Mitigating DRDoS Network Attacks via Consolidated Deny Filter Rules

Todd Booth[1] and Karl Andersson[2*]
[1]Digital Services and Systems
[2]Pervasive and Mobile Computing
Luleå University of Technology, 97187 Luleå, Sweden
{Todd.Booth,Karl.Andersson}@Ltu.Se

Abstract

This article is concerning distributed reflection denial of service (DRDoS) attacks. These DRDoS attacks are more frequent and large scale, and are one of the biggest threats on the Internet. This paper discusses the best way to defend from these attacks using public cloud defenses, such as Amazon AWS, Google GCP, and Microsoft Azure, at a very low cost. Our mitigation strategy takes advantage of the fact that the attacker does not have full control to change the source IP port to anything they want, when used in these reflective attacks. We propose to have the customer host their Web servers and other types of supporting servers in the public cloud. The cloud provider then reserves a /CIDR block of IP addresses, which will be protected. The cloud providers customers who opt in, will be allocated an IP address from this block. This block will be used as the source IP address deny portion of the firewall rule-sets. Then the public cloud providers will use BGP4 Flow-Spec or some scripting solution, to have their IP service provider neighbors perform the actual filtering of the DRDoS attack traffic concerning attacks against these servers.

Keywords: DDoS, DRDoS, BGP4 Flow-Spec, Cloud security

1 Background and Motivation

Distributed denial of service (DDoS) attacks are a subset of denial of service (DoS) attacks [24]. DDoS reflective attacks (DRDoS) are a subset of DDoS attacks. Our paper is concerning a subset of DRDoS reflective attacks, which are volumetric in nature (DRDoS Volumetric), which is a layer 3 network attack. There exist both UDP and TCP DRDoS reflective attacks. This paper is concerning such UDP and TCP attacks and as an example, we'll use the attacks which abuse the memcached protocol. One such DRDoS Volumetric attack, is the memcached UDP amplification attack and a recent attack was clocked at 1.7 Tb/sec [18]. Assume that the ISP's customer's on premise network is attached via an ISP with a 10Gb/sec connection. If the customer's on premise server is attacked, their ISP connection would have been overwhelmed. Even if the customer purchased an extra 10Gb/sec connection, their ISP connect would have been overwhelmed.

The only possible defense for this size attack is to filter the traffic before it reaches the customer's on premise network. One possible approach is to try and filter the attack traffic at the customer's ISP. There is a protocol which is designed for this, which is called BGP4 Flow-Spec. The customer's BGP4 signals the ISP's BGP 4, as to what traffic should be filtered. However, most ISP's don't support BGP 4 Flow-Spec for their customers. Also, it is often the case that there is such a high bandwidth of attack traffic, that it would overwhelm the ISP.

Another approach is to use BGP4 to reroute the customer's incoming traffic to go via a DDoS scrubbing service, which will try to filter at least most of the attack traffic. However, there is a delay in time,

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 6, Article No. 2 (October 1, 2020)
*Corresponding author: Tel: +46 (0)910 585364

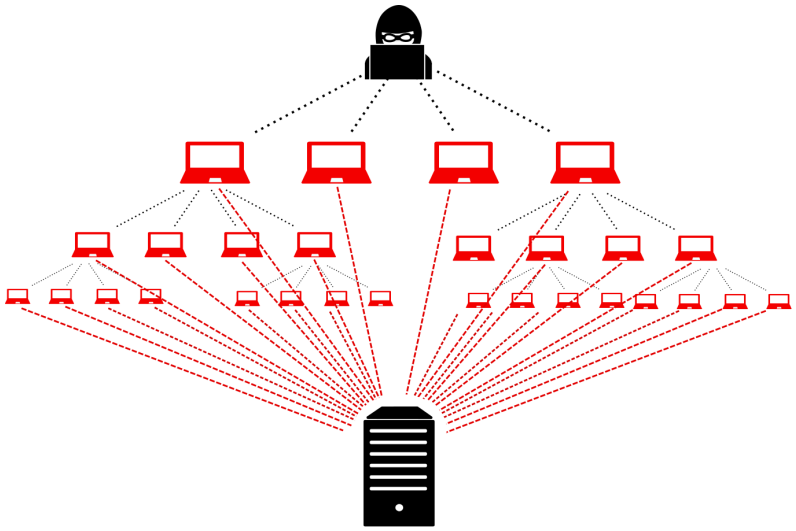


Figure 1: DRDoS Volumetric Attack with a Botnet

which is measured in minutes, for the traffic to reroute. In the meantime, the site will be down and this strategy is quite expensive.

Another approach is for the customer to run their servers in a public cloud, which is part of the solution proposed by this article. One benefit is that via the Microsoft Azure cloud, for example, the default speed for instances is 10Gb/sec per VM guest, or instance. If the customer wants more bandwidth, they can attach multiple interfaces to the instance or run multiple instances. Still, the attack traffic bandwidth could easily overcome this speed. However the 10Gb/sec is independent of traffic filtered by the network security group (NSG) firewall. For example, if 20Gb/sec was filtered by the network security group (NSG) firewall, it would not effect the allowed 10Gb/sec to the instance. For a short introduction to DRDoS Volumetric attacks, with the memcached protocol, see [1]. If a cloud infrastructure is attacked, there may be charges for incoming bandwidth, which can be extremely high.

General DRDoS attacks work as follows. The network attacker spoofs their source IP address, to be that of the intended victim. They then send requests to a third-party server, which most often is innocent. An example DRDoS attack is shown in figure 1.

When the third-party server replies to the spoofed IP address, the response goes to the victim server. Since the attacking clients spoof their IP addresses, it is extremely difficult to find out the attacking clients' true IP addresses. DRDoS attacks are often performed via botnets, which can include thousands of clients, which are infected with malware. DRDoS Volumetric specific attacks work as follows: A small request may be sent to the third-party server, and the server will respond (to the victim) with a much larger payload. For example, with the memcached UDP DRDoS attack, the attacker can send requests to the third-party server, which will generate much more traffic in the reply, which goes to the victim. According to the USA Cert authority, the amplification of attack data can be up to 51,000 times for a DRDoS memcached UDP attack [8].

It is very easy to launch a DRDoS Volumetric attack, but very hard to defend against. There are even services which will launch these attacks for you, for a small fee. You just surf to the right anonymous Web site, provide your credit card, enter information on what you want attacked and when.

There exist various DRDoS protection services, however they are expensive. Microsoft has an Azure DDoS Protection Standard service [14], which costs $2,944 USD/month plus $0.05/GB of incoming traffic. For an overview of Azure's DDoS protection, see [17]. Amazon's AWS has a DDoS protection service named AWS Shield [2] which costs $3,000 USD/month with some additional traffic-based costs.

This article's goal is to mitigate DRDoS Volumetric attacks in the public cloud, for no additional cost, other than, for example, the virtual guest instance servers cost, which starts at about $30 USD/month per server instance.

To understand our proposed solution, you will need to understand how IPv4 IP addresses are used and how ports are classified. IPv4 ports are classified, as shown in figure 2. A list of some of the most

Port Number	Description
0 - 1023	Well known ports
1024 - 49151	Registered ports
1025 - 5000	Ephemeral or Dynamic ports (earlier used by Windows)
9152 – 65535	Ephemeral or Dynamic ports (IANA recommendation)

Figure 2: Classification of IPv4 Ports

common UDP ports used in larger DRDoS Volumetric attacks are found in figure 3.

Protocol	Port	Amplification Factor
DNS	53	54
NTP	123	557
CharGEN	19	359
QOTD	17	140
RIPv1	520	131
Memcached	11211	51,000

Figure 3: Popular UDP Ports and Amplification Factor Used in DRDoS Attacks

A list of some of the most common TCP ports used in larger DRDoS attacks are found in figure 4

Protocol	Port
FTP	20, 21
SSH	22
Telnet	23
SMTP	25
HTTP	80, 443
POP3	110
NNTP	119
NetBIOS	137, 139
IMAP	143
IPP	631
MySQL	3306
SIP	5060-5061
IRC	6660-6669, 7000
Memcache	11211

Figure 4: Popular TCP Ports and Amplification Factor Used in DRDoS Attacks

Here is a specific example, of a DRDoS Volumetric attack. Assume the following network: client with malware, which is part of a botnet: IP Address: 100.0.0.1. Third-party server (which are most

often innocent): 200.0.0.1, running the memcached server listening on UDP port 11211. Victim server: 300.0.0.1. Assume the victim server is running a Web server, on both ports 80 and 443. We are trying to protect the victim server, which in our scenario, is in the Azure public cloud. The malicious client could craft packets with a spoofed source IP address of 300.0.0.1 and send those packets to the third party memcache server, with a destination port of 11211. The server would try to reply by sending traffic to the received source IP address of 300.0.0.1, with a source port of 11211. However, the return traffic would not go to the client, since its IP address was spoofed, to be that of the victim server. The return traffic would go to the victim server. Further, the victim server would think it's being attacked by the 3rd party server, since that is the source IP address received.

All valid client to Web Server traffic should have the client's source IP ports, which would normally be in the dynamic port range of 1025 - 5000, or 49152 – 65535. So the Web client should use these dynamic port ranges for valid requests. For a DRDoS UDP attack, for the traffic received by the victim, the incoming source port will correspond to the third-party server service listening UDP port, in the range from 0 - 49151, with the exception of valid addresses, with some specific examples shown in figure 3 and figure 4. When using the memcached server DRDoS attack, the source port would be 11211. We use this to our advantage, when creating the proposed firewall rules defense solution.

The rest of this article, is as follows. We have a related works section 2, followed by our proposed method section 3, then our conclusions 4.

2 Related Works

In [4], we present one solution to eliminate UDP DRDoS attacks, for TCP services. In [5], we show that it is not feasible to defend against large DRDoS attacks at the company's premises or at most ISPs. In [6], we introduce the ability to perform redirects, but only for authenticated clients, to mitigate DDoS attacks. In [7], we show how authenticated users can be broken into small groups, and put on different servers. This then allows us to hide the IP addresses of all the servers, as a means to mitigate the DDoS attacks. In [12], Kuhrer, et. al. present the TCP reflection amplification attacks problem, but no solution was presented.

In [23], Wei, et. al provide an algorithm to detect Reflection DDoS attacks. In [13], Lyu, et. al. present the problem of Reflective DDoS attacks from IoT devices. In [10], Fachkha, et. al. propose a novel approach to infer and characterize Internet-scale DNS Distributed Reflection Denial of Service attacks by leveraging the darknet space. In [11], Jonker, et. al. provide a characterization of DRDoS attack phenomenon, and of countermeasures to mitigate the associated risks.

In [9], Chen, et. al. describe how software defined networking (SDN) can be used to detect DRDoS attacks. In [21], Thomas, et. al. describe their experience in setting up honeypots to learn how scanning of hosts works, to find innocent hosts, which can be used as DRDoS servers. In [19], Sharma, et. al. propose a self-aware communications architecture for IoT using mobile fog servers. The proposed approach is evaluated against DDoS attacks for analyzing it sustainability. In [27], Zargar, et. al. present a survey of DDoS flooding attacks and claim these are one of the biggest concerns for security professionals. They classify existing counter-measures, but they don't discuss our proposed solution. In [20], Silva, et. al. present a survey of botnets, which are used to perform DDoS attacks. They include a discussion of persistent research problems which remain open.

In [22], Wang, et. al. find that SDN technology can help enterprises to defend against DDoS attacks. In [25], Yan, et. al. show how although SDN can be used as a DDoS defense, a DDoS can attack the SDN technology. They discuss a number of challenges that need to be addressed to mitigate DDoS attached in SDN with cloud computing. In [26], Yu, et. al. propose a dynamic resource allocation strategy to counter DDoS attacks against individual cloud customers. They propose a solution, by scaling up,

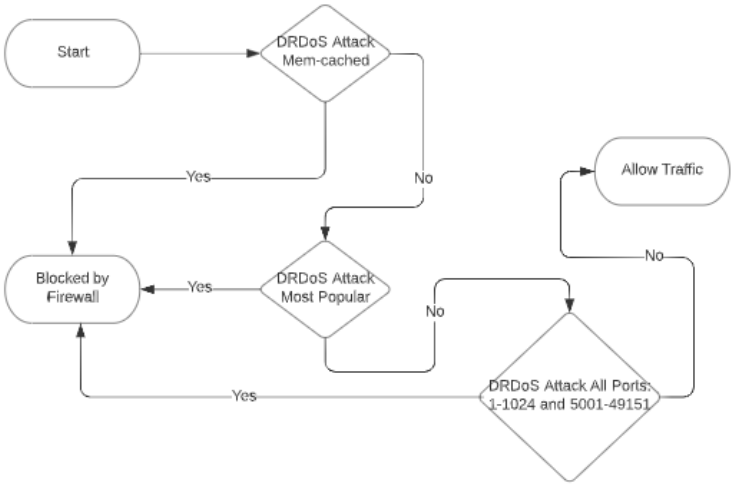


Figure 5: Firewall Process Flow for DRDoS Attacks

during an attack by using idle resources.

3 Proposed Method

The figure 5 shows the firewall process flow, which is now presented step by step, in the next few pages.

With Azure, we can have both deny and allow network security group (NSG) rules and the rules can include the source ports. For a detailed description of Asuzre's NSGs, see [16]. However, with Amazon's AWS security groups, we can only have allow rules, and can't filter (deny) on source ports, see [3]. For this paper, but just for the moment, we will now focus on Azure. Just the incoming bandwidth would otherwise easily over saturate the incoming Azure instance NIC speed, which is 10 Gb/sec. With the Azure Accelerated Networking feature, the interface speed is limited to about 25 Gb/sec per virtual instance. We could add multiple interfaces to the instance to increase bandwidth.

Let's suppose we want to stop the DRDoS attacks, which use the third-party services, with TCP ports shown in figure 4 and with UDP ports shown in figure 3. Let's assume that the Web server, that we wish to protect, has an IP address of 11.0.0.1. Let's assume that we wish to allow ICMP traffic which is going to the Web Server. Consider the corresponding firewall filter rule-set in figure 6.

IP Destination Address	Protocol	Destination Ports	Action
11.0.0.1	TCP	80,443	Allow
11.0.0.1	ICMP	N/A	Allow
11.0.0.1	Any	Any	Deny

Figure 6: Firewall Rule-set 1

One might think that the rule-set would stop all DRDoS traffic to the Web server, however we'll show an issue with it. Let's assume that the attacker sends the following attack traffic:

Type=TCP, **source IP**=victim server, **source port**=443, **destination IP**=3rd party server, **destination port**=DNS

The 3rd party server would answer with a packet such as the following:

Type=TCP, **source IP**=3rd party server, **source port**=DNS, **destination IP**=victim server, **destination port**=80,443

The above attack traffic would bypass our above firewall rule-set. Assume that we wish to stop TCP and UDP DRDoS attacks which abuse the memcached services, which listen on port 11211. In this case, we could filter based on the TCP and UDP source port of 11211, using NSG stateless rules. If a third party memcached server is used for the DRDoS attack, the source port would be 11211. This would completely defeat the DRDoS reflective attack via the 3rd party memcached server. Note that our proposed solution shown later will do for a large group of Web servers in an efficient manner, and we will filter it at Azure's IP neighbors before the traffic is even received by Azure.

We will now generalize our strategy to stop other non-memcached DDRoS attacks. We could use the same strategy to stop third party DNS attacks, by also filtering on TCP and UDP source port 53. We could use the same strategy to stop DRDoS attacks for NTP, CharGEN, QOTD RIPv1, FTP, Telnet, SMTP, etc., by filtering on their TCP and UDP source ports. We are using the specific UDP firewall rule, so that we can get statistics on the number of UDP packets dropped. We are using a specific firewall drop all rule so that we can get statistics on those packets. Since the Web server does not need to receive any UDP traffic, we'll just use a rule to drop it all. Note that ICMP does not use source or destination ports. The NSG firewall rule-set is shown in figure 7.

IP Destination Address	Protocol	Source Ports	Destination Ports	Action
11.0.0.1	TCP	53,123,19,17,520,11211	80,443	Deny
11.0.0.1	TCP	Any	80,443	Allow
11.0.0.1	ICMP	N/A	N/A	Allow
11.0.0.1	UDP	Any	Any	Deny
11.0.0.1	Any	Any	N/A	Deny

Figure 7: Firewall Rule-set 2

We could use the same strategy to filter attacks from all well known source ports of 0-1023. We could use the same strategy to filter attacks from all registered ports of 1024 - 49151, with the exception of 1025 - 5000, which are the old Windows valid dynamic source IP addresses. Instead of stating which are all the hundreds of ports which are disallowed, we'll provide a range of which ports are allowed and then deny the rest. Here are our proposed Azure NSG filter stateless rules, which can filter and drop all of the malicious DRDoS traffic which use the third-party services listed in figure 3 and figure 4. Note that Azure does not charge anything extra for this incoming DRDoS incoming filtered traffic, which is filtered via the Azure NSG feature [15]. We will not lose any valid traffic since the valid Web client source ports are found in that rule-set. This more comprehensive firewall rule-set is shown in figure 8

IP Destination	Protocol	Source Ports	Destination Ports	Description
11.0.0.1	TCP	1025 - 5000	80,443	Allow
11.0.0.1	TCP	9152 - 49151	80,443	Allow
11.0.0.1	ICMP	N/A	N/A	Allow
11.0.0.1	UDP	Any	Any	Deny
11.0.0.1	any	Any	Any	Deny

Figure 8: Firewall Rule-set 3

This will filter out UDP and TCP DRDoS attack traffic. Our proposed Azure NSG rules would

filter DRDoS TCP and UDP memcached and other server malicious traffic, via simple stateless filtering rules, which execute very fast. On the other hand, filtering with stateful firewall rules can sometimes be executed more slowly, since the state tracking table would need to be inspected for each packet.

So even if there was 1 Tb/sec of an incoming DRDoS volumetric attack traffic, there would be no additional charge for this bandwidth and the traffic would not hit our virtual server instance. Again the alternative is Microsoft's DDoS Protection Standard service, for $2,944 USD/month, which of course has some additional features, and we have solved the problems in another way, which is much less expensive.

Likewise, as previously shown for Memcache servers, our technique would filter the following server DRDoS traffic: FTP, HTTP, IMP, IPP, NetBIOS, and all other protocols shown in the figure 3 and figure 4. Further, our solution would also filter all DRDoS traffic from servers who listen on any standard TCP or UDP port, which are within the range 0 – 1024, 5001 - 9151, and 49152 - 65,535.

This, in effect, mitigates any and all DRDoS attacks where the third-party server is configured to listen on any of these given ports. With this solution, the DRDoS would be filtered by the NSG using stateless firewall rules. Therefore, the Web server would not need to process any of the attack traffic, which would otherwise easily overwhelm the server. So far, with our proposed method, it would be Microsoft's Azure which both receives the malicious traffic and filters the malicious traffic. We will now greatly improve upon our firewall rule-set design. We propose that Microsoft does not receive the malicious traffic to begin with. We propose that Microsoft requests their IP neighbors to filter this traffic instead of sending the malicious traffic to Microsoft. There are various approaches which can be used by Microsoft to inform their IP neighbors as to what should be filtered.

One approach is that Microsoft and its Internet IP neighbors implement the BGP4 Flow-Spec protocol. BGP4 Flow-Spec would allow Microsoft to signal its IP neighbors to perform the filtering. The other approach is to use scripts to create the firewall rule-sets. For example, Microsoft could create scripts for the popular network router products used by their IP neighbors.

One problem with our rule-set in figure 8, is we would need a different rule-set for each of Microsoft's customers. For example, if Microsoft's 2nd customer had the IP address 13.0.0.3, the firewall would need to be updated to accommodate to reflect that, as shown in figure 9.

IP Destination	Protocol	Source Ports	Destination Ports	Description
11.0.0.1	TCP	1025 - 5000	80,443	Allow
11.0.0.1	TCP	9152 - 49151	80,443	Allow
11.0.0.1	ICMP	N/A	N/A	Allow
11.0.0.1	UDP	Any	Any	Deny
11.0.0.1	any	Any	Any	Deny
13.0.0.3	TCP	1025 - 5000	80,443	Allow
13.0.0.3	TCP	9152 - 49151	80,443	Allow
13.0.0.3	ICMP	N/A	N/A	Allow
13.0.0.3	UDP	Any	Any	Deny
13.0.0.3	any	Any	Any	Deny

Figure 9: Firewall Rule-set 4

If there was one Microsoft customer who desired this feature, five rules would be required. If there were two Microsoft customers who desire this feature, ten rules would be required. If there were 10,000 Microsoft customers who desire this feature, 50,000 rules would be required, which is much larger than BGP4 Flow-Spec supports. For example, Cisco only supports 3,000 filter rules to be configured with their BGP4 Flow-Spec rule-sets.

We are now ready to explain our main contribution in this article. Our proposed solution would operate as follows: Microsoft would reserve a single class B block of IP addresses, which could be used by all Web server customers, who wish to be protected from these DRDoS attacks. In advance, Microsoft would ask its IP neighbors to filter DRDoS traffic to this block of IP addresses. By IP neighbors, we mean the service providers that have a direct Internet connection to Microsoft. For example, let's assume the class B block of IP addresses is 100.0.0.0/16. The rule-set in figure 10 that Microsoft would ask its IP neighbors to execute:

IP Destination	Protocol	Source Ports	Dest. Ports	Description
100.0.0.0/16	TCP	1025 - 5000	80,443	Allow
100.0.0.0/16	TCP	9152 - 49151	80,443	Allow
100.0.0.0/16	ICMP	N/A	N/A	Allow
100.0.0.0/16	UDP	Any	Any	Deny
100.0.0.0/16	Any	Any	Any	Deny

Figure 10: Firewall Rule-set 5

With the class B IP address block, Microsoft could support 65,534 customers with the above five filter rules. When Microsoft's customers provision an instance, for example a Web server, the customers would have one additional configuration option. This option would allow the customers to opt in to our proposed Web server DDoS protection service. If the customer opted in, they would be allocated an IP address, from this reserved class B IP address block. This way, as new customers opt in, Microsoft does not require its IP neighbors to make any additional firewall rule-set changes, to protect the new customers, I.E., those customers provisioned with an IP address, from the class B block, after Microsoft's IP neighbors have implemented the requested rule-sets. The above can be implemented via BGP4 Flow-Spec, which allows Microsoft's IP neighbors to send monitoring data to Microsoft, for example how many packets were filtered.

However, perhaps not all of Microsoft's neighbors support BGP4 Flow-Spec. Or perhaps Microsoft's neighbors don't want to implement this with BGP4 Flow-Spec. In this case another approach would be required. One such alternative if for Microsoft and its IP neighbors to use some type of network scripting approach, which is often used as an alternative to BGP4 Flow-Spec.

Note that Microsoft and other public cloud providers offer content delivery networks (CDN), which provide numerous CDN hosts throughout the world. They are used to, for example, cache web pages closer to end users. To achieve this, IP anycast is used, which redirects users to the closest CDN edge location. Therefore, Microsoft would need to ask their neighbors, at each CDN edge location to perform the filtering rule-sets proposed.

The filtering would prevent management of the servers via direct connections, for example SSH or RDP. One way to support management, would be to update the rule-set, as shown in figure 11.

An alternative would be to allow management of the Web server, via the use a bastion host. The above places the burden of filtering on Microsoft's IP neighbors. However, via recursion, Microsoft's IP neighbors could ask their IP neighbors to perform the filtering. And so on, and so forth, the traffic filtering could be moved closer to the 3rd party server, which is reflecting the malicious traffic.

For example, Microsoft's neighbors could also send the BGP4 Flow-Spec messages to their neighbors, etc., to have the filtering performed before they receive the malicious traffic. We could even have ISPs, who service normal end users implement the filter rules, even if their upstream neighbors don't. This could be done via loading filter rules from static files, with scripts created by Azure and other public cloud vendors.

IP Destination	Protocol	Source Ports	Destination Ports	Action
100.0.0.0/16	TCP	1025 - 5000	80,443,22,3389	Allow
100.0.0.0/16	TCP	49152 - 65535	80,443,22,3389	Allow
100.0.0.0/16	UDP	1025 - 5000	3389	Allow
100.0.0.0/16	UDP	49152 - 65535	3389	Allow
100.0.0.0/16	ICMP	N/A	N/A	Allow
100.0.0.0/16	Any	Any	Any	Deny

Figure 11: Firewall Rule-set 6

If the Microsoft customer wishes to opt-out, they could then be assigned a new IP address, outside of the filter range, which is outside the class block B IP address set.

Even though Amazon AWS does not support filtering via source ports, BGP4 Flow-Spec does, so AWS could push Flow-Spec filtering rules to it's neighbors, just like we've illustrated via Microsoft's Azure. Or AWS could use scripting with their IP neighbors. This would give AWS a similar feature which it currently lacks. This brings us new opportunities in efficiency.

4 Conclusion

DRDoS attacks are common and powerful. DRDoS high bandwidth attacks are not able to be defeated, at the on premises location. Defenses are often expensive costing $3,000 USD/month plus traffic costs. We have shown a defense in depth strategy, whereby the DRDoS reflective attack traffic can eliminated, without any additional costs, over the basic virtual instance costs. Our solution specifically prevents DRDoS attacks from servers which listen on ports 1 - 1024 and 5001 - 49151, which is by far most of the ports used in DRDoS attacks.

We have even found a way that the public cloud provider can have their IP neighbors perform the malicious traffic filtering. This way, the malicious traffic does not reach the public cloud providers, in the first place. Attackers will quickly realize that their DRDoS attacks are not successful, and they will normally stop these attacks.

Attackers will soon realize that to perform a DDoS against the cloud Web servers will require them to perform direct attacks, instead of indirect DRDoS attacks. Attackers will then trying sending traffic directly, but spoofing their source IP address. However about 70% of the ISPs prevent their clients from spoofing their IP addresses. For the botnets that use the this 70%, their clients will need to send traffic without spoofing, which will expose their IP address to the attacked servers. This makes it easier to add the malicious botnet clients, via their exposed IP addresses, to lists of malicious traffic to be filtered. This makes it much easier for anti-malware to remove the malware from these botnet clients and easier to setup filters to filter traffic from these systems. In summary, when the attackers realize we have such a strong defense, they will most likely move on to attacking some other softer targets.

Our study was to mainly to protect Web servers. However, our proposed solution also solves the problem when you wish to protect a server which is running other UDP or TCP services. Our case study example was with the Microsoft Azure cloud, but applies to all public cloud providers.

References

[1] Akamai. Memcached UDP Reflection Attacks, `https://blogs.akamai.com/2018/02/memcached-udp-reflection-attacks.html`, [Online: Accessed on June 25th, 2020].

[2] Amazon. Amazon AWS DDos Shield, https://aws.amazon.com/shield, [Online: Accessed on May 21st, 2020].

[3] Amazon. Amazon AWS EC2 Security Group Rules, http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html#security-group-rules, [Online: Accessed on June 17th, 2020].

[4] T. Booth and K. Andersson. Elimination of DoS UDP reflection amplification bandwidth attacks, protecting TCP services. In *Proc. of the 1st International Conference on Future Network Systems and Security (FNSS), Paris France, LNCS*, volume 523, pages 1–15. Springer International Publishing, Springer-Verlag, May 2015.

[5] T. Booth and K. Andersson. Network Security of Internet Services: Eliminate DDoS Reflection Amplification Attacks. *Journal of Internet Services and Information Security (JISIS)*, 5(3):58–79, August 2015.

[6] T. Booth and K. Andersson. Network DDoS Layer 3/4/7 Mitigation via Dynamic Web Redirection. In *International Conference on Future Network Systems and Security*, pages 111–125. Springer International Publishing, October 2016.

[7] T. Booth and K. Andersson. Critical Infrastructure Network DDoS Defense, via Cognitive Learning. In *Proc. of the 14th IEEE Consumer Communications & Networking Conference (CCNC'2017), Las Vegas, USA*, pages 1–6. IEEE, January 2017.

[8] U. Cert. US Cert UDP-Based Amplification Attacks, https://www.us-cert.gov/ncas/alerts/TA14-017A, [Online: Accessed on June 23rd, 2020].

[9] C.-C. Chen, Y.-R. Chen, W.-C. Lu, S.-C. Tsai, and M.-C. Yang. Detecting amplification attacks with Software Defined Networking. In *Proc. of the 2017 IEEE Conference on Dependable and Secure Computing (DSC,2017), Taipei, Taiwan*, pages 195–201. IEEE, August 2017.

[10] C. Fachkha, E. Bou-Harb, and M. Debbabi. Inferring distributed reflection denial of service attacks from darknet. *Computer Communications*, 62:59–71, May 2015.

[11] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti. Millions of targets under attack: A macroscopic characterization of the DoS ecosystem. In *Millions of targets under attack: A macroscopic characterization of the DoS ecosystem*, volume Part F131937, pages 100–113, November 2017.

[12] Kuhrer, et. al. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks, https://www.usenix.org/system/files/conference/woot14/woot14-kuhrer.pdf, [Online: Accessed on May 13th, 2020].

[13] M. Lyu, D. Sherratt, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman. Quantifying the reflective DDoS attack capability of household IoT devices. In *Proc. of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'17), New York, USA*, pages 46–51. ACM, July 2017.

[14] Microsoft. Microsoft Azure DDoS Protection, https://azure.microsoft.com/en-gb/services/ddos-protection/, [Online: Accessed on June 19th, 2020].

[15] Microsoft. Microsoft Azure Pricing Details Bandwidth, https://azure.microsoft.com/en-us/pricing/details/bandwidth/, [Online: Accessed on June 19th, 2020].

[16] Microsoft. Microsoft Azure VNet Security Overview, https://docs.microsoft.com/en-us/azure/virtual-network/security-overview, [Online: Accessed on May 10th, 2020].

[17] Microsoft. Microsoft DDoS Protection Overview, https://docs.microsoft.com/en-us/azure/virtual-network/ddos-protection-overview, [Online: Accessed on May 21st, 2020].

[18] Netscout. Memcached Attack 1.7 Tb/sec Security Center Integration, https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era, [Online: Accessed on June 19th, 2020].

[19] V. Sharma, J. Lim, J. Kim, and I. You. SACA: Self-Aware Communication Architecture for IoT Using Mobile Fog Servers. *Mobile Information Systems*, 2017, April 2017.

[20] S. Silva, R. Silva, R. Pinto, and R. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, February 2013.

[21] D. Thomas, R. Clayton, and A. Beresford. 1000 days of UDP amplification DDoS attacks. *2017 APWG Symposium on Electronic Crime Research (eCrime)*, April 2017.

[22] B. Wang, Y. Zheng, W. Lou, and Y. Hou. DDoS attack protection in the era of cloud computing and Software-

Defined Networking. *Computer Networks*, 81:308–319, April 2015.

[23] W. Wei, F. Chen, Y. Xia, and G. Jin. A Rank Correlation Based Detection against Distributed Reflection DoS Attacks. *IEEE Communications Letters*, 17(1):173–175, January 2013.

[24] Wikipedia. Wikipedia DoS, `https://en.wikipedia.org/wiki/Denial-of-service_attack`, [Online: Accessed on June 23rd, 2020].

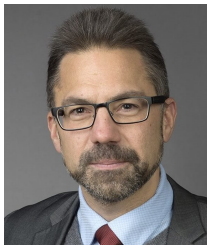
[25] Q. Yan and F. Yu. Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Communications Magazine*, 53(4):52–59, April 2015.

[26] S. Yu, Y. Tian, S. Guo, and D. Wu. Can we beat DDoS attacks in clouds? *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2245–2254, July 2014.

[27] S. Zargar, J. Joshi, and D. Tipper. A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks. *IEEE Communications Surveys and Tutorials*, 15(4):2046–2069, March 2013.

Author Biography

Todd Booth has over 20 years experience in computer networking and computer security. He has a B.Sc. in Mathematics/Computer Science from UCLA with honors and has taught networking and programming classes at UCLA. He has a M.Sc. in Computer Science related Information Security from the University of Technology (in Luleå, Sweden). He has been teaching masters level information security courses full time, for nine years, and has received the teacher of the year award. He has been recognized by IBM as a visiting scientist. He is now a Ph.D. student at Luleå University of Technology, Sweden with a research discipline of Computer Science related network security, including DDoS and network authentication.



Karl Andersson has a M.Sc. degree in Computer Science and Technology from Royal Institute of Technology, Stockholm, Sweden and a Ph.D. degree in Mobile Systems from Luleå University of Technology, Sweden. After pursuing postdoctoral research at the Internet Real-time Laboratory at Columbia University, New York, USA and National Institute of Information and Communications Technology, Tokyo, Japan, he is now Associate Professor of Pervasive and Mobile Computing at Luleå University of Technology, Sweden. His research interests include Green and Mobile Computing, the Internet of Things, Cloud Technologies, and Information Security.