Deep Reinforcement Learning for Building Honeypots against Runtime DOS Attack

Selvakumar Veluchamy (vselvakumar0720@gmail.com)
Chennai Institute of Technology
RubaSoundar Kathavarayan
P S R Engineering College

Research Article

Keywords: Honeypot, DoS, DL techniques, IDS and event tracking

Posted Date: March 24th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-207770/v1

Version of Record: A version of this preprint was published at International Journal of Intelligent Systems on October 20th, 2021. See the published version at https://doi.org/10.1002/int.22708.

Deep Reinforcement Learning for Building Honeypots against Runtime DOS Attack

Mr. Selvakumar Veluchamy[1*], Mr. RubaSoundar Kathavarayan[2]

[1]Assistant Professor, Computer Science and Engineering Department, Chennai Institute of Technology, Chennai, Tamil Nadu, India*

[*]Email: vselvakumar0720@gmail.com

[2]Professor, Computer Science and Engineering Department, P.S.R. Engineering College, Sivakasi, Tamil Nadu, India

Abstract

Honeypot is a network environment used to protect the legitimate network resources from attacks. Honeypot creates an environment that impresses attackers to inject their activities to steal resources. This is a way to detect the attacks by doing attack detection procedures. In this work, Denial of Service (DoS) attacks are effectively detected by proposed honeypot system. Machine Learning (ML) and Deep Learning (DL) methods evolve in many areas to build intelligent decision making systems. This work uses DL approaches and secures event validation procedures for finding predicting DoS attacks. The proposed system called Deep Adaptive Reinforcement Learning for Honeypots (DARLH) is implemented to monitor internal and external DoS attacks. In the honeypot environment, the proposed DARLH system implements DARL based IDS (Intrusion Detection System) agents and Deep Recurrent Neural Network (DRNN) based IDS agents for monitoring multiple runtime DoS attacks. These techniques support for dynamic IDS against DoS attack. In addition, the DARLH creates protected poison distribution and server side supervision system for keeping the monitoring events legitimate. This work is implemented and performance is evaluated. The results are compared with existing

systems like GNBH, BCH and RNSG. In this comparison, the proposed system provides 5% to 10% better results than other systems.

Keywords: *Honeypot, DoS, DL techniques, IDS and event tracking*

1. Introduction

Honeypot is a secure machine or a collection of machines to appeal invaders into it. Honeypot systems are the part of any enterprise networks but deployed as independent blocks. These systems are having a portion of data in any machine to attract the attackers to start their hacking activities. Honeypot machines monitor the activities injected by an attacker or any intruder continuously. Honeypot systems maintain sufficient number of hardware and software resources to execute the attack monitoring activities. Also, the systems contain the huge active and passive attacker datasets [1-3].

Notably, honeypot creates a duplicate and isolated network to validate the external traffics. This helps to improve the security of main network. Honeypot helps to implement well defined Network Intrusion Detection System (NIDS). This honeypot based NIDS implementation can be deployed centrally or distributed manner. Many related research works put their effort in honeypot based IDS to detect various attacks. Among the attacks, DoS is a serious active network attack to dump the network functions. The recent researches developed IDS against DoS using rule based classification techniques, Support Vector Machines (SVM) and other Machine Learning (ML) techniques.

The techniques with sufficient level of trained dataset provide additional intelligence to honeypot systems. This improves attack detection rate. Though, the existing techniques provide good support to honeypot, they lack in complex activity validations.

At the vulnerable point, an intelligent attacker can execute several decent cryptanalysis techniques and compromised activities. The changes inside the network may not be monitored by this ML based honeypots.

This is taken as research problem which motivates the proposed system to develop Deep RL engines to build bidirectional honeypots. This RL enabled bidirectional honeypots monitor both internal and external attackers. In this regard, these honeypot systems are implemented in Virtual Local Area Network (VLAN) which seems to be a directly connected physical LAN. This provides all types of data that looks like legitimate to both internal and external users. By this trap, the proposed honeypot protect the strong security layer between the data and the attackers.

The proposed DARLH system contributes protected honey pot implementation against DoS attacks. This DARLH system monitors both internal and external attacks using multi-level DL techniques. It uses protected poison distribution for event management system. In addition, it extends the contribution to implement, event distribution and supervision practices and DARLH models (Level-1 and Level-2) creation. In Level-1, the proposed DARLH system creates secure deep RL networks for monitoring both events and clients. In the next level, the system creates DRNN and DARL IDS agent integration modules for effective runtime attack detections. For DoS attack scenario, this work used KDD dataset (Knowledge Data Discovery) patterns [4][5].

This article is organized as follow from related works. The rest of the sections cover proposed system architecture, technical implementations, event distribution models and DoS monitoring models. The section 4 elaborates implementation details and the results produced for proposed system' evaluations.

2. Related Works

Honepot implementation and security provisioning in any enterprise network play major role in any enterprise network systems. Even though many attack detection procedures emerge around the world, the DL based honeypot implementation is really needful to construct dynamic Network IDS (NIDS). Attacks inject various attacks can be injected to harm network performance. The attackers are the users those who steal or harm the resources. The attacks such as DoS, Spoofing, Wormhole, Black hole, identity theft are known as serious attacks in network world. Among the attacks, DoS is a type of active attack which degrades overall network performance.

Many research works contributed on DoS detection procedures and honeypot security schemes. They delivered various designs and implementation setups on detecting DoS in different network natures. Bingham et al. [6] proposed automated honeypot system for providing security features. This work concentrated on automated network administration strategies in honeypot and honey net environment. The system consisted of multiple LANs and resources with several attackers. The attacker events and real-time activities of this honeypot unit were monitored by systematic procedures. This work gave base level automated functions in honeypot scenario without making any ML or DL systems.

In the same manner, Negi et al. [7] discussed IDS and Intrusion Prevention System (IPS) implementations for cloud based honeypot systems. This system mainly focused on malicious activities and malicious users in cloud honeypot environment. Besides this research work implemented new type of honeypot system that diverts malicious attacks thorough different isolated path. However this work was not more active against runtime attacks and it had lack of deep attack analysis techniques.

Banday et al. [8] proposed captcha and timestamp based honeypot security techniques. Tis work completely talked about text based security provisions for protecting honeypots. This work uses data mining and data processing techniques to evaluate temporal data features. On the way, Ormazabal et al. [9] implemented finger print evaluation schemes for detecting DoS attacks in network. This finger print detection techniques were evaluated using image processing procedures for identifying malicious activities. Both techniques worked perfectly in attack detection but with conventional properties.

Shrivastava et al. [10] and Shi et al. [11] concentrated on DoS attacks honeypot systems. The first work used game theory and Naïve Bayes techniques for detecting DoS attacks at runtime. This system used ML logics in finding DoS using training sets. This work used basic ML techniques not DL approaches. The second work proposed block chain based DoS attack detection in honeypot systems. This techniques was executed for distributed network environments. This gave effective performance in terms of security. Yet, the technique is not suitable for more active or centralized systems.

Kaur et al. [12] proposed DRNN based signature generation procedures for finding attacks in honeypots. As DRNN is an effective DNN, it creates more complex signatures for all events raised in honeypots. At the same time, this wok implemented DRNN for signature generation procedures only. This DRNN could be extended for monitoring the events and attacks. Several techniques provided well defined ML and DL based attack detection procedures in honeypot systems. Yet they assumed that the internal events are legitimate. This leads to internal attacks in honeypots. Particularly, internal DoS attacks are complicated to be identified and isolated.

Khan et al. [13] proposed multi-level botnet detection techniques using ML classifiers such as Naïve Bayes, Support Vector Machine and Self Organizing Maps. This work evaluated

network domains, web events and the conversations of network elements. Rao et al. [14] and Sharma et al. [15] discusses various security techniques and DoS types. These works delivered vast network principles, attack problems, cybercrimes in cloud based systems. In addition, both works provided useful information about attack nature and motives at various vulnerable points of the network. The information gathered from these works were really useful for designing proposed DARLH system.

Huang et al. [16] designed the proposed system using RL based IDS. This system proposed adaptive honeypot principles using RL and markov process structures. This gave more cost effective honeypot management system in terms of Quality of Service (QoS). However this work dealt with QoS factors not with security credentials. At the same time, this system used ML approached not DL methodologies. In relation with DoS attack, Kotey et al. [17] clarifies various attack schemes and procedures that affect network resources.

From the vast analysis of related research work, the proposed system finds the lack of DL adaptations in runtime DoS detection.

The proposed DARLH system solves the issues and complexities in DL based honeypot security system. The next section describes the proposed DARLH system details.

3. DARLH System

The Proposed ARLH system is designed and implemented to monitor both internal and external attackers. The attackers may inject DoS attack into any system deployed in honeypot environment. This environment has one or more monitoring resources (Servers and Clients) which are enabled with active IDS agent. The IDS agent running inside the monitoring resources scrutinize each and every network activities (traffics). The proposed DARLH contains following technical aspects for building more reactive honeypots.

- Secure Honeypot Traffic Distribution Model
- Server Distribution and Supervising Model
- Honeypot with DARLH Level-1 Model
- Honeypot with DARLH Level-2 Model

Secure honeypot traffic model helps to distribute the events from server to parallel clients. Clients monitor the events and send reports to server. Server supervising model is used to monitor client's activities. DARLH Level-1 and DARLH Level-2 models are implemented for doing DoS attack detection activities using DARL and DRNN IDS agents.

3.1 Secure Honeypot Traffic Distribution Model

A proposed honeypot traffic model is implemented for building traffic collection methodology. Honeypot environment contains database server, a database, communication devices (switches and routers) and monitoring nodes. The monitoring nodes are the collection of servers and client machines. Client monitoring machines run their IDS agent unit to monitor the activities injected from any side of network. The IDS agent has been equipped with the proposed DARLH procedures. The attacker database attached with each client machine independently to do attacker identification. The reports and alert messages are delivered from client machines to servers.

The monitoring servers evaluate the reports delivered from various monitoring nodes. The servers in honey pot are implemented with Supervised IDS agents. This agent monitors the activities of client machines and the attack reports. In addition, the network traffic is injected in to honey pot queues through servers [18, 19]. Then the server distributes the network traffics among clients connected to it. The honeypot traffic model follows authentic poison distribution model and protected queuing model.

Definition1*:* A Honeypot, H (P) containsnsimilarmonitoring servers and mclient machines. In this system, n servers collect the network events at the rate, T^e and generate attack reports at the rate, R^{μ}. At time τ, the monitoring resources of honeypot are utilized by the events at the rate, U^i. As the client machines are connected to servers, U^iimplies the aggregated utilization rate. Equation (1) and (2) illustrate traffic models and utilization factors respectively.

$$U^i = \frac{T^e}{R^{\mu} n} \tag{1}$$

$$U^i = S^i + C^i \tag{2}$$

U^i-Honeypot utilization factor

T^e-Average traffic rate at honeypot event queues

R^{μ}-Average response rate of each server

n- Total number of monitoring servers

S^i-Utilization factor of honeypot server

C^i-Utilization factor of honeypot client machines

$$C^i = C^1 + C^2 + \cdots \ldots . + C^n \tag{3}$$

Equation (3) denotes the individual utilization rate of honeypot clients. In each client machine, traffic queues are available to get the network events continuously. There are two types of events queue simple mented inside each machine. This queue setup is common to servers and client machines. The types of queues are given below.

- Internal Traffic Queues
- External Traffic Queues

In each server traffic queue (internal or external), $e\ X\ n$traffic events wait for server response. At the same time, server distributes the traffic events to the client machines. In each client traffic queue (internal or external), $e^i X\ m^i$ traffic events wait for the completion of monitoring process.

m^i denotes the client machine identifiers. e^i illustrates the distributed network events to each client machine in honeypot.

Let take on$Q^{eI}\ and\ Q^{eE}$, the number of internal and external traffic queues in each client machine respectively.. Then the number of internal and external traffic queues in each monitoring server are $Q^{SI}\ and\ Q^{SE}$ respectively. In this case, $Q^{SI} \gg Q^{eI}$ and $Q^{SE} \gg Q^{eE}$. Now, the honeypot traffic arrival and distribution models are given in equation (4). The honeypot poison model is determined as follow.

$$H^P = e^{-\tau^e} \frac{(\tau^e)^l}{n!} \pm b \tag{4}$$

l-Total number of traffic event arrivals

b-Event behavior factor

The poison model used for honeypot traffic distribution uses secure procedures which are incorporated in IDS agent modules. Algorithm 1 describes the credentials based poison distribution. In this technique, the events and the sources are validated before the event distribution. This kind of validation is completed at honeypot monitoring servers.

Algorithm 1: Honeypot Traffic Handling Using Authentic Poison Distribution

---

Input: Traffic Events (Internal and External)

Output: Received Events at Client Queues

Begin

Step 1: Get events, REQ (e^{ID}, e^{RID}, S^{ID})

e^{ID}-Event identifier, e^{RID}- Event Request Identifier, S^{ID}-Event source credentials

Step 2: Execute the procedure,$H^P$for all valid events

Step 3: Repeat $H^P$for all $Q^{SI}$and Q^{SE}

End

In this honeypot circumstance, the servers and clients maintain multiple parallel; queues. The queuing processes are defined by honeypot multi-system queuing model, $M / M / n,\ n > \tau^e$. This is determined for all finite number of internal and external traffic events. Therefore, the queuing model becomes, $M / M / n / \infty / l$, with the rate of $(e\ X\ n)$ at servers. At the same time, this rate is determined at clients as, $e^i X\ m^i$. The honeypot traffic distribution model is used to distribute the internal and external events into honeypot servers and clients respectively. The next section describes about server distribution and supervising model.

3.2 Server Distribution and Supervising Model

This model describes secure traffic distribution and client supervising tasks executed at server units. As discussed, there are n supervising servers available for monitoring m clients. The IDS enabled client machines are deployed for monitoring the events distributed from n serers. Authentic poison distribution model helps to share the events among various client machines. In order to deliver the events to clients, the server verifies the credentials of client machines and other resources.

Definition 2: A Honeypot gets continuous events from internal and external participants at time τ^i. The internal and external events are denoted as e^I and e^Erespectively. The events are collected at multiple $Q^{SI}$and Q^{SE}at regular intervals. The gathered events are distributed to Q^{eI} and Q^{eE}. The total events delivered to each queue is e^i. Honeypot servers validate m^i client credentials before the event distribution process. Algorithm 2 denotes server distribution and supervising model procedures.

The secret shared key communication between servers and clients is enabled using shared secret key, S^K.

$$S^{KH} = SHA^{512}(S^{IP}||C^{IP}||T||N)$$

Algorithm 2: Distribution and Supervision Model

Input: e^{I} and e^{E}

Output: Events at Q^{eI} and Q^{eE}

Begin

Step 1: Get internal and external events, e^{I} and e^{E} at $Q^{SI}$and Q^{SE}respectively

Step 2: Check for client monitoring requests, ReQ ($C^{IP}, T^{C}, S^{K}, C^{P}, C^{T}$)

C^{IP}-Client IP Address, T^{C}-Client TCP credentials, S^{KH}-Secret SHA key (512 Bits)

C^{P}-Honeypot Client policy credentials and C^{T}-Request Timestamp

Step 3: If ReQs available in client request queues

Then validate ReQ for all credential

Step 4: Server sends a request, ReQS($S^{IP}, T^{S}, S^{K}, S^{P}, S^{T}, Q^{A}$)

S^{IP}Client IP Address, T^{S}-Server TCP credentials, S^{KH}-Same Secret SHA key (512 Bits)

S^{P}-Honeypot Server policy credentialsS^{T}-Request Timestamp, Q^{A}–Status (Q^{eI} and Q^{eE})

Step 5: Client gives response to server with Q^{A}status

Step 6: Call H^{P}based on Q^{A}status

Step 7: Complete the event distribution

i.e $Q^{SI}$to $Q^{eI}$and $Q^{SE}$to Q^{eE}

Step 8: Enable Client Supervision Mode (CSM)

Step 9: Request the clients to put them in Promiscuous Mode (PRM)

End

The CSM helps the servers to monitor their associated clients when the clients put themselves in open promiscuous mode for appropriate servers. Servers are attached with main attacker database (DoS) and client activity control system. The client activity control system validates each and every actions of client machines based on preconfigured rules (rule based classification). This helps to identify and supervise the legitimate activities of client machines. In addition, the server keep master DoS database (KDD Dataset) in safe zone for supervising fault detection rates.

3.3 Honeypots with DARLHLevel-1Model

DARLH system creates event driven IDS agents in each monitoring resources. This IDS agent functions are implemented using DARL techniques. DARL approaches are designed with the help of event based reward functions as shown in figure 1.

Figure 1 illustrates DARL based IDS agent execute actions against DoS attacks injected in honeypot environment. The environment provides rewards (positive or negative) for each action taken by IDS agent. Based on the state wise rewards provided to IDS agent, it trains itself to handle next action against DoS. In this figure both internal and external attackers inject DoS attacks into honeypot environment [17].

The servers located in monitoring section of honeypot distribute the events into clients as illustrated in Algorithm 1 and 2. In case, the attack is effectively detected, the environment generates positive reward to DARL IDS agent. Otherwise it generates negative notifications. Algorithm 3 gives DARLH level-1 functions and DARL agent activities in honeypot environment. In the environment, DoS monitoring system, data resources and other devices are deployed.

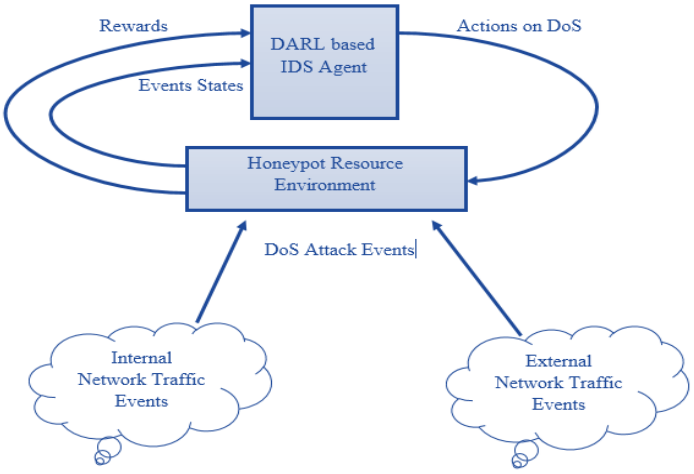

Figure 1: DARLH Level -2 Model

Algorithm 3: DARLHLevel-1 Model

Input: Events from Q^{eI} and Q^{eE}

Output: DARL action on DoS Events

Begin

Step 1: Set client at PRM and delivers,P^T, the traffic port to server

Step 2: Activate DARL agent function in each client machine, m^i

Step 3: Determine Q^{eI} and Q^{eE}sizes with waiting events in queue.

Step 4: Construct DNN for evaluating e^I and e^E.

Step5: Determine internal and external event functions $m^iIE_{(Q)}$and $m^iEE_{(Q)}$ respectively.

$$m^iIE_{(Q)} = Q_0\,(States,\ e^I\,, T^S) \tag{6}$$

$$m^iEE_{(Q)} = Q_0\,(States,\ e^E\,, T^S) \tag{7}$$

Step 6: Call procedure DoS(Data items_KDD) for all e^I and e^Eon various states.

Step 7: Do match function DARL(DoS)

If $((m^iIE_{(Q)}\&\&m^iEE_{(Q)})==KDD^{DOS})$

Then set action as “Alert”

Else action as “Allow”

Step 8: Compute internal and external RL action function. a, the action.

$$m^{i}IA_{(Q)} = Q_0\ (\text{Action States},\ e^{I}, a, T^{S}) \quad (8)$$

$$m^{i}EA_{(Q)} = Q_0\ (\text{Action States},\ e^{E}, a, T^{S}) \quad (9)$$

Step 9: Store the deep Q values for all $m^{i}IA_{(Q)}$ and $m^{i}EA_{(Q)}$

Step 10: Create internal and external alert repots, Report^{IA} and Report^{EA} for all m^{i}

Step 11: Deliver, Report^{IA} and $\text{Report}^{EA}$to servers

Step 12: Server generates rewards to each actions at different states and sends to clients.

Step 13: Clients’ RL IDS agent receives reward, R^{S}.

Step 14: Train the RL IDS agent function based on received R^{S}.

End

The DARLH agent runs in each client machine. This agent follows deep RL structure which has multiple layers of hidden layers and complex event analysis functions to detect DoS. In addition, the DoS detection results are forwarded to supervisor server units deployed in honeypot environment. Particularly, this deep IDS agent follows markov process for finding sequential observations. In this proposed DARLH system, the deep Q values are used to specify each event particulars. At the same time, the events are predicted using current Q values of events. As shown in equation (10).

$$EC_{i(Q+1)}(S_i, a_\tau) = EC_{(Q)}(S_i, a_\tau) + \alpha(S_i, a_\tau) \times \{(ma^{S_i} + mi^{S_i}) + R^{S}\} \quad (10)$$

In this equation, $EC_{i(Q+1)}(S_i, a_\tau)$ predictable network events at the state S_i with an action at the

time a_τ. $EC_{(Q)}(S_i, a_\tau)$ and R^S represent current event particulars and reward respectively. In addition, this RL based DoS detection uses learning rate at each state, $\alpha(S_i, a_\tau)$.

$$R^S = R^S{}_{(Q+1)} \pm \omega - EC_{(Q)}(S_i, a_\tau) \tag{11}$$

Equations (11) and (12) describe reward functions and the action taken after getting server side reward. ω specifies reward variation factor. In equation (12),$Q(e)_O$ is event action on reward and σ *is an action bias*.

$$Q(e)_O = \sum_{i=1}^{L} f\,(R^S \pm \sigma) \tag{12}$$

The DARL IDS agent functions are effectively utilizing the server reward function and event attributes for finding DoS patterns. At the same time, this DARL has well trained DNN and deep Q values. However, the DARLHLevel-1 model uses only DARL IDS agent against dynamic DoS attacks. This leads a technical problem for more dynamic events. In order to solve this issue, the proposed DARLH system implements DARLHLevel-2 mechanism.

3.4 Honeypots with DARLH Level -2 Model

DARLH Level-2 mechanism helps to integrate both DRNN and DARL based IDS agents to monitor DoS attacks in honeypot. The dual DNNs are connected in a pipeline manner to accomplish more accurate detections. In this case, both networks are feedback structured networks. This dual DNN is implemented with IDS agent of honeypot system. Figure 2 depicts the DNN connectivity of DARLH Level -2 Model. Though DARL supports for dynamic arrival of network event, it lacks at more uncertain conditions. In order to solve this problem DRNN is required against real-time uncertainties.

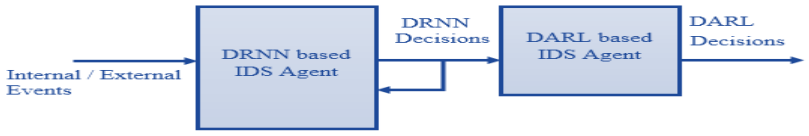

Figure 2: DARLH Level -2 Model

Algorithm 4: DARLH Level -2 Model

Input: Events from Q^{eI} and Q^{eE}

Output: DARLH Level-2 action on DoS Events

Begin

Step 1: Call event initialization steps and functions in Q^{eI} and Q^{eE}

Equations (6) and (7)

Step 2: Call procedure DoS(Data items_KDD) for all e^{I} and e^{E}on various states.

Step 3: Compute, Recurrent event observations,

$$e_T = \tan e(\omega_{ee}.e_{T-1}, \omega^{e}) \quad (13)$$

ω_{ee}-RNN weight function, e_{T-1}-Previous observations and ω^{e}-Input event weight

Step 4: Construct DRNN using hidden weight functions

Do match function DRNN(DoS)

If (($m^{i}IE_{(Q)}$&&$m^{i}EE_{(Q)}$)==KDD^{DOS})

Then set action as "Alert"

Else action as "Allow"

Step 5: Collect and store RNN observations as level-1 events

Step 6: Call DARLH level -1 procedures (Algorithm 3)

End

Algorithm 4 illustrates DRNN and DARL combinations for detecting DoS attacks in honeypot. In this algorithm, DRNN is a level-1 DNN and DARL is a level-2 DNN. DRNN generates the decisions on DoS using KDD training sets and the rule based classifications. The events and results generated by DRNN are evaluated with the help of DARL IDS agent. Both

IDS agents are using KDD DoS features and real-time policy irregularities happen due to attacks. DRNN and DARL networks are trained themselves in each iteration of honeypot events. In addition, the dual DL techniques support for implementing complex attack analysis processes.

Algorithm 4 describes the steps involved in DRNN and DARL networks to detect DoS attack detection. The proposed honeypot environment creates more complex DL techniques for enquiring both internal and external DoS attacks. In this regard, it deals with separate event queues in client and server units. In this proposed DARLH system, client machines are monitoring DoS events on both sides parallel. The event distribution and reward management functions are controlled by server units. This is a completely protected environment against DoS attacks. The next section describes implementation details and results produced for evaluating the performance of proposed system.

4. Experimental setup And Results

The proposed DARLH system is implemented with 10 server units and a generic database. Each server in honeypot controls 25 client machines. The proposed DARL and DRNN IDS agents are implemented in servers and clients. At the same time, event distribution model and client control principles are deployed at servers only. For detecting DoS attacks, the IDS agents run rule based classification procedures and signature mapping techniques. In this environment, the KDD' 99 dataset is deployed at serer points and clients points. Server points hold main dataset but client machines hold training datasets and test datasets.

This experiment environment is built using Weka 3.0 and Python 3.7 tools. Weka 3.0 is used to pre-process the KDD' 99 dataset. This removes unwanted data and missing data from original dataset. Python 3.7 is used to implement proposed DARLH techniques. The performance of proposed system is evaluated using metrics such as True Positive Rate, False Positive Rate,

Classification accuracy and error rate. This is evaluated for both internal and external DoS attacks. Then the results are compared with the existing schemes like, Game and Naïve Bayes Honeypot (GNBH), Block chain Honeypot (BCH) and RNN based Signature Generation and Detection (RNSG).

GNBH technique uses game theoretical schemes and Naïve Bayes schemes for detecting DoS attacks in honeypot systems.

This technique classifies and isolates DoS attack from the overall events. The ML techniques used in this work delivered significant outcomes but limited accuracy at dynamic situations of honeypot traffics. Moreover, the techniques are conventional tactics against runtime attacks like DoS. In the same manner, BCH is used to construct secure and distributed honeypot system. Block chain is an emerging technology used to provide authentic transaction in community networks. It uses complicated cryptography techniques to protect the data. This work used this technology for building protected honeypot systems. This method did not evaluate attack scenarios in the experimental setup. However, this system evaluated security aspects in the performance evaluation.

RNSG was a DL based signature generation system to protect the network against attacks. A complicated signature pair provide more security in network transactions. The signature credentials were created using DRNN structures for detecting intrusions. This system used signature based pattern recognition techniques for intrusion detection. This system implemented signature generation and pattern recognition techniques using DRNN. This is an effective technique for detecting intrusion detection. Yet, it has limitations related to runtime dynamics such as traffic variations and parallel monitoring complications.

Figure 3 illustrates the comparison between the existing techniques and proposed DARLH system. In this figure, True Positive Rate is taken as measurement against number of attacks raised in honeypot.

In this comparison, DARLH performs well than other existing systems. As DARLH maintains multi-level monitoring system (DARL and DRNN), it effectively detects DoS attacks. The other systems use only standard techniques against runtime DoS dynamics. In the existing systems, RNSG is only technique provides better true positive rate against higher rate of attacks. GNBH and BCH are delivering distorted performances against dynamic honey pot traffics.

Figure 4 and 5 provide the details of internal and external circumstances. Figure 4 illustrates the details of internal and external DoS attacks injected into honey pot resources.

In this figure, comparing to inter DoS injections external DoS injections are higher. Figure 5 gives the honey pot utilization rates. In this proposed DARLH system, servers and clients are implemented to manage DoS monitoring tasks. In this monitoring process, external utilization rate is higher in both client and server units compared to internal rates.

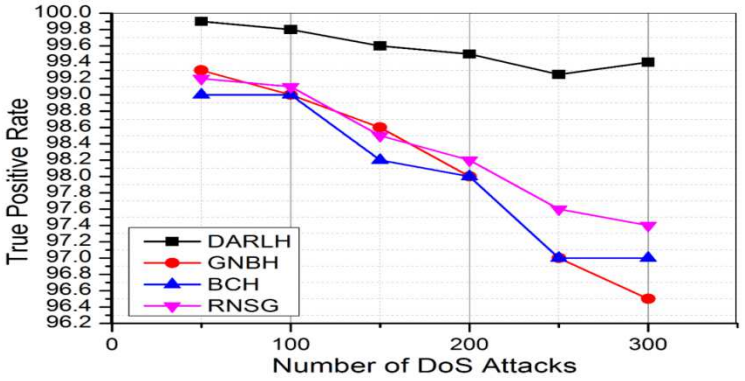

Figure 3: True Positive Rate

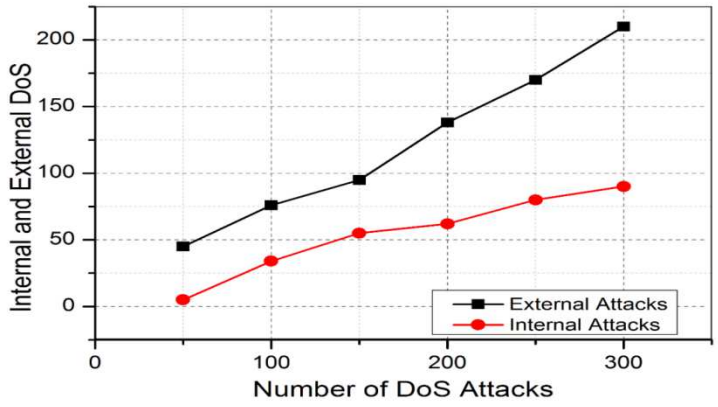

Figure 4: Internal and External DoS Attacks

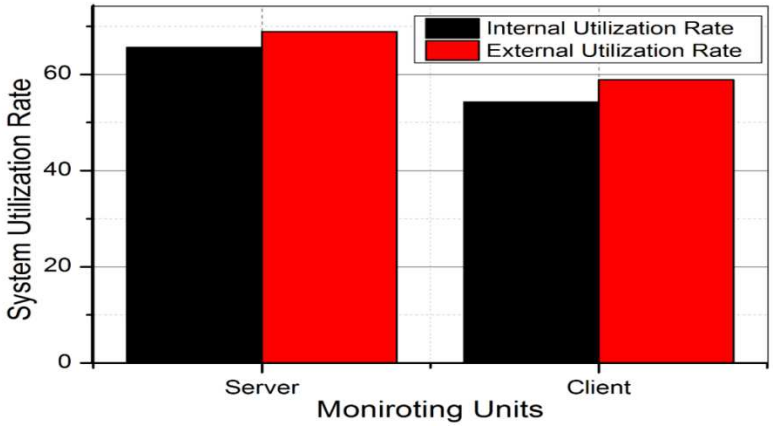

Figure 5: Honeypot Monitoring and Utilization Rate

Figure 6 and 7 are providing the details of internal and external DoS detection rates. In this experiment, number of internal and external DoS attacks is increased between 100 and 600. In both cases, the proposed DARLH system detects the attacks effectively. Since it has individual event management queues, the events of honeypot are neatly distributed and monitored. In addition, both internal and external attacks are identified using appropriate authentication schemes. Comparing existing systems, RNSG is the technique provides closer performance rate to proposed DARLH system than other systems. The reason is RNSG is a kind of DL approach. In overall performance comparisons, the proposed DARLH system produces optimal and effective results against runtime attacks in honeypot systems [21, 22].

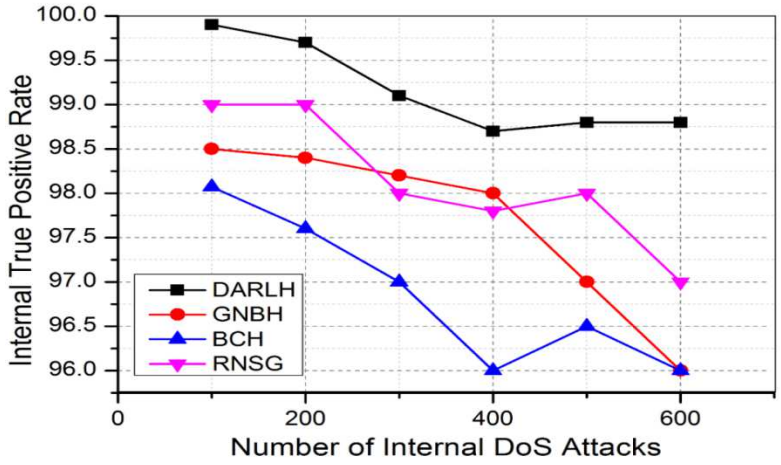

Figure 6: Internal DoS Attack

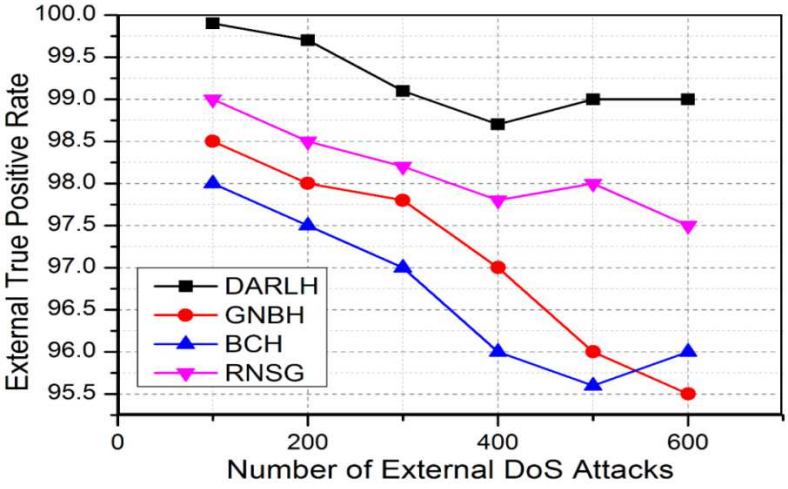

Figure 7: External DoS Attack

5. Conclusion

In this work, the proposed DARLH system was designed and implemented for detecting runtime DoS attack in honeypot environment. This proposed system developed secure event distribution techniques, server side monitoring techniques, DARL based level-1 DoS detection techniques, DARL and DRNN based level-2 DoS detection techniques. The techniques are implemented and compared with the existing techniques such as GNBH, BCH and RNSG. In this comparison, the proposed DARLH outperformed other techniques in all aspects against DoS attack. This work is

limited with single attack detection methodologies. This can be improved in future for multiple attack detection strategies.

Data availability statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Disclosure of potential conflicts of interest*:* Authors declare that they have no conflict of interest.

Declaration of interest Statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

References

[1] Hamada, A. O., Azab, M., & Mokhtar, A. (2018). Honeypot-like moving-target defense for secure iot operation. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 971-977. IEEE.

[2] Kumar, G., Saha, R., Singh, M., & Rai, M. K. (2018). Optimized packet filtering honeypot with snooping agents in intrusion detection system for WLAN. *International Journal of Information Security and Privacy (IJISP)*, *12*(1), 53-62.

[3] Veena, K., & Meena, K. (2019). Implementing file and real time based intrusion detections in secure direct method using advanced honeypot. *Cluster Computing*, *22*(6), 13361-13368.

[4] Doshi, R., Apthorpe, N., & Feamster, N. (2018). Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, 29-35. IEEE.

[5] Nguyen, S. N., Nguyen, V. Q., Choi, J., & Kim, K. (2018). Design and implementation of intrusion detection system using convolutional neural network for DoS detection. In *Proceedings of the 2nd international conference on machine learning and soft computing*, 34-38.

[6] Bingham, S. J., & Shirley, M. R. (2020). *U.S. Patent No. 10,560,434*. Washington, DC: U.S. Patent and Trademark Office.

[7] Negi, P. S., Garg, A., & Lal, R. (2020). Intrusion Detection and Prevention using Honeypot Network for Cloud Security. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 129-132. IEEE.

[8] Banday, M. T., & Sheikh, S. A. (2020). Improving Security Control of Text-Based CAPTCHA Challenges using Honeypot and Timestamping. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 704-708. IEEE.

[9] Ormazabal, G. S., & Schulzrinne, H. G. (2014). *U.S. Patent No. 8,689,328*. Washington, DC: U.S. Patent and Trademark Office.

[10] Shrivastava, R. K., Ramakrishna, S., & Hota, C. (2019). Game Theory based Modified Naïve-bayes Algorithm to detect DoS attacks using Honeypot. In *2019 IEEE 16th India Council International Conference (INDICON)*, 1-4. IEEE.

[11] Shi, L., Li, Y., Liu, T., Liu, J., Shan, B., & Chen, H. (2019). Dynamic distributed honeypot based on blockchain. *IEEE Access*, *7*, 72234-72246.

[12] Kaur, S., & Singh, M. (2019). Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. *Neural Computing and Applications*, 1-19.

[13] Khan, R. U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N. A., & Alazab, M. (2019). An adaptive multi-layer botnet detection technique using machine learning classifiers. *Applied Sciences*, *9*(11), 2375.

[14] Rao, N. S., Sekharaiah, K. C., & Rao, A. A. (2019). A survey of distributed denial-of-service (DDoS) defense techniques in ISP domains. In *Innovations in Computer Science and Engineering* (pp. 221-230). Springer, Singapore.

[15] Sharma, S., & Kaul, A. (2018). A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud. *Vehicular communications*, *12*, 138-164.

[16] Huang, L., & Zhu, Q. (2019). Adaptive honeypot engagement through reinforcement learning of semi-markov decision processes. In *International Conference on Decision and Game Theory for Security* , 196-216. Springer, Cham.

[17] Kotey, S. D., Tchao, E. T., & Gadze, J. D. (2019). On distributed denial of service current defense schemes. *Technologies*, *7*(1), 19.

[18] Su, L., & Ye, D. (2018). A cooperative detection and compensation mechanism against denial-of-service attack for cyber-physical systems. *Information Sciences*, *444*, 122-134.

[19] Kommareddy, C., Bhattacharjee, S., Shayman, M. A., & La, R. (2013). *U.S. Patent No. 8,397,284*. Washington, DC: U.S. Patent and Trademark Office.

[20] Gandhi, U. D., Kumar, P. M., Varatharajan, R., Manogaran, G., Sundarasekar, R., & Kadu, S. (2018). HIoTPOT: surveillance on IoT devices against recent threats. *Wireless personal communications*, *103*(2), 1179-1194.

[21] Kemppainen, S., & Kovanen, T. (2018). Honeypot Utilization for Network Intrusion Detection. In *Cyber Security: Power and Technology*, 249-270. Springer, Cham.

[22] Arifianto, R. M., Sukarno, P., & Jadied, E. M. (2018). An SSH Honeypot Architecture Using Port Knocking and Intrusion Detection System. In *2018 6th International Conference on Information and Communication Technology (ICoICT)*, 409-415. IEEE.

Figures

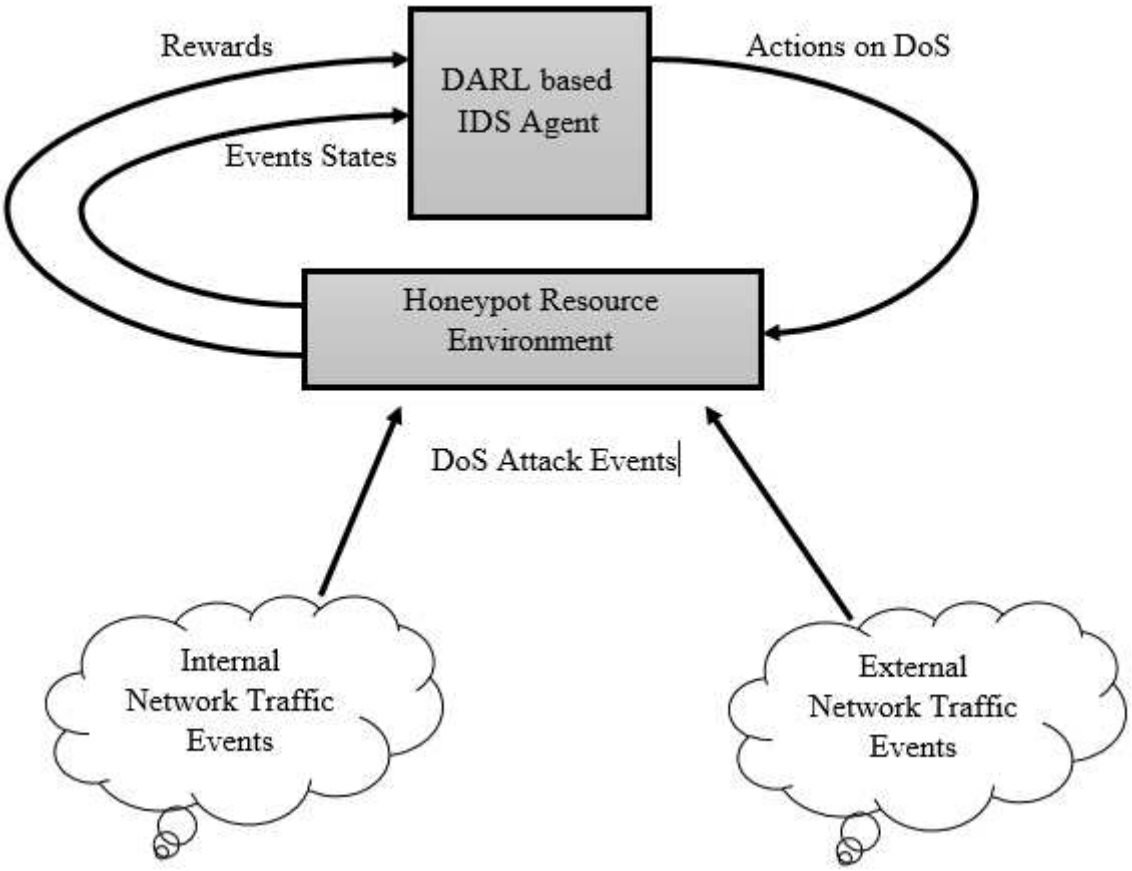

Figure 1

DARLH Level -2 Model

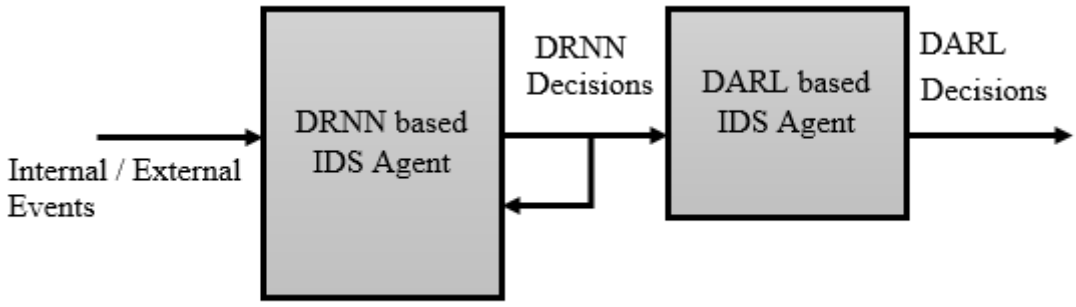

Figure 2

DARLH Level -2 Model

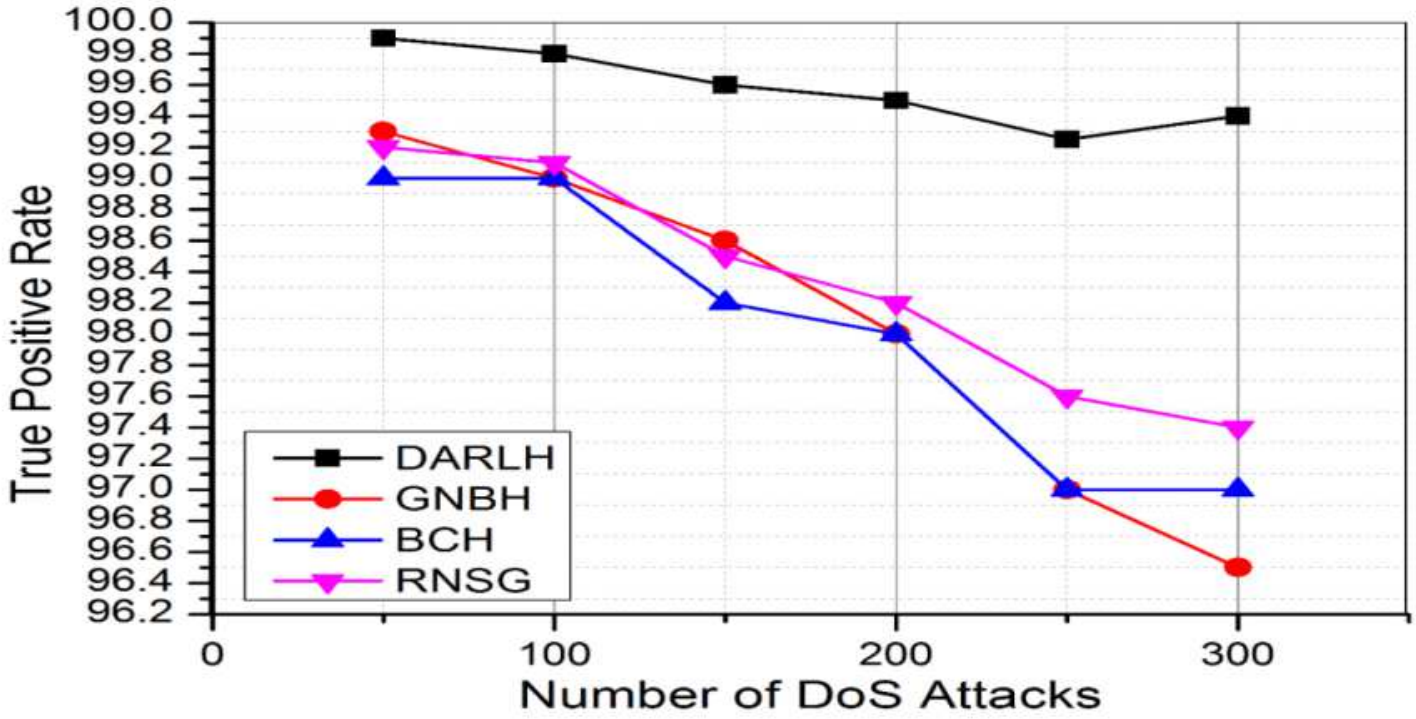

Figure 3

True Positive Rate

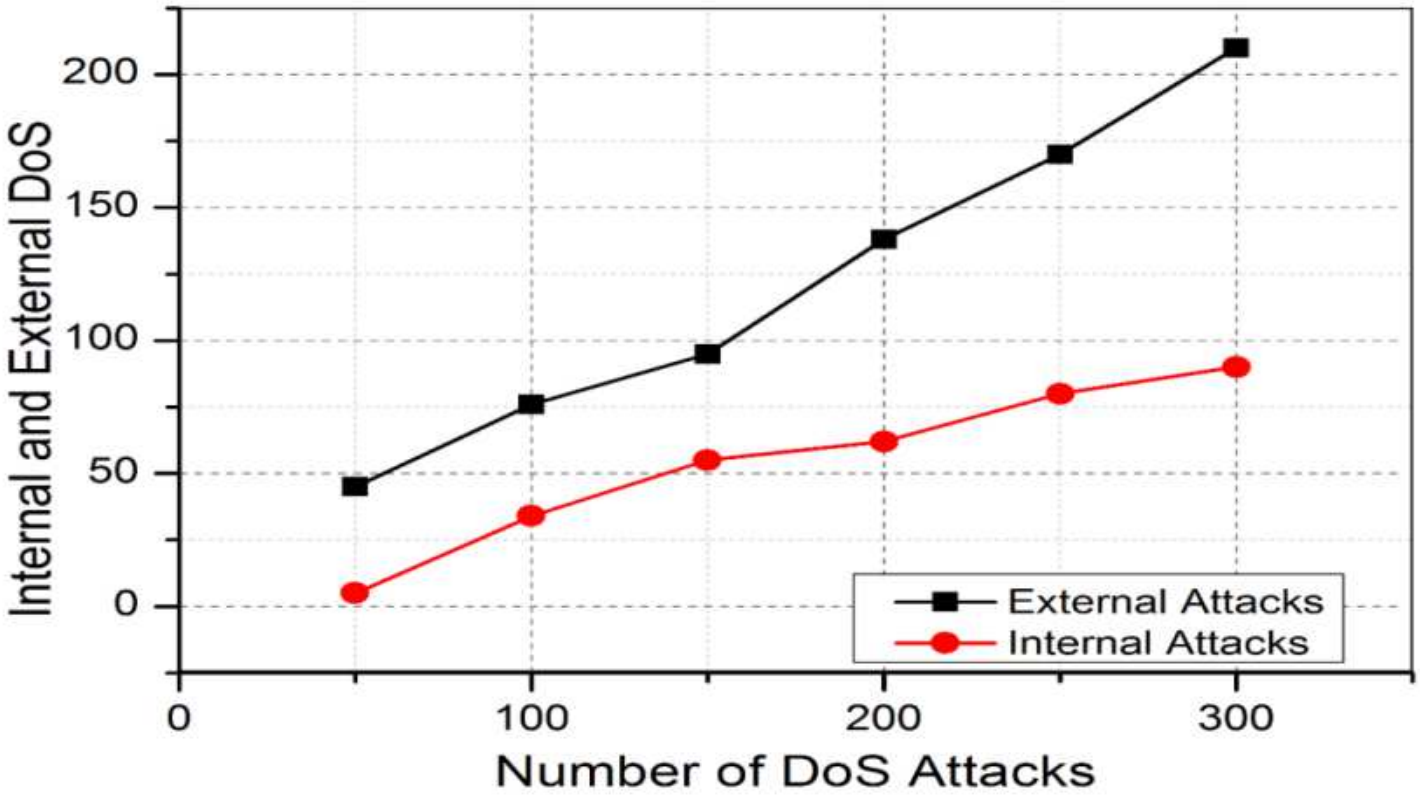

Figure 4

Internal and External DoS Attacks

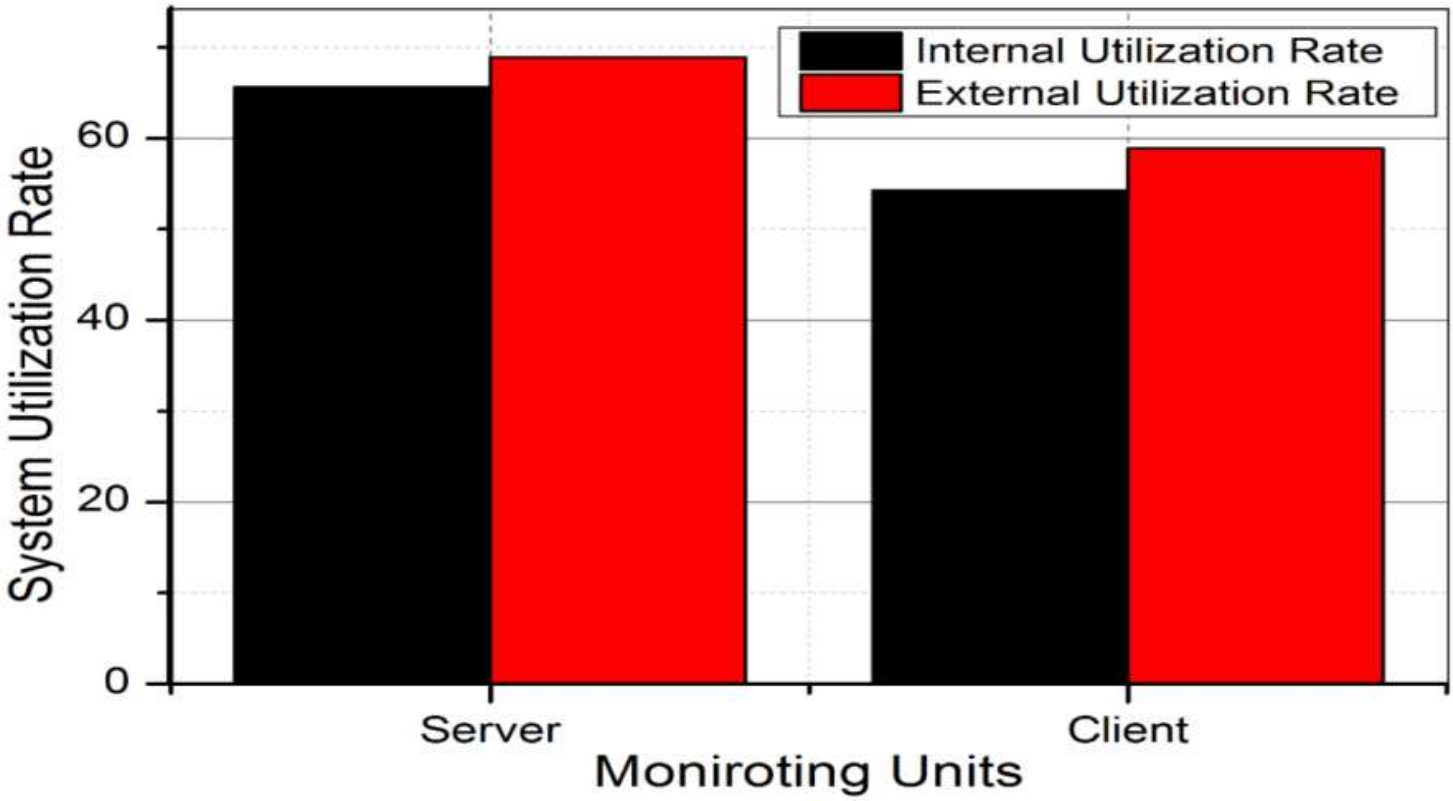

Figure 5

Honeypot Monitoring and Utilization Rate

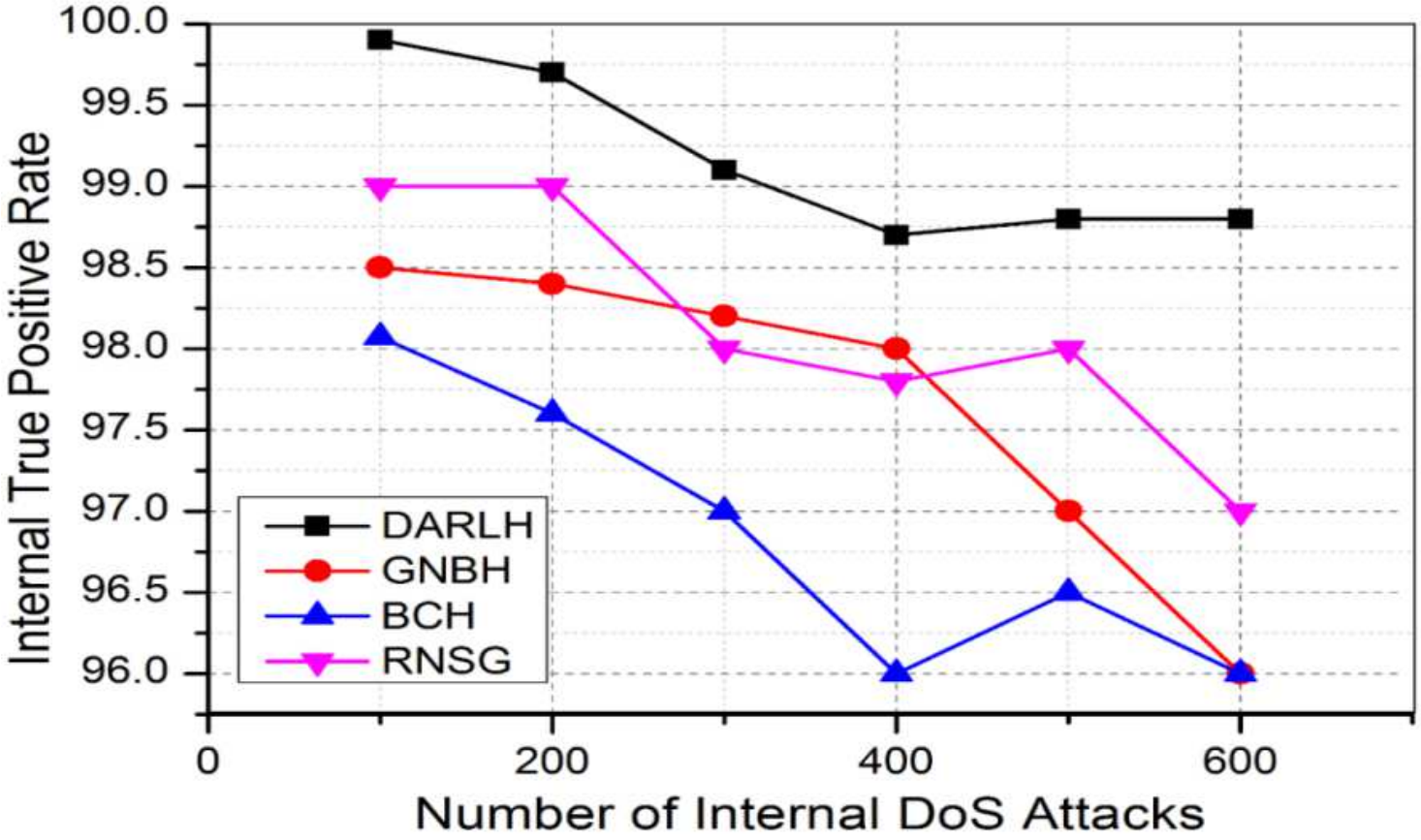

Figure 6

Internal DoS Attack

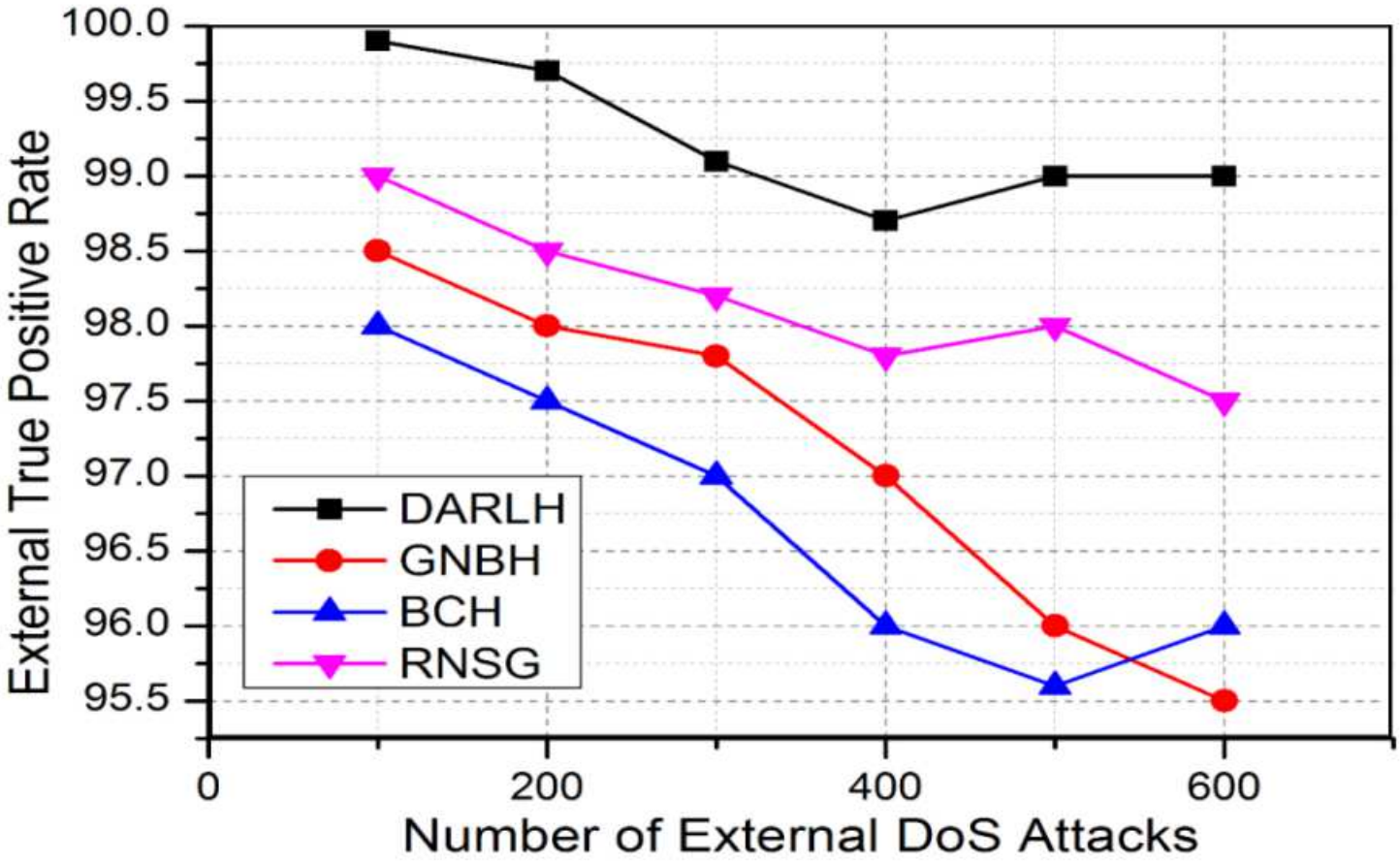

Figure 7

External DoS Attack