Article

NAT++: An Efficient Micro-NAT Architecture for Solving IP-Spoofing Attacks in a Corporate Network

Prakash Veeraraghavan , Dalal Hanna and Eric Pardede

Department of Computer Science and Information Technology, La Trobe University,
Melbourne VIC 3086, Australia; D.Hanna@latrobe.edu.au (D.H.); e.pardede@latrobe.edu.au (E.P.)

* Correspondence: P.Veera@latrobe.edu.au; Tel.: +61-3-9479-1547

Received: 27 July 2020; Accepted: 11 September 2020; Published: 14 September 2020

Abstract: The Internet Protocol (IP) version 4 (IPv4) has several known vulnerabilities. One of the important vulnerabilities is that the protocol does not validate the correctness of the source address carried in an IP packet. Users with malicious intentions may take advantage of this vulnerability and launch various attacks against a target host or a network. These attacks are popularly known as IP Address Spoofing attacks. One of the classical IP-spoofing attacks that cost several million dollars worldwide is the DNS-amplification attack. Currently, the availability of solutions is limited, proprietary, expensive, and requires expertise. The Internet is subjected to several other forms of amplification attacks happening every day. Even though IP-Spoofing is one of the well-researched areas since 2005, there is no holistic solution available to solve this problem from the gross-root. Also, every solution assumes that the attackers are always from outside networks. In this paper, we provide an efficient and scalable solution to solve the IP-Spoofing problem that arises from malicious or compromised inside hosts. We use a modified form of Network Address Translation (NAT) to build our solution framework. We call our framework as NAT++. The proposed infrastructure is robust, crypto-free, and easy to implement. Our simulation results have shown that the proposed NAT++ infrastructure does not consume more than the resources required by a simple NAT.

Keywords: PrECast; Micro NAT; DoS; DNS-amplification

1. Introduction

The Internet Protocol (IP) version 6 (IPv6) was developed by the Internet Engineering Task Force (IETF) and was ratified as an Internet Standard in 2017. However, until today, only 15% of Internet traffic is due to IPv6 [1]. Google statistics show that depending on the day of the week, between 25–30% of its users are using IPv6 addresses [2]. However, the BGP table data shows that the IPv6 penetration was only 11% as on 20 July 2020 [3]. Thus, as of today, the Internet Protocol (IP) version 4 (IPv4) is the most deployed Internet protocol. IPv4 has several known vulnerabilities. One of the important vulnerabilities is that the protocol does not validate the correctness of the source address carried in an IP packet. Users with malicious intentions may take advantage of this vulnerability and launch various attacks against a target host or a network. This attack is popularly known as IP-Address Spoofing or simply IP Spoofing. IP address spoofing is the creation of IP packets with a false source IP addresses to impersonate another host or a network device. The false IP address may belong to the local subnet or a remote network. The IP address spoofing is primarily employed to launch a Distributed Denial of Service (DDoS) attack. Packets with spoofed IP addresses are more difficult to filter in the target network since each spoofed IP packet appears to come from a legitimate IP address. It is estimated that individual DDoS attacks can cost an organization up to US\$ 50,000 [4]. However, due to these attacks, large enterprises not only lose substantial money but also their credibility and lose their market share [5,6]. Thus, protecting a network from these attacks is an utmost priority.

IP Spoofing is one of the well-studied areas since 2005. There are several research articles and commercial software systems available to protect a network from IP address spoofing attacks. However, until today, no network is immune to IP Spoofing attacks. IP Spoofing is not only exclusive to the IPv4 network but also applicable to the IPv6 network in the absence of IPSec. Even though there are several research articles and commercial products available in the literature to stop this attack, most of the article reviewed by us assumes that the attack originated from an external network. None of the research work known to us in the literature address IP spoofing attacks that originate internally to the network either through a malicious insider host, internal nodes that colluded with external hosts, or compromised internal nodes. This form a strong motivation for this work.

This paper addresses the IP-Spoofing problem that arises from the internal network. This consists of two scenarios: an inside host that spoofs an IP address that belongs to the same corporate network; an inside host that spoofs an IP address that belongs to an external network. In both scenarios, we provide an efficient and scalable solution to stop the IP spoofing problem. Our solution uses a modified form a Network Address Translation (NAT) architecture. The NAT was originally invented to address the shortage of IPv4 address space. However, in this paper, we use NAT only to hide the network and hosts from the external network. We call our proposed system as NAT++. NAT++ does not need any modification to the protocol stack at the client-side. The NAT++ requires that every corporate switch support Layer-3 and above and can run and maintain the other services of the NAT++ solution.

Our contribution to this paper is as follows: The NAT++ system that solves the IP-Spoofing problem that arises from an inside host.

In our earlier paper [7], we introduce the PrECast infrastructure for a corporate network that solves various poisoning problem that leads to the Man-in-the-middle (MITM) attack in a Local Area Network (LAN). The NAT++ solution is implemented on top of the PrECast infrastructure to provide robust security.

The following are the advantages of the proposed NAT++ system:

1. NAT++ effectively eliminates several important network attacks that are due to IP-address Spoofing.
2. Due to the distributed nature of our design, NAT++ consumes very less system and network resources. This is evident from our simulation.
3. Our solution is modular and scalable. Thus, we may able to offer a solution to unknown attacks in the future without much of a redesign.
4. NAT++ does not use any crypto-algorithms and it does not need any modification to the protocol suite. Thus, our scheme does not violate any protocol standards.
5. As an indirect benefit, NAT++ controls the network diameter whereby avoiding chaining at the access layer.

The rest of the paper is organized as follows: In Section 2, we present the state-of-art in IP-address spoofing. We also outline the “port scanning attack”. In Section 3, we explain why we choose NAT as a candidate in solving the IP-Spoofing problem. We then also outline why NAT requires a modification. Section 4 deals with our proposal. We start this section with a brief overview of the PrECast framework. In this section, we also highlight what attack vectors our proposal is immune to. In Section 5, we evaluate the performance of NAT++ against various standard NAT implementations. We also evaluated the CPU workload incurred for a similar IP-spoofing solution. Section 6 deals with the conclusion and future direction.

In this paper, we follow the Open Systems Interconnection model (OSI model) due to the International Organization for Standardization (ISO). Whenever we mean Layer-2, it denotes the Datalink Layer and Layer-3 denotes the Network Layer of the OSI stack.

2. The State-of-Art in Address Spoofing

In this section, we review some of the key works available in this area. Network communication involves two types of addresses. One is the Data link-layer address called the Media Access Control Address or simply the MAC address. This address is also sometimes referred to as a Physical address. MAC address is a unique identifier assigned to a Network Interface Controller (NIC). MAC addresses are primarily assigned by device manufacturers and are referred to as Hardware address. MAC address is stored in hardware, such as the network card's read-only memory, or by a firmware mechanism. Until a few years ago, MAC addresses are hardcoded in a ROM and cannot be changed. For all new generations of NIC cards, the default MAC address can be changed temporarily through software. Malicious users may take advantage of this feature and launch MAC-address spoofing attacks. Since the focus of this paper is at the IP-layer, we will not focus on this type of attack in this paper.

The second address is the IP-address, called the logical address. IP addresses are assigned to hosts either manually or dynamically through a Dynamic Host Configuration Protocol (DHCP). Communication between any two hosts on the Internet requires the use of source IP address and the destination IP address. The IP protocol (both v4 and v6) does not validate the correctness of the source address carried in an IP-packet. Only the destination IP address is needed for routing the IP packet from the source to the destination. The source IP address is used only by the destination host for sending the reply. In the absence of any validation, IP address spoofing involves the creation of IP packets with a false source IP addresses to impersonate other hosts or network devices in a network. The malicious intent may be the main motive behind IP address spoofing. IP address spoofing is used primarily to launch DDoS attacks. IP address spoofing is also effectively employed whenever there is a trust-relation that exists between hosts in a corporate network. In a corporate network, it is common that one or more servers may establish inherent trust relations among them. If a user successfully logged into one machine, then they can automatically connect with other machines without the requirement for a username and password. By spoofing a connection from a trusted machine, an attacker on the same network may be able to access the target machine without authentication. There are numerous solutions available to thwart this attack. Most of them use some form of digital certificates and the use of encryption. Kerberos is an example of such a system [8].

The majority of IP spoofing will result in a DDoS attack [9,10]. Whenever an IP packet carries an impersonated IP address, it is termed as a "spoofed IP packet". A spoofed IP packet does not cause any harm to the path routers; rather the intention is to only harm the victim or his network. The research community started working on combating IP spoofing as early as 2000 [11,12]. The early work involves IP filtering at the ingress and the egress routers. Research on IP spoofing can be classified into broadly three categories namely:

- Identifying spoofing packets
- Mitigating spoofing attacks
- Pinpointing an attacker's real location

A classical survey on the above topics is due to Ehrenkranz and Li [13]. After this work, unfortunately, there are no up to date surveys available on this topic. This may form a motivation for our future work. In a recent paper [14], Lichtblau et al. proposed a detection, classification, and analysis of inter-domain traffic with a spoofed IP address. Their work involves filtering at the border routers based on known IP ranges used inside the network. Since the backbone speed is increasing day-by-day, filtering at the border router may introduce unwanted delays. Also, Cisco's network design [15] recommends no filtering done at the Core-layer of the network. This solution may stop an inside host in spoofing an IP address that is external to the network. However, this solution will fail to detect an inside host spoofing an address that belongs to the same corporate network.

In 2007, Wang et al. [16] proposed a hop-by-hop filtering mechanism for mitigating IP Spoofing. In the absence of any incentive for path routers to perform packet filtering, this proposal works well only in the ingress network. Hop-by-hop filtering is better in terms of detecting spoofing than the

algorithm presented in [14]. It can detect a host spoofing an IP address that belongs to another subnet, but not on the same subnet.

In Reference [17], Mirkovic et al. proposed a self-learning algorithm for filtering spoofed packet in the upstream traffic. Their algorithm learns to detect spoofed traffic upstream from the victim by combining information about the traffic's expected route and the sender's response to a few packet drops. Even though their algorithm may work well in a lighter load, on a heavy DDoS attack, detection may not be of any use; rather immediate mitigation strategies are needed. This algorithm exhibits the same weakness as that of [14] in detecting internal IP address spoofing.

The lack of authentication in the Internet's data plane facilitates IP spoofing attacks. Fonseca et al. [18] proposed a method to establish a hop-by-hop authentication mechanism to track down the source of spoofed IP packets. Tracking down the source may thwart further attacks from the same attacker. However, this scheme does not effectively eliminate IP spoofing. In addition to this, establishing authentication in the data plane is difficult due to the amount of workload in processing individual IP packets (or IP flows). This algorithm behaves like [16] in detecting IP spoofing attacks.

The papers such as [14,16–18] and the articles reviewed in the survey [13], the authors assume that no host spoof an IP address that belong to the same IP subnet. This deficiency forms a strong motivation in providing our NAT++ framework.

Broadcasting is one of the essential features of the IPv4 protocol. This is replaced with multicasting in the IPv6 protocol. Two essential services, namely the Address Resolution Protocol (ARP) and the DHCP are built based on IPv4 broadcast services. To launch a DDoS attack, IP spoofing must be combined with the IPv4 broadcasting feature. IP spoofing alone may not result in a DDoS attack [9,10].

The following are some of the important classes of attacks that are due to the combination of the broadcast feature in the IPv4 protocol and IP-address spoofing. These attacks are classified under "Amplification Attacks":

- The Smurf Attack
- The Fraggle attack
- The DNS-Amplification attack

There are other amplification attacks such as Network Time Protocol (NTP), The Simple Service Discovery Protocol (SSDP) and The Simple Mail Transfer Protocol (SMTP) amplification are possible by exploiting the vulnerabilities of the IPv4 protocol. A detailed description of these attacks is presented in [7].

For all the above-mentioned attacks, botnets [19] may be used as an attacker to increase the amplitude of these attacks. DNS-amplification attack is hard to detect as they appear to originate from valid servers and targeted towards a legitimate host. For Smurf, Fraggle, and DNS-amplification attacks, the attacker can be anywhere on the Internet.

Figure 1 provides a taxonomy of attacks that exploits the IP-broadcasting and IP-Spoofing vulnerabilities in the IPv4 protocol. Figure 2 depicts a mechanism through which an attacker launch DoS attacks mentioned in the taxonomy.

The broadcast nature of the IPv4 protocol (without IP address spoofing) alone can be exploited to launch several attacks in a Local Area Network (LAN). These attacks are broadly classified under "Poisoning attacks". The Address Resolution Protocol (ARP)-Poisoning attack is one of the classical examples. The main motive behind poisoning attacks to become a Man-in-the-Middle (MITM), thus an attacker can attract the victim's Layer-3 traffic towards him. A detailed review of various poisoning problem is presented in [7].

Figure 1. Taxonomy of attacks due to IPv4 broadcasting [7].

Figure 2. Typical DoS attack.

Port Scanning Attack

This attack is different from the two types of attacks (spoofing and poisoning) described above. This attack is not launched by exploiting any vulnerabilities in the Internet protocol. Every cyberattack starts with the port scan attack, whereby the attackers scan the target network for any potential vulnerabilities or running unpatched services. In our earlier paper [20], we provided a solution to mislead an attacker. The middleware proposed in [20] will provide falsified information about the network and hosts to an attacker. However, our earlier proposed solution will not stop an insider in fingerprinting hosts.

Network Address Translation (NAT) provides a seamless solution to all the attacks mentioned above. This is by hiding the campus hosts from an external network. We describe the strengths and weaknesses of NAT in terms of security in the next section.

3. Network Address Translation (NAT)

NAT [21] is a method of remapping one IP address space into another by modifying the address information in the IP packet header while they are in transit. NAT was originally invented to address IPv4 address depletion and scaling in routing [21]. The address reuse solution is to place NAT at the border of stub domains. Each NAT box has a table consisting of pairs of local IP addresses and globally

unique addresses. The IP addresses inside the stub domain are not globally unique and thus can be reused. The main advantage of NAT is that it can be installed without any changes to routers or hosts.

Even though NAT was originally designed to address the IPv4 address depletion problem, it provides some level of security to end-hosts. A NAT box hides the internal hosts from external networks. However, the traditional security mechanisms such as firewalls, Intrusion Detection System and Intrusion Prevention System cannot be ignored. By hiding the host's IP address, NAT provides the following advantages:

1. DoS protection: This is because the host's private addresses are not known to public hosts. Using the same reasoning, NAT protects infrastructure.
2. NAT can be used to effectively limit the use of protocols and ports.
3. NAT can eliminate network and port scanning done by rogue devices from the Internet.
4. NAT blocks unsolicited connections that are initiated from an external network (external to the NAT realm)

Even though NAT has the above advantages, it has the following disadvantages:

- NAT box itself is a bottleneck. Attackers may launch DoS attacks against NAT boxes that will stop the entire network operation.
- Since NAT limit the use of protocols and ports, it is hard to detect any potential malware that infected the inside network. This is because; no external network can detect malware's heartbeat.

The following property made us consider NAT as a potential candidate for protecting a corporate against an IP-Spoofing attack. Whenever a connection is made from a host in a private realm to a public realm, for every communication, an entry is created in a NAT look-up table based on <Protocol, Source IP, Source Port, destination IP, destination port>, where Destination IP and port belong to a public realm. Replies arriving from <Destination IP, port, protocol> are forwarded to inside hosts. If there are no matching entries, packets are silently dropped. Thus, any unsolicited replies originating from the public realm are blocked. This important security feature is used along with the PrECast infrastructure to provide robust security in a corporate network.

3.1. Attack against Nat Boxes

In this subsection, we outline a few attacks that are targeted against NAT boxes that can potentially stop the entire network operations. Thus, NAT by itself cannot be used for protecting a network from IP-based attacks.

Traditionally NAT boxes are deployed at the gateway of an IP subnet (that uses a private realm). This IP subnet may represent a building, department, or unit in a corporation. It may consist of several hosts ranging from 20 to 100. When using NAT boxes, all traffic between NATed subnets and the Internet must go through the NAT gateway. For this reason, the network administrator must ensure that NAT boxes have greater throughput and low latency. The NAT boxes should also have fast processors that can examine and translate packets quickly. We now discuss some of the vulnerabilities that are not addressed by NAT. For our discussion, let us assume that M be a malicious host and V be a victim. Both M and V are inside the same NATed network.

1. Since NAT boxes hide inside the network, it protects the inside network from DoS. However, consider the following scenario:
 - (a) M spoofs V's private IP address and create a Smurf or a Fraggle attack through an external amplifier. The NAT box will faithfully create a forwarding table and forward it to the amplifier network. The destination field in the forwarding table is the broadcast address of the amplifier network. One spoofed IP packet from M will result in 1000s of packets from the amplifier network targeted towards V. The NAT box will keep dropping them

after checking the source IP address (no match with individual host addresses from the amplifier network). Thus, this is effectively a DoS attack targeted towards the NAT box. Since the NAT box is processing all the incoming packets, it may not be able to perform its translational task.

- (b) Whenever M spoofs V's IP address and launch a DNS-amplification attack through compromised DNS servers, all the incoming packets are faithfully translated and forwarded towards the victim V, as there is a match between the source and the destination IP in the forwarding table. Thus, the DNS-amplification DoS attack is still possible in a NATed network if the actor node is inside the NATed network.
2. NAT effectively stops port scanning attacks originating from the Internet. However, it cannot protect the network from a port scanning attack originating from the inside network. As before, we also demonstrate that if there is a malicious actor node V inside the NATed network, he can open up the port for an external host to come through. Let M_1 be a malicious inside host, M_2 be an outside host and V be the victim host. Let us assume that M_2 wishes to connect to V on port P. Under normal circumstances, this is not possible in a NATed network. Any request to V on port P will be silently dropped by the NAT box as there was no forwarding entry. A forwarding entry may be created with the help of M_1 . As a first step, M_1 sends a spoofed IP packet with V's IP address and P as its source port to M_2 . NAT box will faithfully create a forwarding entry for V towards M_2 . Once this spoofed IP packet is received by M_2 , he can then connect with port P of his victim V.

Using this method, any malicious inside host may facilitate an external host in performing port scanning or connect to a service.

Thus, NAT by itself does not offer security. However, the interesting feature of NAT is that it hides the inside network from the outside network. In this paper, we modify NAT and augment with the PrECast infrastructure to offer more robust security for corporate networks.

There are three different types of NAT translations. The majority of NAT boxes map several hosts with private IP realm to one public IP address. This public IP address is assigned to an outgoing interface of the NAT box. The NAT box on that network has a private address in that address space. This type of NAT boxes is known as Many-to-one NAT or a Port Address Translator (PAT). Home gateways are an example of this type of NAT implementation. Even large networks can be connected to the Internet using a single public IP address. This type of NAT may have performance limitations, such as high latency, does not support enough applications, and so on. To address this limitation, we have many-to-many NAT. The Many-to-Many NAT policy allows us to translate a group of addresses into a group of different addresses. Usually, the private realm has more addresses than the public realm. This may sometime result in starvation. To avoid this problem, we have one-to-one NAT. In this type of NAT boxes, one private IP address is mapped on to one public IP address. RFC 2663 [21] refers to this type of NAT as Basic NAT. In this type of NAT, only the IP addresses, IP header checksum, and any higher-level checksums that include the IP address are changed.

In a NAT scheme, if the private IP realm has more than one host, then any arbitrary host can launch IP-spoofing attacks that are mentioned above. Thus, to address a solution, we wish to limit the number of hosts in a private IP realm to one. The only NAT scheme that provides this feature without wasting IP address space is the one-to-one NAT. Also, among the three different types of NAT, one-to-one NAT offers lesser latency and higher throughput. Thus, in our NAT++ architecture, we prefer to use one-to-one NAT due to various advantages, both in terms of performance and security.

A one-to-one NAT alone does not solve the issues mentioned above. By definition, a one-to-one NAT associates one private IP address to one public IP address. This process does not guarantee that there is exactly one physical host in the private IP realm. Thus, one-to-one NAT alone does not solve the IP-Spoofing problem. NAT++ is a one-to-one point-to-point micronatting architecture that is compatible with the PrECast protocol. We present its detail in Section 4.

4. The Nat++ Architecture

In this section, we describe the detailed operation of NAT++ architecture.

Since NAT++ is deployed on top of the PrECast infrastructure, we provide its key operational features.

4.1. The Precast Infrastructure

In our earlier paper [7], we provided the PrECast infrastructure that eliminates the poisoning problems in a LAN. This is by effectively converting IPv4 broadcasting into a “secure multicasting”. The PrECast protocol operates through the following three phases:

1. Bootstrap Phase
2. Secure multicast tree (called the PrECast-tree) construction phase
3. The Network operation phase

We now explain these phases in brief. For the detailed operation of the PrECast protocol, readers may refer to our original paper [7].

During the bootstrap phase, each switch in the LAN starts creating the PrECast table. This table is similar to the MAC-Address table kept in every switch. The PrECast table is specific to each switch. This table consists of the following information:

- Port/Switch ID: If a host is connected to a local switch port, this field represents the port-number to which the host is connected. If a host is connected with a remote switch, this field represents the “Switch ID”. Each switch in a LAN has a unique Switch ID.
- MAC Address: This field stores the MAC address of the host associated with this Port/Switch ID.
- IP Address: This field stores the IP address of the host associated with this Port/Switch ID.

The heart of the PrECast protocol is the secure multicast tree that consists of all campus LAN switches to which hosts are connected. In our earlier paper [22], we presented an efficient algorithm for creating and maintaining a secure multicast tree using a Core-based Tree. This tree is used for distributing broadcast messages between LAN switches securely. We call this tree as a PrECast-tree.

The heart of the network communication is the two protocols namely the ARP and the DHCP. During the bootup stage, every host must obtain an IP address to communicate with the rest of the world.

Whenever a host broadcasts a DHCP-Discover message, the switch to which the host is connected will note the host’s MAC address and unicast the request to the registered DHCP server available in this network. The switch ports to which the requesting host and the DHCP-Server are connected will tunnel the broadcast communication. Once the DHCP handshake is complete, the allocated IP address is mapped against the host’s MAC address in the PrECast table. By doing this way will eliminate the DHCP-poison problem.

The PrECast protocol handles the ARP process much like the conventional process, except that the ARP broadcast requests are multicasted among the switches. The immediate switch to which the requested host is connected will answer the ARP-request based on the information available in its PrECast table. If the requested information is not available in its local table, the switch will then multicast the ARP-request to every LAN switches. Eventually, the switch to which the host is connected will “reply”. This information is entered into the PrECast table by all LAN switches.

We now provide a detailed overview of the NAT++ architecture and how NAT++ can be integrated with the PrECast infrastructure to eliminate several attacks that are due to the two important vulnerabilities mentioned in this paper. Figure 3 represents how NAT++ is integrated with the PrECast infrastructure. The proposed NAT++ framework addresses the IP-Spoofing problem that arises from internal hosts. However, NAT++ by itself cannot eliminate the “poisoning problem” in a LAN. Thus, we propose to implement NAT++ on top of the PrECast infrastructure.

Figure 3. Integration of NAT++ with the PrECast Infrastructure

The block diagram of our proposed architecture is presented in Figure 3. Every host is connected to a switch-port. The connection between a host and its switch-port is regarded as a point-to-point connection. A NAT instance is running on every switch-port (as a software service) that translates the host's private IP address to a public IP address. All modern switches support this feature.

4.2. the Micro Natting Architecture

In Section 3, we presented important advantages of NAT in terms of offering security by hiding inside-network from an external network. However, as we discussed in Section 3.1, a malicious inside host can open a port, so that an outsider can launch an attack. In our proposed NAT++ architecture, we consider one-to-one NAT, where each switch port run a NAT engine.

System Bootstrap stage: During the system setup phase, the network administrator selects one or more LAN switches in the network as the Core switches for the multicast delivery tree. Based on our previous work [7], all LAN switches form a secure multicast tree. This tree is used for data communication as outlined in our PrECast paper [7].

As outlined in Section 4.1, every LAN switch stores a table that consists of the following information:

- Port ID: This is the port number of the current switch the host is connected. For hosts that are connected with remote switches, this field represents the Switch ID.
- MAC Address: This field holds the MAC address of the host connected to this port/Switch ID.
- IP Address: IP address of the host that has this MAC address.

In a corporate network, there will be a few hosts that use static IP addresses. For example, DHCP servers, primary and secondary DNS servers, default gateways, file, and print servers. These hosts are connected with certain switch ports. Information about these hosts is manually populated by the network administrators on the respective switches' PrECast table. It is important to note that important devices in the network that have static IP addresses will not participate in the NAT++ process. Thus, the switch ports to which these devices are connected will not run the NAT++ engine. Since these devices are trusted devices, they will not launch an IP-Spoofing attack. Except for these switch ports, by default, all other switch ports will run the NAT++ engine.

Stage 1: Every switch port runs a NAT engine. For our discussion, let us assume that P_1 be a switch port of a LAN switch S_1 . Let H_1 be a host connected to the port P_1 . Then the connection between P_1 and H_1 can be regarded as a point-to-point (P2P) connection. The switch port P_1 also acts as a proxy between H_1 and the rest of the network. All LAN hosts except servers obtain an IP address through DHCP servers. They all participate in the NAT++ process. If any malicious host tries to connect to the network using a static IP address, the NAT++ engine will silently drop its IP packets.

We now explain the DHCP-DORA process through our NAT++ mechanism:

DHCP-Discover Process: In this step, the host H_1 sends a DHCP-discover message. This message is a Layer 2 and 3 broadcast message. Once this message is received by switch port P_1 , the switch S_1 issues a modified DHCP-Discover message. This is a unicast message sent at Layer-2. The IP source address (IP SRC) is still 0.0.0.0 (as H_1 has not yet obtained any IP address) and the IP destination address (IP DES) is 255.255.255.255. However, the source MAC (MAC SRC) is the MAC address of port P_1 and the destination MAC address (MAC DES) is the MAC address of the DHCP server connected to this network. Rather than issuing a network-wide broadcast, this packet is sent to the DHCP-server as a unicast Layer-2 frame. Doing this way protects the network from the DHCP-poisoning attack in the network.

DHCP-offer process: When a DHCP-server receives a DHCP-Discover message from a client, the DHCP server reserve an IP address (say A.B.C.D) and makes a lease offer by sending a DHCP-OFFER message to the client. This message contains the client's ID (traditionally a MAC address; in our case, it is the MAC address of port P_1), the IP address that the server is offering (i.e., A.B.C.D), the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

At Layer-3, the source address is the IP address of the DHCP-server. The destination address is 255.255.255.255. This is a Layer-3 broadcast packet. At the Layer-2 frame, MAC SRC is the MAC address of the DHCP server and the MAC DES is the MAC address of P_1 . At Layer-2, it is a unicast frame.

The received message is translated by the switch S_1 and send to H_1 as follows: In the translated message, the Layer-3 IP-SRC is the IP address of P_1 . According to our NAT++ design, for all switch ports, their IP address is 192.168.0.1 with the subnet mask of 255.255.255.252 and the default gateway of 192.168.0.1. The primary and the secondary DNS address provided by the DHCP server is copied to the translated packet. However, the offered public IP address of A.B.C.D is not passed on to H_1 ; rather a private IP address of 192.168.0.2 is passed on to the host. With our proposed design, every host in the network will have the same private IP address 192.168.0.2. For the Layer-2 frame, the MAC-SRC is the MAC address of P_1 , and the destination MAC address is the MAC address of H_1 .

DHCP-Request Process: Once the DHCP-offer message is received by H_1 , it replies with the DHCP-request process confirming the allocated IP address. For this reply message, the IP-SRC is still 0.0.0.0 and the destination IP address is 255.255.255.255; MAC-SRC is the MAC address of H_1 and the destination MAC address is the MAC address of P_1 . As before, this received message is translated by switch S_1 .

Once the DHCP server receives this message from Switch S_1 , it will then send a DHCP-ACK message. Switch S_1 will translate and send it to H_1 .

At the end of this DHCP phase, the client H_1 is issued with an IP address 192.168.0.2 with a subnet mask of 255.255.255.252 and the default gateway of 192.68.0.1. This process has segmented the entire LAN into a micro-segment that consists of only P_1 and H_1 . Thus, the new broadcast domain consists of only two hosts namely H_1 and P_1 . At Layer-3, the connection between P_1 and H_1 are point-to-point.

The port P_1 acquired a public IP address of A.B.C.D from the DHCP server. This address is used for NATting the connection from H_1 . Thus, by design, our NAT is a one-to-one NAT and it hides every host from every other host in the network.

We present the sequence diagram depicting the DHCP address process in Figure 4. In this paper, we used the private IP network 192.168.0.0/30 for the P2P connection between a host and its switch port. However, this may be replaced with any private IP space with a subnet mask of 255.255.255.252 (or /30).

Figure 4. Sequence Diagram for the DHCP Process.

4.3. The Network Operation

In this subsection, we outline some of the essential network operations performed under our proposed design.

ARP-Request and Reply are precursors to communication between any two hosts in a network. Thus, we address how ARP-Request and replies are handled by our NAT++ protocol.

ARP Request and Reply: From the host point of view, the network is a P2P network connection between the host and the switch port. Thus, ARP-request and replies are not relevant in this network.

Case 1: H_1 wishes to communicate with $A_1.B_1.C_1.D_1$ where the destination host is within the campus LAN.

As a first step, P_1 will evaluate if $A_1.B_1.C_1.D_1$ lie within the campus LAN. This is by comparing its Network ID based on its public address A.B.C.D and the network ID of $A_1.B_1.C_1.D_1$.

Since P_1 is a NATting point, it will first create an entry in the NAT-translational table. The source address is 192.168.0.2 and the destination address is $A_1.B_1.C_1.D_1$. The ports are copied as it is. P_1 then NAT it and forward it to the network using a modified ARP process specified in the PrECast infrastructure. As the first step, S_1 will check its PrECast table if there is a binding entry for <MAC, $A_1.B_1.C_1.D_1$ > binding. If it already exists, S_1 will transmit the IP packet encapsulated with a Layer 2 frame.

If there is no binding information available in its PrECast table, S_1 will multicast an ARP-request for the IP address $A_1.B_1.C_1.D_1$. The switch T to which the host $A_1.B_1.C_1.D_1$ is connected will multicast the ARP-reply to all LAN switches, including to S_1 . S_1 will then will transmit the IP packet encapsulated with a Layer 2 frame. All other switches, including S_1 , will add T's Switch ID, MAC of $A_1.B_1.C_1.D_1$ and the IP address $A_1.B_1.C_1.D_1$ to their PrECast-table.

If no switches in the network knew about $A_1.B_1.C_1.D_1$, the ARP-request will go for a timeout. After ARP-timeout, P_1 will issue an ICMP-host not found error to H_1 and remove the relevant entry from the NAT translational table.

Case 2: H_1 wishes to communicate with $A_1.B_1.C_1.D_1$ where the destination host is outside the campus LAN.

The switch S_1 checks the network ID of the IP packet destined to $A_1.B_1.C_1.D_1$. In this case, $A_1.B_1.C_1.D_1$ is outside of the subnet A.B.C.D. S_1 will NAT and forward the packet to its default gateway as in Case 1.

In both cases, the NAT++ engine checks if the source address is 192.168.0.2. If not, it will drop it silently. Using this mechanism, we effectively eliminated IP spoofing without incurring any extra

overheads. There is no complex source address checking or crypto-algorithm used in our scheme. The source address authentication is inherent from the P2P relation.

4.4. Network Maintenance

It is a practice in a corporate LAN that network administrators frequently push OS and software patches to every LAN hosts. With our new design, this service may not be affected. The network administrator may set local policy to obtain updates from a corporate server (that resides within the corporate LAN) frequently. Thus, this maintenance service is not affected by our proposed design.

Another task of a network administrator is to perform port scanning on all LAN hosts (mostly the corporate hosts and not BYO machines). The purpose of this exercise is to identify any vulnerable ports that are open to the external network.

Port Scanning has its legitimate uses inside the network, and therefore, needs to be supported by the proposed solution. The information gathered by a port scan has many legitimate uses including network inventory and the verification of the security of a network. The network scanner involves sending some especially crafted TCP/IP packets to the target host and observing their reply. The network scanning is always initiated from a client host towards a target host.

In our proposed NAT++ implementation, all end hosts apart from the important servers are hidden inside a NATed network. Any network host, including an administrator trying to reach an inside host, their connection will be considered to be “unsolicited” and will be dropped by the NAT engine. This is one of the strengths of our proposal. However, to facilitate port scanning for a legitimate purpose, we need to propose an alternative method.

In a NATed environment, port scanning is still possible from a host within the NAT realm. Since in our case, we have a P2P network, port scanning must be initiated only from 192.168.0.1. 192.168.0.1 is assigned to each switch ports to which end-hosts are connected. Thus, port scanning must be initiated only from the switch, not from any other hosts. Since the network administrators have legitimate access to corporate switches, they can connect with individual switches and perform port scanning across all connected hosts. By design, no other hosts can successfully perform a port scan.

4.5. The Attack Vector

In this subsection, we discuss various possible attacks against the NAT++ system.

IP Spoofing from Internal hosts: By design, this type of attack is not possible. Whenever a NAT++ port detects an outgoing IP packet with the return address not belonging to its only host, the packet will be dropped immediately without forwarding to the network. Thus, preventing spoofed IP address being generated inside the corporate network.

Address Spoofing by an External Host: We consider an external host H spoofing the corporate IP address A.B.C.D. If the host network does not implement the NAT++ protocol or other IP-address spoofing solutions, the IP packet will be forwarded to the destination host B faithfully. Based on the spoofed source address, Host B will send its reply to A. Being an unsolicited reply, the NAT++ engine will drop this packet silently.

Address spoofing that leads to DDoS attack: As in the previous case, in the absence of any forwarding entry, the NAT port will drop all the unsolicited replies arriving for the destination Host A. If this attack leads to “resource starvation”, then only the particular switch to which Host A is connected will be affected. The rest of the network can function without any problem. However, if the state-information is maintained at the corporate firewall, all the unsolicited replies can be dropped even at the gateway; thus, not affecting any network or host services.

Port Scanning attack: Every public-facing host, or a server may be subjected to this attack. By our design, all these hosts have private IP addresses. Since they are behind a NAT port no external host can probe their TCP or UDP ports.

Network Chaining: Controlling the network diameter in a corporate network provides low and predictable latency. The strict control of the network topology at the access layer should be maintained [15]. Users at the access layer tend to add switches to the existing network topology inappropriately to get connectivity to their personal devices. This problem is commonly known as “chaining”. Our proposed NAT++ infrastructure seamlessly solves this problem without any extra requirements.

4.6. Sharing Resources in a Corporate Network: Potential Challenges

In a corporate network, it is necessary to share resources across its users. This includes printers and shared storage. With the invention of Cloud-based services, many corporates are moving towards cloud-based storage services that provide more convenience, security, and redundancy. One such example is Microsoft’s OneDrive. However, an organization may be interested in sharing printers and share local files (ex. Group drives). The “Server Message Block” (SMB) Protocol [23] is a Network resource sharing protocol native to Microsoft Windows and widely supported under various platforms through Open Source Samba project [24]. The SMB protocol operates at the Application Layer of the OSI stack. It relies on lower-tier protocols for its transport. SMB uses NetBIOS over TCP/IP. The SMB protocol implements a client-server architecture. The current version (3.1.1) supports AES-128-GCM for encryption. This protocol is effectively exploited in every corporate network to share network resources. SMB uses TCP port 445 and UDP ports 139.

To evaluate potential challenges and configuration issues, we conducted two experiments under different environments.

1. We implemented a one-to-one NAT in a small network as outlined in Figure 5.
2. We implemented a one-to-one NAT in our University segment (as in Figure 6) and evaluated the ease of accessing shared services that are already available in our network.

Figure 5. Potential challenges and Configuration issues: Experiment 1.

Figure 6. Potential challenges and Configuration issues: Experiment 2.

Experiment 1: One-to-one NAT as per Figure 5.

In this experiment, we use the SMB protocol for creating a shared folder and a print server. We use the Windows Active Directory to create three access groups namely Student, Staff, and Admin. We use access control policies on the shared folder and the print server. Students have no access to the shared folder and print services. The Staff has read access to shared folders and no print services. Admin has full read and write access to the shared folder and full print services.

We split this experiment into two scenarios: In the first scenario, we hide the file server and the print server inside a one-to-one NAT. In the second scenario, file and print servers were connected to a switch port with no one-to-one NAT involved, as per our proposal. In this case, both the machines were given a public IP address within the corporate IP range.

Challenges faced in Scenario 1: With the default one-to-one NAT configuration on Switch 3, the hosts were unable to access both print and file server, irrespective of the user access right at the host. This is because, one-to-one NAT on Switch 3 was denying access to servers, as these connections were considered to be “unsolicited”.

Lesson Learned: We created a TCP hole punching at the NAT engine at Switch 3 for port 445 and UDP port 139 for the SMB protocol to listen for any incoming connections. However, hole punching is a manual process that needs to be done carefully (not to open other TCP or UDP ports by accident) by the network administrator. The hole punching permits the NAT-engine to accept unsolicited connections to these TCP and UDP ports.

Once we created a hole punch, we were able to access these services according to the access control policies we established for users. However, we incurred an average delay of 100 ms before we initiate these services.

Challenges faced in Scenario 2: Since the print and file servers have a public IP address, accessing them was easy compared with Scenario 1. Compared to Scenario 1 with hole punching, the latency in accessing these services could not be detected.

Challenges faced in Scenarios 1 and 2: In the absence of any “service catalog”, it is difficult to know what services are available in the network.

Lesson Learned: We created a simple intranet web page that consists of a catalog of network services available in the network along with its “Server name”. The name server is configured to resolve the Server name to its IP address to access these services.

Based on two scenarios, we noted that while file and print servers were configured with a public corporate IP address and no NAT, they provide less latency. Since all internal end-hosts (other than important servers) are protected by NAT++, there is no threat to these services (including poisoning or spoofing). With proper firewall rules, these services may be protected from external threats. This validates our proposal of not hiding essential servers under NAT++.

Experiment 2: One-to-One NAT integrated with the Campus network (Figure 6)

In this experiment, we were able to access all shared resources without any issue or with any noticeable latency.

In this experiment, we also launched two attacks, one originating from the host and another towards a host.

IP-Spoofing originating from a host: We created a spoofed IP packet with a source IP address that belongs to the public IP range. Since this address was not the same as 192.168.0.2, the NAT engine silently dropped; thus, preventing IP-Spoofing originating from inside hosts.

IP-Spoofing originating from the external host: We created a spoofed packet with a public source IP address that is being used by PC connected with Switch 1. This packet was then launched from an

external network where there is no NAT++ exists. The destination host faithfully sends its reply to the corporate host. Since this IP packet is deemed to be an unsolicited reply, the one-to-one NAT engine silently dropped this packet; thus, preventing IP-Spoofing originating from an external network

5. Simulation and Performance Analysis

In this section, we perform two different performance benchmarking to evaluate the strength of the proposed NAT++ framework.

- Analyse the performance of the NAT++ protocol against various NAT standards [21]. The purpose of this simulation exercise is to evaluate the additional workload introduced by the proposed NAT++ architecture in comparison with various standard NAT implementations.
- Compare the Central Processing Unit (CPU)-load on the first-hop network devices for an existing IP-spoofing solution and the NAT++.

There are two important parameters related to the performance of an IP-flow between two devices. The performance may either apply to end-host devices or the intermediate network devices such as a router:

1. CPU-load on the intermediate devices.
2. IP-Processing delay on the intermediate devices.

Whenever a NATting mechanism is implemented on an intermediate device, the delay due to the NAT process is added along with the IP-processing delay. As we discussed before, One-to-One NAT, Many-to-Few NAT and Many-to-One NAT (PAT) introduce different delays due to the complexity of their process.

In this section, we compare our proposed NAT++ scheme against the existing NAT standards through the above parameters.

Experimental Setup:

To evaluate the performance of the proposed NAT++ protocol, we created a small campus network given in Figure 7. We have connected 12 PCs each to Subnet 1 and Subnet 2 through Cisco 3560 switches. Cisco 2800 series routers were used at the access layer. ISR4000 series routers were used for the distribution layer and the gateway.

Figure 7. Experimental Topology.

Since we have no access to the Cisco IOS API while performing this work to implement our proposed protocol, we implemented the entire architecture using the OPNET modeler [25] to evaluate the strength. OPNET is a commercial discrete event simulation and modeling software that provides a comprehensive suite of vendor-models, in particular, Cisco switches and Routers. The use of these

switches and routers in modeling provides identical performance benchmarking as if we are using the original equipment. Thus, OPNET is used mostly in the commercial environment for designing large-scale corporate networks. Due to its flexibility, power, and the availability of vendor-specific models, we used OPNET for conducting this research.

Traffic Modelling

Each PC fetch web pages from the server. Once a page-fetch is complete, the host waits for random seconds (follows an exponential distribution with a mean of 120 s) before making another page request. For each connection request, the latency is recorded.

Experiment 1: Baseline experiment

In this experiment, we evaluate the IP processing delay due to the routing process at the router and the CPU load on the router. This experiment is conducted without any NAT. These baseline results are used for benchmarking purposes.

Experiment 2: One-to-One NAT

In this experiment, we created a one-to-one NAT at the subnet router (Subnet 1 and 2). These routers handle NAT while an IP-packet exit through its exit-interface connecting Subnets 1, 2, and the Distribution Layer router. The router also needs to maintain the translational state-information in a table for translating a reply from the server. Whenever a TCP handshake happens between the client and a server, the router performs the NATting task using the translational table. This is an additional load compared with the baseline experiment.

Experiment 3: Many-to-Few NAT

In this scheme, the total available IP address is less than the number of hosts. Whenever all public IP addresses are in use, any new outgoing connection is put on a queue until an on-going connection is closed after page-fetch. This may affect the Queuing delay.

Experiment 4: Many to One NAT or PAT

In this experiment, the IP address of the exit interfaces of the Subnet 1 and 2 routers are used for PAT. There is more processing involved in this type of NAT compared with one-to-one and many-to-few NAT.

Experiment 5: The NAT++

This experiment implements our NAT++ protocol. This scheme is similar to one-to-one NAT; however, every switch port handles only one NAT connection compared with the router (or traditionally NAT boxes) that handle all NAT connections. The NATed connections are directly switched on to the router's ingress interface (similar to VLAN trunking). In a traditional one-to-one NAT, all the NATting load is handled by the gateway router along with routing overload. In the case of NAT++, the NATting process is offloaded to LAN switches, and thus gateway routers incur no additional load due to the NATting process. Therefore, due to this experiment, the load on the gateway router is similar to the baseline test.

In all other experiments except NAT++, the LAN switches work as a Layer-2 device. However, in NAT++, it has to perform the NATting process. Thus, edge LAN switches incur additional CPU load and queuing delay due to the NATting process.

Experiment 6: Performance Benchmarking with a Similar solution

We implemented a solution similar to the one proposed in [16,18]. To detect IP spoofing more effectively and efficiently, the IP Source address checking process is implemented at the Ingress of the subnet routers. Every host that generates an IP packet will “digitally sign” the packet header to establish its identity. As soon as this packet is received by the router’s ingress link, the digital signature is verified. Once the identity of the source host is verified, the router then checks the source IP address. If there is a mismatch, the packet will be dropped silently.

This task involves processing at the host side and the verification at the router side. Signing and verification require CPU cycles. In this experiment, we evaluate the CPU load incurred at the host and the router side.

5.1. The Results

In this subsection, we present the results obtained through our experiments.

CPU Workload on the router:

In Figure 8, we presented the CPU load on the gateway router. As we expected, the baseline experiment without any NATting incurred the least CPU load. The CPU load due to our proposed NAT++ infrastructure is also similar to the baseline test. As we explained before, this is because the NATting process is done at the LAN switch. The next higher CPU load was due to one-to-one NAT followed by Many-to-few and many-to-one NAT.

Figure 8. CPU Workload on the router.

In Figure 9, we presented the CPU load on the 3560 LAN switch that was due to the NAT++ process. We then wished to compare the total CPU load incurred by the NAT++ process. Thus, we added the CPU load incurred at the Switch and the gateway router and compared it with one-to-one

NAT. The comparison is presented in Figure 10. As we can see from this result, the total IP processing delay is similar to one-to-one NAT.

Figure 9. CPU Workload on the switch.

Figure 10. Total CPU Workload on the router and the switch.

IP Processing delay:

From the results presented in Figure 11, it can be seen that IP processing delay at the router for No-NAT and NAT++ are similar. This is because NAT++ needs no translation similar to the no-NAT process. This is followed by one-to-one NAT and many-to-few NAT. Many-to-few NAT incurs the largest IP processing delay.

Traditionally, routers implement fast switching feature. Fast switching still uses the CPU, but after a packet has been forwarded, information about how to reach the destination is stored in a fast-switching cache. Whenever another packet to the same destination arrives at the router, the router will use the cache entry to forward through the correct interface.

Figure 11. IP Processing delay.

Benchmarking with a similar solution:

We implemented a solution similar to the one proposed by Wang et al. [16] and Fonseca et al. [18]. To detect IP spoofing more effectively and efficiently, IP Source address checking is implemented at the Ingress of the subnet routers. Every host that generates an IP packet will “digitally sign” the packet header to establish its identity. As soon as this packet is received by the router’s ingress link, the digital signature is verified. We implemented 1024-bit RSA as our underlying cryptosystem.

This task involves processing at the host side and the verification at the router side. Whenever an IP packet is received by the router’s ingress interface, as a first step, the integrity of the received packet is checked using the host’s public key.

Signing and verification require CPU cycles. In this experiment, we evaluate the CPU load incurred at the host and the router side. This is presented in Figure 12. The first hop header authentication incurred a total CPU load of 14.5%, whereas the NAT++ incurred only a CPU load of 4.5% on an average. This is because NAT++ does not use any cryptographic mechanism to check the validity of the source IP address.

Figure 12. CPU load based on first-hop authentication.

6. Conclusions and Future Direction

In this paper, we proposed a simple and scalable solution to thwart the IP-Spoofing problem that arises from an internal network. Our solution uses a modified form of NAT maintained at every

router port. The proposed NAT++ architecture effectively eliminates DoS attacks originating from inside malicious nodes. Since every internal host are hidden from every host, including external hosts, IP-spoofing attacks from the external network is not possible. The proposed architecture also solves the port-scanning attack from the external and internal networks. If the corporate firewall is configured with an effective flood protection mechanism, we can effectively eliminate DoS attacks that are originating from the external network. We implement NAT++ along with the PrECast infrastructure that eliminates MITM attacks in a LAN. Together with NAT++ and PrECast infrastructure, a network can defend against several known attacks.

In this paper, we offered a solution to protect a corporate network from attacks that are due to vulnerabilities in the IPv4 protocol. However, if the network implements a dual-stack architecture containing both IPv4 and IPv6 protocols, an attacker may still penetrate the network by exploiting some of the vulnerabilities found in the IPv6 protocol. Solving this problem is not within the scope of this paper. Our future work involves solution to IPv6 vulnerability issues.

Author Contributions: Conceptualization, writing—original draft preparation, editing P.V.; investigation, D.H.; supervision, writing—review E.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Graziani, R. *IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6*; Cisco Press: Indianapolis, IN, USA, 2017.
2. Google IPv6 Statistics. Available online: <https://www.google.com/intl/en/ipv6/statistics.html> (accessed on 7 July 2020).
3. BGP Table Data. Available online: <https://bgp.potaroo.net/index-bgp.html> (accessed on 20 July 2020).
4. Here's How Much Money a Business Should Expect to Lose If They're Hit with a DDoS Attack. Available online: <https://www.techrepublic.com/article/heres-how-much-money-a-business-should-expect-to-lose-if-theyre-hit-with-a-ddos-attack/> (accessed on 20 July 2020).
5. Cyber Security Market Size, Share & Trends Analysis Report, by Component, by Security Type, by Solution, by Service, by Deployment, by Organization, by Application, by Region, and Segment Forecasts, 2020–2027, Grandview Research (June 2020). Available online: <https://www.grandviewresearch.com/industry-analysis/cyber-security-market> (accessed on 7 September 2020).
6. Kammoun, N.; Bounfour, A.; Özyaygen, A.; Dieye, R. Financial market reaction to cyberattacks. *Cogent Econ. Financ.* **2019**, *7*, 1645584. [CrossRef]
7. Hanna, D.; Veeraraghavan, P.; Pardede, E. PrECast: An Efficient Crypto-Free Solution for Broadcast-Based Attacks in IPv4 Networks. *Electronics* **2018**, *7*, 65. [CrossRef]
8. Kerberos: The Network Authentication Protocol. Available online: <https://web.mit.edu/kerberos/> (accessed on 20 July 2020).
9. Steven Bass, Spoofed IP Address Distributed Denial of Service Attacks: Defence-in-Depth (2001) SANS Reading Room. Available online: <https://www.sans.org/reading-room/whitepapers/threats/paper/469> (accessed on 29 August 2020).
10. IP Spoofing: Simple Manipulation of Data Packets by Attackers. Available online: <https://www.ionos.com/digitalguide/server/security/ip-spoofing-fundamentals-and-counter-measures/> (accessed on 29 August 2020).
11. Park, K.; Lee, H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. *ACM SIGCOMM Comput. Commun. Rev.* **2001**, *31*, 15–26. [CrossRef]
12. Bremner-Barr, A.; Levy, H. Spoofing prevention method. In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), Miami, FL, USA, 13–17 March 2005; Volume 1, pp. 536–547.
13. Ehrenkranz, T.; Li, J. On the state of IP spoofing defense. *ACM Trans. Internet Technol.* **2009**, *9*, 6. [CrossRef]

14. Lichtblau, F.; Streibelt, F.; Kruger, T.; Richter, P.; Feldmann, A. Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses. In Proceedings of the 2017 Internet Measurement Conference, London, UK, 1–3 November 2017; pp. 86–99.
15. Oppenheimer, P. *Top-Down Network Design*; Cisco Press: Indianapolis, IN, USA, 2012.
16. Wang, H.; Jin, C.; Shin, K.G. Defense Against Spoofed IP Traffic Using Hop-Count Filtering. *IEEE/ACM Trans. Netw.* **2007**, *15*, 40–53. [[CrossRef](#)]
17. Mirkovic, J.; Kline, E.; Reiher, P. RESECT: Self-Learning Traffic Filters for IP Spoofing Defense. In *Proceedings of ACSAC 2017*; ACM: New York, NY, USA, 2017.
18. Fonseca, O.; Cunha, Í.; Fazzion, E.; Meira, W.; Junior, B.; Ferreira, R.A.; Katz-Bassett, E. Tracking Down Sources of Spoofed IP Packets, CoNEXT'19. In Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies, Orlando, FL, USA, 9–12 December 2019; pp. 51–53.
19. Rodriguez-Gomez, R.A.; Macia-Fernandez, G.; Garcia-Teodoro, P. Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.* **2013**, *45*, 45. [[CrossRef](#)]
20. Hanna, D.; Veeraraghavan, P.; Soh, B. SDMw: Secure Dynamic Middleware for Defeating Port and OS Scanning. *Future Internet* **2017**, *9*, 67. [[CrossRef](#)]
21. RFC 2663, IP Network Address Translator (NAT) Terminology and Considerations. 1999. Available online: <https://tools.ietf.org/html/rfc2663> (accessed on 29 August 2020).
22. Veeraraghavan, P.; Hanna, D.; Pardede, E. Building Scalable and Secure Multicast Delivery Infrastructure in a Local Area Network. *Electronics* **2019**, *8*, 1162. [[CrossRef](#)]
23. Microsoft SMB Protocol and CIFS Protocol Overview. Available online: <https://docs.microsoft.com/en-us/windows-server/storage/file-server/file-server-smb-overview> (accessed on 7 September 2020).
24. The Samba Protocol. Available online: <https://www.samba.org/> (accessed on 7 September 2020).
25. Yang, H.; Lu, Z. *Unlocking the Power of OPNET Modeler*; Cambridge University Press: Cambridge, UK, 2012.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).