterogeneity in critical infrastructure IoT make the centralized security methodologies unsuitable for large-scale deployment. Therefore, distributed security approach is necessary to protect the critical infrastructure IoT from increasing number of cyber-attacks.

Various methodologies, such as intrusion detection system (IDS), firewall, antivirus, intrusion prevention system (IPS) and intrusion response system (IRS) are utilized to prevent, detect and respond to cyber-attacks in critical infrastructure [Locasto et al. 2005, Marchetti et al. 2009, Keromytis et al. 2004]. However, the involvement of IoT in critical infrastructure increases the complexity of existing security approaches for critical infrastructure. Moreover, IoT enabled critical infrastructure allows multi domain communication and connectivity through handheld devices which widen the attack surface for the attackers to exploit. Security of critical infrastructure involving IoT has drawn increased attention in recent years due to the rapid usage of IoT devices in critical infrastructures [Sandor et al. 2017, Ghani et al. 2014]. The security mechanism of critical infrastructure should intelligently interpret both the known attacks and zero day attacks from the voluminous data traffic generated by the IoT devices which are incorporated in it. Learning from these massive data is essential to identify the security events and their associations by correlating the internal and external information to identify the attacks and to view a wider picture of threats in critical infrastructure.

Autonomic computing technology has got growing attention in the era of IoT. It facilitates the computing system with Self-* properties such as self-configure, self-protect, self-heal to mention a few [Kephart et al. 2003]. Since IoT devices are deployed both in managed and unmanaged environments, the autonomic computing supports IoT for performing its functions with less or no human intervention. The self-protect functionality of autonomic computing can be adopted by the security methodology in critical infrastructure to predict, detect and respond to cyber-attacks. Another major factor that influences the need for self-protect characteristics is increasing number of IoT devices in critical infrastructure, increases the attack surface exponentially. Hence, the devices have to protect themselves from cyber-attack by using their own capability. The self-protect capability is proved to be efficient in detecting and mitigating the

attack with minimal human intervention [Wailly et al. 2012, Hariri et al. 2005]. However, the devices in critical infrastructure range from simple sensors to high computational system, so highly complex security mechanisms and algorithms cannot be employed in resource constrained devices.

Most of the IoT applications process the data from the end devices at the cloud. The cloud based security methodologies have trade off between offloading processing and latency and it is unsuitable for critical infrastructure. Fog computing is an efficient technology to fulfill the requirement of autonomic security system for critical infrastructure. Fog computing is a paradigm developed by CISCO that shifts the data and services to the edge of the network from cloud. It provides fast and actionable decisions from vast amount of real-time data streams generated from IoT environment [Bonomi et al. 2014]. The distributed intelligence of fog computing can be used for providing self-protect mechanism in critical infrastructure IoT by offloading computations and data from critical infrastructure devices to fog nodes instead of cloud server for providing a faster security mechanism which is highly required in critical infrastructure.

The aim of the proposed work is to design an autonomic security system with intelligent self-protect mechanism at distributed fog nodes to predict, detect and respond to cyber-attacks in critical infrastructures.

- Designing self-protect mechanism to predict, detect and respond to cyber-attacks in critical infrastructure.
- Incorporating intelligence in distributed fog nodes for providing self-protect functionality in critical infrastructures.

The rest of the paper has been organized as follows. Section 2 provides the existing research performed so far in this domain. In Section 3, the brief introduction about the proposed system is presented. Section 4 presents the detection module, Section 5 details the forecasting module and Section 6 presents the response module. Section 7 depicts the experimental setup and the results to evaluate the performance of the proposed work. Finally, the conclusion is given in Section 8.

2 Related Work

Various research works have been performed on designing an autonomic computing with self-protection capability. A distributed self-protection system was proposed to automatically isolate malicious nodes by [Claudelet al. 2006]. In this system, various sensors are used to detect the malicious nodes and they also isolate the malicious nodes as defense mechanism. Similarly, a self-protection system for cloud infrastructure has been proposed and it uses self-organizing map to perform behavior analysis for identifying the unknown anomalies[Dean et al. 2012].

A self-configuration system along with self-protection capability was proposed to detect Denial-of-Service (DoS) attacks [Qu et al. 2010]. This system uses automated online monitoring tools along with feature selection mechanism for anomaly detection and protection. The major drawback of this system is that, it can protect only from limited type of DoS attacks. A self-protection architecture for detecting and handling the cyber attacks in cloud environment was proposed [Wailly et al. 2012]. This architecture protects the virtual environment using variety of control structures and by providing adaptable security policies.

The attacking flows from the compromised nodes are traced to identify and act against DoS attack was proposed by [Gelenbe et al. 2007]. This self-protection system autonomously detects and filters the malicious packets from the upstream using traceback of attack flows and free the resources in downstream. A self-defense framework has been proposed to detect the anomaly by using Quality of Service (QoS) parameter of network flows [Hariri et al. 2005]. It has online monitoring capability and it has been designed by using Hotelling T2 methodology to detect the malicious data by analyzing the distance between QoS parameters of the normal network flows . A self-protect approach has been proposed for static wireless sensor networks [Wang et al. 2008]. In this approach, a sensor is monitored by other active sensors involved in the same network. This approach is suitable only for static networks. However, none of these works is specific to IoT applications which involve heterogeneous devices and multiple technologies. A framework for self-protect mechanism was designed for IoT ecosystem [Chen et al. 2014]. It is based on centralized control mechanism which is not suitable for critical infrastructure with heterogeneous IoT devices connecting multiple domains in dynamic environment. Hence, the critical infrastructure needs a distributed autonomic approach which can learn and adapt to dynamic environment of IoT applications.

The novelty of the proposed system is the self-protect mechanism implemented at fog nodes to detect and defend the cyber-attacks in critical infrastructure at faster rate with minimal human intervention.

3 Proposed System

The proposed system uses fog computing to deploy the autonomic security system. Fog computing is based on distributed computing in which data processing, storage and services are handled at the edge devices of the network [Bonomi et al. 2014]. The emerging IoT generates voluminous data from millions of connected devices and it requires analysis with minimum latency. This requirement can be satisfied by fog computing. Fog nodes have abstraction layer which hides the heterogeneity among the devices and provides uniform programmable interface by virtualization. Fog computing requires orchestration to

manage the services and resources among the fog nodes. The IoT application architecture based on fog computing is considered to have 3-tiers: end device, fog and cloud as shown in Figure 1. The end device tier contains the IoT sensing devices from simple sensors to all kinds of devices that can be connected to the Internet. The main purpose of this tier is to collect the data from its environment and pass them to fog tier. Fog tier is based on distributed computing in which data processing, storage and services are handled at the edge devices of the network such as access point, gateway and routers. The cloud tier receives the data from the fog nodes and performs global data management. It also provides final presentation of data based on the requirement IoT application.

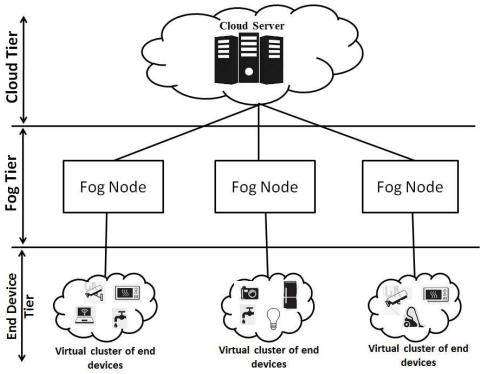

Figure 1: Fog computing based IoT application architecture

In the proposed system, group of IoT devices forms an IoT virtual cluster. These clusters are connected to the fog nodes which in turn is connected to the cloud server. The fog nodes in Figure 1 are incorporated with the proposed self-protect mechanism containing forecasting module, detection module and response module as shown in Figure 2. The IoT devices send and receive data to and from the cloud server through the fog nodes. These data may contain malicious packets that can disrupt the normal operation of critical infrastructure. To protect the critical infrastructure from cyber-attacks, the fog node implements the self-protect mechanism to predict, detect and react to attacks. The detection module detects the known attacks using OS-ELM. The forecasting module predicts the unknown attacks at early stage by using Gaussian process regression. If an attack is predicted or detected by forecasting module or detection module, an alert is generated and sent to the response module. Based on the type of alert message, the response module selects suitable response using fuzzy logic. The orchestration services of fog nodes provide stability for the proposed security system by enabling resilience through resource sharing and load balancing among the fog nodes [Wen et al. 2017, Bonomi et al. 2014]. In the following sec-

tions, the detection, forecasting and response modules of the proposed system are discussed in detail.

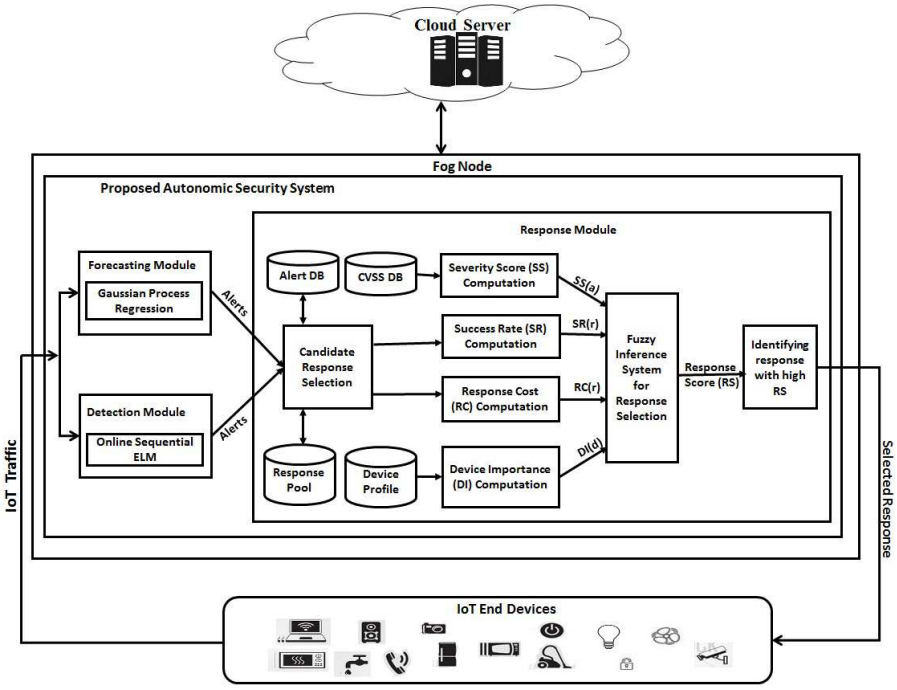

Figure 2: Architecture of autonomic security system

4 Detection Module

Critical infrastructure IoT dynamically connects heterogeneous devices across multiple domains. Therefore, the security mechanism should be a faster learning approach which learns the environment quickly and adapts to changes for detecting the security threats at faster rate. The streaming nature of critical infrastructure IoT traffic favors the need for Online Sequential Extreme Learning Machine (OS-ELM). OS-ELM is the online variant of basic ELM for handling online applications [Liang et al. 2005]. Extreme Learning Machine (ELM) is a single hidden layer feed forward neural network (SLFN). It is a faster learning algorithm which analytically computes the output weights using randomly chosen input weights and biases unlike the traditional gradient based learning which involves many iterations in parameter tuning [Zhu et al. 2006].

The basic ELM is modeled from the given training set with N arbitrary samples containing n attributes and m classes such that (x_i, t_i) where input

vector $x_i = (x_{i1}...x_{in})^T \in R^n$ and the expected output vectors $t_i = (t_{i1}...t_{im})^T \in R^m$. The SLFN with $\tilde{N}$ hidden neurons and $g(x)$ activation function can be mathematically formulated as

$$f_N(x_j) = \sum_{i=1}^{N} \beta_i g(w_i.x_j + b_i) = o_j, j = 1...N \tag{1}$$

where w_i is the n-dimensional weight vector connecting i^{th} hidden neuron with input neurons, β_i is the n-dimensional weight vector connecting i^{th} hidden neuron with output neurons and b_i is the n-dimensional threshold of i^{th} hidden neuron. $w_i.x_j$ is the inner product of w_i and x_j. If N samples are approximated with zero error, then there exist β_i, w_i and b_i such that

$$f_N(x_j) = \sum_{i=1}^{N} \beta_i g(w_i.x_j + b_i) = t_j, j = 1...N \tag{2}$$

The equation 2 can be rewritten as

$$H\beta = T \tag{3}$$

where H is the hidden layer output matrix as in equation 4, β is the output weight matrix as in equation 5 and T is target output matrix as in equation 6.

$$H = \begin{bmatrix} g(w_1.x_1 + b_1) & \dots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \dots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}_{NX\tilde{N}} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix}_{\tilde{N}Xm} \tag{5}$$

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{NXm} \tag{6}$$

The training phase is performed in online manner by streaming the data sequentially delivered from IoT devices. OS-ELM provides fast learning model which can adapt to new data from IoT devices quickly along with a good generalization power.The intrusion detection system at fog nodes uses OS-ELM to identify the attacks in incoming traffic from IoT clusters. The OS-ELM algorithm classifies the incoming packet as normal or attack based on the training

data. The OS-ELM is trained using the training data along with the initial parameters containing the number of neurons at each layer, random input weights, biases and the activation function. The basic ELM is modified in matrix H of equation 3 and $rank(H) = \tilde{N}$ is considered as the rank of hidden neurons. The pseudo inverse of H is derived as

$$H^{\dagger} = (H^T H)^{-1} H^T \tag{7}$$

using the fact $H^{\dagger}H = I_{\tilde{N}}$.

The estimation is calculated as

$$\hat{\beta} = (H^T H)^{-1} H^T T \tag{8}$$

which is the least-square solution to $H\beta = T$. The sequential implementation of the least-square of equation 8 is referred as recursive least square algorithm which provides the solution of OS-ELM.

The learning in OS-ELM involves two phases: initialization and sequential learning . The initialization phase is similar to training in basic ELM but with small amount of data. For the initial training data $N_0 = (x_i, t_i)_{i=1}^{N_0}$, weights w_i and bias b_i are randomly assigned. The initial hidden layer matrix is computed as

$$H_0 = \left[h_1 \dots h_N\right]^T \tag{9}$$

Using the hidden layer matrix values, the initial output weights are calculated as

$$\beta_0 = (H_0^T H_0)^{-1} H_0^T T_0 \tag{10}$$

The initial training data are replaced with chunk by chunk online data $N_1 = (x_i, t_i)_{i=N_0+1}^{N_0+N_1}$ for continuous online training. The latest hidden layer output matrix is computed in the sequential learning phase as

$$H_{k+1} = \left[h_1 \dots h_N\right]^T \tag{11}$$

The recursive least square algorithm is used to compute the output weights for new training data as

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}} \tag{12}$$

where $M_k = (H_k^T H_k)^{-1}$. The generalized output weights are computed as

$$\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (t_i^T - h_{k+1}^T \beta_k) \tag{13}$$

Algorithm 1 gives the steps involved in the intrusion detection at the fog nodes using OS-ELM algorithm.

Algorithm 1 Cyber-attacks detection at fog node.

Input:

Activation function $g(x)$

Number of hidden layer neurons $\tilde{N}_f$

Chunk size

Initialization:

For the initial training data $N_{f0} = (x_{fi}, t_{fi})_{fi=1}^{N_{f0}}$

randomly assign weights w_{fi} and bias b_{fi}

Calculate the initial hidden layer matrix:

$$H_{f0} = \left[h_{f1} \dots h_{fN}\right]^T$$

Calculate the initial output weights:

$$\beta_{f0} = (H_0^T H_{f0})^{-1} H_{f0}^T T_{f0}$$

Sequential Learning:

For next chunk of data $N_{f1} = (x_{fi}, t_{fi})_{i=N_{f0}+1}^{N_{f0}+N_{f1}}$

Calculate the latest hidden layer matrix

$$H_{fk+1} = \left[h_{f1} \dots h_{fN}\right]^T$$

Calculate the latest output weight based on recursive least square algorithm

$$\beta_{fk+1} = \beta_{fk} + M_{fk+1} h_{fk+1} (t_i^T - h_{fk+1}^T \beta_{fk})$$

Detection:

Based on the learning, the attacks are identified and classified from the incoming traffic data

5 Forecasting Module

The forecasting method for IoT should consider the uncertainty prevailing in the critical infrastructure IoT. A Gaussian process is a collection of random variables and any finite number of these random variables has a joint Gaussian distribution [Rasmussen et al. 2006]. It works well in high uncertainty. Gaussian process regression identifies the relationship between the input and output and also provides predictions with associated uncertainty measure based on the Bayesian inference [Shumway et al. 2000]. The fog nodes use Gaussian process regression to forecast the cyber-attack locally for the IoT cluster under its control. It is performed by predicting the IoT traffic characteristics and they are represented as a latent function f

$$y_i = f(x_i) + \varepsilon_i \tag{14}$$

where x_i are the traffic parameters of the IoT application and $f(x_i)$ is the learning function to transform input x_i to the target y and ϵ_i is the Gaussian noise. The learning function $f(x_i)$ is Gaussian process which is fully specified by the mean and covariance function. To make the prediction for the new traffic x_* the

joint distribution of the observed target of the predicted function is

$$\begin{bmatrix} y \\ \bar{\mathrm{f}}_* \end{bmatrix} \sim N\Big(0, \begin{bmatrix} k(X,X)+\sigma_n^2 I & k(X,x_*) \\ k(x_*,X) & k(x_*,x_*) \end{bmatrix}\Big) \tag{15}$$

Based on the conditional probability,

$$p(y|X,k) = N(0, K+\sigma_n^2 I) \tag{16}$$

the joint Gaussian distribution gives predicted mean $\bar{\mathrm{f}}_*$ and variance $V[x_*]$ of IoT traffic as in equations 17 and 18

$$\bar{\mathrm{f}}_* = \mathrm{k}_*^T (k+\sigma_n^2 I)^{-1} y \tag{17}$$

$$V[x_*] = k(x_*,x_*) - \mathrm{k}_*^T (k+\sigma_n^2 I)^{-1} \mathrm{k}_* \tag{18}$$

where $k_* = k(X,x_*)$ and $K = K(X,X)$.

The choice of covariance function and its corresponding parameter are important to accurately model the cyber-attack forecasting using the Gaussian process regression. The parameters of covariance function are called hyperparameters. To model the given problem efficiently, the covariance function should reflect the traffic characteristics in the IoT application. In the proposed system to model the covariance function, three basic traffic characteristics of IoT are considered: fluctuation in the traffic, dependency within the traffic and periodicity of the traffic . A composite covariance function i.e. combination of covariance functions is used. The stationary covariance functions are considered for modeling the IoT traffic characteristics. Based on the definition, stationary process in the input domain T is a function of $r = t - t^{'}$ where $t \in T$ and $t^{'} \in T$. Therefore, the covariance function is denoted by $k(r;\theta)$. IoT traffic has fluctuations, due to large number of heterogeneous devices with or without synchronization. To model the fluctuation of the IoT traffic, the rational quadratic covariance function K_{RQ}, is used as given in equation 19.

$$Fluctuation : K_{RQ}(r_1;l_1,\alpha) = \sigma_1^2 \left(1 + \frac{r_1^2}{2\alpha l_1^2}\right)^{-\alpha} \tag{19}$$

where l_1 is the length scale parameter, α is the shape parameter and σ_1 is the variance.

The IoT application also exhibits both long term and short term dependencies in the network traffic. As a result, sum of two isotropic squared exponential covariance functions K_{SE} are used as given in equation 20.

$$\begin{aligned} Dependency : K_{SE}(r_2;l_2) + K_{SE}(r_3;l_3) \\ = \sigma_2^2 \left(-\frac{r_2^2}{2l_2^2}\right) + \sigma_3^2 \left(-\frac{r_3^2}{2l_3^2}\right) \end{aligned} \tag{20}$$

where l_2 and l_3 are the length scale parameters and σ_2 and σ_3 are the variances. To model the periodicity and the cyclic behavior of IoT traffic, product of periodic covariance function $K_{Periodic}$ and isotropic squared exponential covariance K_{SE} are used as shown in equation 21.

$$\begin{aligned} Periodicity : K_{Peroidic}(r_4; l_4, p) * K_{SE}(r_5; l_5) \\ = \sigma_4^2(-\frac{2\sin^2(\frac{nr_4}{p})}{l_4^2}) * \sigma_5^2(-\frac{r_5^2}{2l_5^2}) \end{aligned} \tag{21}$$

where l_4 and l_5 are the length scale parameters, p is periodic parameter, σ_4 and σ_5 are the corresponding variances.

Thus, the composite covariance function used for the proposed Gaussian process regression modeling at the fog node of the IoT application traffic is given in equation 22 which is the combination of equations from 19 to 21.

$$\begin{aligned} K_{FOG} = Fluctuation + Dependency + Periodicity \\ K_{FOG} = K_{RQ}(r_1; l_1, \alpha) + (K_{SE}(r_2; l_2) + K_{SE}(r_3; l_3)) \\ +(K_{periodic}(r_4; l_4) * K_{SE}(r_5; l_5)) \end{aligned} \tag{22}$$

The accuracy of the proposed Gaussian process regression model depends on the values of the parameters involved in the above covariance function. These set of values of the covariance function, that influences the accuracy of the model, are called hyperparameters θ. By maximizing the log marginal likelihood function give in equation 23, the optimal parameters for θ are estimated.

$$\begin{aligned} \log(p(Y|s,\theta)) = \frac{1}{2}Y^T(K + \sigma_n^2 I)Y - \frac{1}{2}\log|(K + \sigma_n^2 I)| \\ -\frac{n}{2}\log(2n) \end{aligned} \tag{23}$$

The local forecasting of the cyber-attack is performed by comparing the predicted traffic values with the predefined threshold. The threshold values are selected under normal conditions of IoT application. The local forecasting is used to alert the IoT application prior to the attack and it can initiate a response mechanism to prevent the attack. Algorithm 2 gives the steps involved in the forecasting of cyber-attacks in fog node.

6 Response Module

The critical infrastructure IoT needs an efficient automated response mechanism to act fast against attack incidents and to reduce the attack impacts. The existing traditional response selection techniques are not suitable for critical infrastructure which are equipped with heterogeneous technologies and increased

Algorithm 2 Cyber-attack forecasting.

Given:

Covariance function: K_{FOG}
Hyperparameter θ
Training data: $D = \{x_i, y_i\}_{i=1}^{n}$

At each Fog nodes:

1. Apply the identified covariance function K_{FOG} and Hyperparameter θ for training with IoT traffic data.
2. Perform local Gaussian process regression to predict the posterior mean and variance using the prior values from the training data
   $\bar{\mathrm{f}}_* = \mathrm{k}_*^T(k + \sigma_n^2 I)^{-1} y$
   $V[x_*] = k(x_*, x_*) - \mathrm{k}_*^T(k + \sigma_n^2 I)^{-1}\mathrm{k}_*$
3. Compare the predicted value with defined thresholds of the IoT traffic parameters to identify the possibility of attack.
4. If the posterior values violate the defined traffic thresholds, then the fog node declares it as attack and sends alert message to response module.

number of resource constrained devices deployed in a large-scale providing data with uncertainty. Hence, it needs an intelligent distributed response selection mechanism to handle attack by using the imprecise data from the uncertain environment. Fuzzy logic is used to handle the imprecise information from the critical infrastructure IoT devices, during the attack for decision making about selecting the suitable response. When an attack or anomaly is detected by detection module or forecasting module, it generates alerts. On receiving the alerts, the response module, first identifies the set of candidate responses to handle the attack. The candidate responses are selected from the response pool by matching the generated alerts. From the candidate responses, fuzzy logic based decision module selects the appropriate response based on the response selection metrics using fuzzy logic.

6.1 Response Selection Metrics

The response selection metrics are used to identify the most appropriate response for the generated alert. The metrics have been defined based on the IoT device characteristics, attack severity, the cost induced by implementing the response and the success rate of the response. Hence, the intervention of selected response does not cause harmful negative impact in critical infrastructure. These metrics are obtained at the fog nodes which are closer to the end devices.

6.1.1 IoT device importance

This metric provides the importance of victim IoT device in terms of : the value of data generated by the device, the criticality of the function it performs and device failure effect scalability. The fog node maintains profiles of all the devices under its control and stores all the details required to calculate the device importance. The IoT device importance is calculated based on the general weighted average equation. The aforementioned IoT device characteristic x is weighted by the following criteria

$$w(x) = \begin{cases} 0.1, & \text{if } 0 \leq x \leq \mathrm{t}_1 \\ 0.5, & \text{if } t_1 \leq x \leq \mathrm{t}_2 \\ 1, & \text{if } t_2 \leq x \leq 1 \end{cases} \tag{24}$$

where t_1 and t_2 are the thresholds to decide the weights for the specified device characteristics. 0.1 gives low weight for specifying the minimum criticality value, 0.5 for moderate criticality value and 1 for high criticality. These thresholds are decided based on the IoT application in which the devices are involved. The function to calculate the device importance of the victim device d is given as

$$DI(d) = \frac{\sum(w(v) \times v)(w(f) \times f)(w(s) \times s))}{\sum w(v) \times w(f) \times w(s)} \tag{25}$$

where v is the data value generated from the device, f is the criticality of function that the device performs and s is the device failure effect scalability. The value of DI is computed on a scale from 0 to 1. The DI value 0 indicates that the device has low importance and the values nearer to 1 indicates that the device has higher importance.

6.1.2 Severity score

The severity score is used to measure the impact of the exploit. There are various standard sources such as Common Vulnerability Scoring System, MITRE, NIST,Secunia and so on to provide this measure for the existing attacks. In the proposed system, this score is calculated based on the Common Vulnerability Scoring System (CVSS). The CVSS contains three metric groups: Base metric, Temporal metric and Environmental metric. The base metric represents the fundamental characteristics of a vulnerability and they are constant with the time. Likewise, the temporal metric represents the characteristic that varies with time and the environmental metric represents the characteristics that are relevant to environment. All the three metrics are considered to calculate the severity score of the IoT application. A trust value α is added to the severity score because the severity of the attack will change based on the network and application.

Therefore, based on the application in which the device works, the trust value is assigned from 0 to 10, with 10 as the highest trust value. The severity score SS of attack x is calculated at the fog node as

$$SS(x) = \frac{vs(x) + \alpha(x)}{2} \tag{26}$$

where vs is the vulnerability score calculated from CVSS database and α is the trust value. The value of SS is computed on a range from 0 to 10. The SS value 0 indicate that the attack has low impact and the values nearer to 10 indicate that the attack has high impact on the IoT.

6.1.3 Response cost

This metric measures the cost for implementing the response for the identified attack. In IoT environment, most of the devices involved are resource constrained devices and so the response cost considered is the amount of resources additionally utilized for deploying the response. The major resources considered to measure the response cost in the IoT environment are amount of energy, RAM and CPU which are utilized for deploying the response. The response cost RC for response r is given as

$$RC(r) = \sum \left(\frac{E + M + P}{300} \right) \times 10 \tag{27}$$

where E is the percentage of energy , M is the percentage of memory and P is the percentage CPU required to deploy the response. The response cost RC is scaled from 0 to 10, where response with lower values of RC is more preferable in resource constrained IoT environment.

6.1.4 Success Rate

This metric provides adaptability to the response selection by adjusting the selection based on the dynamic environment of IoT. The response selection is made based on the success of previously selected response. Each response is associated with success counter maintained in the response pool. If the selected response successfully reduces the impact of the attack, then the success counter is incremented automatically. If the selected response does not block the attack then the success factor remains unchanged. The success rate SR of response r is calculated as

$$SR(r) = \left(\frac{s}{n} \right) \times 100 \tag{28}$$

where s is the value in the success counter which gives the number of times the selected response is successful and n is the total number of times the response is selected.

6.2 Fuzzy Inference System

Unlike the classical logic system, fuzzy logic does not require complete data for understanding of system. It provides a conceptual framework for solving the problems in an environment of uncertainty and imprecision. This ability of fuzzy logic is used by the proposed response selection model to identify the accurate response in uncertain IoT environment by calculating the response score. The output of the fuzzy inference system is the response score of the given candidate response. The response model selects the response with high response score to handle the attack scenario.

In general, the fuzzy inference system consists of four steps. First, the fuzzification step converts the crisp inputs into fuzzy set using the fuzzification function. In the proposed system, triangular membership functions are used to represent the inputs. The triangular membership functions are used for their simplicity in representation by specifying only the lower and upper bounds and yield optimal solution for imprecise information in IoT environment. The fuzzy inference engine in the proposed system is modeled with four input parameters: device importance DI, severity score SS, response cost RC and success rate SR with three linguistics states: low, medium and high to describe the input parameters as given in Table 1. Three linguistic states varying from low, medium and high have been chosen to describe the characteristics of critical infrastructure namely criticality of device, attacks and their impacts. The trapezoidal membership function is used to represent the output Response Score (RS) in the proposed system. The randomness and uncertainty of the input in IoT environment produce output with high randomness and it can be efficiently modeled using the trapezoidal membership function with two linguistics states: Not selected and Selected. Their ranges are given in Table 2.

Table 1: Input variables and the ranges of membership function

Input	Low	Medium	High
Device Importance (DI)	0 − 0.3	0.25− 0.7	0.65−1.0
Severity Score (SS)	0 − 3.5	3.0 −7.5	7.0 − 10.0
Response Cost (RC)	0 − 3.5	3.0 − 7.5	7.0 − 10.0
Success Rate (SR)	0 − 30	25− 70	65 − 100

Table 2: Output variable and the ranges of membership function

Output	Not-Selected	Selected
Response Score (RS)	0 − 5.5	4.5− 10

Next, the fuzzy rules are applied on these values to generate output values for every rule in the rule base. The fuzzy rules contain the condition to map the

input values to output values using IF-THEN rules. The number of rules in the rule base depends on the number of input and number of fuzzy set of each input. In the proposed model, there are four inputs with 3 members in each fuzzy set. Hence, the total number of rules in the rule base is

$$\text{Number of Rules} = \Pi_{i=1}^{n} f_i = 3 \times 3 \times 3 \times 3 = 81 \tag{29}$$

where f is the number of input and n is the number of fuzzy set of each input. The sample set of fuzzy rules defined for the proposed response selection model is given Table 3.

Table 3: Sample Fuzzy rule set of the proposed model

DI	SS	RC	SR	Output
High	High	Medium	High	Selected
Low	Low	High	Low	Not Selected
High	Low	High	Medium	Selected
Medium	Medium	High	Low	Not Selected
Low	High	Low	Medium	Selected
Medium	Medium	Medium	Medium	Selected

The proposed system uses Mamdani inference mechanism which uses simple min-max method to generate the fuzzified output. These output values are averaged to generate a single fuzzy output. Finally, the defuzzification is applied on the averaged output value to generate the final crisp output. The proposed system uses centroid or center of gravity method for defuzzification. The final response score is computed using the centroid defuzzification method based on the equation 30.

$$\text{Response Score (RS)} = \frac{\int x_i \mu(x_i)}{\int \mu(x_i)} \tag{30}$$

where $\mu(x_i)$ is the membership function and x_i is the fuzzy value assigned to that membership function.

7 Experiments and Results

In this section, the proposed system is evaluated and the performance is measured for each module i.e. detection, prediction and response modules. Finally, the effectiveness of fog computing is measured by implementing the autonomic security system in cloud server.

7.1 Simulation setup

As a Proof-of-concept, the fog based IoT architecture is implemented using Network Simulator 2.34 (NS 2.34) simulation tool, Azure cloud service along with

Table 4: Criticality level setup for sensors

Sensor Type	Low	Medium	High
Pressure Sensor	Hospital Bed	Blood Analyzer	Anesthesia Delivery Machine
Temperature Sensor	Room Monitoring	Digital Thermometer	Kindney Dialysis Machine
Bio Sensors	Drug Abuse Testing	Blood Analyzer	Cancer Diagnosis
Position Sensor	Hospital Bed	Surgical equipment	Heart Pacemaker
Flow Sensor	Room Monitoring	Oxygen concentrator	Respiratory monitoring

Table 5: Simulated network parameters of wireless nodes

Parameter	Value
Routing Protocol	DSR
Size of the Packets	512 Bytes
Transmission Pattern	Constant Bit Rate
Bandwidth	10 Mbps

Matlab, AWK and python scripts. In this experiment, 150 wireless sensor nodes of healthcare system are simulated using NS 2.34. Five types of health care application sensor nodes are simulated by considering the criticality of the devices applied at various scenarios of healthcare system as shown in Table 4. The sensor nodes are simulated on three environments each containing 50 nodes such that 10 sensors of each type are provided in Table 4. Three sink nodes are simulated and each connects 50 wireless nodes by forming an IoT cluster. The simulation parameters of the network are given in Table 5. The sink nodes are treated as fog nodes and they are emulated on three computers with the configuration of Quad core, AMD Opteron 2354, 2.20GHz, 32GB RAM, $2 \times 500GB$ HDD. These computers are connected to the Azure cloud service with the computing resource $4 \times Dual$ Core AMD Opteron 2218, 2.6GHz, 32GB RAM, $6 \times 146GB$ HDD. Scripts are coded to generate the required values by the sensors to simulate the healthcare infrastructure. Figure 3 shows the simulation setup to evaluate the proposed technique.

7.2 Performance of Detection Module

The detection module is evaluated at the fog nodes and the experimental results are viewed in terms of accuracy, response time and network load. NSL-KDD benchmark dataset is used for intrusion detection to evaluate the proposed system. This dataset contains 24 types of attacks and they are grouped under four categories: Denial of Service (DoS), Remote to User (R2L), User to Root (U2R) and probing attack [Tavallaee et al. 2009]. The DoS attacks make the resource unavailable for the legitimate users. The attacks under DoS category are Land, Neptune, Smurf, Ping to death, Tear drop and Back. In U2R, the

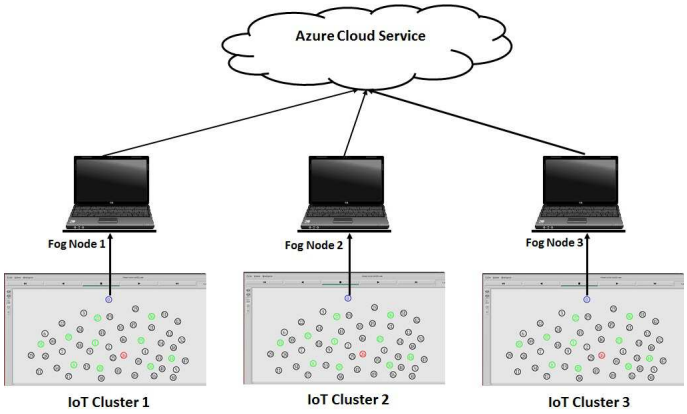

Figure 3: Simulation setup

attackers aim to gain the root access of the target system. The U2R includes Buffer overflow, Rootkit, Load module and Pearl attacks. In R2L, the goal of the attacker is to gain access to the target system or network by gaining the legitimate user's remote access. The attacks included in it are Warezclient, Guess password, Warezmaster, Imap, Ftp write, Multihop, Phf, and Spy. In Probing category, the attacker aims to gather information about the target machine or network and the attacks included in this category are Satan, Ipsweep, Portsweep and Nmap.

From the experimental results, it is observed that to efficiently model the OS-ELM, higher number of hidden layer neurons and smaller chunk size are needed. Therefore, 35 hidden layer neurons and 1000 chunk size with sigmoid as activation function are used to model the OS-ELM in the proposed method to yield best result. The detection accuracy of the OS-ELM is measured based on accuracy, detection rate and false alarm rate as given in equations 31 to 33 by using the following confusion matrix containing False Positive (FP),False Negative (FN), True Positive (TP) and True Negative (TN) values. Table 6 shows the accuracy measurement of binary classification and Table 7 shows the accuracy measurement of multi-class classification.

		Predicted	
		Attack	Normal
Actual	Normal	TN	FP
	Attack	FN	TP

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{31}$$

$$DetectionRate = \frac{TP}{TP + FN} \tag{32}$$

$$FalseAlarmRate = \frac{FP}{TN + FP} \tag{33}$$

Table 6: Accuracy measurement for binary classification

Overall Accuracy	**Detection Rate**	**False Alarm Rate**	**TPR**	**FPR**
97.36	96.92	1.53	97.72	0.37

Table 7: Accuracy measurement for Multi-class classification

Detection Rate	Normal		Probe		DoS		U2R		R2L		Overall Accuracy
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	
96.08	98.63	4.74	84.2	0.81	96.61	2.32	23.81	0.52	71.87	0.20	96.54

The detection module achieves 97.36% of accuracy with reduced false alarm rate of 0.37% in approximately 13 seconds of training time.

7.3 Performance of Forecasting Module

In this section, the forecasting module is evaluated by simulating new kind of flooding attack that does not have pattern in NSL-KDD dataset. To validate the prediction module, flooding attack is generated. The test case considered in the experiment uses the routing based flooding attack in Dynamic Source Routing (DSR) protocol. The route recovery mechanism is used by the attackers to generate flooding attack as shown in Figure 4.

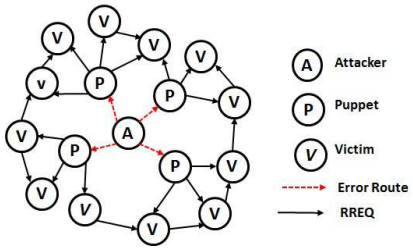

Figure 4: Flooding attack

The attacker node adds unavailable node to the route and sends Route error (RRER) message in the network. When the intermediate node cannot find the

node in the specified route, it generates Route request (RREQ). Likewise, the attacker sends messages to the multiple error routes and makes other nodes as puppets to flood the network with RREQ for unavailable nodes. This type of flooding attack is also called as puppet attack. The effect of puppet attack is more and difficult to detect compared to the conventional flooding attack because it uses the legitimate nodes to generate attack.

To generate this attack scenario, one node in each of the three simulation environment is made malicious to make 5,10,15 and 20 nodes as puppet to flood the network with 50 RREQ packets per second. The simulation time is 200 seconds. The attack can be identified by measuring the packet delivery rate, which is the ratio of successfully received packets by the destination to the packets generated by the source. The packet delivery rate is measured for 30 simulation runs and the average is used to plot the data in the graph.

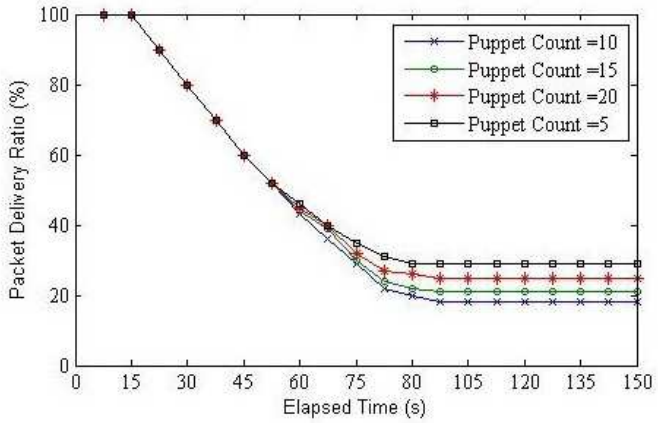

Figure 5: PDR for varying number of puppets

The performance of the network is measured by varying the number of puppets in the generated flooding attack as shown in Figure 5. The packet delivery rate (PDR) falls down with the increase in number of puppets. From the experiment, for 5 puppets, PDR drops to 30 percent and for 10 puppets, it falls to 19 percent. From the simulation results, it is found that for 15 puppets and 20 puppets, the PDR slightly increases because the attacking packets will not reach the puppet nodes, due to many RREQ flood from the puppets.

To evaluate the prediction module, 10 puppets for each IoT cluster is considered. The 10 puppets flooding attack reduces the PDR to 19%, which is equivalent to the effect of higher number of puppets. For this case, the simulation result of the proposed prediction algorithm at single fog node is shown in Figure 6. To predict the attack based on PDR, the training time of 40 seconds is

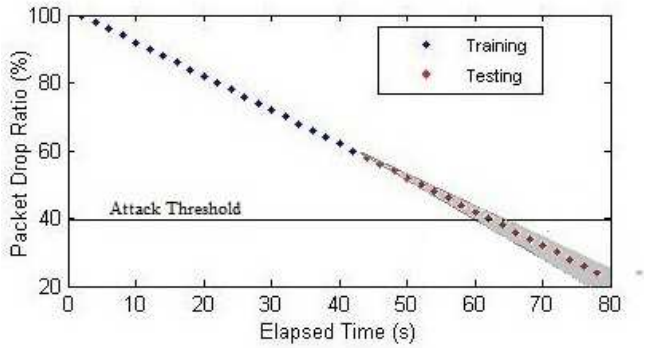

Figure 6: Prediction at fog node

considered because PDR is reduced to below 60%, during this time as shown in Figure 5. From the simulation results of Figure 5, it is inferred that PDR drops to below 40%, during the attack occurrence and therefore, the threshold of PDR is assigned as 40% to predict the attack. The training data and the test data are shown in different colors in Figure 6. The highlighted area in the Figure 6 shows the predicted result of the simulated training data by applying the proposed prediction algorithm. The highlighted area demonstrates the probabilistic prediction of the PDR during the occurrence of attack. The graph shows that the average of predicted range matches well with the test data.

7.4 Performance of Response Module

The response module is evaluated for the generated flooding attack in the forecasting module. The candidate set of responses for handling the generated flooding attack is given in the Table 8. The proposed response selection metric values are varied at all ranges to consider the different cases of IoT scenario and to fully evaluate the proposed response selection technique.

Table 8: Response action for flooding attack

Response	Action Detail
Packet Filtering	Attack packets are filtered out and dropped based on IP address.
Rate Limiting	Allow a router to control the transmission rate of specific flows.
Port Changing	The port is changed and informed only to legitimate users.
Network Disconnection	The attacker node is disconnected from the network.
Process Termination	Current process is terminated.

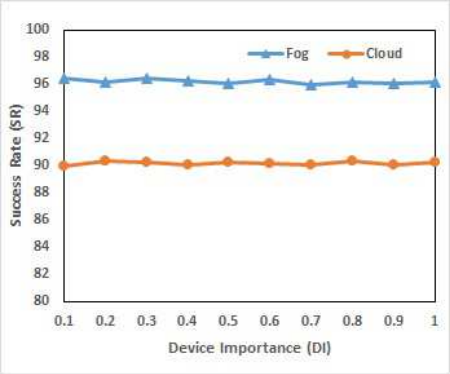

Figure 7: Success rate versus Device importance

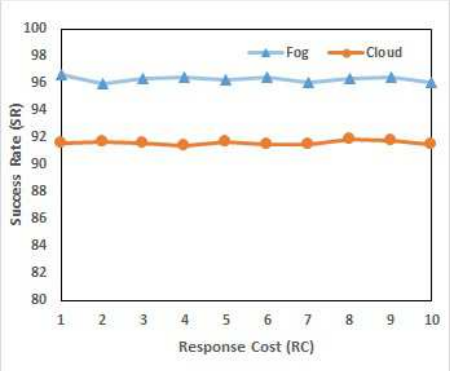

Figure 8: Success rate versus Response cost

To evaluate the efficiency of the response module, the proposed system in fog node is compared by implementing them in Azure cloud service as centralized system. The efficiency of the implementing proposed fuzzy inference for response is measured in terms of success rate for each defined metric. Figures 7-9 shows that the change in metric values (low or high or medium) i.e., change in the IoT environment does not show large variations in the success rate. The major inference from Figures 7-9 is that the success rate of the response is high, when the proposed fuzzy inference is performed at the fog compared to the cloud. The fog nodes, which are closer to the end, activate the response faster than the cloud. The delay between the detection and the response is high in cloud based implementation. Hence the severity of the attack would increase and complicates the attack scenario by making the selected response inefficient. In the fog based

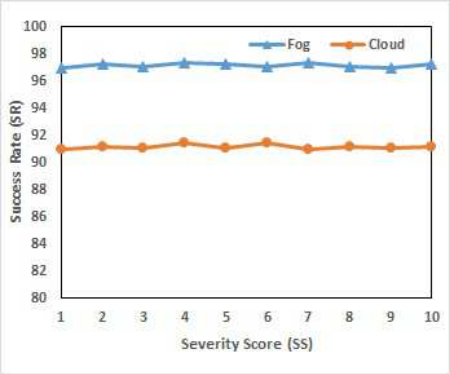

Figure 9: Success rate versus Severity score

implementation, the delay is less by making the success rate of the response higher than the cloud based implementation.

7.5 Performance Evaluation of Fog Computing

The performance of the proposed fog based autonomic security system is measured by comparing it with the cloud based implementation in terms of response time and network load.

7.5.1 Response Time

The latencies of the proposed fog based system and cloud based system are measured to show the impact of the self-protection mechanism in fog computing. The response time of the proposed system is compared with the cloud based implementation with varying network bandwidths as shown in Figure 10. The response time to identify and handle cyber-attack is nearly 25% less in the proposed fog based approach compared to the cloud based implementation because fog nodes are closer to the IoT end devices.

7.5.2 Network Load

As the number of devices increases in the IoT application, the amount of data generated also increases which in turn increases the network load. In the case of fog based approach, the load is shared by distributed fog nodes and hence, the network usage is considerably reduced compared to the centralized cloud based approach. As the amount of data increases, the network load is also reduced

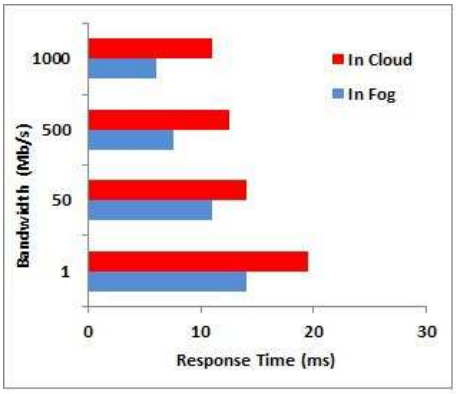

Figure 10: Response time comparison between fog based and cloud based implementation.

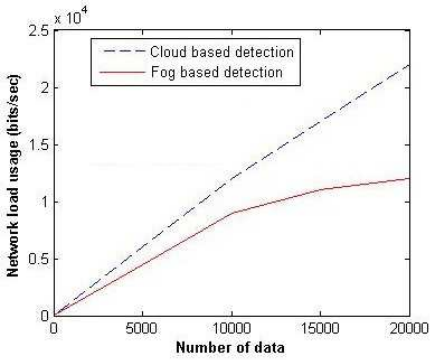

Figure 11: Network usage comparison between fog based and cloud based implementation.

significantly as shown in Figure 11 for the fog based system compared to the cloud based system.

At the outset, the proposed cognitive fog based self-protection mechanism efficiently predict, detect and react to cyber-attack with reduced response time and network load

8 Conclusion

In this paper, an autonomic security system with self-protect mechanism incorporated on fog nodes has been proposed for critical infrastructure involving IoT

devices. The proposed system predicts, detects and reacts to cyber-attack intelligently at faster rate with less human intervention which is highly required for critical infrastructures. The experimental results emphasize that the proposed system predicts and detects the cyber-attack with high accuracy and also learns the new attacks from the voluminous IoT traffic. Moreover, the fog nodes increase the success rate of selected response for the attack scenario and overall latency of the self-protect mechanism is reduced compared to the cloud based implementation.

The proposed work provides an initial step for designing an autonomous security system based on fog computing for critical infrastructure. The future work is to enhance the proposed system by implementing and testing in the real-time critical infrastructure.

References

[Atzori et al. 2010] Atzori, Luigi, Antonio Iera, and Giacomo Morabito., The internet of things: A survey, Computer networks 54(15) (2010) 2787-2805.

[Bonomi et al. 2014] Bonomi, F., Milito, R., Natarajan, P. and Zhu, J., Fog computing: A platform for internet of things and analytics, In Big Data and Internet of Things: A Roadmap for Smart Environments,Springer International Publishing (2014) 169-186.

[Chen et al. 2014] Chen, Qian, Sherif Abdelwahed, and Abdelkarim Erradi., A model-based validated autonomic approach to self-protect computing systems, IEEE Internet of Things Journal 1(5) (2014) 446-460.

[Claudelet al. 2006] Claudel, Benoit, Noel De Palma, Renaud Lachaize, and Daniel Hagimont., Vishal Misra, and Dan Rubenstein., Self-protection for distributed component-based applications, SSS 6 (2006) 184-198.

[Dean et al. 2012] Dean, Daniel Joseph, Hiep Nguyen, and Xiaohui Gu., Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems, In Proceedings of the 9th international conference on Autonomic computing, ACM (2012) 191-200.

[Gelenbe et al. 2007] Gelenbe, Erol, and George Loukas., A self-aware approach to denial of service defence, Computer Networks 51(5) (2007) 1299-1314.

[Ghani et al. 2014] Ghani, H., Khelil, A., Suri, N., Csertan, G., Gonczy, L., Urbanics, G. and Clarke, J., Assessing the security of internet-connected critical infrastructures. Security and Communication Networks, 7(12), (2014) 2713-2725.

[Hariri et al. 2005] Hariri, Salim, Guangzhi Qu, Ramkishore Modukuri, Huoping Chen, and Mazin Yousif., Quality-of-protection (QoP)-an online monitoring and self-protection mechanism, IEEE Journal on selected areas in communications 23(10) (2005) 1983-1993.

[Kephart et al. 2003] Kephart, Jeffrey O., and David M. Chess., The vision of autonomic computing, Computer 36(1) (2003) 41-50.

[Keromytis et al. 2004] Keromytis, Angelos D., Vishal Misra, and Dan Rubenstein., SOS: An architecture for mitigating DDoS attacks, In International Conference on Information Security, IEEE Journal on selected areas in communications 22(1) (2004) 176-188.

[Liang et al. 2005] Huang, G.B., Liang, N.Y., Rong, H.J., Saratchandran, P. and Sundararajan, N., On-Line Sequential Extreme Learning Machine. Computational Intelligence,(2005)232-237.

[Locasto et al. 2005] Locasto, Michael E., Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo., Towards collaborative security and p2p intrusion detection, In Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC (2005) 333-339.

[Marchetti et al. 2009] Marchetti, Mirco, Michele Messori, and Michele Colajanni., Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale, In International Conference on Information Security Springer Berlin Heidelberg (2009) 475-490.

[Ng et al. 2006] Ng, H. S., M. L. Sim, and C. M. Tan., Security issues of wireless sensor networks in healthcare applications, BT Technology Journal 24(2) (2006) 138-144.

[Qu et al. 2010] Qu, Guangzhi, Osamah A. Rawashdeh, and Dunren Che.,Self-Protection against Attacks in an Autonomic Computing Environment. IJ Comput. Appl. 17(4),(2010) 250-256.

[Rasmussen et al. 2006] Rasmussen, Carl Edward, and Christopher KI Williams., Gaussian processes for machine learning, Cambridge: MIT press (2006).

[Rinaldi et al. 2001] Rinaldi, Steven M., James P. Peerenboom, and Terrence K. Kelly., Identifying, understanding, and analyzing critical infrastructure interdependencies, IEEE Control Systems 21(6) (2001) 11–25.

[Sandor et al. 2017] Sandor, Hunor, Piroska Haller, Bela Genge, and Zoltan Katai. Optimally scheduled interventions in the presence of vulnerabilities for modern cyber-physical systems. in IEEE 15th International Conference on Industrial Informatics (INDIN), (2017) 115-120.

[Shumway et al. 2000] Shumway, Robert H., and David S. Stoffer.,Time series analysis and its applications, Studies In Informatics And Control 9(4) (2000) 375-376.

[Tavallaee et al. 2009] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A., A detailed analysis of the KDD CUP 99 data set. In Computational Intelligence for Security and Defense Applications,CISDA 2009. IEEE Symposium (2009) 1-6.

[Wailly et al. 2012] Wailly, AurÃl'lien, Marc Lacoste, and Herve Debar., Vespa: Multi-layered self-protection for cloud resources, In Proceedings of the 9th international conference on Autonomic computing , ACM (2012) 155-160.

[Wang et al. 2008] Wang, Y., Li, X.Y. and Zhang, Q., Efficient algorithms for p-self-protection problem in static wireless sensor networks. IEEE transactions on parallel and distributed systems, 19(10) (2008), 1426-1438.

[Wen et al. 2017] Wen, Zhenyu, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos. Fog orchestration for internet of things services. IEEE Internet Computing 21(2),(2017), 16-24.

[Whitmore et al. 2015] Whitmore, Andrew, Anurag Agarwal, and Li Da Xu., The Internet of Things A survey of topics and trends, Information Systems Frontiers 17(2) (2015) 261-274.

[Zhu et al. 2006] Huang, G.B., Zhu, Q.Y. and Siew, C.K., Extreme learning machine: theory and applications, Neurocomputing, 70(1) (2006) 489-501.