# Autonomous Dynamic Honeypot Routing Mechanism for Mitigating DDoS Attacks in DMZ

Anjali Sardana
*Department of Electronics and Computer Engineering*
*Indian Institute of TechnologyRoorkee Roorkee 247 667,*
*India* anjledec@iitr.ernet.in

R. C. Joshi
*Department of Electronics and Computer Engineering*
*Indian Institute of Technology Roorkee Roorkee 247 667,*
*India* joshifcc@iitr.ernet.in

***Abstract*- DDOS attacks generate flooding traffic from multiple sources towards selected nodes and cause obstruction in flow of legitimate information within a network. If the victim node is a server in DMZ requiring fast information processing, the entire network operation stops. We use various lines of honeypot based defence against such attacks. The first line of defence detects the presence of attacks and tags attack flows in real time. The work in this paper concentrates on the next line of defence, where a model for autonomous dynamic honeypot routing has been proposed in response to identified attack flows. We propose the automatic generation of adequate server nodes to service client requests and honeypots to interact with attackers in contained manner. The judicious mixture of servers and honeypots in DMZ at different time intervals provide stable network functionality even in the attacked network. We validate the effectiveness of the approach with modelling on Internet type topology and simulation in ns-2 on a Linux platform**

***Index Terms*– Distributed Denial of Service, Dynamic Honeypot, Autonomous, Mitigation, Security.**

## I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks arguably represent the most serious threat to the security and stability of any network including the Internet. The power of a DDoS[1] attack is based on the massive number of distributed and coordinated attack sources which aim at overwhelming a target server with immense volume of useless traffic. These DDoS attacks degrade or completely disrupt services to legitimate users by eating up communication, computational and/or memory resources of the target through sheer volume of the packets. DDoS attacks are amplified form of DoS attacks where attackers direct hundreds or even thousands of compromised "zombie" hosts against a single target [2].

Network level [3] DDoS attack aim at congesting the network, such as link capacity and router buffers, by flooding them with bogus packets. In service level [3] DDoS attacks, a large number of attack machines manage to acquire services from victim server consuming both service level resources such as server memory and processing time, as well as network level resources along the path outward from the server.

Our DDoS research focuses on three tracks: (1) Classification of the attack which move the classification away from server , (2) Mitigation which aim to mitigate the effects of DDoS attacks by isolating the illegitimate flows, (3) Attack tracking which try to identify DDoS attack sources. Firstly, entropy – based detectors at Point of Presence (POP) detect the presence of DDoS traffic. Next, the flows that fail the entropy tests are tagged as attack in subsequent time windows.

In this paper, we focus on the second line of defense, i.e. attack mitigation using autonomous dynamic honeypot routing and redirection to protect the information stored on servers in DMZ. The honeypots change in number and locations, providing deterrence from public domain servers. The flows tagged as attacks at POP are directed to honeypots. As the attack flows are isolated, it ensures smooth information flow for legitimate clients even in network under high attack. Connection retention with attack flow is to obtain information about the attackers by logging their actions which can by used for trace back. The network is self organizing and adopts a proactive behavior to circumvent any anticipated attacks.

Honeypots [4] serve two purposes; they strengthen entropic detection and act as mediation tool. Honeypots [5] are computer systems placed on a network to attract attackers, allowing administrators to capture and analyze current attack methodologies and use that information to harden their systems and networks. One of the biggest challenges when deploying honeypots is in maintaining the functionality of the total system and information protection on servers as the network behavior changes.

With honeypots [5], one of the configuration issues is how many honeypots to deploy. Also, it must be made sure that honeypots blend into the system [6] and appear like any other entity on the network. Our work addresses these issues as described. Our aim is to protect an FTP server from DDoS attacks while providing smooth information flow. FTP services have been replicated on all nodes to make honeypot act like FTP server responding in contained manner to attack flows. The work proposes a means for analyzing traffic, rapidly detecting attacks, and dynamically generating honeypots appropriate for that attack to interact with. Our honeypots are generated on the information gathered about the network to turn it into a set of honeypot definitions, and provides a means of deploying adequate number of honeypots.

We use GT-ITM, an ISP level topology generator and ns-2 [7] as test bed for simulations. We show that entropy captures variable rate DDoS attacks [4] coupled with honeypots providing stable response in attacked network. The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the proposed mechanism. Section 4 explains the simulation testbed. Results and discussion is presented in Section 5. Finally section 6 concludes the paper.

## II. Related Work

The number and assortment of both attacks as well as defense mechanisms is monstrous. Array of schemes proposed against DDoS attacks [3] are either based on point solutions or perimeter models. In case of a sustained high bandwidth attacks, it is not possible to contain the attack at border gateway or firewalls. Mostly, the attacks are dealt with after the mammoth damage has been caused to resources. This results in costly downtime and single point of failure. Moreover, identifying the attack streams alone is not viable due to their distributed nature.

In order to minimize the loss caused by DDoS attacks, attacks must be mitigated before underway. Most solutions to the DDoS attacks try to develop a packet filter that can distinguish legitimate packets from illegitimate ones and hence drop illegitimate packets only [3]. The D-WARD defense system [8] is deployed at source-end networks, and autonomously detects and stops attacks originating from these networks. Wide deployment of D-WARD will motivate service-level attacks. In [9], Bohacek has suggested a mitigating approach that relies on routers filtering enough packets so that the server is not overwhelmed while ensuring that as little filtering as possible is performed. The drawback is that legitimate traffic packets may also be dropped en route to the destination causing high collateral damage.

In the Pushback framework [10], once a router suffers from sustained congestion, it tries to detect flow aggregates that are contributing the most to congestion. Defining aggregate is difficult. Also, pushback requires contiguous deployment; to overcome these limitations, Selective Pushback [11] proposes to send rate-limiting requests to routers sending traffic with higher than "normal" rates. The detection of these routers and the profiling of normal traffic are performed via an enhanced but costly probabilistic packet marking scheme.

In [12], Kalantari et al. have proposed a proactive method for mitigation of the effects of DDoS attacks wherein each router maintains a partition of active TCP flows into aggregates. Each aggregate is probed to estimate the proportion of attack traffic that it contains. Packets belonging to aggregates that contain significant amounts of attack traffic may be subject to aggressive drop policies to prevent attack at the intended victim. Again, in this case too, legitimate packets face the risk of being dropped.

Static honeypots [5] have been used to defend against a variety of attacks. However because of their deployment at fixed and detectable locations, they may be compromised by sophisticated attacks. Moreover, a compromised static honeypot can be used to attack servers in the network. Sherif et. al. [13], proposed the proactive server roaming mechanism, which is a secure and light-weight mechanism to proactively change the location of the active server within a server pool. However, the scheme incurred an overhead that caused performance degradation both in the absence of attacks and under low attack loads. In [4], because of sacrificing some servers to act as honeypots, distributing the load on all the servers outperformed the roaming honeypots scheme in the case of a high legitimate client load combined with a low attack load.

Our autonomous dynamic honeypots approach requires servers and honeypots to continuously change their number and location. IP hopping [14] protects a public server; as the server changes its IP address after detecting it is under attack without changing its location. However, as physical location is not changed, physical connections are not broken and states of attacked systems remain unclean. IP hopping can be used both reactively and proactively. IP hopping does not block a persistent attacker which looks up the new IP address using DNS. By physically moving the service from one server to another and cleaning the state of the old server avoids the limitation of IP hopping.

Mutable Services [15] is a framework to allow for reactively relocating service front-ends and informing only pre-registered clients of the new location through a secure DNS-like service. Service replication and load balancing diffuses the load of DDOS attacks over a number of simultaneous replicas. However, replication is not DDOS attack-tolerant in the same degree as it is fault-tolerant. For instance, replication alone cannot tolerate large-volume attacks that can clog all the replicas. Moreover, they starve legitimate clients due to overburdened servers. It is observed that most of the mitigation techniques in practice today, suffer from the following drawbacks:

1. They are reactive in nature.
2. They deploy packet dropping policies at the routers wherein even legitimate packets face the risk of being dropped.
3. The topology of the network needs to be known in advance.

Our scheme uses autonomous dynamic honeypots that overcome the above limitations of IP hopping and use replicated honeypots for attack tolerance along with replicated servers for load balancing.

## III. Proposed Mechanism

We use an autonomic safeguard mechanism against DDoS attacks which generates dynamic honeypots for attack isolation and load balancing. The total budget (total number of machines) get partitioned into two groups, servers and honeypots. The good traffic is routed to the servers, while the attack is diffused across replicated honeypots. The number of servers ($N_s$) and honeypots ($N_H$) is adaptive and intelligently mapped according to the client load (CL) and attack load (AL) respectively in real time. The locations of servers and honeypots change at variable time intervals or eons depending on the network conditions to mitigate the effects of the attack. The dynamic nature of number of honeypot and server help perform load balancing.

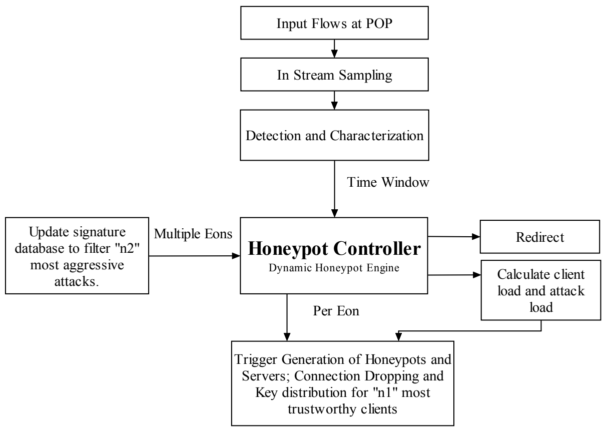

Input Flows at POP
In Stream Sampling
Detection and Characterization
Time Window
Update signature database to filter "n2" most aggressive attacks.
Multiple Eons
Honeypot Controller
Dynamic Honeypot Engine
Redirect
Calculate client load and attack load
Per Eon
Trigger Generation of Honeypots and Servers; Connection Dropping and Key distribution for "n1" most trustworthy clients

Fig. 1 Functionality of Honeypot Controller Engine

Our approach for dynamic honeypot generation is in response to flows identified as legitimate and attacks (using entropy detectors and characterization [16]) in the flow list (FL) maintained at POP. A honeypot controller (HC) has been modelled at the POP and performs three functions. First, it keeps a track of FL and state of flows over moving time windows. Using flow list (FL maintained during detection and characterization process) as the input it determines the values of client load (CL) and attack load (AL) [4].

Second, depending on the attack load and client load, dynamic honeypot engine (DHE) triggers the generation of appropriate number of servers and honeypots and coordinates their timings and location. A dynamic honeypot engine (DHE) uses CL and AL and generates a judicious mixture of active servers and honeypots from the server pool after varying time intervals called "eon" for replication and load balancing.

Table 1 is used for mapping CL and AL to optimum combination of servers and honeypots, $N_S$ and $N_H$ respectively.

TABLE I
MAPPING LOAD TO HONEYPOTS AND SERVERS

| Attack Load (AL) | Client Load (CL) | Number of Honeypots ($N_H$) | Number of Servers ($N_S$) |
|---|---|---|---|
| Low | Low | Low | 2 moderate - low |
| Low | Moderate | Low | 2 moderate - low |
| Low | High | Low | 2 moderate - low |
| Moderate | Low | Moderate | Moderate - low |
| Moderate | Moderate | Moderate | Moderate |
| Moderate | High | (Moderate – low* ; Moderate) | (Moderate; Moderate + low*) |
| High | Low | 2 moderate - low | Low |
| High | Moderate | (moderate; moderate + low**) | (moderate – low**; Moderate) |
| High | High | Moderate | Moderate |

The values of low, moderate and high client and attack loads are defined according to the bandwidth (BW) resource:

TABLE 2
DEFINING LOW, MODERATE AND HIGH LOADS

| | **Low** | **Moderate** | **High** |
|---|---|---|---|
| **Client Load** | Bottleneck Link BW (link utilization <1) | Bottleneck Link BW (link utilization=1) + 20 % BW | Bottleneck Link BW (link utilization=1 ) + 50 % BW |
| **Attack Load** | < 30 % BW | 30% - 65 % BW | > 65 % BW |

The parametric values of "low" and "*moderate*" in computing $N_S$ and $N_H$ in table 1 depend on the total budget $N$ s.t. *moderate=N/ 2* ; *low* = $\lceil \leq 30\% N \rceil$ ; *low** = $\lfloor \leq 30\% N \rfloor$; *low*** alternates between $\lfloor < 30\% N \rfloor$ and $\lceil < 30\% N \rceil$ to give priority to client requests.

*Deciding the location of servers and honeypots*

Having identified the number of servers and honeypots, next we utilize backward hash chain to calculate the locations and achieve roaming which is inherent in autonomous dynamic honeypots. This scheme builds on connection migration mechanism for issuing the roaming trigger proactively.

$N_S$ active servers continuously change their location within a pool of $N$ homogenous servers to proactively defend against DDoS attacks. Service time is divided into eons; at the end of each eon, $CL$ and $AL$ is mapped to $N_S$ and $N_H$ using table 1 and the service migrates from one set of servers to other in the server pool. A long hash chain is generated using a one way hash function $H()$, and used in backward fashion similar to Pay-Word scheme [17]. The last key in the chain, $K_n$, is randomly generated and each key, $K_i$ $(0<i<n)$, in the chain is computed as $H(K_{i+1})$ and is used to calculate both the length, $R_i$ , of service eon $E_i$, and set of current active servers $F_i$ during $E_i$.. Let $S$ represent the set of indexes of an array $NA[]$ of servers. Also let $P_{NS}(S)$ represent an ordered set of all possible subsets of $S$ with cardinality $N_S$. The cardinality of $P_{NS}(S)$ is $N_P = (N_S\,!/(N!\,*\,(N_S - N)!))$, where $N$ is the number of servers in $NA[]$. Then for each service eon $E_i$, the set of current active servers is $P_{NS}(S)[MSB_{\lg NP}\ H'(K_i)]$, where $H'()$ is a one way hash function and $MSB_x(y)$ are the $x$ most significant bits of $y$.

*Deciding the length of eon*

The length of eon should be as large as possible so that file requests are handled in same eon to avoid TCP migration, connection reestablishment and TCP slow start phase. However, in case of attacked network, smaller eon is favoured to quickly change the location of servers and clean their state. The length $R_i$, of the service eon $E_i$ is uniformly distributed in the interval $[u, m+u]$ seconds as follows: $R_i = u + MSB_m(H''(K_i))$, where $m$ is the design parameter that represents the current threat level signified by $N_H$ and changes according to $N_H$. Value of $m$ is inversely proportional to the threat level, i.e. it is directly proportional to $N_S$ and inversely proportional to $N_H$ and is derived from table 3.The value $u$ represents a lower bound on the idle time of a server and should be long enough to analyse attacks. HC module keeps account of $N_S$, $N$, $u$, $NA[]$, $m$ and $P_{NS}(S)$ as roaming updates. Values of K are distributed among clients based on their trust levels over multiple eons which is used to derive the location of active servers and length of eons.

TABLE 3
MAPPING $N_S$ AND $N_H$ TO M

| *m* | | $N_S$ | | | | |
|---|---|---|---|---|---|---|
| | | **L** | **M-L** | **M** | **M+L** | **H** |
| **$N_H$** | **L** | 5 | 5 | 5 | 5 | 5 |
| | **M – L** | 4 | 4 | 3 | 3 | 4 |
| | **M** | 3 | 4 | 3 | 4 | 3 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | **M + L** | 2 | 1 | 1 | 2 | 2 |
|  | **H** | 1 | 1 | 1 | 1 | 1 |

Third, the legitimate traffic is routed to one of the randomly selected active server in server pool by variations in the dynamic field like destination address. The controller establishes a dedicated connection and adjusts routing information so that the attack traffic is forwarded to the randomly selected honeypot for interaction.

### *The redirection algorithm*

The redirection algorithm performs the per-time window (<eon) treatment of each flow in the Flow List (FL) at POP. The pseudo code is as follows:

```
HoneypotControllerTimeWindow ( FL, Time window size)
L: For each flow in FL
If (Tag = attack)
        Parse the primary packet and search source and destination address ($F_{DA}$ and $F_{SA}$)
        $P_{DA} = F_{DA}$
        C1: If ($P_{DA}$ = Destination address of honeypot)
                Forward the packet to $P_{DA}$
        Else
                        Replace $P_{DA}$ by destination address of honeypot
                        Forward the packet to $P_{DA}$
        N1: If (Next Time Window)
                Goto L
        Else If (More Fragment = 0)
                Goto N1   //wait for next time window
        Else
                        Parse next header of the flow for $P_{DA}$
                Goto C1
Else
Parse the primary packet and search source and destination address ($F_{DA}$ and $F_{SA}$)
        $P_{DA} = F_{DA}$
        C2: If ($P_{DA}$ = Destination address of active FTP server)
                Forward the packet to $P_{DA}$
        Else
                        Tag = attack
                        Forward the packet to $P_{DA}$
        N2: If (Next Time Window)
                Goto L
Else If (More Fragment = 0)
                Goto N2   //wait for next time window
        Else
                        Parse next header of the flow for $P_{DA}$
                        Goto C2
```

Fig. 2 Redirection Algorithm

Hence the attack traffic is isolated from legitimate traffic and diffused over honeypots instead of interfering with active servers in the pool. All data from this interaction is logged at a remote database located inside DMZ for later analysis. A higher-level responsibility of the honeypot controller is to manage and prioritize the selection of server set.

### *Mathematical model for three operating points*

At any active ftp server node i, the arriving traffic is the aggregate of several legitimate (*l*) and possibly of some attack *(a)* flows, where $l = (l_1, l_2, \ldots, l_j, \ldots, l(N_{fl}))$ and $a = (a_1, a_2, \ldots, a_i, a(N_{fa}))$ where $FL = \{aUl\}$. The total traffic rate $\lambda_i$ arriving to node is composed of two parts:

$$\lambda i = \sum_{n} \lambda_i^n, n + \sum_{d} \lambda_i^d, d \qquad (1)$$

where $\lambda_n^i, n$ is the legitimate incoming traffic rate which belongs to normal flow *n*, and $\lambda_d^i, d$ is the arrival rate of attack packets belonging to flow *d*.

Some attack traffic may be mistakenly taken to be normal traffic while some normal traffic will be mistakenly thought to be attack traffic. Any traffic characterized as attack is redirected at one of the randomly selected honeypot servers. Thus, a fraction $f_i$, n of normal traffic (the probability of false alarms) and a major fraction of attack traffic *di,*d (the probability of correct detection) will be redirected to honeypot as it arrives to the honeypot controller. If the attack detection mechanism were perfect we would have *fi,*n = 0 and *di,*d = 1. However, if characterization mechanism [20] is such that probability of false negative is 0 i.e. practically *di*,d = 1,this is termed as best defence. If probability of false alarm is 0, i.e. $f_i$,n=0, it is called naïve defence. The intermediate operating point is the normal defence. However there may be some false positives, so 1>*fi*,n >0. Once a packet is admitted into a honeypot controller, it is queued and then forwarded based on its selected destination address. During redirection, ideal situation at honeypot is described by equalities, $f_i, n = 0$ and $d_i, d=1$.

TABLE 4
OPERATING POINTS: BEST AND NAIVE

| *i* | **Best Defense (Min false Negatives)** | **Naïve Defense ( Min False Alarms)** |
|---|---|---|
| **Active server** | $((1-d_i,d)=0)$ ; $(1-f_i,n)$ | $((1-f_i,n)=1)$; $(1-d_i,d)$ |
| **Honeypot** | $d_i,d=1$; $f_i,n>0$ | $d_i,d<1$; $f_i,n=0$ |

## IV. SIMULATION TESTBED

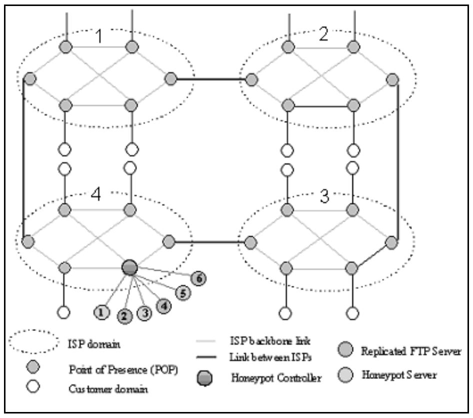


Fig. 3 Experimental Testbed

Our test bed is an ISP level network with four cooperative ISP domains (1, 2, 3, and 4) where each domain has 10 POPs represented by single node each as shown in fig. 3. One customer domain is attached to each POP which consists of legitimate and attacking hosts. Two POPs in every ISP are attached to other ISPS. ISP domain 4 has additional POP for connecting to our protected server. Our aim in is to protect the server located in DMZ of ISP domain 4 from DDoS. Transit domain routers are represented as POPs and stub domains as customer domains attached to POPs. The POP link to server bandwidth for honeypot server is kept same as FTP server so that it exactly imitates the FTP server. Table 5 gives simulation parameters for topological expansion in fig. 4.

### *Topological Expansion*

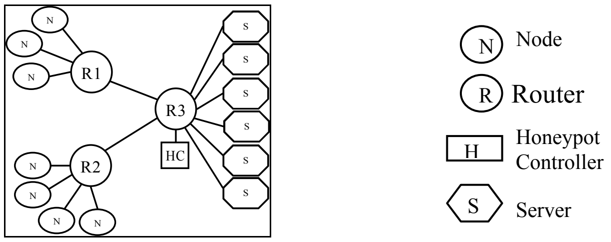


Fig. 4 Topological Expansion

TABLE 5
BASIC PARAMETERS OF SIMULATION

| Parameter | Value | Description |
|---|---|---|
| Client Load | 0.1-1 Mbps | Relative load issued by client requests |
| Attack load | .1-5Mbps | Relative load due to attacker traffic |
| Simulation time | 0-350 sec | Simulation duration |
| Attack time | 50 – 300 sec | Attack duration |
| Legitimate Traffic type | TCP | File Transfer Protocol |
| Attack Traffic Type | UDP | Constant Bit Rate |
| Client-Router BW/Delay | 5 Mbps / 20sec | Bandwidth |
| Attack-Router BW/Delay | 5 Mbps / 20sec | Bandwidth |
| Router-Router BW/Delay | 5 Mbps / 20 sec | BW at 1st level |
| Router-Router BW/Delay | 15 Mbps / 20 sec | BW at 2nd level |
| Router-Server BW/Delay | 45 Mbps / 2 sec | Bandwidth |
| Number of attackers | 70 | Number of attackers |
| No. of Legal sources | 35 | Number of clients arriving according to Poisson process |
| Total Budget | 6 | Number of Servers in the Pool |

## V. RESULTS AND DISCUSSION

We overloaded the network with extremely high client and attack loads and analysed Mean Time between Failure and Average Response Time to determine behaviour of network for three operating points: best, normal and naïve (table 4).

*Mean Time between Failures*

MTBF is attributed to the "useful life" of the device. Calculations of MTBF assume that a system is "renewed", i.e. fixed, after each failure. It is an attribute than quantifies attack tolerance of the network. System is said to have failed when the ART becomes infinite. However, a system must also guarantee a promised QOS i.e an upper limit on the ART. For the purpose of simulation, we assume the system to have failed if ART becomes greater than the sum of last five eons. The worst case downtime of a network would be duration of an eon.

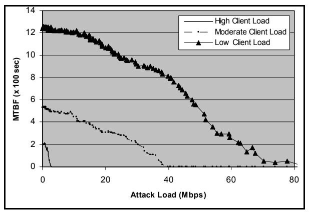


Fig. 5 Variation of MTBF with AL

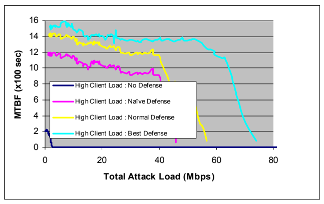


Fig. 6 Variation of MTBF with AL; High CL

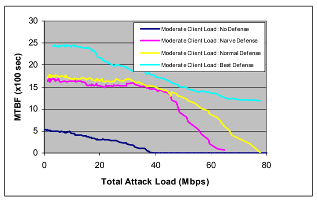


Fig. 7 Variation of MTBF with AL; Moderate CL

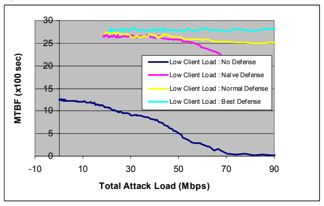


Fig. 8 Variation of MTBF with Low CL

Figure 5 shows the variation in MTBF of a network with the increase in AL in case no defense against DDoS is used. In the presence of high CL, even a slight increase in AL causes the system to abrupt failure. Even at low CL, the system fails in the presence of a very low AL.

Next, we study MTBF with the proposed scheme for three operating points, namely naïve, normal and best defense. Fig. 6 shows that in case of high CL, best defense gives better stability as compared to naïve defense. It is so because in best defense, there are no false negatives and all the attack flows are directed to honeypots. Whereas in case of naïve defense, many attack flows will be directed to active server (false negatives) causing disruption in services and ultimately leading to the failure of the network with increase in attack load. At moderate and low client loads, best defense has greater MTBF than naïve defense due to similar reasons as described. This is shown in figures 7 and 8 respectively. This demonstrates that the proposed scheme has the potential to give stable network functionality even in presence of attacks.

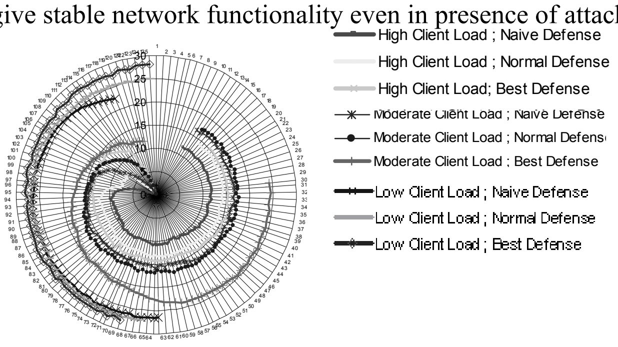


Fig. 9 Variation of MTBF with Client load: Naïve, Normal and Best Defence : The Radar Plot

In figure 9 the end point of radar plot curves concentrated at center show failure. In the presence of high CL, even a slight increase in attack load causes the system to abrupt failure. The network is comparatively more stable with respect to moderate andlow client load. Without DDOS defense, even at low client load, the system fails in the presence of a very low AL (not shown in fig.).

*Average Response Time*

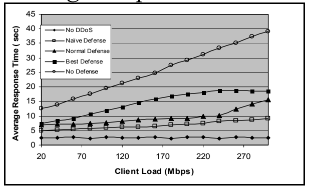


Fig. 10 Variation of ART with CL (Low AL)

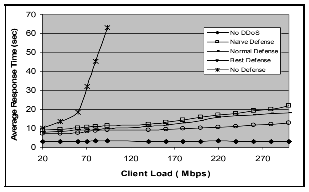
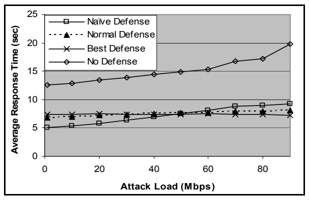


Fig. 11 Variation of ART with CL (High AL)



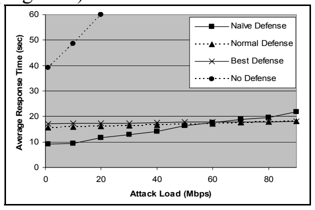


Fig. 12 Variation of ART with AL (Low CL)

Fig. 13 Variation of ART with AL; (High CL)

In the proposed scheme, no failure was reported for very large simulation intervals at very low AL as shown in fig. 9 There was response to requests in same or subsequent eons. So, the variation in ART has been studied with varying CL at low AL as in fig. 10.

Fig. 10 shows that in absence of defense, the ART increases with an increase in CL even in the presence of low AL. The naïve defense gives lowest ART values because as CL increases, false positives increase. These flows are directed to the honeypots before redirected to active servers.

Fig. 11 shows the variation in average response time with increasing client load at attack load. With high attack load, best defence gives better ART values as compared to naive defence. At high AL, best defence is able to identify and isolates server from attack load, thus maintaining stable ART.

Fig. 12 shows in the presence of dynamic honeypots, in case of naïve defence, the ART increases linearly with increase in AL. For normal and best defence, ART remains consistent even with the increase in AL. This is so because active servers are isolated from attacks due to negligible false negatives.

Fig. 13 shows variation in ART with AL in the presence of high CL. In the presence of dynamic honeypots, all attack traffic is diverted to honeypot relieving the active servers from the attack. For naïve defense, ART increases linearly with AL and ART remains consistent for normal and best defense. Hence, our scheme is insensitive to the AL and is capable of giving stable network functionality for high AL.

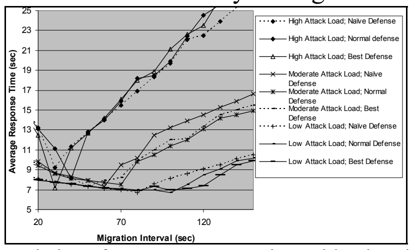


Fig. 14 Variation of average response time with migration interval

Fig. 14 shows the effect of migration interval on ART. Upto a certain value of migration interval, ART decreases and then increases. Below a threshold, small values of migration intervals lead to frequent connection breakdowns and TCP slow start phase and hence increased ART. After a particular value of migration interval, ART again starts increasing, more so at high client load because attack starts to build up causing network delays. The ART graphs in above section attained optimum value of migration interval *m* before results were taken, thus giving stable network functionality.

The mechanism is autonomic as it uses a control loop to self heal and configure the network parameters It dynamically adapts its offered services and resources to meet changing network demands. It uses reasoning to optimize its behavior and policy based management to govern the behavior of the control loop. Values of CL and AL determine $N_S$ and $N_H$ which is mapped to eon (m) which further control CL and AL.

## VI. Conclusions

The work in this paper investigated autonomous dynamic honeypot defence method that isolates the attacks before they reach the potential targets in DMZ. Evaluation of the scheme on mean time between failure and average response time under three modes of operation show that scheme gives desirable QoS in case of smooth change in client load. In case of abrupt changes, it adapts itself to changing network

The scheme experiences a favourable attack load independent behaviour and self optimizes the detection parameters. The scheme mitigates DDoS attacks keeping collateral damage and overheads negligible. The results show significant improvement in how the network deals with DDoS attacks. Dynamically changing judicious mixture of honeypots and servers has the potential to mitigate DDoS while maintaining stable network functionality.

Some of the avenues for further experimentation are with larger and heterogeneous networks. Policy to distribute keys depending upon trust level can take the management off the HC engine at the expense of increased computation and control messages. Both of them hold promise for evaluating and improving our DDoS defense method.

## References

[1] CERT Coordination Center. *Denial of Service Attacks*. http://www.cert.org/tech_tips/denial_of_service.html.

[2] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovi, "Distributed denial of service Attacks", In *Proc. of IEEE International Conference on Systems and Cybernetics* vol. 3, pp. 2275-2280, 2000.

[3] J. Mirkovic, and P. Reiher, "A Taxonomy of DDoS Attack and DDoS defense Mechanisms", *ACM SIGCOMM Computer Communications Review,* vol. 34, no. 2, April 2004.

[4] A. Sardana, K. Kumar, R. C. Joshi, "Detection and Honeypot Based Redirection to Counter DDoS Attacks in ISP Domain," *IEEE Proceedings of IAS'07*, pp. 191-196, Aug-2007.

[5] L. Spitzner, "Honeypots: Simple, Cost-Effective Detection", 30 April 2003. URL: http://www.securityfocus.com/infocus/1690

[6] M. Kuwatly, M. Sraj, Z. A. Masri, and H. Artail "A Dynamic Honeypot Design for Intrusion Detection", *IEEEProc. 24th ICDCS'04, 2004*

[7] NS Documentation. http://www.isi.edu/nsnam/ns

[8] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source". In *Proceedings of ICNP 2002*, Paris, France, pp. 312–321, 2002.

[9] B. Stephan, "Optimal filtering for denial of service" *Proc. 41st IEEE Conference on mitigation, Decision and Control*,pp: 1428–1433, 2002

[10] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxon, and S. Shenker, "Controlling high bandwidth aggregates in the network", *SIGCOMM Computer Comm. Review,* ACM Press, vol. 32, pp. 62–73..

[11] T. Peng, C. Leckie, K. Ramamohanarao, "Defending against distributed denial of service attack using selective pushback". *Proc of ICT,* 2002.

[12] M. Kalantari, K. Gallicchio and M. A. Shayman, "Using transient behavior of TCP in mitigation of distributed denial of service attacks". In *Proc 41st IEEE Conference on Decision and Control*, pp. 1422 – 1427, Dec. 2002.

[13] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, D. Mosse', and T. Znati, "Proactive Server Roaming for Mitigating Denial-of-Service Attacks", *In Proceedings of ITRE'03* pp. 286-290.

[14] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek, "The Migrate Approach to Internet Mobility",*Proc. Oxygen Student Workshop,* 2001.

[15] P. Dewan, P. Dasgupta, and V. Karamcheti, "Defending against Denial of Service attacks using Secure Name resolution", In *Proc SAM 2003.*

[16] A. Sardana, R. C. Joshi, T. Kim, "Deciding Optimal Entropic Thresholds to Calibrate the Detection Mechanism for Variable Rate DDoS Attacks in ISP Domain", P*roc. ISA,* 2008, pp. 270-275.

[17] R. L. Rivest and A. Shamir, "PayWord and MicroMint–Two Simple Micropayment Schemes", *Proc. IWSP*, vol. 1189 LNCS, pp. 69–87.Springer,1996.