Computer Communications 69 (2015) 107–115

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

SF-DRDoS: The store-and-flood distributed reflective denial of service attack

Bingshuang Liu[a,b], Jun Li[c], Tao Wei[d], Skyler Berg[c], Jiayi Ye[a], Chen Li[a], Chao Zhang[d], Jianyu Zhang[b], Xinhui Han[a,*]

[a] Institute of Computer Science and Technology, Peking University, Beijing 100080, China
[b] National Computer Network Emergency Response Technical Team/Coordination Center (CNCERT/CC), Beijing 100029, China
[c] University of Oregon, Eugene, OR 97403, USA
[d] University of California, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Available online 22 June 2015

Keywords:
DDoS
DRDoS
Amplification factor
Kademlia
Store-and-flood

ABSTRACT

Distributed reflective denial of service (DRDoS) attacks, especially those based on UDP reflection and amplification, can generate hundreds of gigabits per second of attack traffic, and have become a significant threat to Internet security. In this paper we show that an attacker can further make the DRDoS attack more dangerous. In particular, we describe a new DRDoS attack called **store-and-flood DRDoS**, or **SF-DRDoS**, which leverages peer-to-peer (P2P) file-sharing networks. An attacker can store carefully prepared data on reflector nodes before the flooding phase, to greatly increase the amplification factor of an attack. In this way, SF-DRDoS is more surreptitious and powerful than traditional DRDoS. We present two prototype SF-DRDoS attacks on two popular Kademlia-based P2P file-sharing networks, Kad and BT-DHT. Experiments in real-world environments showed that, this attack can achieve an amplification factor of 2400 on average in Kad, and reach an upper bound of attack bandwidth at 670 Gbps and 10 Tbps for Kad and BT-DHT, respectively. We also propose some candidate defenses to mitigate the SF-DRDoS threat.

1. Introduction

While distributed denial of service (DDoS) attacks have posed a significant threat to Internet security for many years, recently distributed reflective denial of service (DRDoS) attacks have become prevalent and received a lot of attention due to their severity. For example, CloudFlare observed a UDP-based DRDoS attack [1] in February 2014, which generated over 400 Gbps attack traffic through NTP amplification, targeting a French hosting provider. This attack is one of the largest DDoS attacks in history, with a tremendous traffic volume that could bring down virtually any service on today's Internet.

In a typical DRDoS attack, the attacker first sends many requests with a spoofed source IP address (i.e., the victim's address) to intermediate nodes (i.e., *reflectors*), which in turn reply with numerous and often voluminous responses to the spoofed IP, thereby flooding the victim. Two key metrics for measuring the severity of a DRDoS attack are **amplification factor** (**AF**) and **attack ability**. The amplification factor is the ratio between the traffic volume of response packets and that of request packets, reflecting the resource cost ratio between attackers and victims. The attack ability is the absolute amount of attack traffic launched toward the victim. Please note that reflectors are usually meant to provide a legitimate service and can hardly realize that they are being exploited to produce a large attack traffic.

Unfortunately, DRDoS attacks can be even more dangerous than expected. In particular, we notice in our recent studies that peer-to-peer (P2P) file-sharing applications can be leveraged to conduct more powerful UDP-based DRDoS attacks. We observe three features that make P2P applications particularly attractive for DRDoS attacks: (i) P2P applications use UDP messages frequently, such as the index services provided by Distributed hash tables (DHT [2]), making IP address spoofing easy to perform. (ii) All P2P users can freely access and store various data on other nodes in a P2P network, making almost all nodes in the P2P network perfect candidates for DRDoS reflectors. (iii) P2P applications often have a huge user base. At present, the user population of popular P2P file-sharing applications, such as Kad [3] and BitTorent [4], are over millions [5,6].

In this paper, we present a new type of DRDoS attack called **store-and-flood DRDoS** attack, or **SF-DRDoS**. The most notable characteristic of SF-DRDoS is that an adversary prepares and stores carefully crafted data on reflector nodes before issuing spoofed requests to reflector nodes and flooding the victim. This strategy can

* Corresponding author. Tel.: +86 10 82529591; fax: +86 10 62754420.
E-mail addresses: liubingshuang@pku.edu.cn (B. Liu), hanxinhui@pku.edu.cn (X. Han).

http://dx.doi.org/10.1016/j.comcom.2015.06.008
0140-3664/© 2015 Elsevier B.V. All rights reserved.

yield a much higher AF than previously explored DRDoS approaches [7]. The adversary can further adjust the timing, content, and in particular the volume of the responses.

Furthermore, we present a prototype SF-DRDoS attack system based on two popular Kademlia-based [8] P2P file-sharing networks, Kad and BT-DHT. The prototype system consists of a crawler to crawl the network, a node group to store index entries, and another node group to flood the victim. We investigate various factors that may affect the ability of the attack, and build a model to help illustrate the relationship between the attacker's bandwidth cost, the reflectors' response sizes, and the AF.

We further conducted real-world experiments to evaluate the effectiveness and flexibility of SF-DRDoS attacks, and found the average AF in the Kad network is about 2400, much higher than the AF achieved by current DRDoS attacks. The peak AF can reach 4326, making it possible to use only a 205 Kbps bandwidth to generate an attack flow of about 865 Mbps. If an attacker had enough bots, it could initiate an attack that costs only about 280 Mbps to generate a SF-DRDoS attack of more than 670 Gbps. Meanwhile, the average AF in BT-DHT networks is approximately 7. But due to BT-DHT's loose flow control mechanism and its huge user base (more than 16 million), the damage of this attack is much more terrible than expected, which can achieve a SF-DRDoS attack of over 10 Tbps.

Finally, we propose defense solutions to filter out the attack traffic generated by SF-DRDoS attacks. It is worth noting that, the attacker obeys the specifications of P2P networks and the attack traffic has no specific characteristics to be distinguished from legitimate traffic, making the defense very challenging. Our proposed defenses are based on BGP flow specification [9], and are able to filter attack traffic at the upstream links.

The rest of the paper is organized as follows. We first describe the DRDoS attack and discuss related work in Section 2. We then describe how a SF-DRDoS attack works in general in Section 3, and introduce the Kad-based and BT-DHT-based SF-DRDoS attacks in Sections 4 and 5, respectively. Section 6 presents the real-world experiments of SF-DRDoS attacks. We then discuss defenses against SF-DRDoS attacks in Section 7, discuss several observations in Section 8, and conclude the paper in Section 9.

2. Background and related work

2.1. The DRDoS attack

Fig. 1 illustrates the working mechanism of a DRDoS attack. Because the victim can only see that the attack traffic coming from reflectors, the attacker is difficult to locate. To conduct an effective DRDoS attack, the following conditions should be met:

- The transport protocol should be stateless and lack authentication so that the attacker can use spoofed IP addresses in their requests. Otherwise, the potential reflectors cannot be fooled into sending responses to the victim.
- There should be abundant reflectors that are open to all Internet users. Insufficient availability of reflectors caps the attack ability. For example, there are more than 27 million open DNS resolvers on the Internet that can be misused as reflectors [10].
- Some requests should trigger large responses according to the communication protocol in place, thus enabling amplification. Otherwise, the attack ability will be reduced to that of a non-reflective attack or even lower.

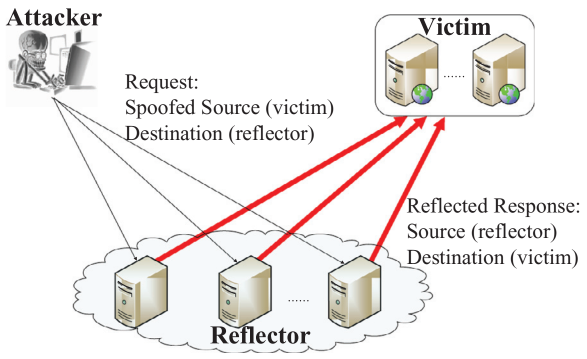


Fig. 1. The working mechanism of a DRDoS attack.

Also desirable for attackers are protocols which are difficult to filter, such as those using non-fixed UDP ports or that encrypt or obfuscate messages by default.

2.2. Related work

2.2.1. DRDoS attack methods

Researchers have conducted comprehensive analysis of the DRDoS attack since 2001 [11]. The earliest well-known DRDoS is probably the smurf attack [12], which sends spoofed ICMP echo requests to subnet broadcast addresses to trigger massive echo responses to a victim. Kumar et al. [13] further investigated the factors that affect the attack ability of the smurf attack and explained the relation between the attack cost, the reflector network and the final amplified attack traffic. TCP-based DRDoS attacks have also been studied [14–16]. These work demonstrated TCP-based protocols can be abused to conduct DRDoS attacks, despite the TCP three-way-handshake mechanism. By dissecting all kinds of unexpected responses received upon sending a single SYN packet, they found that some common TCP-based protocols can generate an amplification factor of 50 $\times$. In this work we focus on UDP-based DRDoS attack instead.

Nowadays the UDP-based DRDoS attacks have become more and more popular and destructive on the Internet. These attacks exploit popular Internet services to greatly amplify the attack traffic. For example, one of the biggest DDoS attacks in history [17] utilizes DNS reflection and amplification [18] to generate a striking DDoS attack at more than 300 Gbps. Another famous DRDoS attack happened in February 2014 by using NTP amplification [1], which generated attack traffic against a French hosting provider at more than 400 Gbps. Recently, Rossow [7] revisited popular UDP-based protocols, including DNS, NTP, SNMP, CharGen, BitTorrent and Kad. The results indicated that 14 protocols are exploitable as reflectors, with the AF reaching as high as 4670 when NTP severs are exploited. However, the computation of the AF only considered UDP payload. When considering the packet header as shown in our paper, the actual AF is only about 700, much lower than that of SF-DRDoS attacks that we will present in this paper.

Czyz et al. [19] has chronicled how NTP-based DRDoS attacks rapidly rose from obscurity to a dominant DDoS vector . They have measured the size of the NTP amplifier pool and found a small number of "mega amplifiers," i.e., servers that generate huge response packets. These mega amplifiers are similar to the Mono Torrent clients found in Section 5.3.

In order to help researchers understand the latest DDoS attacks, Wang et al. [20] gave an in-depth analysis based on 50,704 Internet DDoS attacks. They found a majority of current DDoS attacks were conducted by botnets. Compared with botnet-driven DDoS attacks, the SF-DRDoS attacks that we introduce in this paper do not need any bot program injected into normal hosts; yet, the SF-DRDoS attacks will be more efficient and lethal than botnet-driven DDoS attacks.

2.2.2. P2P-based DRDoS and DDoS attacks

Similar to our study, a few studies investigated the DRDoS attack via P2P networks. The possibility of reflection and amplification in Kad was found in [21], which utilizes `bootstrap` requests to gain

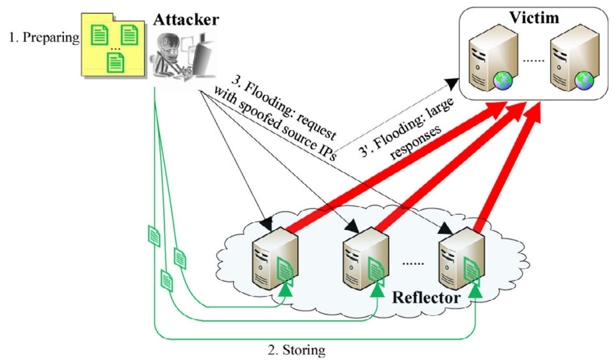

1. Preparing
Attacker
Victim
3. Flooding: request with spoofed source IPs
3'. Flooding: large responses
Reflector
2. Storing

Fig. 2. The working mechanism of a store-and-flood DRDoS attack.

an AF of about 8. Rossow [7] also investigated the AFs in Kad, discovering the average AF (without including packer headers) to be 16.3. On the other hand, most P2P-based DDoS attacks studied so far are not DRDoS, and are mainly focused on deceiving innocent users into flooding messages toward a victim. To do so, they either trick peers into adding bogus neighbors in their routing tables [21,22], or provide bogus index entries that a victim owns popular contents [21–26]. While such DDoS attacks have been found in the wild [27], they only generate tens of megabytes of UDP traffic or tens of hundreds of TCP connections per second at most.

2.2.3. Defenses against DRDoS

Research on DRDoS defenses can be classified into two categories, network source address validation and traffic scrubbing. The first category includes BCP 38 [28], ITRACE [29], SPIE [30], Hop-Count Filtering [31], Passive IP traceback [32] and SAVE [33]. The later includes the statistics-based approach[34], SIFF [35], AITF [36] and StackPi [37]. These solutions, however, only have limited efficacy against DRDoS unless they can be fully deployed.

3. Store-and-flood DRDoS

In this section, we present the methodology of the store-and-flood DRDoS (SF-DRDoS) attack and then analyze its characteristics. We build a model to evaluate its attack ability, with an emphasis on the AF the attacker can achieve.

3.1. Methodology

As shown in Fig. 2, an SF-DRDoS attack typically consists of three stages:

1. Preparing stage: In this step, the attacker prepares data to store at reflectors. To do so, the data must follow the protocol of the exploited services to appear legitimate. Furthermore, the attacker needs to consider how the data may maximize the amplification factor. In P2P networks, for example, such data can be an index entry that contains an extremely long filename.
2. Storing stage: Now that the data is prepared, the attacker must store the data at reflectors. Since the data follows the protocol of the exploited service, storing the data will likely go undetected. For example, in P2P networks, the storing process can be as simple as storing index entries in selected peers. The attacker must be mindful of the data's expiration time to ensure that the data will be available during the next stage.
3. Flooding stage: With data stored at reflectors, the attacker can then trigger flooding traffic toward her victim. To do this, the attacker sends reflectors requests for the previously stored data with the source IP address of the victim. Since the requests appear to be from the victim, the reflectors will send all the responses to the victim. The more requests the attacker generates, the more responses the victim will receive, the less available the victim will become to its legitimate users.

Due to the storing stage before flooding, the AF of the SF-DRDoS attack can be much higher than that of the traditional DRDoS attacks. In traditional DRDoS attacks, the attacker depends on data already stored at reflectors. If there is nothing or little data related to a request, the AF could not possibly be as high as desired. Conversely, in SF-DRDoS, because carefully prepared data are stored on reflectors in advance, the attacker can customize every request to generate a large response, thus significantly increasing the AF.

The SF-DRDoS attack offers great flexibilities to the attacker. First, the attacker can configure how many reflectors to use, how much data to store at each reflector, and how large each response will be. Second, the attacker can also control the timing of its attack, as it could determine when to begin and end the storing stage, when to trigger the flooding stage, and even whether to overlap different stages. Finally, the attacker can easily adjust the attack volume by controlling the number of requests and the size of responses.

3.2. The amplification factor (AF) of SF-DRDoS

While AF is the ratio between the total attack traffic volume launched toward the victim, i.e., *attack ability*, and the total traffic volume invested by the attacker, i.e., *attack cost*, we further introduce two AF metrics to measure the potency of an SF-DRDoS attack: the **attack-time AF** where the attack cost is only the cost during an attack, and the **all-time AF** where the attack cost includes *all* the cost that the attacker invests.

Assume that sending a request of size s will cause a reflector to retrieve the attacker's stored data and generate a response of size r. The attack-time AF will simply be:

$$\text{Attack-time AF} = \frac{r}{s}. \tag{1}$$

While traditional DRDoS attacks can be well measured using attack-time AF, it is also important to use all-time AF for SF-DRDoS where the cost of storing data on reflectors may be non-negligible. Assume that the attacker must use s' worth of traffic volume to store data at each reflector. If each reflector can be used t times after having data stored at it, then

$$\text{All-time AF} = \frac{r \cdot t}{s' + s \cdot t}. \tag{2}$$

The all-time AF represents the actual ratio of victim resource usage to attacker resource usage. The attack-time AF is more indicative of what attacks will be achievable by an attacker. While an attacker may have a moderate all-time AF, it may have a high enough attack-time AF to disable a victim for a short period of time with a relatively low amount of bandwidth. In effect, the attacker is able to pay for preparing the attack over a longer period of time, then suddenly launch a large attack which the victim cannot handle all at once.

4. Store-and-flood DRDoS on Kad

In this section, we introduce a Kad-based store-and-flood DRDoS attack. First we give an overview of the Kad system, with the emphasis on its characteristics that the SF-DRDoS attack will exploit, then we describe the design and implementation of this attack.

4.1. Kad

Kad is a P2P file-sharing network using the Kademlia [8] DHT protocol. In Kad, every participating node has a unique 128-bit identifier called *Kad ID*. Kad supports two types of objects, keywords and files. Every keyword is associated with a 128-bit *key ID*, which is the

Table 1
Amplification factors of Kad operations based on experiments.

Operation	Request (bytes)	Response (bytes)	AF
Bootstrapping	64	480	~ 8
Routing	77	111–336	1–5
Searching an unpopular keyword	64	230	~ 4
Searching a popular keyword	64	27,493	~ 350
Searching an unpopular file	70	260	~ 4
Searching a popular file	70	12,000	~ 160

hash of the keyword, and every file is assigned a 128-bit *file ID*, which is the hash of the file's content. In the 128-bit ID space, Kad calculates the distance of two IDs using bitwise *XOR* operation. Kad supports two primary operations: *publish* and *search*. A node can publish a *keyword-to-file* index (such as ⟨*key ID*, ⟨*filename*, *filesize*, *fileID*...⟩⟩) at *index nodes* whose Kad ID is closest to the *key ID* of the keyword in the ID space, and allow other nodes to search the index using the *key ID*.

Kad has the following characteristics that a store-and-flood DRDoS attack can exploit:

- All Kad operations are UDP-based, and they do not have handshaking mechanisms at the application level either, thus making IP spoofing easy.
- Kad has two million concurrent users [5] and all of them could be reflectors.
- Kad can provide a large amplification effect. Table 1 shows the amplification effects that some operations in Kad can already achieve without much elaboration, i.e., no storing stage. Further, taking advantage of SF-DRDoS, the attacker can get a much higher AF in Kad. In Kad, each Kad node can be easily turned into an index node by publishing appropriate index entries to it. And there is no explicitly limitation on the *string* fields in the keyword-to-file index entry, so one entry could be much larger than expected. If the attacker stores enough enlarged index entries at a Kad node in advance, the Kad node would return all of them when receiving one small searching request. Kad allows at most 300 index entries in one searching response, possibly encapsulated into one or more UDP packets.
- In Kad it is easy to manipulate the size of response packets, the UDP ports used by Kad are not fixed (in order to avoid censorship), and messages are encrypted and obscured, all making traffic filtering difficult.

4.2. Design of Kad-based SF-DRDoS

A Kad-based SF-DRDoS attack consists of three basic components: the crawler, the storing group, and the flooding group. As shown in Fig. 3, the crawler can periodically crawl the entire Kad network to collect Kad nodes online and provide a list of them to the storing group. Every online node is a potential reflector represented by a tuple *(Kad ID, IP, UDP port, Kad version)*. The storing group, upon the receipt of a list of online nodes, prepares 300 large index entries, and stores them to each of these nodes—i.e., the storing stage of an SF-DRDoS attack. For each node, the storing group first selects an appropriate keyword which has a key ID that shares at least the first 8 bits (i.e., prefix) with the node's Kad ID, then constructs 300 keyword-to-file indices all with the keyword and a random, extremely long string for the filename field, and finally publishes all these indices. These indices, once stored, will stay valid for 24 h, and this storing process can be repeated every 24 h to support a persistent attack.

When it is the time to launch the attack as desired by the attacker, the flooding group can then issue a large number of searching requests to look for the keywords for which the storing group has stored indices in Kad. All these requests carry the spoofed source IP

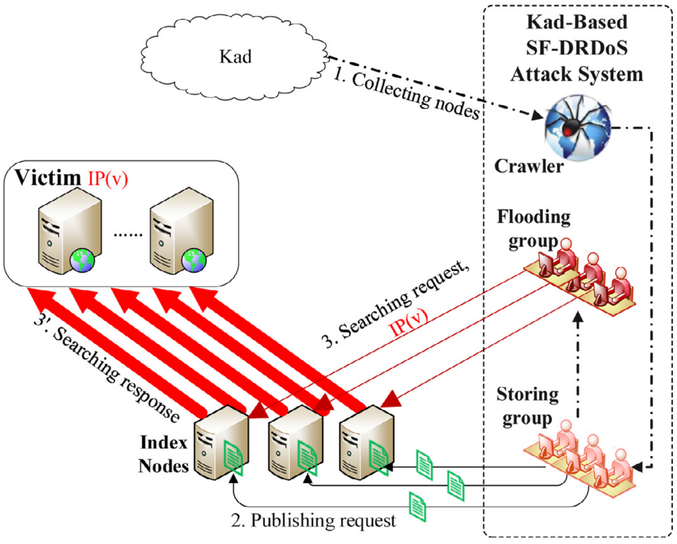


Fig. 3. The design of Kad-based store-and-flood DRDoS attack.

address of a victim (IP_V in Fig. 3). Due to the lack of any handshaking mechanism in Kad, these reflectors cannot verify the authenticity of the source IP. Every request is only 64 bytes and can trigger up to 300 index entries, thus generating massive response packets toward the spoofed source IP address. Consequentially, the links at the victim will be clogged and the victim will be successfully DDoS'ed.

4.3. Implementation of Kad-based SF-DRDoS

We implemented the Kad-based SF-DRDoS attack system using aMule [38], an open-source application which works with Kad. We created a customized client with only the necessary Kad components, and using this client we spawned many Kad nodes to perform the crawling, storing, and flooding. The Kad crawler is a specific implementation of our crawling algorithm proposed in [39], which according to our experiments can gather over 2 million nodes in about 3 min. However, since we use a single crawler, rather than a distributed crawler as proposed in [39], our crawler takes longer. Also, we limit the crawler to only collect nodes which can be used as reflectors, e.g., those that do not use firewalls or NAT.

For the storing process, an important parameter is the maximum length of the filename in keyword-to-file indices, which determines the ultimate AF. By analysing Kad implementations, we found that eMule cannot accept an index where the length of the filename exceeds 2048 bytes. On the other hand, there is no limitation on the filename in aMule. It means the attacker can store indices with arbitrary length, and achieve an infinite AF in the aMule network. In this paper, we uniformly use a smaller size to avoid having any filenames dropped or rejected in both eMule and aMule.

Kad has a flooding control mechanism to limit the rate of request packets from a specific source IP address. Once the request rate from a node exceeds the limit, its requests will be dropped and eventually its IP address will be blacklisted temporarily. For example, every node can issue at most *three* searching or publishing requests every minute. However, during the storing stage, every publishing request can use a different spoofed source IP address, or that of a different bot each time in a large-scale botnet, to stay below the rate limit. This limitation is therefore only relevant to the flooding stage of the SF-DRDoS attack where every request must use one of the IP addresses of the victim. Therefore, in the flooding stage for every IP address of the victim, this attack will send every node three searching requests per minute to maximize the usage of each node as a reflector. If the victim has many IP addresses, such as when an entire subnet is targeted, a lot more searching requests can be issued.

5. Store-and-flood DRDoS on BT-DHT

In this section, we present another SF-DRDoS attack based on the BT-DHT network. Compared with the Kad-based attack, it has several distinctive characteristics, such as the larger user base with over 16 millions users and a looser flow control mechanism. Although BT-DHT-based SF-DRDoS has a smaller amplification effect than the one based on Kad, its attack ability can be more powerful.

5.1. BT-DHT

In essence, both BT-DHT and Kad are specific implementations of the Kademlia protocol. So the working mechanism of BT-DHT is similar to that of Kad, as shown in Section 4.1. Here, we just highlight several important differences.

BT-DHT has only one type of object, i.e., the file object. Both nodes and objects share a common 160-bit ID space. We call the mapping between the file ID and the connection information of the file owner, i.e., $\langle \textit{file ID}, \langle IP1: Port1, IP2: Port2, \ldots \rangle \rangle$, the file-to-source index. In the BT-DHT-based SF-DRDoS attack, the attacker can only use the file-to-source index. Due to the fixed length of every field in this kind of index, the AF of SF-DrDoS would be naturally restricted.

In BT-DHT, the get_peers message provides the searching function. When a node receives a get_peers request for a specific (file) object and it has records of the object in its index table, it replies with the information of a certain amount of file owners (i.e., peers). On the other hand, file owners use the announce message to publish file objects toward index nodes. Unlike Kad, an index node in BT-DHT does not examine whether an announced file object is adjacent to itself in the ID space, and the file ID in the announce message and the node ID of an index node does not need to have an 8-bit prefix in common. This feature makes it easy for the storing group in an SF-DRDoS attack to construct and announce file-to-source indices, store them at index nodes, and further use these index nodes as reflectors in an SF-DRDoS attack. And in BT-DHT, if some node wants to announce something to an index node, it must first send a get_peers request. Then the index node sends back a random *token* in the get_peers response, which will be used by the initiator to send the announce request. At last, the index node checks the token when receiving the announce request. It refuses the storing request with a wrong token.

Like Kad, BT-DHT can also be a good candidate as an exploitable reflector network. According to the measurement work in [39], the number of users concurrent online in BT-DHT is more than 16 millions, and the number of unique nodes seen in 24 h exceeds 100 million. This huge user base provides even more reflectors for an SF-DRDoS attack. We have also measured amplification effects in the natural BT-DHT network. As shown in Table 2, the get_peers message is a good candidate to be exploited in an SF-DRDoS attack. Though the size of one file index is small and fixed, this attack can leverage the fact that one request can retrieve dozens of index entries. Finally, like the Kad network, the BT-DHT network also provides other desired characteristics, such as UDP transport, flexible UDP ports, and message encryption.

Table 2
Amplification factors in the real BT-DHT network.

Category	Request (bytes)	Response (bytes)	AF
Ping	109	100	~ 1
Find_node	145	329	~ 2
Get_ peers	159	367–695	2–4
Announce	222	100	~ 0.5

5.2. Design of BT-DHT-based SF-DRDoS

As shown in Fig. 3, the principle of the BT-DHT-based SF-DRDoS attack is basically the same as that of the Kad-based SF-DRDoS, Below we only discuss their differences. First, at the storing stage, the storing group utilizes announce messages to store enough file-to-source index entries on reflector nodes. To choose reflector nodes to use, it can take snapshots of the whole or certain parts of BT-DHT network, such as by using the splitting crawler described in [39]. Then at the flooding stage, it sends reflectors continuous get_peers requests with spoofed source IPs that point to the target, causing massive get_peers responses toward the target area and thus DDoS occurrence.

5.3. Implementation of BT-DHT-based SF-DRDoS

Unlike Kad, most BT-DHT clients are not open source and they implement the BT-DHT protocol in different ways. We implemented the BT-DHT-based SF-DRDoS attack system based on libTorrent [40], an open source library of the BitTorrent protocol. We have conducted a series of measurements on BT-DHT to see how an SF-DRDoS attack can dynamically adjust various BT-DHT parameters for different client versions, in order to to fully exploit the BT-DHT network and achieve the most serious damage.

The first parameter is the number of index entries triggered by a get_peers request. In the experiment, we crawled 23,000 nodes randomly in BT-DHT and stored enough index entries on them, and then sent get_peers requests to them to trigger corresponding responses.

Fig. 4 presents the complementary cumulative distribution (CCDF) of the number of returned indices. The result shows the average number of indices is 92. So we can deduce that the average AF is nearly 7 $(= (367 + 92 * 8)/159)$ (See Table 2). Most nodes (nearly 60%) return less than 100 index entries in one get_peers response. However, there are still about 11% of nodes returning more than 150 index entries, and exploiting these nodes can achieve an AF of about 10. In addition, one special type of BT-DHT client, MonoTorrent, has almost no limitation, and can return as many as 2016 indices.

The second parameter is the validity period of file-to-source indices stored on BT-DHT clients. We have measured some mainstream clients, including uTorrent, BitTorrent, libTorrent, Bitspirit and Bitcomet. The result shows that the index entry in the uTorrent client has the longest period of validity, i.e., more than 36 h. For libTorrent, the duration is 45 min and all others are 30 min. With the guidance of these measurements, the storing group in SF-DRDoS can then execute re-publish operations more accurately.

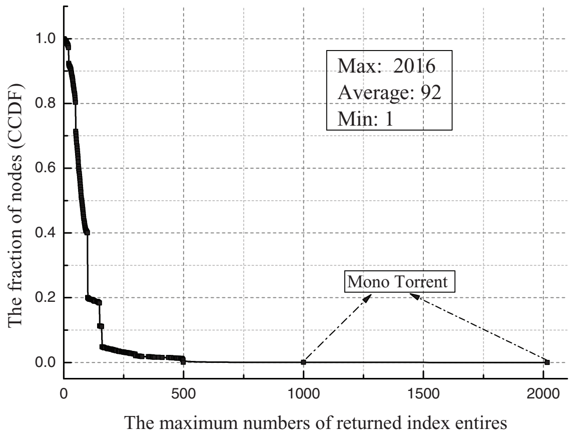


Fig. 4. The distribution of the number of index entries returned in BT-DHT.

Table 3
The results of flow control mechanisms (maximum packets per second (pps)) in three BT-DHT clients.

Client ersion	uTorrent 3.5.0.275	Bitcomet 1.13 Stable Release	Bitspirit 1.8.3
ping	10	15	15
find_node	2	20	9
get_ peers/announce	2	25	10

The last parameter is the flow control mechanism in BT-DHT. By checking the source code of libTorrent, we found it has no flow control. We thus measured other three mainstream BT-DHT clients, including uTorrent, Bitcomet and Bitspirit. Table 3 presents the flow control mechanism details in each client. Because each `announce` request needs a `get_peers` request first, so the rate limitations of the two requests are identical. The results show the flow control of uTorrent is much more rigorous, with the rate of `get_peers` and `announce` only 2 pps. Nevertheless, the overall flow control mechanisms in BT-DHT are more relaxed than in Kad, where the pps of searching requests or publishing requests is just 0.05 (every Kad node can issue at most 3 searching or publishing requests every minute).

6. Experiment results and analysis

In this section, we describe our experiments with the SF-DRDOS attack system and our analysis of the results.

6.1. Experiment setup

We deploy the attack system on a server with two Intel Xeon CPUs (E5645, 2.40 GHz, 24GB RAM) located on a university campus. It spawns 40 customized P2P nodes for the storing group as well as 40 for the flooding group. Each node uses an independent IP address to directly connect to the Internet. All nodes are distributed evenly across the Kad and BT-DHT ID space. Every node has a 4 Mbps uplink, rented from a special network where BCP 38 [28] is not deployed, allowing every node to spoof their source IP address.

6.1.1. The Kad-based attack

We conducted a Kad-based experimental attack in May 2013 under real-world conditions toward a victim with a dedicated download bandwidth of 1 Gbps at another participating university. We carefully controlled the attack traffic volume to avoid causing any unexpected network failures. During the experiment, the crawler collected Kad nodes (i.e., reflectors) online from two 8-bits ID zones (0x51 and 0x2E) on an hourly basis, and was able to continuously provide about 20,000 live nodes for the storing group and the flooding group. The storing group then published 300 keyword-to-file index entries to each node collected, where the length of every filename was 1500 bytes. The storing process lasted 30 min; as the average Kad node stays online for 165 min [41], the 30 min duration is appropriate to ensure enough nodes remain available with the stored data during the flooding stage. We then launched the flooding stage that lasted for 15 min, during which we recorded all attack traffic towards the predefined UDP port at the victim machine. For every node used as a reflector, because it stays online for 165 min on average and can accept 3 searching requests per minute from each source address, assuming there are $|V|$ different IP addresses of the victim, the node can then receive totally $165 \cdot 3 \cdot |V|$, i.e., $495|V|$, requests.

6.1.2. The BT-DHT-based attack

In the BT-DHT-based attack, the experiment setup was almost the same. About three thousand reflectors come from a 12-bit ID zone, 0x91A. The number of index entries and the rate of sending requests are consistent with the measurement results in Fig. 4 and Table 3. In

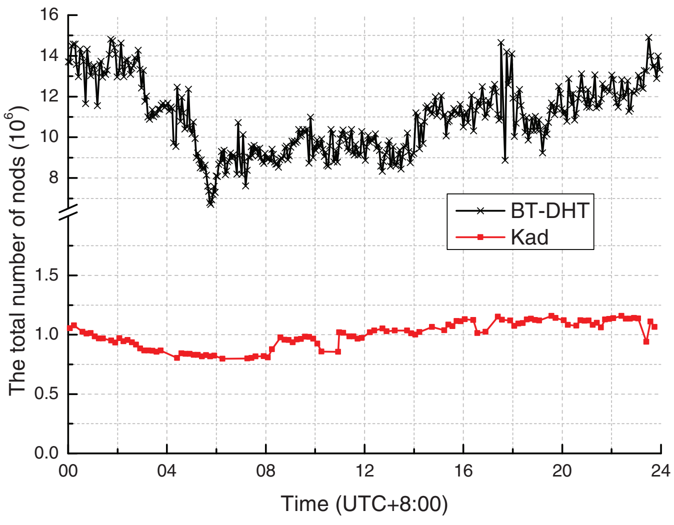


Fig. 5. The total number of nodes simultaneously online in Kad and BT-DHT on June 22, 2013.

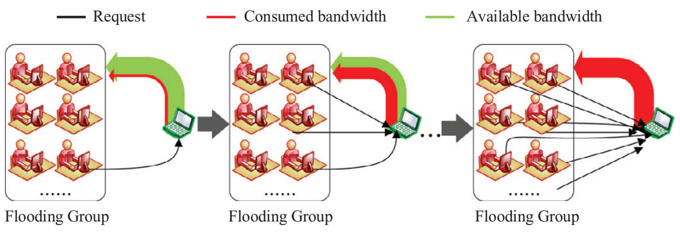


Fig. 6. The schematic diagram of the bandwidth estimation technique.

BT-DHT, a node stay online for 174 min on average and can accept 120 `get_peers` requests (for uTorrent) per minute from each source address, so the node can then receive totally $174 \cdot 120 \cdot |V|$, i.e., $20880|V|$, requests. Benefiting from the looser flow control mechanism, the BT-DHT-based attack can utilize the stored contents more fully than the Kad-based one's.

6.2. Environmental parameters

Two environmental parameters are important for our experiments: the network size and the uplink bandwidth of nodes in Kad and BT-DHT. The former determines the total number of usable reflectors, and the latter is the upper limit of the DRDoS traffic on an individual reflector. We thus estimated both as follows.

To determine the network size , we ran our crawler continuously for 24 h. The crawler could collect all nodes in the Kad or BT-DHT network in an average of 15 min. Fig. 5 presents the total number of usable reflectors in the two Kademlia networks. Though the number fluctuates over time, useable nodes in each network are always sufficient to conduct a serious DDoS attack. In BT-DHT, even not at peak hours, there are still about 12 million nodes online simultaneously. And in Kad, it is at least 0.8 million all the time and is often over 1 million.

For the uplink bandwidth of Kademlia nodes, we developed a customized bandwidth estimation method based on the self-loading periodic streams (SLoPS) technique [42], with its results shown in Fig. 6. Our assumption is that senders will not receive responses from a receiver once the available uplink bandwidth of the receiver is all used up, and the uplink bandwidth of a node is thus the maximum bandwidth consumed by all the response packets to all the senders. After storing enough index entries at a Kad or BT-DHT node, we set up 170 flooding nodes to send searching requests with gradually increasing rates directly to the Kademlia node's IP, and then measure the maximum response bandwidth. Because Kademlia does not limit the total

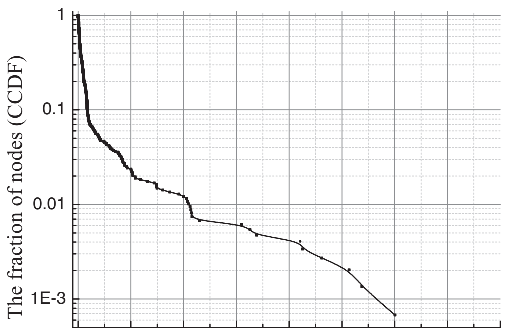


Fig. 7. The distribution of uplink bandwidth of Kad nodes.

amount of traffic used in DHT communications, this bandwidth is exactly the uplink bandwidth of the node in question.

Fig. 7 presents the distribution of the uplink bandwidth for 2150 randomly chosen Kad nodes. The result shows that the uplink bandwidth of 92% of nodes is lower than 1 Mbps, and the average value is 0.67 Mbps. However, there are still 2.3% of nodes whose uplink bandwidth is higher than 5 Mbps. When conducting a real SF-DRDoS attack, the attacker can adopt a method, similar to TCP Slow Start [43], to fully utilize the uplink bandwidth of every node/reflector.

6.3. Results and analysis

We now present the results from our experiments under real world conditions. Here, the results and analysis mainly focus on the Kad-based attack for simplicity.

6.3.1. The Kad-based attack

Fig. 8a presents the attack bandwidth toward the victim over a 15 min attack window. The peak reaches 865 Mbps, and the average is 480 Mbps. Meanwhile, as the average attack cost during the attack is only 0.2 Mbps, the average attack-time AF is 2400. Fig. 8b further presents the attack-time AF throughout the attack window, where the maximum AF is 4326. Note that due to node overloading and packet loss, this maximum is lower than the theoretical maximum, which is 5980 for 1500-byte filenames, and if we use 1900-byte filenames the maximum AF can be even higher, with a theoretical maximum of 9548. However, the measured AF value of 4326 is already significantly large. In a previous DDoS attack using Kad [21], it utilizes bootstrap requests and gains an AF of only about 8.

The all-time AF for this attack can also be quite high. The storing cost, s', for 300 index entries each with a filename taking 1500 bytes will be $1500 \cdot 300$ bytes. Considering the flooding control mechanism in Kad, in order to fully utilize the uplink bandwidth of reflectors, the target area must have 11 IP addresses. Assuming that the number of IP addresses in the target area $|V|$ is 11, then the number of times a reflector can be used, t, is $495 \cdot 11$. The size of a request to a reflector, s, is 64 bytes and this request will generate a response of size $r = 300 \cdot 1500$ bytes. Now, referring to Eq. 2 in Section 3.2, we have

$$\text{All-time AF} = \frac{(300 \cdot 1500) \cdot (495 \cdot 11)}{(300 \cdot 1500) + 64 \cdot (495 \cdot 11)} = 3069. \tag{3}$$

Now we estimate the total attack ability of the Kad-based SF-DRDoS attack and the corresponding attack cost. The number of usable reflectors in Kad is approximately one million, of which the average uplink bandwidth is 0.67 Mbps. We use the attack-time AF of 2400 from our experiment. Assuming the attacker can utilize her reflectors with the average reflector uplink bandwidth, the maximum attack ability would be approximately 670 Gbps (i.e., 0.67 Mbps multiplied by one million), sufficient to disable most web sites on the Internet. Meanwhile, the attack cost at the flooding stage would be just 280 Mbps, easy for an attacker to obtain.

6.3.2. The BT-DHT-based attack

The result of the BT-DHT-based attack shows that the attack bandwidth toward the victim is 13.3 Mbps. Meanwhile, the average attack cost is 1.97 Mbps. So the average AF is nearly 7, which is consistent with the result in Fig. 4.

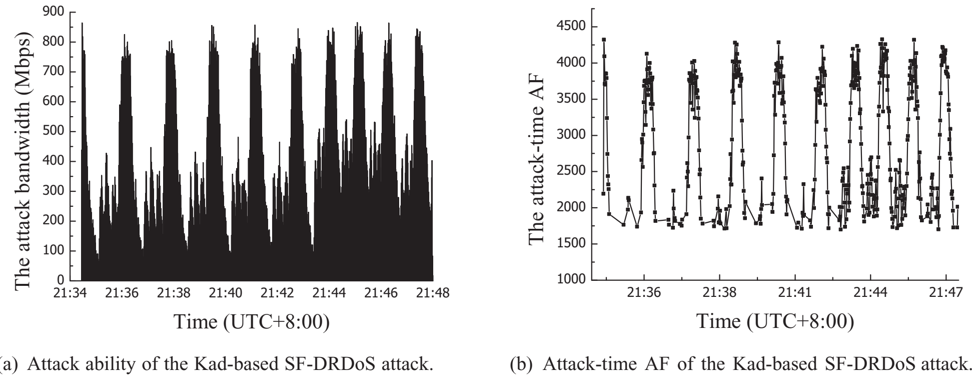
Although the amplification effect is not as good as the Kad-based one's, some distinguishing features of BT-DHT could make the BT-DHT-based attack more powerful under certain conditions. Taking advantage of the looser flow control mechanism and the huge user base of more than 16 million users, the full potential of the BT-DHT network is enough to carry out 10 Tbps-level SF-DRDoS attacks (i.e., 0.67 Mbps multiplied by 16 million). If the attacker can obtain an bandwidth of 1.4 Tbps, it will be likely to happen. Honestly speaking, it is difficult for most attackers to gain such a big Internet bandwidth. Nevertheless, it is worth noting that numerous botnets emerge in today's Internet and the attacker can easily obtain a bandwidth of 100 Gbps by controlling them. Thus, a SF-DRDoS attack of 1 Tbps would not be very far from us.

7. Detection and defenses

In this section, we propose possible methods for detection and defense against SF-DRDoS attacks.

7.1. Detection

By adopting random ports and encrypted payloads, Kademlia-based SF-DRDoS attacks are hard to detect. We propose that we can deploy enough "honey nodes" in these P2P networks to detect such



(a) Attack ability of the Kad-based SF-DRDoS attack. (b) Attack-time AF of the Kad-based SF-DRDoS attack.

Fig. 8. Attack ability and AF in the Kad-based SF-DRDoS attack conducted on May 21st, 2013.

attacks. These honey nodes would act just the same as other normal nodes, and each honey node keeps statistics about publishing and searching events that it is involved in. According to features of store-and-flood attacks, it is easy for honey nodes to detect abnormal behaviors such as by seeing large index entries (with long file names) published at honey nodes and then subsequent searching requests for the same index entries at a high rate.

In order to capture most publishing and searching events in P2P networks, we should consider the deployment number and location of honey nodes in the ID space of Kad. According to the idea of the eclipse attack [44], deploying one honey node in each small ID space would allow them to listen and collect most Kad messages toward the region. Experiment results show that the size of each small ID space should be 15 bits, i.e., all nodes in that space share a common prefix of at least 15 bits. So the number of honey nodes is $2^{15} = 32K$. Given 1.5 to 2 million nodes in the current Kad network [5], the population of honey nodes would not disturb the normal functions of Kad. Regarding how to set up numerous honey nodes using a small number of physical machines, readers can refer to our prior work in [45].

7.2. Defenses

We make the following recommendations to defend against SF-DRDoS. First, Kademlia networks should make some changes, including answering requests only after validating their sources, limiting the string length and the number of index entries triggered by one request. However, it is difficult to deploy these incompatible modifications in the current network.

Next, as recommended in BCP 38 [28], every ISP should deploy ingress filtering to eliminate the possibility of source IP spoofing as well as reflection attacks, including SF-DRDoS. Ingress filtering requires that when an IP packet departs from a network to enter the Internet, it must carry a source IP address belonging to the ISP. Unfortunately, while nearly 80% of the Internet deploys some type of ingress filtering, the remaining 20% are reluctant to deploy ingress filtering due to technical and economic reasons [46].

Then assuming IP spoofing will not be eliminated in the foreseeable future, we believe that effective traffic filtering is key to defending against SF-DRDoS attacks. Though the design of a comprehensive filtering system is beyond the scope of this paper, we provide an overview of what such a system would require in order to succeed, and give operational examples of such a system based on BGP flow specification [9].

Before discussing challenges in designing a comprehensive system, we consider a scenario in which a TCP server would like to defend itself against SF-DRDoS attacks. Designing a flow specification in this scenario is simple: The TCP server can send a rule to its switch asking that all UDP packets be dropped. The switch then propagates the rule further upstream. This should mitigate any UDP-based SF-DRDoS attacks.

A more comprehensive system, meant to defend systems which cannot afford to simply block all UDP traffic, must be able to effectively deploy filtering rules which can distinguish malicious traffic from legitimate traffic. Such a system would need to either automatically generate suitable rules or listen for rules from a client and then reliably propagate said rules to upstream links. Fig. 9 shows an example of such a system based on flow specification propagating rules to filter packets exceeding a set size.

8. Discussion and open issues

To exploit *all* nodes in the Kad network, the attacker would need a bandwidth of nearly 280 Mbps. And for the BT-DHT attack, the bandwidth cost will be strikingly up to 1.4 Tbps. While this cost may seem prohibitive at first, with sufficient funds, attackers may achieve a high bandwidth by renting botnets. In recent years, botnet attacks have

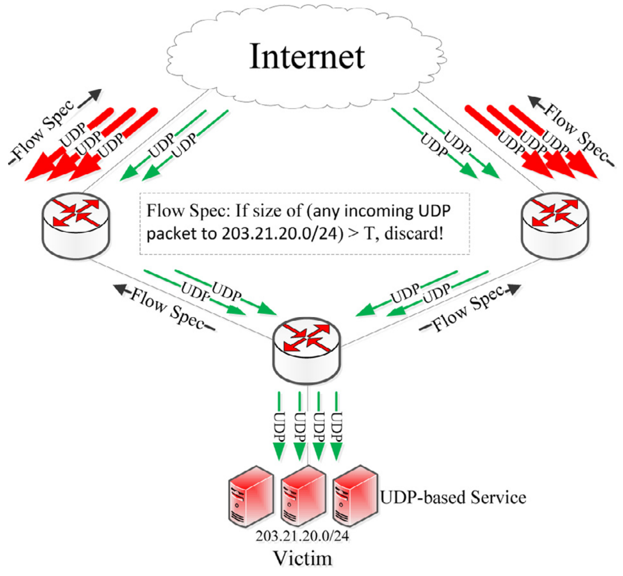


Fig. 9. An example of protecting UDP-based services using BGP flow specification.

become quite prevalent [47–49], and botnets can be as cheap as $100 for 10,000 bots [50]. Furthermore, botnets with hundreds of thousands of bots are readily employable. With such low costs and a high volume, attacks exploiting every node on a network may be feasible from both a technical and economic standpoint.

Also, the store-and-flood DRDoS attack is not limited to P2P file-sharing networks, such as Kad. A thorough examination of all public UDP protocols is beyond the scope of this paper, however, the same methodology we used could be applied to develop attacks on other UDP protocols. Proprietary UDP protocols for which specifications are not publicly available could be vulnerable to this type of attack.

This paper presents the SF-DRDoS attack and points to upstream filtering as a promising defense. However, the creation of such a system is left as future work. Such a system would need to meet the following demands: (i) it must be able to produce high quality filtering rules; (ii) it must scale to be able to handle many rules from many servers; and (iii) it must be easily deployable. A system meeting these requirements would not only provide defense against SF-DRDoS attacks, but could be generalized to help mitigate any type of DDoS.

9. Conclusion

Distributed denial of service (DDoS) attacks have been around for many years, yet they continue to pose a serious threat to the security of the Internet. Worse, as the Internet adds new applications with very large user bases, the soil for DDoS becomes ever more fertile, sometimes even leading to new, more devastating DDoS attacks.

We elucidate one such new DDoS attack in this paper, which we call *store-and-flood* distributed reflective denial of service attack, or SF-DRDoS attack. The attack can leverage popular peer-to-peer (P2P) applications to flood a victim with an unusually large amplification factor. The attacker can first store carefully prepared data on a large amount of P2P nodes, and then issue specially prepared requests to these nodes to generate responses toward innocent victims. The timing, content, and in particular the volume of the responses are all under the control of the attacker.

We implemented prototypes of SF-DRDoS on the Kadamlia P2P networks, and conducted real-world experiments. Compared with the state-of-the-art amplification factor in DRDoS attacks, the Kad-based SF-DRDoS can achieve a much higher attack-time amplification factor of 2400 on average, with an attack bandwidth as high as 670 Gbps—sufficient to take down most web sites on the Internet.

What is more, the BT-DHT-based attack can even conduct a terrible DDoS of 10 Tbps, which definitely means a disaster to today's Internet.

We further discussed defenses against SF-DRDoS attacks, including injecting honey nodes into Kademlia networks, deploying BCP 38, and employing BGP flow specification. Together with other DDoS attacks, SF-DRDoS attacks signal the urgent need for new, effective defense solutions.

Acknowledgments

This research was supported in part by National Natural Science Foundation of China (Grant no. 61402125).

References

[1] A 400 gbps ntp amplication DDoS attack, https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/ (accessed 15.02.14).
[2] M. Zhang, M. Dusi, W. John, C. Chen, Analysis of UDP traffic usage on internet backbone links, in: Proceedings of 9th Annual International Symposium on Applications and the Internet (SAINT), 2009.
[3] The eMule Project, http://www.emule-project.net (accessed 10.02.14).
[4] B. Cohen, Incentives build robustness in BitTorrent, in: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, 2003.
[5] M. Steiner, E.W. Biersack, T. Ennajjary, Actively monitoring peers in kad, in: Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS), Bellevue, WA, USA, 2007.
[6] J. Yu, P. Xiao, Z. Li, Y. Zhou, Toward an accurate snapshot of DHT networks, IEEE Commun. Lett. 15 (1) (2011) 97–99, doi:10.1109/LCOMM.2010.110310.101027.
[7] C. Rossow, Amplification hell: Revisiting network protocols for DDoS abuse, in: Proceedings of the 21th Annual Network & Distributed System Security Symposium (NDSS), 2014.
[8] P. Maymounkov, D. Mazieres, Kademlia: a peer-to-peer information system based on the xor metric, in: 1st International Workshop on Peer-to-peer Systems (IPTPS'02), 2002.
[9] P. Marques, Dissemination of flow specification rules, in: RFC 5575, 2009.
[10] Open Resolver Project http://openresolverproject.org (accessed 02.03.14).
[11] V. Paxson, An analysis of using reflectors for distributed denial-of-service attacks, ACM SIGCOMM Comput. Commun. Rev. 31 (3) (2001) 38–47.
[12] CERT advisory CA-1998-01 smurf IP denial-of-service attacks, http://www.cert.org/advisories/CA-1998-01.html (accessed 01.02.14).
[13] S. Kumar, Smurf-based distributed denial of service (DDoS) attack amplification in Internet, in: Proceedings of the Second International Conference on Internet Monitoring and Protection (ICIMP), 2007.
[14] Distributed reflection denial of service, http://www.understandingcomputers.ca/articles/grc/drdos_copy.html (accessed 02.08.13).
[15] M. Kührer, T. Hupperich, C. Rossow, T. Holz, Exit from hell? reducing the impact of amplification ddos attacks, in: Proceedings of the USENIX Security Symposium, 2014.
[16] M. Kührer, T. Hupperich, C. Rossow, T. Holz, Hell of a handshake: abusing tcp for reflective amplification ddos attacks, in: Proceedings of the USENIX Workshop on Offensive Technologies (WOOT), 2014.
[17] The largest DDoS attack in history, http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet (accessed 27.03.13).
[18] An analysis of DrDos DNS reflection attacks, http://storage.pardot.com/9892/101226/Analysis_of_DrDoS_DNS_Reflection_Attacks_White_Paper_US_031513.pdf (accessed 02.02.14).
[19] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, M. Karir, Taming the 800 pound gorilla: The rise and decline of ntp ddos attacks, in: Proceedings of the 2014 Conference on Internet Measurement Conference, ACM, 2014, pp. 435–448.
[20] A. Wang, A. Mohaisen, W. Chang, S. Chen, Delving into internet ddos attacks by botnets: characterization and analysis, in: IEEE International Conference on Dependable Systems and Networks (DSN), 2015.
[21] J. Yu, Z. Li, X. Chen, Misusing Kademlia protocol to perform DDoS attacks, in: Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2008.
[22] N. Naoumov, K. Ross, Exploiting P2P systems for DDoS attacks, in: Proceedings of the 1st international conference on Scalable information systems, 2006.
[23] E. Athanasopoulos, K.G. Anagnostakis, E.P. Markatos, Misusing unstructured P2P systems to perform DoS attacks: the network that never forgets, in: Proceedings of the Applied Cryptography and Network Security, 2006.
[24] K. El Defrawy, M. Gjoka, A. Markopoulou, BotTorrent: misusing BitTorrent to launch DDoS attacks, in: Proceedings of the 3rd USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet, 2007.
[25] X. Sun, R. Torres, S. Rao, DDoS attacks by subverting membership management in P2P systems, in: Proceedings of the 3rd IEEE Workshop on Secure Network Protocols (NPSec), 2007.
[26] K.C. Sia, DDoS vulnerability analysis of BitTorrent protocol, Technical report, University of California, Los Angeles., 2007.
[27] Z. Li, A. Goyal, Y. Chen, A. Kuzmanovic, Measurement and diagnosis of address misconfigured P2P traffic, in: Proceedings of IEEE INFOCOM, 2010.
[28] P. Ferguson, Network ingress filtering, in: Proceedings of the RFC 2827, 2000.
[29] ICMP traceback messages, http://tools.ietf.org/html/draft-ietf-itrace-04 (accessed 01.01.14).
[30] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, W.T. Strayer, Hash-based IP traceback, ACM SIGCOMM Comput. Commun. Rev. 31 (4) (2001) 3–14.
[31] H. Wang, C. Jin, K.G. Shin, Defense against spoofed IP traffic using hop-count filtering, IEEE/ACM Trans. Netw. (TON) 15 (1) (2007) 40–53.
[32] G. Yao, J. Bi, Z. Zhou, Passive IP traceback: capturing the origin of anonymous traffic through network telescopes, ACM SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 413–414.
[33] J. Li, J. Mirkovic, M. Wang, P.L. Reiher, L. Zhang, SAVE: source address validity enforcement protocol, in: Proceedings of IEEE INFOCOM, 2002.
[34] L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred, Statistical approaches to DDoS attack detection and response, in: Proceedings of 2003 DARPA Information Survivability Conference and Exposition, 2003.
[35] A. Yaar, A. Perrig, D. Song, SIFF: a stateless internet flow filter to mitigate DDoS flooding attacks, in: Proceedings of 2004 IEEE Symposium on Security and Privacy, 2004.
[36] K.J. Argyraki, D.R. Cheriton, Active internet traffic filtering: real-time response to denial-of-service attacks., in: Proceedings of the USENIX Annual Technical Conference, 2005.
[37] A. Yaar, A. Perrig, D. Song, StackPi: new packet marking and filtering mechanisms for DDoS and IP spoofing defense, IEEE J. Selected Areas Commun. 24 (10) (2006) 1853–1863.
[38] The aMule Project, http://www.amule.org (accessed 02.08.13).
[39] B. Liu, S. Wu, T. Wei, C. Zhang, J. Li, J. Zhang, Y. Chen, C. Li, Splider: a split-based crawler of the BT-DHT network and its applications, in: Proceedigns of the 11th Annual IEEE Consumer Communications & Networking Conference (CCNC), 2014, p. 9pages.
[40] An open source library of BitTorrent, http://libtorrent.rakshasa.no/ (accessed 02.03.14).
[41] M. Steiner, T. En-Najjary, E.W. Biersack, Long term study of peer behavior in the KAD DHT, IEEE/ACM Trans. Netw. 17 (5) (2009) 1371–1384.
[42] M. Jain, C. Dovrolis, End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput, in: ACM SIGCOMM Computer Communication Review, 32, 2002, pp. 295–308.
[43] W. Stevens, Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, in: RFC 2001, 1997.
[44] M. Kohnen, M. Leske, E.P. Rathgeb, Conducting and optimizing eclipse attacks in the kad peer-to-peer network, in: NETWORKING 2009, Springer, 2009, pp. 104–116.
[45] B. Liu, T. Wei, C. Zhang, J. Li, J. Zhang, Improving lookup reliability in kad, Peer-to-Peer Netw. Appl. (2013) 1–15.
[46] R. Beverly, A. Berger, Y. Hyun, et al., Understanding the efficacy of deployed internet source address validation filtering, in: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, 2009.
[47] J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.
[48] F.C. Freiling, T. Holz, G. Wicherski, Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks, in: Proceedings of the 10th European conference on Research in Computer Security, 2005.
[49] D. Dagon, G. Gu, C.P. Lee, W. Lee, A taxonomy of botnet structures, in: the Twenty-Third Annual Computer Security Applications Conference (ACSAC), 2007.
[50] J. Caballero, C. Grier, C. Kreibich, V. Paxson, Measuring pay-per-install: the commoditization of malware distribution., in: USENIX Security Symposium, 2011.