Principles of Network Security Protocols Based on Dynamic Address Space Randomization

Vladimir Krylov and Kirill Kravtsov

Institute of Radio Electronics and Information Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod 603950, Russia

Abstract: To create network security solutions and protocols, we introduced IP Fast Hopping. Our approach is based on theory of convoluted multiaddress networks, which describes principles of dynamic network address space randomization. Our technique aimed to protect network nodes against various types of network threats by hiding of node's network address from malefactors. Existing networks are still vulnerable to attacks like Denial of Service despite a number of proposed defense approaches. Therefore, developing of a new alternate approach in this area is still actual problem. Our solution in this area can be easily deployed as a software solution without significant impact of existing network architecture. The main idea of our work is isolation of network nodes from a malicious traffic by the specific addressing policy where the network address is not unique identification of physical location of the node. This correlation should be dynamic and available only for legitimate terminals. In such case, a malefactor cannot acquire access to the protected server and initiate an attack.

Key words: Network security, Internet of Things, DDoS (distributed denial of service), network protocols, IP Fast Hopping.

1. Introduction

In this article, we discuss security challenges in networks based on the OSI (open systems interconnection) model and, above all, TCP/IP protocol stack. Such networks are vulnerable to various types of network attacks like traffic eavesdropping, IP Spoofing Attacks, Denial of Service etc. (see Ref. [1] for a more detailed classification). The current study focuses on protection against DDoS(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 malefactor generates a separate data stream to a victim. In addition, the suggested technique increases confidentiality of a communication session by hiding of the destination of this session.

A 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 have been proposed. 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. An increase in the number of terminals inevitably entails an increase in the flow of requests created, therefore, whichever level of server performance has 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

Corresponding author: Kirill Kravtsov, Ph.D. candidate, research field: network security.

more groups to be found [4, 5].

However, generally, we can see that a greater part of existing DDoS protection mechanisms are based on a wide range of traffic analysis and filtering algorithms. They use a reactive strategy, where, on the first stage, these methods analyze traffic, tries to detect active DDoS attack and after that filters malicious packets. In these approaches, the network 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 reaches the 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: resistance against DDoS attacks can be achieved via randomization of this tract by means of a specific addressing policy where a network address is only a temporary conditional identifier of a victim server. This technique is called convoluted multiaddress networks and is discussed in the next paragraph. In today's networks, there are a several practical solutions based on manipulations with network addresses (like tunneling protocols (IPsec) or Tor anonymity network). They are focused, primarily, on data protection and hiding traffic source/destination. In this paper, we demonstrate a general approach of network address manipulation that leads to boosting the robustness of networks and data against a wide range of different threats like DDoS attacks, traffic eavesdropping and analysis, etc.

In this paper, we describe theory of convoluted multiaddress networks and its implementation in TCP/IP networks, IP Fast Hopping. We demonstrate IP Fast Hopping Protocol, initial results of experiments and major practical constraints of our approach. We discuss Thing Lakes, the application of IP Fast Hopping Protocol in the Internet of Things paradigm.

2. Convoluted Multiaddress Networks

To demonstrate our network security approach, consider a mathematical model of network traffic traversing with decomposition of mapping of instantaneous traffic intensity from a network node with address x to a 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 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 incoming traffic (Fig. 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 $out(n): A_x^{in}(t) = A_y^{out}(t-n)$ ensures direct traffic translation from source x to receiver y with delay n .

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 j is related to receiver y : $out(j, y): A_x^{in}(t) = a(t, i, j)$.

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

Fig. 1 Model of traffic traversing in networks.

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 :

$$\begin{aligned} R(i, j) &= (i, hi(t, j)); out(j, y) : y \\ &= ho(t, j); hi(., .) = ho(., .) \end{aligned} \quad (1)$$

We call the solution (1) address hopping. 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 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:

$$A_y^{out}(t) = \sum_{x \in X} rout(n_x) A_x^{in} \quad (2)$$

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 breaks additivity (2) in the following way: traffic intensity from a particular ("legitimate") source x_l is transmitted to a particular receiver y without losses, but for all other sources $x \in X, x \neq x_l$, intensity of traffic targeted to node with address y will be decreased in K times. We can see that this task has a solution, when mappings are defined by the following way:

$$\begin{aligned} in(x, i) : a(t, i, j) &= A_x^{in}(t); R(i, j) = (i, hi(t, i)); \\ out(j, y) : y &= ho(t, j); \\ hi(t, k) &= ho(t, k) \forall (t, k) \text{ if } x = x_l \text{ else } hi(t, k) \neq ho(t, k) \end{aligned}$$

In fact, according to commutativity of the diagram of mappings, traffic from a legitimate source x_l 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 decreasing factor is defined by the following expression:

$$K = \frac{|T_m|}{|T_h|}, \text{ where } |T_m| \text{ is the cardinality of a set of}$$

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.

This abstract model has shown that application of the special addressing policy in a network provides a way to decrease intensity 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. 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.

3. IP Fast Hopping Approach

IP Fast Hopping technique [6, 7] is intended to make the real destination of a client's communication session invisible for all external terminals and, consequently, to prevent DDoS attacks and unauthorized access from illegitimate clients. IP Fast Hopping method is based on the model of convoluted

multiaddress networks and is an implementation of this model in TCP/IP networks. In IP Fast Hopping approach, a server acquires random IP address corresponding to each particular client at each time moment. After this method has been applied, for an external observer in a man-in-the-middle position, a communication session between the client and the server does no longer look like a packet stream between these two Internet terminals. Instead, the 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 among packets inside the stream.

IP Fast Hopping is similar to the principles of 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 bring noise into the 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 this 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.

3.1 Abstract Model of IP Fast Hopping

To illustrate the basic idea of applying convoluted multiaddress networks in TCP/IP networks, consider a system model, with a server with address s , a set of clients $C=\{c_1, \dots, c_l\}$, a subset of network address

space $IP_V=\{ip_1, ip_2, \dots, ip_N\}$ where $s \notin IP_V$ and a set of gateways $R=\{r_1, \dots, r_M\}$ where $M \leq N$. Each client $c_i \in C$ has a representation of a server's address s : s' or "initial address". Each message a from the whole set of messages A from client c_i to server s is 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 in three independent addressing subsystems: (1) subsystem (subnetwork) W_1 from client c_i to input point P_{in} of route p ; (2) subnetwork W_2 of route p ; (3) subnetwork W_3 from output point P_{out} of route p to server s . When message a is transmitted through a particular subnetwork W_z , a has a specific destination address IP_{dstW_z} . Obviously, $IP_{dstW_1}=s'$, $IP_{dstW_2}=x$, $IP_{dstW_3}=s$. To make the entire system model consistent, each W_z has function F_z that maps the destination address of message a in preceding W_{z-1} to the destination address in the current subnetwork W_z (Fig. 2).

We can consider the common client-server architecture in Internet networks as a system with only mapping $route(n)$ (Fig. 1), therefore the destination address of message a is always equal to s , which means that $P_{in}=c_i$ and $x=s$. So, our abstract system model (shown in Fig. 2) undergoes simplification as shown in Fig. 3. In this case, function F_z is trivial and, thus, all messages from c_i addressed to s' are 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, consequently, 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)

Fig. 2 Abstract system model.

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

$B=\{b_1, \dots, b_k\}$ can initiate a brute-force DDoS attack on address s (Fig. 3).

As has been reflected before, nowadays most of DDoS prevention techniques suggest installing firewalls and filtering solutions in a network between a set of clients C and victim server s . With reference to our model (Fig. 2), we can treat gateway P_{out} as a firewall that carries out a traffic analysis and filtering according to one of the 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 a 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, the system does not contain direct mapping $route(n)$ which ensures trivial traffic translation from client c_i to server s ; also, there is $s'=y$ 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' (local representation of server's address s) equals to the address of this gateway, all messages a targeted to the server with address s will achieve this gateway. 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-initiator of this packet, ID_{c_i} , as a private key. The system model under consideration has a pseudo-random function $H(t_a, ID_{c_i})=n \in [1; N]$ that

determines valid virtual address ip_n of the server for particular message a . This function is regarded as a hopper function because it defines the hopping of the IP address of the protected server in convoluted multiaddress networks. Since 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 W_2 . Therefore, $F_2=H(t_a, ID_{c_i})$. An address subspace IP_V has a number of disjoint subsets IP_V^m , where $m \in [1; M]$ and $IP_V = \bigcup_1^M IP_V^m$. Each address from IP_V^m is related to corresponding gateway r_m , thereby W_2 has M paths p_m for messages from client c_i (Fig. 4). Gateway r_m validates all incoming packets and maps destination address of these messages IP_{dst} on address in W_3 by using function F_3 (NB: zero address means here that the message should be treated as malicious):

$$F_3(t_a, ID_{c_i}) = \begin{cases} IP_r, & \text{if } IP_{dst} = ip_n \\ 0, & \text{if } IP_{dst} \neq ip_n \end{cases},$$

where

$$n = H(t_a, ID_{c_i}) \quad (3)$$

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 among independent terminals in the network.

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 various

Fig. 4 IP Fast Hopping model.

possible implementations, each of them can have particular limitations, but in this section, we review major constraints of our technique:

- If IP pool IP_V is not large enough, a botnet can start an attack on each IP address using malicious traffic masquerading as a legitimate data stream by using 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 thousands of addresses related to a number of different routers in the Internet.

- The method is not state-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 Implementation

IP Fast Hopping has a variety of possible implementations. In this section, one of simplest approaches is described. In the simplest case, the timestamp of TCP header of each packet is used as timestamp t_a for hopper function $H(t_a, ID_{c_i})$. Also, unique identification ID_{c_i} of the originator of packets is defined during authorization. To make this approach consistent with other network protocols, DNS servers should contain 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 despite the fact that our technique requires adding several network terminals (gateways), which are responsible for calculation, validation and changing the destination IP address of packets. This additional functionality can be added to network routers. Therefore, as an example of implementation, we implement IP Fast Hopping mechanism as a kernel module of OS GNU/Linux. In this case, installation of this module on GNU/Linux-based routers is enough to deploy the suggested system.

Linux kernel contains built-in firewall Netfilter, which is responsible for packet filtering and forwarding according to the rules predefined 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 in the destination field of outgoing packets and in the 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 in 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 the index of the correct address in 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 for 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 value in 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 address by one of virtual addresses according to the current value of the hash-function.

3.4 Initial Experimental Results

The described basic implementation of IP Fast Hopping approach has been validated on a 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 (the implemented Netfilter module was installed on both machines), 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 (Fig. 5). For generating DDoS attack we used third-party application, LOIQ.

Even with such minimal testing, it is easy to see that application of IP Fast Hopping methods ensures filtering of malicious traffic stream from the incoming traffic of a protected server.

3.5 IP Fast Hopping Protocol

As was shown above, IP Fast Hopping is easily deployable software solution. To apply this approach for protection of server s , network should contain a set of routers $R=\{r_1, \dots, r_M\}$, which support IP Fast Hopping rules (for instance, by installing the discussed Netfilter module) and have a set of addresses IP_V^m . Depending on hopper function, the same set R and the same IP address pool IP_V can be used to protect several independent servers. According to the fact that our approach is not state-less, the proposed basic implementation requires installing of a special control module, IPFH Controller, which is responsible for managing communication sessions established in accordance to IP Fast Hopping. In this section, we discuss IP Fast Hopping protocol to

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

demonstrate how such system can operate (especially, connection establishing and message transmission between client and server). The message sequence chart of such protocol is shown on Fig. 6.

The suggested implementation of IP Fast Hopping modifies common process of establishing a connection between a client and a server. Here, DNS server contains IP address of an authorization server instead of the protected server s . So, to acquire access to s , client c should prove his legitimacy on the authorization server. After successful client

authorization, this server requests handshake between client's IP Fast Hopper (the special utility installed on client's terminal, which includes enhanced Netfilter) and the IPFH Controller. During such handshake, the controller associates unique identification ID_{c_i} with this particular client and selects pool of addresses IP_V for address hopping. The controller sends these parameters of address hopping to client's IP Fast Hopper and to all gateways $R=\{r_1, \dots, r_M\}$, each of them is responsible for address hopping among related subset IP_V^m . After this stage, IP Fast Hopping

Fig. 6 Message sequence chart of IP Fast Hopping Protocol.

handshake is completed and client c can now acquire access to the protected server. Therefore, authorization server redirects client's session to s' , initial address of server s .

Client c communicates with the server using static initial address s' . At the same time, IP Fast Hopper captures all packets with such destination address and performs destination address hopping. This module calculates the index $n \in [1; N]$ of IP address from IP address pool IP_V using pseudo-random function $H(t_a, ID_{c_i})$ and puts the result address ip_n into the destination address field of the packet. After that, the original packet with modified destination is sent to the network. According to existing routing protocols and the fact that $ip_n \in IP_V^m$, the message is transmitted to router r_m . On this router, IP Fast Hopper module validates destination address of the packet using procedure (2). If the destination is valid, the packet is transmitted to protected server with address s . The same operations, but in reverse order and for the source address field, is performed for all responses of the server to the client.

The discussed protocol provides a way to hide a server behind a set of independent network gateways based only on manipulations with address fields of packets, and such logic can be implemented as a software module. The protocol does not affect existing network architecture and can be integrated without far-reaching impact on it.

3.6 Bidirectional IP Fast Hopping

As can be seen from the preceding paragraphs, IP Fast Hopping is an adaptation of convoluted multiaddress networks for TCP/IP client-server architectures. This method provides a way to filter malicious traffic targeted to a protected server. However, it does not conceal the network address of clients, so a botnet can initiate an attack on a client. In a classical client-server scenario, this limitation is not critical, but in a number of practical cases, each peer of communication is important from a network

security perspective, so we discuss protection of peer-to-peer networks. Such practical cases are tackled below (for example, inter-lake network security in Thing Lakes architecture of the IoT). The need for such protection requires certain modifications in IP Fast Hopping, which eventually has to transform into Bidirectional IP Fast Hopping. Coming back to the abstract model of the suggested approach, the bidirectional modification randomizes a client's addresses c_i in subnetwork W_2 . In which case, a malefactor terminal located in subnetwork W_2 has information neither about a server's address, nor about a client's address.

To describe the basic idea of Bidirectional IP Fast Hopping method, we should consider the mirrored model from Fig. 4 with a subspace of network address space $IP'_v = \{ip'_1, ip'_2, \dots, ip'_N\}$ where $c_i \notin IP_V$ for $\forall i$; a set of gateways $G = \{g_1, \dots, g_{M'}\}$ where $M' \leq N'$; a set of representations of a server's addresses $Y = \{y_1, \dots, y_{M'}\}$ where $y_{m'}$ is the network address of gateway $g_{m'}$; a pseudo-random function $H'(t_a, ID_{c_i}) = n' \in [1; N]$ and c'_i is a representation of the address of client c_i . In the bidirectional approach (Fig. 7), gateway P_{in} is replaced by a set of gateways G . Given address hopping is applied for source address of message a , each subnetwork W_z has a pair of hopping functions F_z and F'_z and, in a common case, they are not equal.

In bidirectional IP Fast Hopping, each messages a targeted to server with address s achieves subnetwork W_2 through a gateway $g_{m'}$ according to pseudo-random function $F_{W_1} = func(t_a)$. F_{W_1} defines address of a gateway $g_{m'}$, which performs hopping of the source address of this particular packet between addresses from subsets $IP_{V'}^{g_{m'}}$ of set IP_V . Gateway $g_{m'}$ applies a pair of hopping functions (F_2, F'_2) for each packet a , where $F'_2 = H'(t_a, ID_{c_i})$. After that packet a in subnetwork W_2 acquires destination address $IP_{dst} = ip_n$ and source address $IP_{src} = ip'_n$, where $ip'_n \in IP_{V'}^{g_{m'}}$. Gateway r_m maps the source address

Fig. 7 Bidirectional IP fast hopping.

Fig. 8 Thing lakes architecture.

of valid message a according to function F'_3 :

$$F'_3(t_a, ID_{c_i}) = \begin{cases} c', & \text{if } IP_{dst} = ip_n, \text{ where } n = H(t_a, ID_{c_i}) \\ 0, & \text{if } IP_{dst} \neq ip_n \end{cases}$$

In IP Fast Hopping the destination address of responses from the server to clients c_i is always equal to c_i . But in the bidirectional approach, the destination address is defined by the same hopper functions F'_2 and F'_3 , but in reverse order: $IP_{dst} = ip'_{n'}$ in W_2 and, in W_1 ,

$$IP_{dst} = \begin{cases} c_i, & \text{if } IP_{dst} = ip_{n'}, \text{ where } n' = H(t_a, ID_{c_i}) \\ 0, & \text{if } IP_{dst} \neq ip_{n'} \end{cases}$$

As a result, in the considered model, communication session becomes a set of random communication sessions between a set of independent peers.

4. Related Work

Our method should not be confused with IP hopping. IP hopping implies that 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, a DDoS attack may be initiated on the domain name instead of the IP address of a victim to avoid IP hopping protection mechanism.

In addition, researches have suggested several DDoS mitigation solutions based on dynamic IP address changing and network address space randomization. For example, the paper [8] introduces a DDoS defense mechanism based on the dynamic change of a server's IP address. A server's 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 for a relatively long time (suggested period is around 5 minutes); accurate time synchronization is required for the calculation of each next IP address since an external timestamp is used.

Another paper [9] suggests a NASR (network address space randomization) 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 is meant 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. Therefore, the scope of NASR implementation is limited to local regions.

5. Thing Lakes

In this paper, we introduce the secure architecture of the Internet of Things networks called Thing Lakes. Thing Lakes is designed to ensure protection of such networks against DDoS attacks and traffic eavesdropping by applying IP Fast Hopping logic and convoluted multiaddress networks principles within

the Internet of Things paradigm.

According to the article [10], one of the major characteristics of the Internet of Things is a unique Internet Address by means of which each connected physical object and device are identified, and therefore they acquire an ability to communicate with one another. However, as has been pointed out above, when each thing has a unique and globally accessible address, this architecture is vulnerable to various types of network attacks as any other network would be. In our work, we suggest applying convoluted multiaddress networks within the Internet of Things paradigm to increase the IoT resistance against network attacks and intrusions. Researches suggest that IPv6 is the most preferable addressing protocol for the Internet of Things installations [11, 12]. Therefore, in our work, we focus on applying IP Fast Hopping (an implementation of convoluted multiaddress networks in TCP/IP networks) in the IoT architecture.

In our paper, we discuss security of the cloud centric Internet of Things [13]. In this case, the IoT can be divided into three levels: a level of things, a cloud level and an application level. This work focused on security challenges only on two first levels here and it is assumed that protected access of authorized clients is ensured. In this case, there can be two types of attacks: interlayer (when an attacker tries to prevent communication between levels) and intra-layer. Commonly, all things are connected to a common shared network and the IoT structure here is a logical structure above a physical network. Taking into account very limited performance of things, the level of things is highly vulnerable to network attacks such as DDoS. Such a problem can be addressed by using trusted private networks for intra-layer installations. In any case, interlayer communication is being performed through shared global networks. Therefore, the level of things should be still protected against malicious traffic from sources outside of things networks and a communication channel

between a level of things and a cloud should be protected against intrusions and other types of destructive actions from illegitimate terminals.

It is also worth a mention that the level of things is not homogeneous. Nowadays, there is a wide range of different types of smart devices and other “things”, many of them having their own communication and control protocols, and a user interface. Integration of such diversity into the IoT requires developing gateways [14]. Furthermore, some researchers suggest (for example, see Refs. [15-17]) installing security gateways to increase security of IoT solutions. Thus, IoT can be considered as a hierarchical architecture (Fig. 8) with different clusters on a lower level connected to a shared network via gateways. These clusters can be treated as independent modules of the IoT, each of them having a gateway as a single point, which ensures connection with the outside. These special nodes are responsible for protection of clusters against malicious traffic initiated by external network terminals. We call each cluster of things a Thing Lake. In this paper, we describe security solutions for inter-lake (a communication channel between lakes and a cloud) and intra-lake (each single thing in a lake) levels of such an architecture.

In terms of inter-lake network security, there is a network of gateways and cloud servers. Here a network attack scenario (like DDoS) has the same form as shown in Fig. 3. But each node (not only a particular lake or cloud) should be protected against DDoS attack, so we suggest applying the Bidirectional IP Fast Hopping method (Fig. 7) to protect a communication channel between gateways (and the corresponding lakes) and a cloud.

In terms of intra-lake network security, we should take into account the following facts: (1) things can be installed long before network infrastructure is set; (2) things have very limited power supply and calculation resources. On the one hand, IPFH does not contain any heavy cryptographic algorithms. Since illegitimate clients do not have information about a valid address

in a particular period, legitimacy of each client is verified by TCP/IP suite.

On the other hand, the described method (including its bidirectional modification) requires installation of special network equipment (gateways that are responsible for address hopping). However, such gateways are necessary for IPFH only in case of DDoS protection because they need to process a big amount of packets during a brute-force DDoS attack. Obviously, taking into account very limited performance of things, their low price and big quantity, we consider protection against traffic eavesdropping and intrusion more important than defense against DDoS. Randomization of traffic routes is enough to achieve this goal. So, on the level of things we suggest implementing a simplified Bidirectional IP Fast Hopping, where gateways r_m and $g_{m'}$ are implemented as local software modules in each thing. In this case, a set of IP addresses can be configured for one network interface of a thing. After that, Lake is still vulnerable to DDoS attacks from Lake's network infrastructure, but illegitimate clients cannot acquire access to things or their traffic due to the fact, that only one network address is valid at each particular time for each particular packet and a malefactor cannot predict valid IP address of the victim thing at future time moments.

6. Conclusions

The demonstrated network addressing policy provides an easy way to protect terminals from DDoS attacks and traffic eavesdropping. In our work, we considered these common network threats from alternative point of view. Usually, researches work on this problem as a security issue of single direct network tract (on physical level this tract can contain a number of sub routes). In such conditionals, we would have to find a way to defend ends of this pipe (by creating a firewall) and/or a way to protect legitimate data stream in such tract (by creating a filtering solution against malicious traffic). Instead, our goal is

to isolate and hide such virtual path from a malefactor. Obviously, it is practically impossible on physical level, but a network is not only physical infrastructure, it is also a logical space of interconnected nodes and each node has an address. The described idea of convoluted multiaddress networks aims to make these nodes and the entire space fluid. We can tell that on this abstract level, network becomes vulnerable to attacks like DDoS or eavesdropping only when every node can predict future location of all other nodes. IP Fast Hopping method is intended to make such prediction impossible or very difficult for all undesired nodes in TCP/IP networks. In our paper, we demonstrated a basic implementation in form of easily deployable software solution. Our method can be applied in the existing networks without impact on global infrastructure of the Internet.

Acknowledgment

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).

References

- [1] Blessy, R., and Deepa, A. J. 2015. "A Survey on Network Security Attacks and Prevention Mechanism." *Journal of Current Computer Science and Technology* 5 (2): 1-5.
- [2] Zargar, S. T., Joshi, J., and Tipper, D. 2013. "A Survey of Defense Mechanisms against Distributed Denial of Service (DDoS) Flooding Attacks." *IEEE Communications Surveys & Tutorials* 15 (4): 2046-69.
- [3] Prasad, K. M., Reddy, A. R. M., and Rao, K. V. 2014. "DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms—A Survey." *Global Journal of Computer Science and Technology* 14 (7): 15-32.
- [4] Gupta, B. B., Joshi, R. C., and Misra, M. 2010. "Distributed Denial of Service Prevention Techniques." *International Journal of Computer and Electrical Engineering* 2 (2): 268-76.
- [5] Ankita, P., and Khatiwala, F. 2015. "Survey on DDoS Attack Detection and Prevention in Cloud." *International Journal of Engineering Technology, Management and Applied Sciences* 3 (2): 43-7.
- [6] Krylov, V., and Kravtsov, K. 2014. "IP Fast Hopping Protocol Design." In *Proceedings of the 10th Central and Eastern European Software Engineering Conference in Russia*, Moscow.
- [7] Krylov, V., and Kravtsov, K. 2014. "DDoS Attack and Interception Resistance IP Fast Hopping Based Protocol." Presented at the 23rd International Conference on Software Engineering and Data Engineering, SEDE 2014, New Orleans.
- [8] Mittal, P., Kim, D., Hu, Y. C., and Caesar, M. 2012. "Mirage: Towards Deployable DDoS Defense for Web Applications." arXiv.
- [9] Antonatos, S., Akritidis, P., Markatos, E. P., and Anagnostakis, K. G. 2005. "Defending against Hitlist Worms Using Network Address Space Randomization." In *Proceedings of the 2005 ACM Workshop on Rapid Malcode*, Fairfax, VA, USA.
- [10] Lake, D., Rayes, A., and Morrow, M. 2012. "The Internet of Things." *The Internet Protocol Journal* 15 (3): 10-9.
- [11] Jara, A. J., Ladid, L., and Skarmeta, A. 2013. "The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities." *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)* 4 (3): 97-118.
- [12] Jara, A. J., Varakliotis, S., Skarmeta, A. F., and Kirstein, P. 2014. "Extending the Internet of Things to the Future Internet through IPv6 Support." *Mobile Information Systems—Internet of Things* 10 (1): 3-17.
- [13] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. 2013. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems* 29 (7): 1645-60.
- [14] Desai, P., Sheth, A., and Anantharam, P. 2014. "Semantic Gateway as a Service Architecture for IoT Interoperability." arXiv.
- [15] Vucinic, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., and Guizzetti, R. 2014. "OSCAR: Object Security Architecture for the Internet of Things." Presented at the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2014).
- [16] Tao, R., Yang, S., Tan, W., and Zhang, C. 2013. "Secure Gateway of Internet of Things Based on AppWeb and Secure Sockets Layer for Intelligent Granary Management System." Presented at the 7th IFIP WG 5.14 International Conference, CCTA 2013, Beijing, China.
- [17] Intel, 2014. "Intel Gateway Solutions for the Internet of Things." [Online]. Available: <http://www.mcafee.com/us/resources/solution-briefs/sb-i-nel-gateway-iot.pdf>. [Accessed 31 January 2015].