Securing the Private Realm Gateway

Hammad Kabir, Jesús Llorente Santos, Raimo Kantola
Department of Communications and Networking
Aalto University
Helsinki, Finland
{hammad.kabir, jesus.llorente.santos, raimo.kantola}@aalto.fi

***Abstract*—The traditional mechanisms to traverse Network Address Translators (NAT) do not scale well to battery powered mobile-hosts: the majority of Internet users today. Private Realm Gateway (PRGW) aims to replace NATs at network edges and overcome the drawbacks of the NAT traversal mechanisms. The solution does not require changes in end-hosts or protocols, and hosts in the private realm can remain globally reachable without polling. PRGW handles incoming connections based on domain resolution of the served hosts. Incoming DNS queries create connection state in PRGW for subsequent packet forwarding. The connection state provides means for access control on the Internet-originated flows. This paper analyses the security of PRGW and introduces mechanisms that protect the served hosts and networks against Internet-borne attacks, in particular: address spoofing and Distributed Denial of Service (DDoS). The paper contributes to establish PRGW as an incrementally deployable network function that offers light-weight NAT traversal and protects the private realm against the inherent Internet threats.**

Keywords— Security; Gateway; NAT Traversal; PRGW; DNS; NAT; Denial of Service; DDoS; Internet threats; Network;

I. INTRODUCTION

According to ITU-T, mobile broadband subscriptions have reached 3.2 billion individuals connected to the Internet [1]. This growing number of mobile users raises challenges for the Internet and further aggravates the IPv4 address space depletion problem. The adoption of NAT at network edges alleviated the IPv4 address space exhaustion at the cost of introducing the reachability problem, which prevents the Internet hosts from unilaterally initiating a connection to hosts in the private realm. The mobile hosts typically reside in the private address space; however the IETF recommended methods for NAT traversal [2] scale poorly to battery-powered hosts [3] and communication applications: 1) device has to periodically wake-up to keep its NAT binding alive; and 2) session setup requires exchanging hundreds of overhead messages per application that seeks global reachability, leading to extra power consumption on the device and delays in the session setup.

In [4], we address these drawbacks of the classical NAT traversal mechanisms and propose the Private Realm Gateway (PRGW) solution. The solution does not require any changes in end-hosts, i.e. clients and servers in the private network can stay globally reachable without applications having to run the code for NAT traversal or to maintain their NAT binding. PRGW can be deployed either as a standalone replacement of NATs or as a component of a customer edge switching [5-6] node, at the network edge.

However, as PRGW makes end hosts reachable in the private realm, it will open new opportunities for the hackers to target the private hosts and their network. The increasing reliance of users on their smart phones and mobile apps have presented mobile networks and their hosts as lucrative targets to Internet hackers. As a result, they are subject to a wide variety of threats possible in the Internet.

The paradigm of Internet security can be viewed as an arms race between attackers and defenders. The possibility of source address spoofing, distributed denial of service (DDoS), traffic floods and network/port scans is inherent in the Internet. Today, hackers often abuse free services, e.g. Google DNS, and employ compromised hosts as reflectors/amplifiers in launching their attacks. The outcome of these attacks may lead to excessive network usage, computing downtime, service unavailability, and ultimately waste of human capital [7]. Societies heavily rely on the Internet, and use it for mission-critical activities. Therefore, the networks shall deploy mechanisms that protect their hosts and resources against Internet-borne attacks, in particular source address spoofing, network scans and DoS, which are often used as launch point for more advanced attacks. Consequently, our threat model in the paper spans to the above attack types.

In this paper, we seek to provide mechanisms that protect PRGW and make it a feasible function in modern IP-networks. As a result, PRGW emerges as a network function that besides overcoming the drawbacks of the NAT traversal solutions [4] is hardened against the inherent Internet threats, i.e. traffic floods, source address spoofing and DoS. The mechanisms adhere to the basic principles of PRGW design and limit all the changes to network edges. This keeps the deployment of PRGW simple, as the upgrade only takes place at the edge nodes, and can be performed one network at a time. We argue that it may possible to take a clean-slate approach, and design a better architecture free of any security weaknesses, at the cost of a huge deployment difficulty. Contrary to this, we take the deployment constraints as the corner stone of our work.

The rest of the paper is structured as follows. Section II discusses the related work. Section III presents vulnerabilities of PRGW in handling the inherent Internet threats. Section IV establishes the basis of our security solutions. Section V and VI describe the security mechanisms and heuristics. Section VII evaluates the security. Section VIII presents the discussion, and Section IX concludes the paper.

II. RELATED WORK

The introduction of NAT at network edges extended the IPv4 address space lifetime. NAT effectively hides the private realm, such that hosts in the private network share a set of public IP address(es) towards the Internet. By default, NAT devices allow outbound connections towards the Internet and create a state to admit subsequent inbound packets of the flow. The connection state enables address translation on packets traversing across the

ISBN 978-3-901882-83-8 © 2016 IFIP

public and private realm, and at minimum contains a 5-tuple: IP and port pair towards public Internet; IP and port pair in the private network; and the transport protocol. Inbound packets that do not have a state in NAT are dropped [8]. As a consequence, connection attempts from the Internet hosts towards the private realm fail, raising the reachability challenge. The current NATs thus employ static port forwarding, or complex NAT traversal mechanisms to admit new connections in the network.

The traditional NAT traversal mechanisms do not scale well to mobile devices [3, 4]. While, static forwarding in NATs can be vulnerable to ills of the Internet, in particular: spoofed flows, network/port scans, and traffic floods from botnets.

Many proposals have attempted to tackle address spoofing and DoS floods. Ingress filtering [9] is a typical solution to the problem of source address spoofing. However, the solution has not been globally adopted, possibly because costs and benefits of ISPs are not well aligned: the receiver or its ISP benefit from spoofing elimination while the other entities bear the expense of configuring and executing the ingress filtering.

IETF proposed the use of SYN Cookies [10] during TCP handshake, to protect the victim host against resource exhaustion from spoofed SYNs. SYN cookie delays the allocation of TCP resources in the host until the sender is verified as *non-spoofed*.

Besides eliminating spoofing, IP puzzles [11] dis-incentivise spurious connection attempts from hosts. The mechanism slows an aggressive host, by requiring the sender to process a received challenge with certain computational effort before it can establish a connection. Similarly, Hop-Count Filtering [12] aims to protect against SYN floods, by comparing the statistics of the received traffic with traffic observed during normal periods. However, these techniques are not in wide use.

Today, an advanced attacker often tricks a large number of hosts to unknowingly participate in launching a DDoS. The *compromised* hosts are mostly bot controlled by the hacker, in a master-slave configuration. Networks typically detect attacks using a set of security approaches, categorized into: Signature detection, Anomaly detection, or a hybrid of both approaches. Upon detecting a DDoS, DoS mitigation proposals typically react by rate limiting the accepted traffic [13]. While it affects the legitimate traffic as well, trace-back techniques are used to locate the malicious entities. An identified attacker is blacklisted and eventually filtered in the admitted traffic.

The research in [14] leverages this understanding of network security to propose a cooperative Feecod architecture. Under this architecture, when a host detects DoS, i.e. from overloading of its resources, the edge router of its ISP rate limits the admitted traffic, so that the total workload for the victim is below its upper bound. A log of each forwarded packet is then sought from the outbound edge to ascertain if no attack originated from its network, upon which the rate-limit is removed. The architecture however requires many changes in end-hosts, as well as in the sender and destination networks, detrimental to its adoption.

The research in [15] tackles DoS through an overlay network that registers the inbound requests before forwarding them to the destination. The proposed indirection infrastructure aims to tackle DoS using P2P networks. The paper in [16] presents various server specific DoS mitigation techniques that require changes in end hosts.

Mobile networks rate availability as the top concern due to high volume of DoS and network/port scans, and typically rate-limit or reset the connections from aggressive hosts [13, 17].

PRGW presents an architecture to overcome the drawbacks of classical NAT traversal solutions. It follows the behaviour of NATs for outgoing connections, such that private hosts connect to Internet sharing a set of public IP addresses. But unlike NATs, it allows Internet hosts to unilaterally initiate connection towards the private hosts using a circular pool of public IP addresses (CPPA). Upon receiving a DNS query for fully qualified domain name (FQDN) of the private host, it temporarily allocates a public IP address from the pool to represent the host in the Internet and creates a temporary *half connection state* that allows forwarding of the subsequent inbound flow to the private host. The client typically initiates the data flow, upon resolving the domain. Upon receiving the first inbound packet from the client, PRGW creates a *full connection state* for the flow and returns the allocated public address to CPPA for future allocations. In this manner, by dynamically assigning an address from CPPA, PRGW protects the private network from direct exposure to the Internet, compared to port forwarding possible in NATs. The half connection state in PRGW applies *endpoint independent filtering* [18] relative to the client, while in the full connection state the filtering is upgraded to *address and port dependent* relative to the client. Since PRGW does not require any changes in end-hosts or remote edge, it avoids the deployment challenge.

III. Security Vulnerabilities

This section analyses the impact of Internet's weaknesses in handling address spoofing and network floods on PRGW. We argue that PRGW does not introduce any explicit security weakness in comparison to the current Internet model, or how NAT allows inbound connections. Like NATs, it also filters to drop the packets that do not have an ongoing connection or a valid state. In addition, 1) the CPPA prevents the private hosts from direct exposure to the Internet, compared to static NATs; and 2) hosts in the private realm are only accessible through their FQDNs, which provides defence-in-depth, in combination with a set of mechanisms that we will introduce in this paper.

Fig. 1 identifies a set of hazardous scenarios where PRGW and the hosts located behind it could be vulnerable to Internet abuses, i.e. in the absence of security mechanisms.

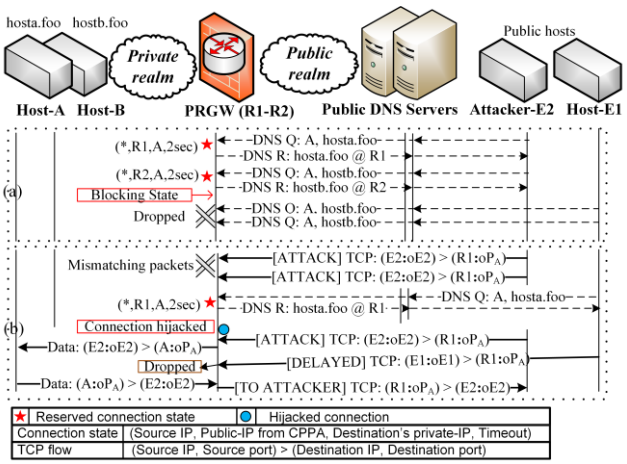

Fig. 1. RGW vulnerabilities to inherent Internet threats

Here we adopt the filtering classification developed in RFC 4787 related to the servers, and use the terms in relation to the clients.

1) Denial of Service: Fig. 1.a illustrates DoS attack from an *aggressive* Internet host that forces PRGW into *blocking* state, by issuing flood of DNS requests for the served hosts. In this state, the CPPA is depleted due to allocation of all its addresses, leaving PRGW unable to accept new incoming connections. The exhaustion of circular pool could also happen due to poor provisioning of the public IP address pool.

2) Connection hijacking: The attacker in Fig. 1.b floods an address R_1 of PRGW. PRGW would drop any packet unrelated to an ongoing connection or valid inbound state. However, on the event that a public host initiates a connection and address R_1 is allocated, there is a window of opportunity when the attacker can claim the connection state. This will result in DoS to the host that originally requested access to the service. Unfortunately, IP address filtering is not a fail proof solution due to the possibility of source address spoofing in the Internet.

IV. Principles Of Security Mechanisms

Attackers often exploit the best effort nature of the current Internet to launch attacks. Disguising under a spoofed identity, the attackers can successfully inject the traffic in the destination network and yet escape the network auditing. Therefore, the key to improve Internet security comes from deploying mechanisms that eliminate spoofing, authenticate the sender, detect malicious hosts, thwart hijacking attempts and thereby grant access only to the legitimate hosts. We define that PRGW must comply with the following principles to tackle the inherent Internet threats:

1) Flow acceptance must be limited to verifiable sources to tackle address spoofing and prevent resource exhaustion.

2) UDP flow initiations are admitted only after a connection has been signalled through a secure channel e.g. SIP(S) [19].

3) To favor deployment, security algorithms and operations shall not require changes to end-hosts, protocols, or application.

4) Under the network stress, resource access should be granted based on the source reputation.

V. Preventing DNS Abuse/Exploitation

PRGW allows unilateral connection initiation to the private realm using CPPA. Since CPPA relies on the inbound domain resolutions, the architecture of PRGW carries a DNS leaf node that is authoritative for the domains located in its private realm.

The state of the art with DNS is such that it uses UDP as transport protocol for majority of its operations. As connection-less protocol, UDP is open to possibility of address spoofing. Attackers often exploit this vulnerability to launch DNS floods, and yet avoid the network audits. Alternatively DNS floods may originate from non-spoofed hosts, under bot control. In addition hackers often use freely-accessible open DNS resolvers, such as Google DNS, as DNS reflectors in launching their attacks.

PRGW is susceptible to this abuse of DNS that can lead to exhaustion of the CPPA resources. To trace *aggressive* host, the current practice in public name servers is not to serve recursive domain requests. As a result, source address of the actual DNS resolver is revealed to the destination. However, the possibility of address spoofing hinders the ability of the destination, i.e. in our case PRGW, to protect itself from DNS floods initiated by the *invisible* attacker.

The resource [20] describes best practices and existing state of the art in the DNS security. Among others, it recommends DNS resolvers/servers to rate limit the domain requests from a source, handle malformed packets, filter requests to not-hosted domains, apply ingress filtering, detect dictionary DNS attacks from hackers i.e. scanning their targets, use of DNSSEC, access control lists (ACL) to filter DNS requests from un-allocated or reserved address spaces, and to drop domain requests originated outside of its network to avoid becoming a DNS reflector.

While these recommendations aim to improve the Internet's resilience against DNS abuses, the ultimate outcome depends on their global adoption by all the network administrators and operators. Realizing this, we attempt PRGW security against the DNS abuses and exploitations by defining set of heuristics and mechanisms, limiting all the changes to the network edges.

A. DNS Relay

We implemented DNS-Relay as a frontend to protect PRGW from direct exposure to the Internet. This is to prevent the CPPA exhaustion from malicious domain resolutions, e.g. inbound DNS floods and spoofed requests. Under this model, PRGW is protected by virtue of *delegating* the DNS security to its ISP.

The DNS relay implementation draws upon the use of DNS reverse proxies in ISP networks and security solutions that aim to secure networks against DNS abuses. In our implementation, we leverage this approach such that the DNS relay forwards an incoming domain request to PRGW and identifies the original sender in the DNS extensions or additional records. The sender tuple identifies: source IP and source port, besides the transport protocol and transaction-ID of the inbound query message. This allows PRGW to identify the original sender, and thus apply its security mechanisms, such as the address allocation model and name server classification. These mechanisms are defined in the subsequent sections. The corresponding DNS response message from PRGW is forwarded by DNS Relay to the actual source, after removing the sender-identification tuple.

The mechanism only requires a few alterations in the edge network, i.e. the ISP name server forwards the inbound domain queries with DNS source information to PRGW. We argue that the changes in the edge network can be motivated by benefits possible from adoption of PRGW, e.g. deployment of servers in private address space, and less-complex session setups. But we consider these aspects beyond the scope of this paper.

The delegation of security to a dedicated DNS-Relay element offers multiple opportunities: 1) it lessens the load of executing the complex DNS security algorithms from PRGW; and 2) the dedicated relay element can independently leverage the existing state-of-the-art and future research in DNS threat detection, to serve the PRGW with legitimate traffic only. As a result, PRGW stays protected against DNS attacks and can allocate the CPPA resources to legitimate hosts.

B. Name Server Classification

When the aforementioned DNS Relay is in attack detection phase, and has not mitigated the DNS attack yet, it is possible that some share of DNS flood is received at PRGW. To prevent the consequent resource depletion, PRGW leverages from the classification of external name servers and allocates the CPPA resources following an Address Allocation Model.

Under this model, PRGW classifies the external DNS servers into: whitelist, greylist and blacklist. Servers on each list are treated differently in PRGW and are promoted/demoted in the classification dynamically, based on the influx of *attack* traffic.

Whitelisting can be based on business contracts and service level agreements (SLAs) between service providers, where the networks that seek *priority* access meet a set of pre-conditions. A whitelist server can meet the specific SLA, by employing the best DNS practices, e.g. active ingress filtering of DNS requests originated in its network, and disabling recursive resolution for external sources. The DNS resolver can also transport domain queries towards PRGW over TCP connection. This eliminates the possibility of spoofing in DNS requests, and on the event that an attack is reported it enables tracing an *aggressive* host back to its network. The terms of whitelisting can be agreed in peering agreements between mobile operators, administrators of the ISPs, or trusted networks, and may stress the networks to employ mechanisms such as DNS/TCP, DNSSEC and ingress filtering to receive whitelist/preferred access.

The whitelist servers are specifically configured in PRGW. By default, the rest of the name servers are *greylist*. This also includes open DNS resolvers and name servers that are freely accessible to Internet hosts, and often serve as DNS reflectors in launching DoS. A greylist name server is therefore offered less resources in PRGW than a corresponding whitelist server.

PRGW actively maintains these lists based on the influx of attack traffic. A name server is demoted to a lower category if states reserved by it repeatedly expire in PRGW. A state expires in the PRGW if it is not claimed by an inbound flow in time T_0. When the state expiration rate for a name server meets threshold R_T, the server undergoes a time penalty T_D in demoted category. A name server that repeatedly exceeds its SLA is blacklisted for time T_B, during which it is barred from accessing the circular pool resources.

C. Circular Pool Address Allocation Model

The CPPA address allocation model responds to an incoming DNS query based on the circular pool load conditions. The model rate limits the number of simultaneous states reserved to a DNS server or for a private host, and manages total allocations of CPPA such that DNS requests from multiple greylist servers only take a portion of the circular pool. For this, the address allocation model operates in conjunction with the name server classification. The model primarily attempts to tackle the DNS floods from *less* secure greylist servers. By prioritizing whitelist servers in address allocation over greylists, the model ensures that whitelist servers always have preferred access to PRGW, particularly under the attack/load conditions.

VI. Filtering Malicious Interent Flows

Internet hackers distribute malicious packets, initiate traffic floods and perform network/port scans to launch their attacks. A hacker can either employ a spoofed identity or hire bots from bot-rental business to launch these attacks. In this section, we introduce a set of mechanisms that attempt to ensure that only a legitimate host gains access to the private realm.

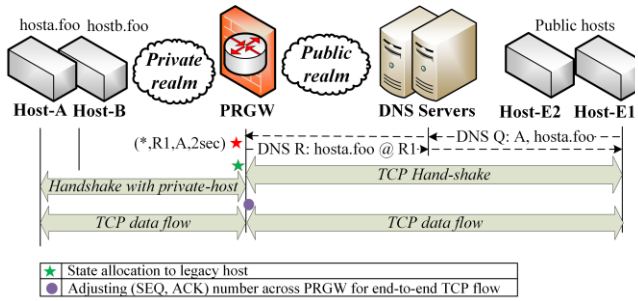

Fig. 2 TCP-Splicing in PRGW

A. TCP-Splice Mechanism

The mechanism ensures that PRGW is secured against hijack attempts from spoofed sources. Fig. 2 presents the mechanism, where an inbound SYN that corresponds to a temporary state is challenged by PRGW with a cookie. Since the TCP handshake only completes on arrival of an ACK bearing the sent cookie, it ensures that the sender is non-spoofed. Next, the PRGW assigns the state to the sender followed by the connection setup with the private host.

Since TCP connection does not complete with spoofed host, PRGW is protected against spoofed sources. PRGW employs a slightly tailored SYN cookie algorithm [10] for computing the initial sequence number (ISN), which is used as a cookie to eliminate address spoofing in the inbound packets.

ISN = time mod-32[5-bits]+MSS encoding[3-bits]+hash{source-IP, destination-IP, source-port, destination-port, SECRET} [24-bits] (1)

The use of the SYN cookie requires that TCP flow is relayed across PRGW. The relay itself must adjust the SEQuence and the ACKnowedgement number on both sides of the PRGW, to maintain the end-to-end semantics of the TCP connection. This is necessary due to the selection of random initial sequence numbers by the private host and PRGW. The translation of SEQ and ACK numbers effectively splices the connection on both sides of the PRGW. By keeping the SEQ number of the SYN to the private host the same as that of the inbound SYN, PRGW saves translation cost on one TCP sequencing.

B. Bot-detection Scheme

Attacks to PRGW could also originate from non-spoofed, i.e. bot hosts. In this section, we present a bot-detection method that attempts to protect PRGW against SYN floods from botnets, and thus complements the limitations of TCP-Splice.

In contrast to the networking elements that simply filter the packets mismatching to a flow or a connection state, PRGW can carry *bot-detection on the dropped packets*. Fig. 3 illustrates the mechanism where PRGW seeks to ascertain if the sender of the repeatedly mismatching SYNs is a non-spoofed entity. When the mismatching packets exceed a threshold in time T_O, PRGW handles the next inbound SYN failing to claim a state as per the SYN cookie algorithm. The subsequent arrival of an ACK bearing the sent cookie establishes the sender as *non-spoofed*. The history of dropped packets together with the non-spoofing check hints at a high likelihood of the sender as a *bot-controlled host*. Following which, the PRGW refuses any state to this host.

1) Implementation Considerations

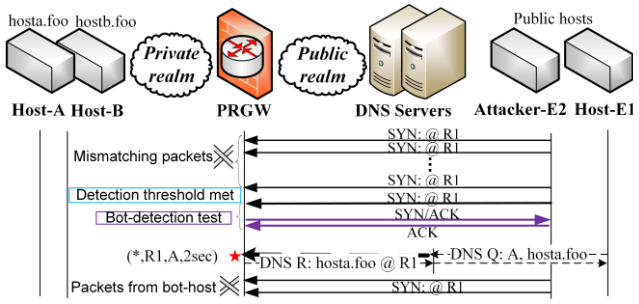

Fig. 3 Bot-detection method on SYN floods from a bot-operated host

An attacker meets the detection threshold, when mismatched packets reach a threshold in time T_O. Attackers typically initiate SYN floods at higher rates than a normal host, which only re-attempts if the previous packet is not responded within a retransmission timeout (RTO). RTO is typically an operating system defined parameter, and we choose a value below it as measurement interval T_O, since it significantly differentiates the legitimate behaviour from an attack. For TCP, UNIX domain sockets and Windows define RTO as 3 seconds [21]. Hackers can also initiate slow-rate SYN floods from various addresses, and thus bypass the bot-detection threshold. This will reveal the lower bound of PRGW security, where PRGW is secure against spoofed flows only. Bot-detection is executed only after an attacker meets the detection threshold, because a continuous monitoring for bot-detection would be too costly.

To realize the impact of our design choices, we classify the source of a mismatching packet into: 1) spoofed host; 2) non-spoofed attacker; or 3) a legitimate host. A packet may arrive from a host whose corresponding state was previously hijacked. However, a legitimate host does not re-attempt (or would not re-attempt x times) within RTO, and thus it would not meet the detection threshold. Similarly, a spoofed address cannot reply to SYN/ACK with the sent cookie, and hence is not blacklisted as attacker. Thus, only a bot-operated host is susceptible to this mechanism after it replies with an ACK bearing the sent cookie.

2) Caveats and Considerations

We realize that Bot-detection is not a fail proof solution and is vulnerable to abuse. Thus, we suggest to dynamically adjust the detection threshold and measurement interval T_O, to prevent the exploitation of the protection mechanism. Despite all the countermeasures, the possibility of a false alarm exists, and thus a bot-suspected host is blacklisted for temporal time T_D.

Since both the TCP-splice and Bot-detection could co-exist in the PRGW, there is a need to differentiate an inbound ACK under Bot-detection from an ACK that is part of TCP handshake with a public host. For this, SYN cookies of TCP-Splice and Bot-detection must differ, e.g. in *SECRET* value of equation 1.

C. Security by Deployment

A carrier-grade realm gateway (CGRG) can improve security of the private realm from a variety of resources at its disposal. For instance, the traffic from white and greylist sources can be accepted over separate sets of interfaces. This is often possible e.g. in mobile networks, where the traffic from other operators or corporate networks is processed on separate interfaces than those for public Internet [13]. This ensures dedicated access for whitelist networks and enables pursuing rather aggressive security on the greylist interfaces.

D. Enhancing the Circular Pool Algorithm

In [22], we present a new algorithm for allocating the public IP addresses of the circular pool, enabling fine-grained access control to flows arriving from the Internet. The new algorithm significantly improves the scalability and security of PRGW.

The underlying idea is to address the services and endpoints simultaneously. To that end, we leveraged the concept of the SRV DNS records and created Service FQDN (SFQDN) to address services on end-hosts. Currently, the use of SRV is only limited to a few applications, whereas the DNS A records are widely in use. SFQDN bridges this gap between DNS A records and the SRV records, and defines simple domain names linked to a specific service. For example, an SSH service at Host A – `a.foo` can be represented as `ssh.a.foo` or it can arbitrarily be defined as a combination of port number and transport protocol as in `tcp22.a.foo`. For aesthetic/security purposes, hosts can hide their SFQDN naming in favour of a more user friendly name, e.g. using CNAME records in DNS as a pointer to other domain names. The SFQDN and its mapping to a port can stay inside the PRGW while the CNAME to PRGW mapping is stored at a DNS server in the ISP network.

Since the SFQDN includes both the endpoint and the service, using the RFC defined terminology, SFQDN resolution allows *endpoint independent but port dependent filtering* in the half connection state relative to the remote host. The more specific half state allows reusing a public IP address for several different services, improving the scalability of CPPA. Theoretically, it implies that a single IP address can be reused as many times as the combination of available ports and protocols. Meanwhile, forcing the blocking state on PRGW becomes more difficult because the hacker must send significantly larger number of DNS requests to reserve the address pool for all the ports. In addition, the hackers must also target the allocated port besides simply flooding the public IP addresses for state hijacking. The temporary half connection state (R_X:oP_H, H:iP_H, P_{proto}, $T_{timeout}$) is unique and carries the IP address and port of the private host (H:iP_H), IP address and port on the public side of the PRGW (R_X:oP_H), the protocol (P_{proto}) and lifetime ($T_{timeout}$) of the entry. Upon the arrival of the first packet of the flow, PRGW upgrades the filtering to address and port dependent.

SFQDN contributes to security due to its more specific address allocation. This increases the attack surface, such that a hacker has more opportunities to meet the detection threshold, as a hacker must scan the entire port range to discover the active services and compromise respective allocations. The increased scalability also makes it more difficult to force the blocking state. Since PRGW solely admits inbound connections based on the domain queries, it becomes simple to temporarily block a service under attack and collect the evidence of misbehaviour.

VII. Security Evaluation

This section evaluates the security of PRGW in tackling the inherent Internet threats: source address spoofing, network/port scans and DNS floods. We implemented the above mechanisms in our PRGW prototype and subjected them to a set of *attacks* to determine the bounds of the PRGW security. The prototype runs in our test network, which is built in Linux environment using standard Linux networking capabilities: linux containers

and switches. The PRGW node in the testbed attends hosts and services located in its private realm, whereas legacy hosts in the testbed either initiate inbound connections or attacks towards the PRGW. The legacy nodes use virtual network interfaces to provide an illustration of many hosts participating in the traffic towards the PRGW.

We utilize Scapy [23] to craft *malicious* packets and launch attacks on PRGW. For our testing, this enables the legacy nodes to: 1) initiate spoofed traffic; and 2) emulate network floods from non-spoofed hosts, i.e. bots. The attack load is measured in SYNs per second from the hacker, whereas the network delay between the nodes is artificially generated. The outcome of the testing reveals the effectiveness and cost of the PRGW security, in terms of the ratio of the hijacked connections and processing delay introduced in the PRGW, respectively.

Fig. 4 demonstrates the PRGW security against DNS abuses. Having pre-configured the whitelist servers, we submit PRGW to DNS flood from multiple greylist servers. In the absence of security, the DNS flood would reserve all the CPPA resources and thus force PRGW in blocking state. However, the address allocation model notes that the DNS source is greylist and limits the resource allocations to a portion of the circular pool.

In this manner, the allocation model prevents the exhaustion of CPPA under DNS floods and ensures that whitelist servers have access to PRGW even under load conditions. A similar resource depletion attack using SFQDN is more challenging, since the high flood rate and amount of domain queries required to force blocking state increases likelihood of attack detection. Moreover, the rate limits on simultaneous domain queries from a DNS server and to a host, hinders the attacker ability to launch DNS floods from a few name servers or open resolvers.

We tested the CPPA enhancement algorithm by designing different inbound traffic patterns that evaluate the improvement in PRGW security due to SFQDN, especially against network and port scan attacks. We designed the following tests:

- Test1: 100% of the inbound traffic has the FQDNs of the destination hosts. On the event that hacker's packet meets an allocated address, the half connection state is claimed.
- Test2: 50% of the inbound traffic is generated using FQDN and the rest employs SFQDN. Hacker must target the right IP and port pair to claim the SFQDN allocation.
- Test3: 75% of inbound traffic is SFQDN; the rest FQDN.
- Test4: 100% of the inbound traffic is SFQDN.

Fig. 5 shows the result of stressing the prototype with above traffic patterns at network delay of 200 msec and a constant load of 4 connections per second. The connection load is distributed among private hosts and follows an exponential distribution. In parallel, a network scan attack at 40 SYNs/sec from the legacy nodes targets the CPPA. The figure reveals that for test1: FQDN initiations only, nearly all the connections are hijacked. This is because the hacker constantly scans the CPPA at high rate and beats the legitimate host in claiming the end point independent state. However, as the share of SFQDN grows and nears 100% in total inbound DNS queries, the ratio of hijacked connections declines and nears zero for an all SFQDN traffic. This is due to the fact that besides scanning the public IP addresses, a probing attacker also has to randomly scan for the allocated port out of 2^{16} possible ports to claim the state. The more specific address allocation for SFQDN enables more opportunities for a hacker to meet the detection threshold, which leads it to blacklisting in Bot-detection and subsequent rise in the legitimate connections.

Next, we evaluate the PRGW security against spoofed flows and network scans. We subjected PRGW to 3 connections (i.e. DNS requests) per second and in parallel launched 40 spoofed SYNs per second from the legacy nodes to CPPA, for hijacking the states. The testing reveals that spoofed SYNs failed to claim the half states due to better filtering enabled by the SFQDN. However, the spoofed SYNs could hijack the FQDN allocations in the absence of security mechanisms, because a hacker would scan the network at a high rate and can compromise states if its packet meets an IP address, allocated in the FQDN state.

In contrast, TCP-Splice successfully thwarts hijack attempts from spoofed sources and prevents leaking of spoofed packets into the private realm. Fig. 6 summarizes the PRGW's delay in assigning the half state to legacy hosts, not considering the link latency. The figure shows that TCP-Splice obviates spoofing in the admitted flows, at the cost of delaying the claim to the half

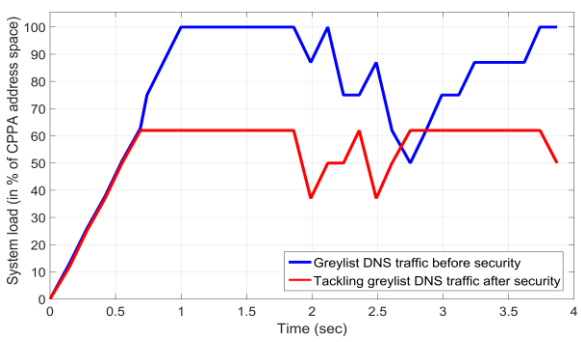

Fig. 4 Allocation model limiting the DNS flood from greylist servers

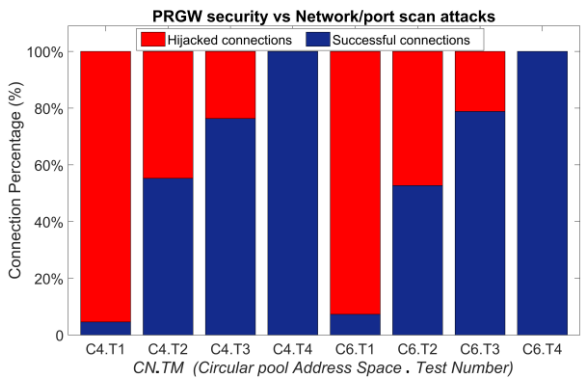

Fig. 5. Impact of inbound traffic type on security, versus network scans

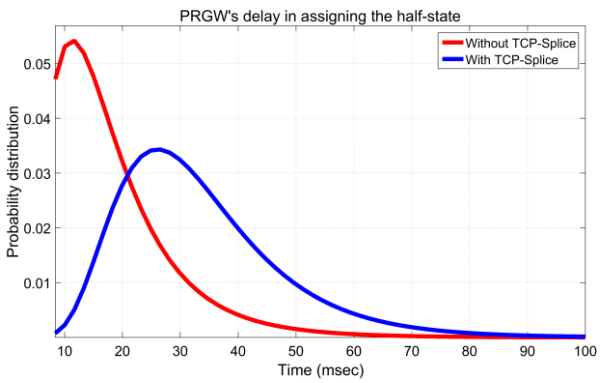

Fig. 6 Delay in assigning TCP half-connection state, before and after security

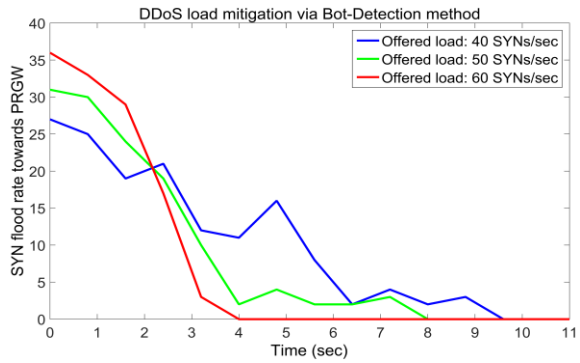

Fig. 7 Mitigating DDoS (SYN flood) via Bot-Detection method

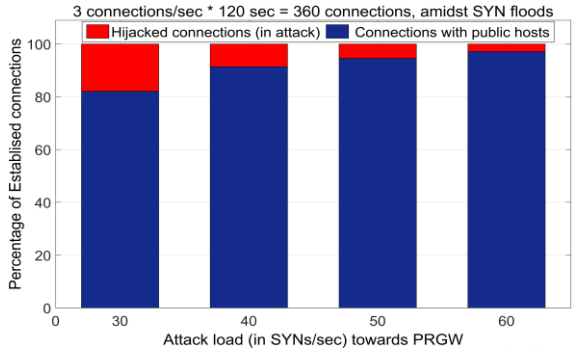

Fig. 8 Securing states against SYN floods from bot-controlled hosts

connection state. This is because to its SYN the sender receives a cookie from PRGW, which must be relayed back in the next inbound ACK to establish the connection, causing the delay.

In terms of performance, this limits the reusability of the public IP address and the port combination by the same duration for the next inbound connection. In a real network, the end-to-end latency for TCP messages would be added to compute the total delay in assigning the half-state. It is possible to reduce the average delay penalty caused by TCP Splice by using it selectively, i.e. on privileged ports, or under network attacks.

Fig. 7 presents an overview of PRGW security against SYN floods from bot hosts, which are non-spoofed sources under a botnet. Without security, an attacker can constantly scan the CPPA at high rate and on the event that its packet meets a half connection state, it will claim the allocation. In comparison, the Bot-detection would constantly track the dropped packets and once they exceed a threshold, the source is blacklisted following a non-spoofing test. As a result, states reserved by legacy clients are protected against the hijacking attempts. The figure shows that Bot-detection is more reactive to high flood rates and filters them earlier, as they quickly meet the detection threshold.

Fig 8 expands on the same result and shows the impact of stressing PRGW with a SYN flood sourced from eight hosts participating in the attack. In parallel, the public hosts initiate 3 connections/second towards the CPPA of three addresses, the network delay is 200 msec and the bot-detection threshold for a source is 12 dropped SYNs in 2 second interval. In practice, this threshold could be chosen during network planning phase, i.e. based on peaks in the traffic statistics graph. The figure reveals that the ratio of hijacked connections decreases as the attack load increases, since an attack with more active bots is filtered earlier, contributing to rise in the legitimate connections. Fig. 9 shows the impact of network delay, where the network delay is

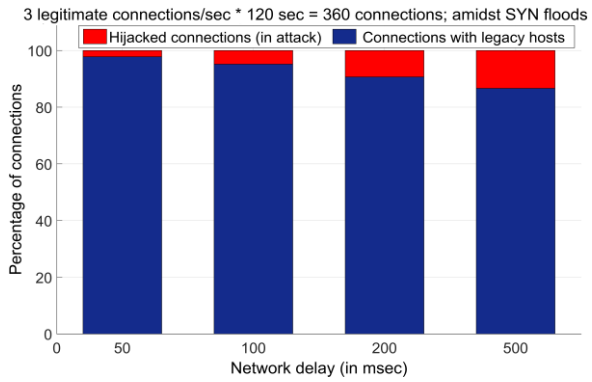

Fig. 9. Impact of network delay on PRGW security

time elapsed from creating a half connection state to the arrival of first packet from the client host.

The outcome of Bot-detection depends on multiple factors. From attack perspective, these are: number of flooding sources; choice of network/port scan strategies, i.e. targeting the known services or random port scans; and flooding rates for attacks or avoiding the detection threshold. On the other hand, the PRGW can improve its defense by dynamically adjusting the detection threshold, allocating more circular pool addresses and allowing SFQDN only. These strategies can provide more opportunities to hackers to meet the detection threshold and get blacklisted.

The paper obviously cannot present the PRGW security as a function of all the parameters. But, the testing generally reveals that Bot-detection reacts the best when attack volume is shared by few hosts. This means that to succeed a hacker must sacrifice rather large number of bots that do not use spoofing, and hence are likely to be identified by the target network's PRGW. The use of Bot-detection together with TCP-Splice guarantees that only legitimate hosts gain access to the private realm.

Fig. 10 compares the security of FQDN initiated connections in PRGW, in presence and absence of the security algorithms. Again,-we subject the PRGW to a load of 3 connections/sec at a network delay of 200 msec, while 8 non-spoofed sources flood CPPA with 40 SYNs/sec. Fig. 10.b shows that the ratio of hijacked connections decreases significantly after the security. The figure also reveals the impact of increasing CPPA address space, which contributes to security by increasing the overall attack surface. This shows that careful network planning and proper dimensioning of the CPPA resources can have positive impact on the PRGW security.

To deeply analyse the security of SFQDN states, we divide the Internet hackers into: 1) probing/scanning hackers; and 2) advanced hackers. A probing hacker scans the entire CPPA address space and port range to discover the available services, IP addresses or NAT mappings. It is quite likely that such an attacker due to its limited victim's knowledge, and thus random network scanning will fail to attack PRGW as shown for Test 4 in Fig. 5. In comparison, an advanced hacker may already know services/ports in the target network, via knowledge sharing among hackers or using botnets that perform the service discovery process. As a result, the hacker can target the SYN floods to the specific ports. We analysed the security of SFQDN allocations against such attacks and depict it in Fig. 11. We use the same test parameters as for Fig. 10. The result in Fig. 11.b shows a rise in the legitimate connections after security. This is because Bot-detection filters the hosts that initiate the floods

towards PRGW, however it is possible that a flood hijacks some states before it is entirely mitigated, as shown in the figure.

Clearly PRGW attains best-case security, when the hacker is unaware and simply scans the network for vulnerable services or IPs, i.e. a probing attacker, while the PRGW accepts SFQDN requests only. Under the premise that the attacks are directed to the served ports, it is perhaps best that SFQDN naming is changed to new service ports. This will force attacker to restart its service/port discovery cycle and help PRGW regain its best case security. Such use of SFQDN is possible in cases where a single administration owns or manages both the remote hosts and the PRGW. For example, Internet of Things (IoT) can emerge as one such use case where the communicating nodes and gateway will fall under single administration. In absence of such a scheme, Fig.11 shows the security of SFQDN allocations against an advanced hacker.

It is pertinent to mention that in our testing no state allocation was compromised by spoofed flows. However, few allocations were hijacked by the packets from the bot-hosts. This is because before a traffic flood is mitigated, some of its packets can beat a legitimate host in claiming the allocated state, and cause DoS to the actual client. Thus the security of PRGW can exhibit false negatives during attack. However, these false negatives reduce as the attack progresses, since the more active bots will be filtered upon exceeding the detection threshold.

The ratio of false negatives can further reduce by: 1) network dimensioning that presents an attacker more opportunities to meet the detection threshold; and 2) dynamically adjusting the detection threshold to prevent exploitation of the protection mechanisms. Though our testing identified few false negatives, PRGW did not exhibit any false positives, i.e. classifying a valid client as attacker. We argue that in the PRGW networks, a false negative is not as severe as a false positive; since a client that suffers hijacks can always re-attempt to access the desired service in the private realm.

Table-I summarizes the mechanisms deployed for securing PRGW against Internet threats and their impact on the PRGW's performance. Whereas, Table-II presents the duration that a received packet is processed in the PRGW security before a decision is reached. The delay values in Table II are computed within PRGW at algorithmic level, i.e. they do not include the time spent in acquisition, packetizing and forwarding of the packet. These values nicely fit with the delay requirements of the end-to-end connection. Hence, PRGW and its hosts can be protected at the cost of minute processing delay.

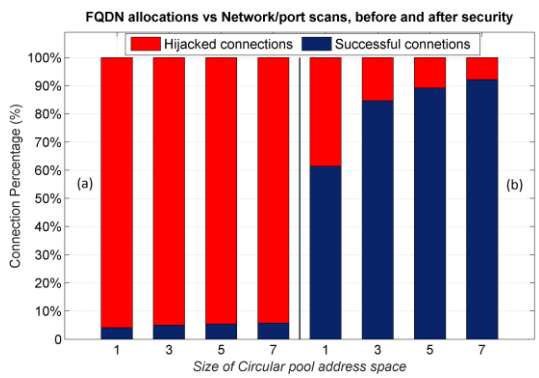

Fig. 10. Security of FQDN allocations, *(a)* without and *(b)* with security

TABLE I. SECURITY MECHANISMS AND THEIR PERFORMANCE

Security threats	*Mechanisms*	*Cost of Security*
Source address spoofing	TCP-Splice	Extends duration of assigning the state
Bot-controlled flows	Bot-detection	Possible False Negatives
Malformed ACK segments	cookie verification	-
DNS-floods	Rate limit simultaneous DNS allocations to hosts and greylist server(s)	Less trusted servers face congestion, under load
Spoofed DNS requests	DNS/TCP, DNS Relay and Ingress filtering	SLA negotiations, and sender's effort

TABLE II. PROCESSING OF INBOUND PACKET/FLOW IN THE PRGW SECURITY

	Processing delay	*Outcome*
Inbound TCP SYN segment	< 0.1 msec	Respond with cookie
TCP-Splice (on non-spoofed)	~1 msec	Eliminates spoofing
Packet not matching any state	~0.01 msec	Processing in bot-detection method
Malformed ACK segments	< 0.1 msec	Accept/Drop
DNS/TCP request	Connection-setup delay for 1st query	Spoofing elimination in the DNS queries
(Greylisted) DNS/UDP request	~ 1 msec	Accept if the load < threshold

The current PRGW prototype employs a minimalistic set of rules, i.e. rate-limiting, to provide the firewall functions. The deployment of PRGW at the network edges would require integrating PRGW with a commercial firewall. We argue that integrating a firewall would further reduce and nearly eliminate the false negatives during an attack, besides hardening the security of PRGW against well-known attacks.

VIII. DISCUSSION

The security testing shows promising results. Though, the implemented mechanisms exhibit false negatives, the proposed firewall integration will present PRGW as a feasible network function. For HTTP, which can set up many flows after a single DNS query, PRGW employs an HTTP reverse proxy to serve the inbound requests. Besides lessening the load on the circular pool, it offers advantages in terms of offloading SSL encryption and load balancing to the proxy [4]. Compared to proxy-server operations in SOCKS [24], TCP-Splice offers an efficient redirection mechanism for admitting the flows, and moves the processing load from caching at application-layer to mere sequence number translation at the transport layer.

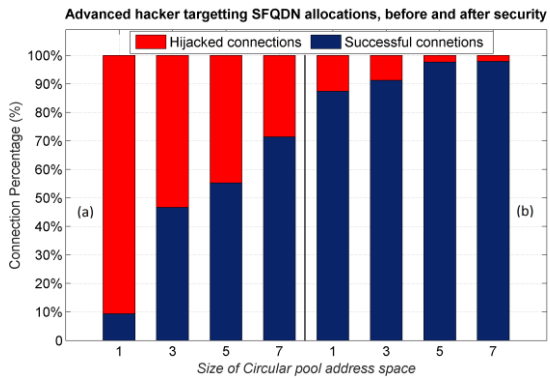

Fig. 11. Security of SFQDN allocations against advanced hackers, *(a)* without and *(b)* with PRGW security

In [4], we introduced PRGW to address the challenges in the Internet and offer a reachability solution that overcomes the drawbacks of the classical NAT traversals. The contribution of this paper is in presenting PRGW as a feasible function in the edge nodes that is well protected against the Internet attacks.

For end host security we can compare PRGW to the case that the application is using the cumbersome but functional IETF NAT traversal mechanisms [2]. To prevent attacks to the hosts that use SFQDNs and to identify the host application, we see the need to integrate an application policy database in PRGW that will link SFQDNs to application parameters, such as proxy name or addresses that can communicate with this SFQDN, and timeouts that will be used to monitor the application traffic, etc. PRGW can consult this database for making address allocation decisions. The idea would be to allow flows only from known entities or allocate most CPPA resources to known entities. The time parameter in the database can also rate limit an application that assumes connection initiation from unknown entities. We believe this would work for example for Peer-to-Peer SIP.

By tying the use of communication service proxies to PRGW via an application policy database, and by monitoring and rate limiting the application traffic, we reach the same level of host protection as in the case of application-specific NAT traversal.

IX. Conclusion

PRGW offers better than NAT service to hosts in the private address space. Unlike NAT, it presents a scalable way to initiate flows from other networks to hosts in the private address space. At the same time, no application-layer NAT traversal code is needed. Private hosts can stay reachable without need for keep-alive signalling to maintain their state, thus reducing the battery consumption. It offers shorter session setup delays, and eases configuring and managing of the port forwarding compared to how it is implemented in NATs, since PRGW can dynamically establish it upon the domain resolution.

This paper complements these advantages of PRGW through a security analysis that presents it as a feasible Internet function. The presented heuristics and mechanisms harden the PRGW against the inherent Internet weaknesses, such as source address spoofing, network/port scans and DNS floods. The mechanisms limit all the changes to network edges to favour the deployment and prevent the resource exhaustion in PRGW, by limiting flow acceptance to verifiable sources only.

PRGW admits inbound connections towards private hosts based on the domain name resolutions. We briefly discuss the current state of the art with DNS and leverage it for securing PRGW against Internet DNS abuses. Besides employing the best practices, we also present a new Bot-Detection algorithm that together with TCP-Splice attempts PRGW security against flows from spoofed and non-spoofed sources.

The security evaluation reveals that PRGW can be protected against the inherent Internet threats, at the cost of minimal processing delay. We briefly discuss the impact of different factors, such as attack strategy and inbound traffic pattern on the effectiveness of PRGW security. By addressing the security limitations of PRGW, this paper further adds to the claim of deploying PRGW at the network edges to address the Internet challenges [4]. We argue this further by briefly comparing NAT and PRGW, and the security of end hosts under both solutions. The adoption of PRGW to networks is simple, since it does not require any changes in end hosts, protocols or applications.

References

[1] ITU-T ICT STATISTICS. Free statistics. [Retrieved on Oct.2015] Available: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf

[2] L. Daigle, IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation, RFC 3424, Nov 2002

[3] G. Camarillo, J. Mäenpää, A. Keränen and V. Andersson, ”Reducing Delays Related to NAT Traversal in P2PSIP Session Establishments,” in Proc. IEEE Consumer Communications and Networking Conference, CCNC 2011, pp.549-553, Las Vegas, NV, USA, 9-12 Jan. 2011.

[4] J. Llorente, R. Kantola, N. Beijar, and P. Leppäaho, "Implementing NAT Traversal with Private Realm Gateway", Communications (ICC), 2013 IEEE International Conference, 2013, pp. 3581-3586.

[5] R. Kantola, “Implementing Trust-to-Trust with Customer Edge Switching,”, AMCA in connection with AINA 2010, Perth, Australia, 20-23 April 2010.

[6] H. Kabir, R. Kantola, and J. Llorente, "Security Mechanisms for a Cooperative Firewall," in Internatinal Symposium on Cyberspace Safety and Security (CSS), Paris, 2014.

[7] "2014 Cisco Annual Security Report," CISCO, 2014.

[8] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[9] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827, May 2000.

[10] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, August 2007.

[11] M. Ma, "Mitigating Denial of Service Attaks with Password Puzzles," in Information Technology: Coding and Computing, 2005, pp. 621-626.

[12] H. Wang, C. Jin, and K. G. Shin, "Defense Against Spoofed IP Traffic using Hop-Count Filtering," in IEEE/ACM, Transactions on Networking, Volume 15, 2007, pp. 40-53.

[13] "SRX Series AS Gi/SGi Firewall for Mobile Network Infrastructure Protection," Juniper Networks, Whitepaper.

[14] H. Beitollahi and G. Deconinck, "A Cooperative Mechanism to Defense Against Distributed Denial of Service Attacks," in 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) 2011, 2011.

[15] R. Lua and K. C. Yow, "Mitigating DDoS Attacks with Transparent and Intelligent Fast-Flux Swarm Network," in IEEE Networks, Volume: 25, Issue:4, 2011, pp. 28-33.

[16] R. R. Robinson and C. Thomas, "Evaluation of Mitigation Methods for Distributed Denial of Service Attacks," in 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2012, pp 713-718.

[17] ”DEFEATING DDOS ATTACKS,” CISCO Systems, Inc., White Paper, 2014.

[18] F. A. Ed. and C. Jennings, ”Network Address Translation (NAT) Behavioral Requirements for Unicast UDP,” RFC 4787, 2007

[19] J. Rosenberg, et al., "SIP: Session Initiation Protocol," RFC 3261, 2002

[20] ”DNS Best Practices, Network Protections, and Attack Identification,” CISCO Systems, White Paper, 2015.

[21] MICROSOFT. TCP/IP and NBT configuration parameters for Windows. [Online]. http://support.microsoft.com/kb/314053 {On: 22.07. 14}

[22] J. Llorente and R. Kantola, "Transition to IPv6 with Realm Gateway 64," IEEE International Conference on Communications (ICC), London, June, 2015.

[23] (2015, Mar.) SCAPY. http://www.secdev.org/projects/scapy/

[24] M. Leech, M.Ganis, Y. Lee, R. Kuris, D Koblas, L. Jones , "SOCKS Protocol Version 5," RFC 1928, March 1996.