Indian Journal of Science and Technology, *Vol 8(36), DOI: 10.17485/ijst/2015/v8i36/90557, December 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Fast IP Hopping Protocol SDI Implementation

Vladimir Krylov*, Kirill Kravtsov and Eleanora Sokolova

Nizhny Novgorod State Technical University Named after RE Alekseev, Nizhny Novgorod, Russian Federation; vkrylov@heterarchica.com, kirill@kravtsov.biz, essokolowa@gmail.com

Abstract

Background/Objectives: This paper suggests a new approach against Distributed Denial of Service attacks and traffic eavesdropping in TCP/IP networks. **Methods/Statistical Analysis**: In our method, DDoS resistance is achieved by introducing a new address policy in IP networks. This policy hides physical location of protected server for all unauthorized clients by dynamic mapping of server's address on a large set of temporary addresses. This paper provides detailed description of the method and its mathematical model. **Findings:** The authors discuss basic implementation, its major practical constraints and results of initial validation. The proposed method is compatible with Software Defined Network and we discuss major possible advantages of IP Fast Hopping application in SDN. **Applications/Improvements:** The demonstrated technique suggests new addressing policy that based on randomizing of server's address. This new addressing policy does not require any global significant changes in the existing Internet architecture and can be implemented as software solution on client and server ends without impact on transit infrastructure.

Keywords: DDoS, IP Fast Hopping, Network Security, SDN, Traffic Intrusion

1. Introduction

In this article, we discuss security challenges in TCP/IP networks and, especially, Software Defined Networks. Such networks are vulnerable to various types of network attacks like traffic eavesdropping, IP Spoofing Attacks, Denial of Service etc[1]. The current study focuses on protection against Distributed Denial of Service attacks (and DoS attacks as a special case) and traffic eavesdropping. However, our method can be applied against other types of attacks when a male factor generates a separate data stream to a victim, also introduced technique increases confidentiality of communication session by hiding of destination of this session.

DDoS attack is one of the major threats to modern networks; the size and frequency of such a type of attacks continue growing despite the fact that quite a number of defense approaches were proposed. Such type of network attack, initiated against controller of software defined networks, can introduce a significant harm and prevent expected behavior of the whole network. During DDoS attacks, a number of malefactor terminals (botnet) and legitimate users are connected to the victim server at the same time. Each bot sends a big number of requests to the victim that create a heavy malicious traffic targeted to the server. Since the increase in the flow of requests is created here increase the number of terminals, therefore, whichever level of server performance has not been achieved, starting from a certain number of bots, they create a flow of requests exceeding the permissible level for any server. In related literature, a certain number of different classifications of DDoS detection and mitigation solutions are presented. According to various principles, they can be divided into groups: Based on the location of their deployment (source-based, network-based or destination-based)[2], based on the type of the applied algorithm (statistical, knowledge-based, soft computing, data mining and machine learning methods[3]) and, according to other principles, there are a couple of more groups to be found[4,5].

However, generally, we can see that a greater part of existing DDoS protection mechanisms is based on a wide range of traffic analysis and filtering algorithms. They uses reactive strategy, where, on the first stage, a method analyses traffic, tries to detect active DDoS attack and after that filters malicious traffic. In these approaches, the network

**Author for correspondence*

address of a node is a unique identifier of this node. Therefore, if a botnet initiates a traffic stream targeted to an address and there is no DDoS protection mechanism, this stream achieves victim node unchanged. We can say that DDoS solutions deal with a direct network tract between a set of sources and a victim terminal (receiver). In this paper, we describe an alternative approach: DDoS attacks resistance can be achieved via randomization of this tract by a specific addressing policy when a network address is only temporary conditional identifier of a victim server. This technique called convoluted multi address networks and is being discussed in the next paragraph. In this paper, we demonstrate a general approach of network address manipulation that lead to boosting robustness of networks and data against a wide range of different treats like DDoS attacks, traffic eavesdropping and analysis, etc.

2. Materials and Methods

2.1 A Model of Convoluted Multiaddress Networks

To demonstrate our network security method, consider a mathematical model of network traffic traversing with decomposition mapping of instantaneous traffic intensity from a network node with address x to network node with address y on three separate mappings: Mapping $in(x, i)$ of an outgoing network traffic to network address space; automorphism $m(n, R)$ of a network address space, where is n is delay in traffic traversing and R is a substitution function on a set of address pairs; mapping $out(j, y)$ from a network address space to an incoming traffic (Figure 1). Consider a matrix with a number of rows and columns, corresponding to the size of the address space of the network, as a model of the network address space. Elements $a(t, i, j)$ of the matrix are the value of traffic intensity from address i to address j at moment t.

Mapping $rout(n): A_x^{out}(t) = A_y^{in}(t-n)$ ensures direct traffic translation from source x to receiver y with delay n.

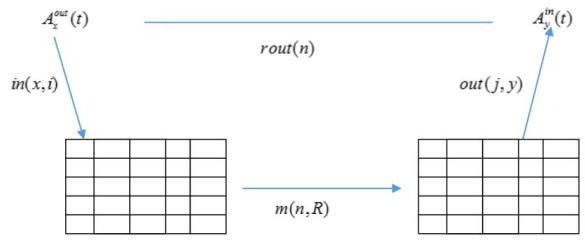

Figure 1. Model of traffic traversing in networks.

Mapping $in(x, i)$ defines address i of source x in the network: $in(x,i): a(t,i,j) = A_x^{out}(t)$. Mapping $out(j, y)$ defines which network address y is related to the receiver y: $out(j,y): A_x^{out}(t) = a(t,i,j)$.

Consider possible substitutions R when our diagram of mappings is commutative. The trivial solution here is $R(i, j) = (i, j)$. In this case, mappings $in(x, i)$ and $out(j, y)$ can be defined in the following way: $in(x,i)$: $a(t,x,y) = A_x^{in}(t); out(j,y): A_y^{out}(t) = a(t,x,y)$. Obviously, this trivial solution is not exclusive. Consider a case when this substitution is not trivial and the value of the second argument is defined by function $f(j)$: $R(i, j) = (i, f(j))$; $out(j, y)$: $y = f(j)$. This solution describes translation of a receiver's address scenario (NAT). Another possible solution here is a substitution with function $hi(t, j)$, dependent on time t: $R(i, j) = (i, hi(t, j))$; $out(j, y)$: $y = ho(t, j)$; $hi(.,.) = ho(.,.)$. We call this solution is referred to as address hoping. There are two hopping functions – hopping of source $hi(t, j)$ and hopping of receiver $ho(t, j)$. If these functions are equal, the commutativity of the diagram of mappings is also retained. Evidently, there are other possible mappings (which retain commutativity), but the above-mentioned solutions are suffice for this study.

Now, consider a case with multiple sources and one receiver. Taking into account the initial mapping for each pair source-receiver (x, y), it is easy to see that traffic intensity at network node y from multiple sources $x \in X$ can be treated as an additive value:

$$A_y^{out}(t) = \sum_{x \in X} rout(n_x) A_x^{in} \quad (1)$$

To protect a network described in the model from distributed denial of service and other types of network attacks with independent streams of malicious traffic, we should find an addressing policy which brakes additivity in Equation (1) in the following way: Traffic intensity from a particular ("legitimate") source x_1 is being transmitted to a particular receiver y without losses, but for all other sources $x \in X\ x \neq x_1$,intensity of traffic targeted to node with address y will be attenuated in K times. We can see that this task has a solution using the following definitions of mappings:

$$in(x,i): a(t,i,j) = A_x^{in}(t);\ R(i,j) = (i, hi(t,i)); out(j,y): y = ho(t,j);$$

$$hi(t,k) = ho(t,k) \forall (t,k) if x = x_1 else hi(t,k) \neq ho(t,k)$$

In fact, according to commutativity of the diagram of mappings, traffic from a legitimate source x_1 will be transmitted unchanged via the network. For all other

sources x, traffic transmission will occur only at particular time moments when the values of hopping functions $hi(t, j)$ and $ho(t, j)$ match. Therefore, if such time moments do not exist, the traffic intensity from the corresponding source for receiver y will be equal to zero. If the values of hopping functions are equal at particular time moments, the network traffic from illegitimate sources will be transmitted to receiver y only at these moments. Average traffic attenuation is defined by the following expression:

$K = \frac{|T_m|}{|T_h|}$, where $|T_m|$ is the cardinality of a set of time moments of message transmission, $|T_h|$ is the cardinality of a set of time moments when the values of the hopping function are equal.

In this abstract model we shown that applying the special addressing policy in a network provides a way to attenuate or completely block malicious traffic in this network. Our approach does not contain analyzing and processing network traffic and, furthermore, is based only on an addressing policy applied in a particular network. In our work, networks, in which such a policy is implemented, are called convoluted multiaddress networks and the addressing policy itself – address hopping. The Convoluted multiaddress network technique is based on the following principle: A network address does not provide a unique identification of a particular network node. Instead, the system has a special set of mapping functions, which builds a dynamic dependence between a physical network node and an address with a set of parameters. As has been demonstrated above, the filtering ability of this method depends on the selected hopping function and the size of address space.

2.2 IP Fast Hopping

IP Fast Hopping technique[6,7] is intended to make the real destination of a client's communication invisible for all external terminals and, consequently, to prevent DDoS attacks and unauthorized access from illegitimate clients. IP Fast Hopping method is based on our model of convoluted multiaddress networks and is implementation of this model in TCP/IP networks. In the IP Fast Hopping approach, a server has a random IP address for each particular client at the each time moment. After this method has been applied, for an external observer close to a client, a communication session between the client and the server does no longer look like a packet stream between these two Internet terminals. Instead of this, an observer detects a packet flow between a client and a number of independent (topologically and physically) terminals in the Internet. None of the streams in this flow has a correlation between packets inside the stream.

IP Fast Hopping is similar to radio systems with frequency hopping. In such systems, a receiver and a transmitter are switching from one frequency to another frequency synchronously during an ongoing data transmission session. A malefactor's transmitter, which is going to introduce a noise into such a session, does not have an actual schedule of frequency hopping; therefore, such an attacker cannot do noticeable harm to the legitimate transmitter defended by the frequency hopping mechanism. In our case, frequency can be treated as an IP address. So, the legitimate client must know the schedule of the server's IP address changing. At the same time, the schedule should be unavailable to non-legitimate clients.

2.2.1 Abstract Model of IP Fast Hopping

To illustrate the basic idea of applying of convoluted multiaddress networks in TCP/IP networks, consider a system model, with a server with address s, a set of clients $C = \{c_1, ..., c_l\}$, a subset of network address space $IP_V = \{ip_1, ip_2, ..., ip_N\}$ where $s \notin IP_V$ and a set of gateways $R = \{r_1, ..., r_M\}$ where $M \leq N$. Each client $c_i \in C$ has a representation of server's address s: s' or "initial address". Each message a from whole set of messages A from the client c_i to the server s is being transmitted via route p with input point p_{in} (gateway) with address y and output point p_{out} (gateway) with address x. In this model we can logically split the system on three independent addressing subsystems: 1. Subsystem (subnetwork) W_1 from the client c_i to input point p_{in} of the route p; 2. Subnetwork W_2 of the route p; 3. subnetwork W_3 from out point P_{out} of the route p to the server s. When a message a is being transmitted through the particular subnetwork W_z, a has a specific destination address IP_{dstW}. Obviously, $IP_{dstW_1} = s'$, $IP_{dstW_2} = x$, $IP_{dstW_3} = s$ To make entire system model consistent, each W_z has a function F_z that maps the destination address of a message a in the preceding W_{z-1} to a destination address in the current subnetwork W_z (Figure 2).

In common client-server architecture of Internet networks, the destination address of message a is always equal to s, which means that $P_{in} = c_i$ and $x = s$. So, our system model simplifies as shown in Figure 2. In this case, function F_z is trivial and, thus, all messages from c_i addressed to s' are being transmitted directly to server s.

In this case, an external observer can easily identify the address of traffic destination and initiate a malicious data stream to address s and this traffic achieves the physical server. Therefore, an unauthorized client can acquire access to the server. In addition, in these conditions, a set of malefactor terminals (botnet) $B = \{b_1, ..., b_k\}$ can initiate a brute-force DDoS attack on address s (Figure 3).

As has been reflected before, nowadays most of DDoS prevention techniques suggests installing firewalls and filtering solutions in a network between a set of clients C and victim server s. With reference to our model (Figure 1.), we can treat getaway P_{out} as a firewall, which performs traffic analyzing and filtering according to one of existing defensive methods against DDoS attacks. Therefore, in these approaches, W_3 subsystem remains unprotected and is treated as trusted. Still, in terms of addressing policy the whole system remains transparent, i.e., all messages targeted to the server always have destination address equal to s. In this paper, it is suggested to conceal s in untrusted subsystems W_1 and W_2 in order to hide the location of a victim server and, as a result, prevent unauthorized access from illegitimate clients.

In IP Fast Hopping, $s' = y$ and s is unavailable outside W_3. Each client c_i is connected to shared global network W_2 via security gateway P_{in}. Due to the fact that s', c_i local representation of server's address s, is equal to the address of this gateway, all messages a targeted to the server with address s will achieve this gateway. The packet a has a pair of keys that uniquely identify this message: 1) timestamp (message creation time) of a, t_a, as a public key and 2) unique identification of the client-originator of this packet, ID_{c_i}, as a private key. The system model under consideration has a pseudo-random function $H\left(t_a, ID_{c_i}\right) = n \in \left[1; N\right]$ that determinates valid virtual address ip_n of the server for particular message a. This function is regarded as a hopper function because of it defines the hopping of the IP address of the protected server in convoluted multiaddress networks. Due to the fact that unique identification of the client is part of the domain of function H, hopping of s can be different for different clients. These virtual addresses ip_n are a representation of s in subnetwork IP_V. Therefore, $F_2 = H(t_a, ID_{c_i})$. An address subspace IP_v has a number of disjoint subsets $IP_V^{r_m}$, where $i \in \left[1; M\right]$ and $IP_V = \mathrm{U}_1^N IP_V^{r_m}$. Each address from $IP_V^{r_m}$ is related to corresponding gateway r_m, thereby W_2 has M paths p_m for messages from client c_i (Figure 4).

Gateway r_m validates all incoming packets and maps destination address of these messages IP_{dst} on address in W_3 using function F_3 (NB: Zero address means here that the message should be treated as malicious):

$$F_3\left(t_a, ID_{c_i}\right) = \begin{cases} IP_r, IP_{dst} = ip_n \\ 0, IP_{dst} \neq ip_n \end{cases}, \text{ where } n = H\left(t_a, ID_{c_i}\right)$$

The same rules are applied for the source address of all responses from the server to clients c_i. As a result, a stream of messages between the server and clients are separated into independent streams between independent terminals in the network.

2.2.2 Implementation

IP Fast Hopping has a variety of possible implementations. In this section, one of simplest approaches is described. In such simple form of our method, the timestamp of TCP header of each packet is used as timestamp t_a for hopper function $H\left(t_a, ID_{c_i}\right)$. Also, unique identification ID_{c_i} of packets' client-originator ID_{c_i} is defined

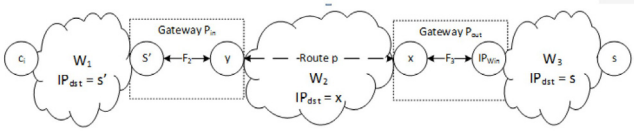

Figure 2. Abstract system model.

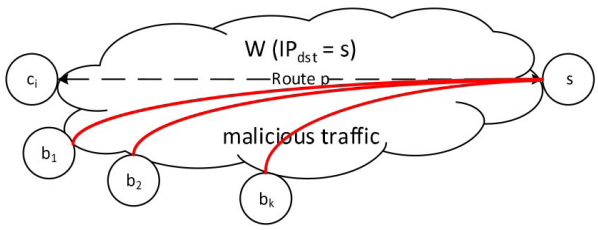

Figure 3. Model of common client-server architecture with brute-force DDoS attack.

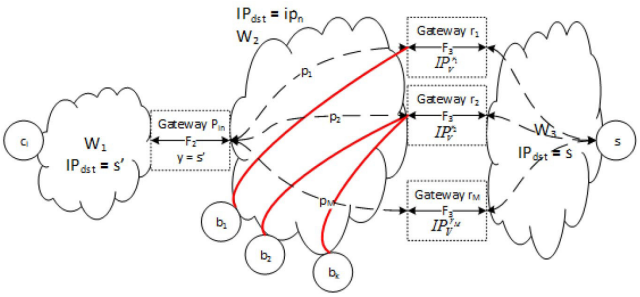

Figure 4. IP Fast Hopping model.

during authorization. To make this approach consistent with other network protocols, DNS servers contains information about the address of an authorization server instead of address *s* of the protected server. In this case, our addressing rules do not change the existing network architecture or protocol except the fact that our technique requires adding several network terminals (getaways), which are responsible for calculation, validation and changing destination IP address of packets. This additional functionality can be added to network routers. Therefore, as example implementation, we implement IP Fast Hopping mechanism as kernel module of OS GNU/Linux. In this case, installing of this module on GNU/Linux-based routers would be enough to deploy the suggested system.

Linux kernel contains built-in firewall Netfilter, which is responsible for packet filtering and forwarding according to predefined rules by iptables utility. Netfilter supports five hooks of rules: Prerouting, Input, Forward, Output, Postrouting. In the implementation suggested above, Netfilter contains a new module, which is responsible for changing of IP address into destination field of outgoing packets and into source field of ingoing packets. This module calculates a new IP address according to the IP Fast Hopping rules (by timestamp field and session UID). During a handshake, an authorization server adds new set of rules into Postrouting hook on a client's terminal and into Prerouting hook on each gateway. This rule activates the kernel module, which implements the following algorithm:

- On the client side this module calculates a hash-function using the timestamps field and session UID for each outgoing packet addressed to the initial IP address. After that, the module uses this result as index of correct address into IP pool, which should be put into the destination field of the packet. For each ingoing packet from the same communication session, the module performs the same actions for the source address field: Checks the current value of the field (by calculating the same hash-function) and changes it on the initial address.
- On the switches side this module calculates the same hash-function using the timestamps field and session UID for each ingoing packet addressed to IP addresses from IP pool. If the current destination address corresponds to the timestamps field and session UID, the real IP address of the server will be placed into the destination field. Otherwise, the packet will be dropped. For all ingoing packets issued by the server, the module will replace the source field by one of virtual addresses according to the current value of the hash-function.

3. Results and Discussion

3.1 Initial Experimental Results

The described basic implementation of IP Fast Hopping approach has been validated on small test stand. The main purpose of our experiments is to show that IP Fast Hopping successfully filters traffic from unauthorized clients. To achieve this goal, we built test stand consisting of several Virtual Machines: Client, client's router and server router (both machines had implemented Netfilter module installed), authorization server, victim server and bot. During the experiment, we measured the average traffic intensity at the server's network interface during active DoS attacks and without attack (see Figure 5). For generating DDoS attack we used third-party application, LOIQ.

Easy to see, that even such minimal testing has shown that applying of IP Fast Hopping methods leads to filtering of malicious traffic stream from incoming traffic of a protected server.

Implementation of IP Fast Hopping in SDN does not require any significant changes in the suggested implementation. Software Defined Infrastructure is frequently based on GNU/Linux solution, so such basic implementation is compatible and easy to adopt. Furthermore, protected SDN controller may be utilized here to balance load between switches which perform validation of packets addresses.

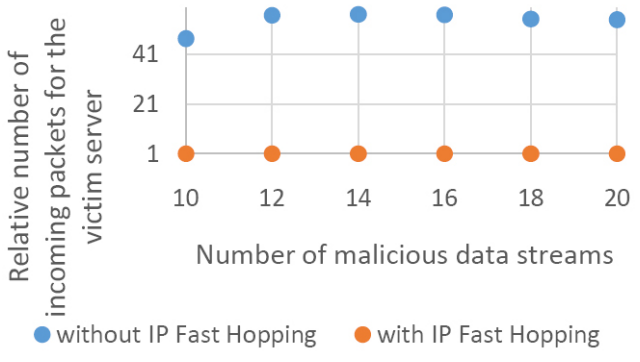

Figure 5. Results of validation of basic implementation of IP Fast Hopping technique.

3.2 Practical Constraints

The model under discussion illustrates the basic idea of IP Fast Hopping mechanism and how this technique ensures hiding the destination of a client's traffic from external observers. The approach has a various possible implementations, each of them can have particular limitations, but in this section, we review basic constraints of our technique:

- If IP pool IP_v is not large enough, a botnet can start an attack on each IP address using masquerading of malicious traffic as legitimate data stream by IP spoofing technique. In this case, gateways redirect part of hateful traffic together with legal traffic to a protected server. Therefore, the IP pool, which is used for IP Fast Hopping, should be large enough to make such an excessive attack way too consuming in terms of resources and inefficient for possible attackers. Obviously, the method will be more efficient in IPv6 systems. In this case, IP pool can contain a thousands of addresses related to a number of different routers in the Internet.
- The method is not session-less; because of the fact that a client unique identification is part of a hopper function's domain.
- IP Fast Hopping requires clients' authorization since a hopper function (especially its domain and codomain) should be available only to legitimate clients. Therefore, the protocol is not applicable for publicly available Internet resources. In addition, our approach does not provide any native ways for secure and DDoS-resistant client's authorization.

3.3 Related Work

Our method should not be confused with IP hopping. In IP hopping approach, the address of a victim server is changed directly after detection of a DDoS attack to make the physical terminal unavailable for malicious traffic. In this case, this sort of replacement will have a much more far-reaching effect and affect DNS server too. On this condition DDoS attack may be initiated to the domain name instead of IP address of a victim to avoid such protection mechanism.

In addition, the community suggested several DDoS mitigation solutions based on dynamic IP address changing and network space randomization. For example, the paper[8] introduces DDoS defense mechanism based on the dynamic change of a server's IP address. Server IP address is changed according to a pseudo-random law known only to authorized clients. In comparison with IP Fast Hopping method, the proposed technique contains the following major differences: The IP address of the victim is changed only during an active DDoS attack on the server; the new IP address is assigned for all client sessions simultaneously on a relatively long time (suggested period is around 5 minutes); accurate time synchronization is required for the calculation of each next IP address since external timestamp is used.

Another paper[9] suggests a Network Address Space Randomization (NASR) technique that is intended to protect enabled networks against hitlist worms. This method presupposes that an address from a global IP address space pool is randomly assigned and this randomization suggested to be performed on protected server directly. So, a basic form of NASR can be implemented by configuring the DHCP server to expire DHCP leases at particular intervals[10]. Therefore, the scope of NASR implementation is limited to local regions.

4. Conclusions

We described novel approach for increasing DDoS and traffic eavesdropping resistance of Software Defined Networks. Our theoretical and basic experimental analysis has shown that IP Fast Hopping filters malicious traffic from incoming traffic of a victim server. The demonstrated technique suggests new addressing policy that based on randomizing of server's address. However, this new addressing policy does not require any global significant changes in the existing Internet architecture and can be implemented as software solution on client and server ends without impact on transit infrastructure.

5. Acknowledgments

This work was supported by the Ministry of Education and Science of the Russian Federation (grant agreement №14.574.21.0034 from 06/17/2014, the unique identifier applied research RFMEFI57414X0034).

6. References

1. Blessy R, Deepa AJ. A survey on network security attacks and prevention mechanism. Journal of Current Computer Science and Technology. 2015; 5(2):1–5.

2. Zargar ST, Joshi J, Tipper D. A survey of defense mechanisms against Distributed Denial of Service (DDoS) flooding attacks. Communications Surveys and Tutorials. 2013; 15(4):2046–69.
3. Prasad KM, Reddy ARM, Rao KV. DoS and DDoS Attacks: Defense, detection and traceback mechanisms – survey. Global Journal of Computer Science and Technology. 2014; 14(7):15–32.
4. Gupta BB, Joshi RC, Misra M. Distributed denial of service prevention techniques. International Journal of Computer and Electrical Engineering. 2010; 2(2):268–76.
5. Ankita P, Khatiwala F. Survey on DDoS attack detection and prevention in cloud. International Journal of Engineering Technology, Management and Applied Sciences. 2015; 3(2):43–7.
6. Krylov V, Kravtsov K. IP fast hopping protocol design. Proceedings of the 10th Central and Eastern European Software Engineering Conference in Russia, CEE-SECR'14; Moscow. 2014. p. 11.
7. Krylov V, Kravtsov K. DDoS attack and interception resistance IP fast hopping based protocol. 23rd International Conference on Software Engineering and Data Engineering, SEDE; New Orleans; 2014. p. 6.
8. Mittal P, Kim D, Hu Y-C, Caesar M. Mirage: Towards deployable DDoS defense for web applications. Networking and Internet Architectures, Cryptography and Security; 2012. p. 16.
9. Antonatos S, Akritidis P, Markatos EP, Anagnostakis KG. Defending against hitlist worms using network address space randomization. Proceedings of the 2005 ACM Workshop on Rapid Malcode, WORM'05; VA, USA: Fairfax. 2005. p. 1–11.
10. Mathi S, Lavanya M, Priyanka R. Integrating dynamic architecture with distributed mobility management to optimize route in next generation internet protocol mobility. Indian Journal of Science and Technology. 2015 May; 8(10):963–74.