On Measuring the Impact of DDoS Botnets

Arne Welzel
VU University Amsterdam
The Netherlands
arne.welzel@gmail.com

Christian Rossow
VU University Amsterdam
The Netherlands
c.rossow@cs.vu.nl

Herbert Bos
VU University Amsterdam
The Netherlands
h.bos@cs.vu.nl

ABSTRACT

Miscreants use DDoS botnets to attack a victim via a large number of malware-infected hosts, combining the bandwidth of the individual PCs. Such botnets have thus a high potential to render targeted services unavailable. However, the actual impact of attacks by DDoS botnets has never been evaluated. In this paper, we monitor C&C servers of 14 DirtJumper and Yoddos botnets and record the DDoS targets of these networks. We then aim to evaluate the availability of the DDoS victims, using a variety of measurements such as TCP response times and analyzing the HTTP content. We show that more than 65% of the victims are severely affected by the DDoS attacks, while also a few DDoS attacks likely failed.

1. INTRODUCTION

In a denial-of-service (DoS) attack, an attacker with financial, political or purely destructive motivation disrupts a service of a victim by adding an excessively high load to the victim's service(s). In a distributed denial-of-service (DDoS) attack, an attacker does not use a single source, but abuses multiple attack sources to hide attack traces or to increase the attack impact. A common way to launch such DDoS attacks are DDoS botnets, i.e., networks of malware-infected computers that are remotely steered to participate in DDoS attacks. The large number of attack sources increases the impact in terms of resources and disguises the identity of the actual attackers (e.g., in terms of IP addresses). Given their big potential impact, prior work has analyzed certain DDoS botnets and evaluated the nature of the attacked targets. A research question that remained largely unanswered, though, is if DDoS attacks are actually effective in that they disrupt the availability of the targeted services. In other words: Do the victim servers always go down, or are most sites able to fend off these attacks? And what do the victims typically do to handle or evade the attack? Do they modify their DNS entries, start serving different content, etc.? Little is known about the real impact of DDoS botnets.

EuroSec'14, April 13 - 16 2014, Amsterdam, Netherlands
Copyright 2014 ACM 978-1-4503-2715-2/14/04 ...$15.00
http://dx.doi.org/10.1145/2592791.2592794.

In this paper, we monitor multiple botnets of two popular DDoS botnet families, namely DirtJumper and Yoddos. We join the command & control (C&C) channels of these botnets and record their DDoS targets. We then monitor the service availability for the time these targets are under attack. For example, we resolve the attacked domains to observe DNS-related changes that the victim takes to defeat the attack. Moreover, we measure the time it takes to connect to the victim via TCP, and in case of HTTP-based targets, we monitor and analyze the content of the web sites.

We then present first steps towards interpreting our monitoring results. For example, from the HTTP responses, we try to understand if the service is functioning normally, or if indicators for web site failures such as empty content or bad HTTP status codes can be found. Our preliminary aggregated measurements show that indeed most (65%) of the monitored victims are severely affected by the DDoS attacks. Lastly, we also aim to share our experiences in monitoring DDoS targets to foster future work in this area.

2. OVERVIEW OF DDOS BOTNETS

This section gives an overview of the two DDoS botnet families that we analyzed – DirtJumper and Yoddos. We describe their attack capabilities and C&C protocols. DirtJumper and Yoddos are two powerful and popular DDoS toolkits that have been used since 2010 with hundreds of C&C servers by now. Given the fact that we observed more than 3700 DDoS targets within less than ten weeks of monitoring shows the significance and representativeness of these two botnets. DirtJumper and Yoddos are also the two most-active DDoS botnet families in our malware analysis system SANDNET [4].

2.1 DirtJumper

DirtJumper is a multi-threaded DDoS bot written in Delphi and the Ararat Synapse library[1]. We analyzed three bot versions that stem from the same code base and indicate that the DirtJumper botnet family is actively maintained.

The oldest version of DirtJumper supports only one attack mode (HTTP `GET` requests). When attacking a host, the bot does not wait for a HTTP response, but closes the TCP connection after the request was sent. The other two versions provide four different attack modes. Besides the above mentioned `GET` request attack, these versions further support an attack where the bot waits until it received the HTTP response from the victim server. Moreover, support for `POST` request attacks was added. Another difference of

[1] `http://synapse.ararat.cz/doku.php`

Cmd ID	Functionality	Target
`0x00000001`	UDP with raw socket. `rand()` spoofed IPs	host/IP
`0x00000002`	Same as `0x00000001`	host/IP
`0x00000004`	Same as `0x00000001`, single thread	host/IP
`0x00000008`	UDP with raw socket. Spoofed IPs	host/IP
`0x00000010`	Same as `0x00000008`	host/IP
`0x00000020`	TCP msgs with `\%d<<<<<I@C<<<<<\%s!`	host/IP
`0x00000040`	UDP with rnd data and msg lengths	host/IP
`0x00000080`	TCP with rnd data and msg lengths	host/IP
`0x00000100`	UDP with rnd data but structured message	host/IP
`0x00000200`	TCP with rnd length for each message	host/IP
`0x00000400`	`connect()` 200 sockets (only once)	host/IP
`0x00000800`	`connect()` 200 sockets (continuously)	host/IP
`0x00001000`	HTTP, Host and Referer fixed, no `recv()`	URL
`0x00002000`	HTTP, path is `/`, no `recv()`, no Referer	host/IP
`0x00004000`	HTTP, `no recv()`, varies path to fetch	URL
`0x00008000`	HTTP, `InternetOpenA()`	URL
`0x00010000`	Custom UDP/TCP data from C&C server	host/IP
`0x00100000`	Change C&C Server	
`0x00200000`	Ping back to server with `0x11223344`	
`0x00400000`	Restart system	
`0x00800000`	Shutdown system	
`0x01000000`	Uninstall service	
`0x02000000`	Stop attack	
`0x04000000`	Download & execute file	
`0x08000000`	`ShellExecute()` with `open` as operation	

Table 1: List and description of Yoddos commands.

these versions is the addition of a randomly generated, persistent bot identifier which is transmitted to the C&C server when requesting new commands. This allows a botmaster to identify bots independently of their source IP address.

During an attack, threads randomly pick a new URL from the list. Therefore, if multiple URLs were provided by the C&C server, the resulting traffic alternates among all available URLs. For each attack request, the `User-Agent` header used in a HTTP request is randomly picked from a hard-coded set of 224 strings included in the bot binary. Bots use multiple threads in order to open concurrent connections to victims. A main thread pulls commands from the C&C server in regular intervals and then creates as many threads as specified in the command.

All DirtJumper versions only support attacks using HTTP requests. The bot implementations do not provide functionality to use plain TCP connections to send either custom, or randomly generated, data. Further, we could not reveal support for attacks using UDP as transport protocol. This also means that DirtJumper never attempts to spoof its source IP address. Finally, DirtJumper does not support download & execute functionality and the botmaster has no possibility to update the bot or download arbitrary executables.

2.2 Yoddos

Yoddos is a multi-threaded DDoS bot with additional functionality for remote administration of the infected host. Compared to DirtJumper, it offers a wide variety of different DDoS attack modes. Moreover, the bot allows the botmasters to download and execute further software on the infected host. For this reason, Yoddos might be considered a remote administration tool with DDoS capabilities.

Yoddos communicates with the C&C server using a non-encrypted custom protocol over TCP. Initially, the bot sends information about the infected host to the C&C server. The bot then waits for the C&C server to reply with a command message. In contrast to the pull-based DirtJumper, the C&C server *pushes* commands to the bot. Also deviating from DirtJumper, Yoddos can only attack a single victim at any time. Further, after a bot received an attack command, the attack is explicitly stopped with a specific termination command sent by the C&C server. This characteristic makes it easy to determine how long a bot was commanded to attack a victim.

Table 1 provides an overview of the attack types that Yoddos supports. The botmaster may combine multiple of these bitmask commands to instruct a bot to execute multiple attack types. Commands with values lower than `0x00100000` present attacks. Values greater or equal to `0x00100000` provide the botmaster with functionality to control an infected machine. For example, Yoddos can download executables, run shell commands, or restart the infected system.

The attack type `0x00010000` instructs the bot to send custom data using either UDP or TCP. Instead of sending predetermined, or randomly generated, messages to the victim, using this attack mode, the C&C server sends payload to the bot that it should use for the attack. This functionality allows the botmaster to send any message using either TCP or UDP. As a concrete example, a botmaster might craft some specific DNS request for a victim and forward it to the bot. In this case, the bot will send the provided content to the victim, which in turn might be a DNS server.

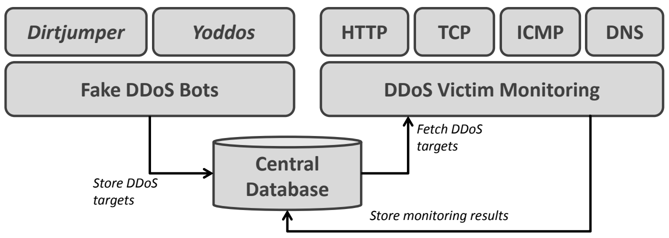

Figure 1: Overview of the monitoring system architecture.

3. BOTNET AND VICTIM MONITORING

We developed a framework to monitor the DDoS botnets and their targets. Our framework consists of two main components: the DDoS C&C monitoring and the DDoS target monitoring. Figure 1 shows the general architecture of our monitoring system. The C&C monitoring component tracks the commands that current C&C servers issue to their bots, parses the commands and stores the DDoS targets in a central database. The victim monitoring component, in turn, starts to monitor the availability of the attack targets. This section describes both parts in more detail.

3.1 C&C Monitoring

To monitor the DDoS commands that a C&C server issues, we implemented fake bot clients for the two DDoS bot families Yoddos and DirtJumper. We used manual reverse engineering to understand the syntax and semantics of the C&C protocol. While the alternative—executing the actual bot and observing the commands it receives—would have been easier, reimplementing the bots provides much better control over the monitoring process. Moreover, this way we can track C&C commands without actually taking part in the DDoS attacks.

Our fake bots connect to all known C&C servers and log the C&C commands. One of the key issues of a monitoring system is thus to have an extensive list of active C&C servers. We obtain a list of C&C servers from our dynamic malware analysis environment SANDNET [4]. Next to our daily feeds from anti-virus vendors, we added malware samples by searching VirusTotal for new DirtJumper and Yoddos samples. We used pattern matching on the resulting network traffic to identify the C&C server communication and the server (i.e., the IP address, port and possibly the URL of the C&C server). For all known C&C servers, we periodically retrieve C&C commands via our bot clients.

3.2 Victim Monitoring

When a new target is found by the C&C monitoring component, we start to periodically monitor the DDoS targets in pre-defined intervals. Our goal is to monitor the availability of services that are currently under a DDoS attack. After an attack on a target stopped, we continue to monitor the targets for another 8 hours with the intention to model the normal behavior of the target. We chose 8 hours as a compromise — the longer we monitor, the more likely it is that we will observe the normal behavior after a victim has resurrected. However, too long measurement periods would also capture regular infrastructural changes that would have happened regardless of the DDoS attack.

We use the following monitoring methods to observe the behavior and availability of DDoS targets:

DNS

We monitor DNS to observe i) if a target reconfigures its domains to evade the attack and ii) to measure if the target's DNS infrastructure is affected by the attack. For all targeted domains we monitor the DNS `A` records. To avoid side-effects due to caching, we use a local DNS server as a recursive nameserver and disabled caching. The IP addresses resolved via DNS are used by subsequent monitoring methods. If at some point the target's IP address changes, we will start monitoring the new address and continue monitoring the previously returned IP address.

TCP Connections

This method records the time to establish a TCP connection to the DDoS target. Technically, we measure the time that the system call `connect()` requires to complete. After a TCP connection is established, we instantly close the connection again. Therefore, an application that might have accepted this testing connection is likely to drop the connection immediately, avoiding side-effects. Our implementation uses a timeout of 60 seconds and the default TCP SYN retransmission settings in Linux.

HTTP

We have observed that most DDoS targets are web servers. However, the TCP measurements only take the operating system into account, i.e., the time it takes to establish a TCP connection. This gives us no indication if the web server was actually functional. We thus also monitor the HTTP response time of attacked HTTP servers and record the HTTP response. HTTP responses provide additional information about the host, such as status codes, HTTP headers and the HTTP body — which may provide insights about the behavior and status of an attacked host.

3.2.1 Measurement Intervals

For our repeated measurements, we chose moderate measurement frequencies so that the negative impact of our measurements on a victim can be considered negligible. That is, compared to the actual attack traffic, the load that our monitoring adds is relatively small and significantly lower than the attack traffic of a single bot. In addition, we want to avoid triggering rate limiting techniques as, for example, `mod_evasive` for Apache[2]. In such a case, the behavior of a victim observed by the monitoring component would not represent the behavior that a legitimate client would observe. We monitor DNS and reverse DNS every four minutes (240s), and establish TCP connections and issue HTTP requests every 7.5 minutes (450s).

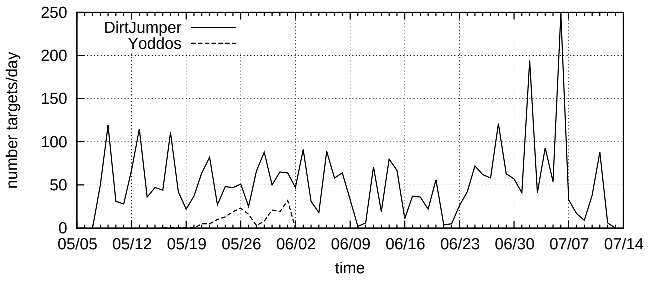

Figure 2: Number of targets per day and botnet family.

3.2.2 Reducing Monitoring Jobs

In principle, DDoS botnets can attack multiple targets (e.g., URLs) all belonging to the same victim host. In the worst case, we would then monitor all of these targets and add a high load to the (single) victim server. As an example that happened during the development and testing phase, one of the DirtJumper C&C servers provided more than 50 different URLs addressing `http://newkn.ru/`, thereby targeting a single host. We take precautions to keep the monitoring frequencies low even for these attacks. To save resources of the victim in these situations, we reduce the number of monitoring jobs. The basic idea is that monitoring jobs do not need to be executed if either a monitoring job that is addressing the same host is currently running, or if a result is available of another monitoring job that ran recently and was addressing the same host.

4. C&C EVALUATION

In this section, we present our results from monitoring the C&C servers of the two DDoS botnet families.

4.1 Attack Targets

We monitored one Yoddos and 13 DirtJumper C&C servers during the period from May 7th, 2012 until July 13th, 2012. Figure 2 shows the number of DDoS targets that we received from these servers during this period. In total, we have recorded 3812 targets, of which 3636 were attacked by DirtJumper and 176 by Yoddos.

The number of new target URLs does not necessarily relate to the number of individual victims. Botmasters of DirtJumper botnets can, for example, attack multiple URLs (i.e., *targets*) pointing to the same host (i.e., *victim*). As discussed, we group multiple targets to a single victim if they point to the same web server. That is, we group attack targets that point to equal IP addresses, resulting in 646 victims. Figure 2 shows the number of targets (and not victims), revealing the general activity of these C&C servers.

[2]`http://www.zdziarski.com/blog/?page_id=442`

Mode	Attack description	Count	Ratio
01	GET request no download	1406	40.0%
02	GET request no download (sync.)	114	3.2%
03	GET request with download	1146	32.6%
04	POST request with download	840	23.9%
05	Unsupported/Unknown	13	1.4%

Table 2: DirtJumper attack mode distribution.

4.2 Attack Types

DirtJumper supports four different attack types. One way to estimate the popularity of each attack mode is to look at all commands received from C&C servers. However, the distribution in this case depends heavily on the length of an attack, or until the botmaster decided to change the attack mode. Another way is to only look at the first command for a given target. In this case, the length of an attack does not influence the distribution. Still, then individual C&C servers, those with a high number of new targets per day, will influence the distribution. The resulting distribution using the latter approach is shown in Table 2.

Most popular is the `GET` request attack without receiving the response from the server. This attack abuses the fact that web servers may spend resources to prepare HTTP responses even if the client does not wait for it. For servers that monitor the clients and stop assembling a response if the client disconnects, the other two attack types (`03` and `04`) provide powerful attack alternatives and were used in more than half of the cases. We have only seen little use of attack mode `02`, in which DirtJumper synchronizes its threads such that all HTTP requests are issued simultaneously. In addition, we observed a small fraction of attacks with mode `05`, which was not present in the DirtJumper binaries we analyzed, revealing a potentially new command.

In Yoddos, the botmaster used the possibility to select multiple attack modes. The most popular bit field (37.6%) is found to be `0xC0` (combination of `0x40` and `0x80`), which enables attack modes using random TCP and UDP messages. The next most frequent (9.9%) bit field is `0x48`, attacking hosts using UDP with the legitimate and spoofed source IP addresses. The following bit field (9.3%), `0xc8` is the combination of all three different attack modes just mentioned. Interestingly, the botmaster used command `0x10000` which allows for overwhelming the victim with custom data in 0.6% of the attacks. The data provided by the C&C server, however, was random in all cases that we observed[3].

4.3 Attack Durations

Observing the C&C commands allows us to determine how long a DDoS attack lasts. For DirtJumper, we observed 545 targets and 934 distinct attacks and for Yoddos 101 targets and 224 distinct attacks[4]. In total, we observed 646 victims and 1158 attacks with no overlap between the two botnet families. However, we have to differentiate how to compute the attack length for both botnet families.

DirtJumper does not have start and stop commands. Instead, DirtJumper C&C servers continue to send the same command until the bots should stop the attack. Additionally, the botmaster can add or remove URLs targeting the same host while the attack is proceeding. Moreover, it is also possible that an attack on a target is stopped, but later on continued. For these reasons, computing the attack duration from the first and last command received for a given target possibly ignores such gaps. Therefore, we grouped the URLs received from the C&C server into sets of related targets and derived the attack duration based on the commands received to attack any of the targets in these sets.

[3]We observed the data: `BA23 4EDA B450`

[4]Note that the number of attacks is higher than the number of targets, as multiple attacks can be launched per target.

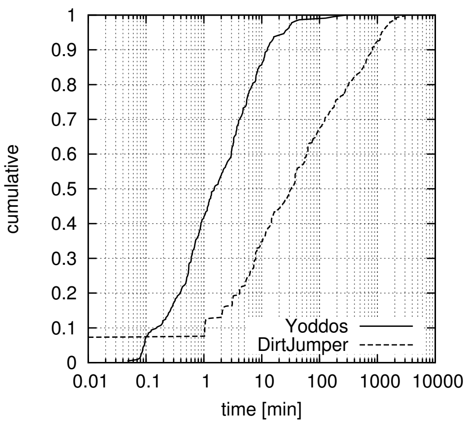

Figure 3: Attack length distribution for DirtJumper and Yoddos.

In contrast to DirtJumper, for Yoddos it is fairly trivial to determine how long a DDoS attack lasted. An attack always starts with the target description and is stopped with an explicit stop command. Thus the length of an attack is simply the difference of the time stamps of these messages.

Figure 3 shows the distribution of the Yoddos and DirtJumper attack lengths. About 96% of the Yoddos attacks last less than 30 minutes. In contrast, the majority of DirtJumper-attacks (51%) last at least half an hour. This may reveal that both botnet families follow a slightly different business model – short attacks may be interpreted as warning shots, while the longer attacks continuously affect services (e.g., to put pressure on the victims and extort them).

About 7% of all attacks have a zero length. This artifact is caused by DirtJumper C&C servers sending a single command only. In contrast, 14% of the attacks last longer than 10 hours. About 69% of the Yoddos attacks last less than four minutes and also 19% of the Dirtjumper attacks are that short. Given our low target monitoring frequencies, it follows that only a single data point is gathered during the actual attack period for these attacks. Of the total of 1158 attack phases, we thus left out 743 phases that lasted less than 30 minutes and thus did not have sufficiently many measuring points. In future work we plan to include an algorithm based on an exponential backoff to determine the measurement interval so that we can also measure the effects of short-lived attacks.

5. VICTIM EVALUATION

Now that we know the targets of the DDoS botnets, we will evaluate to what extent the attack targets are actually affected by the DDoS attacks. These results are based on monitoring the 415 attack phases that lasted for at least 30 min, i.e., we excluded attacks for which our monitoring did not capture significantly many measuring points.

DNS-related change	# of att.	% of att.
Change in IP	48	13.1%
IP-based round-robin	8	2.2%
Consistent loopback IPs	33	9.0%
Temporary loopback IPs	17	4.6%
DNS resolution failed	18	4.9%
No change in DNS	242	66.1%
No DNS info (IP-only attack)	49	n/a

Table 3: DNS observations for attacked victims.

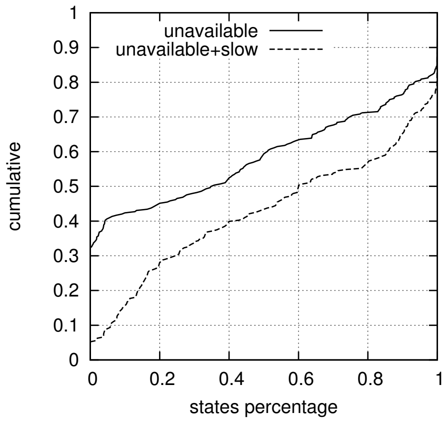

Figure 4: State distribution for attack phases.

5.1 DNS

Table 3 shows how the DNS settings of the victims were affected during the attack phase. The percentages are relative to 366 attacks where a victim was identified by a domain. In 49 cases the victim was identified by an IP address only.

In 34% of the attacks, the attacker provoked a reaction to the DNS of the victims. For example, we have observed that 9% of the victims configure the `A` record of their domain to a loopback address to redirect the DDoS traffic. This, in turn, also means that legitimate clients cannot reach the service anymore. In 5% of the attack phases, the DNS resolution of the victim failed during the attack for more than 50% of the DNS requests. In further 15% of the cases, the `A` record of a domain was changed to another IP address or another set of IP addresses. For example, we have observed victims that redirect traffic to professional load balancing and DDoS protection services.

5.2 TCP Connections

We have shown that DNS is either affected by the attack or presents the victim a technique to redirect attacks. Next to DNS, we also monitored if the attacked services still accepted TCP connections, and if so, we measure the communication latencies. In particular, we focus on those victims with *no* change in the DNS records. When a victim is attacked on multiple ports, we measure the latencies for each port and provide average response times.

Figure 4 shows a CDF of the service availability ("unavailable", solid line) of victims under attack. The figure shows that 40% of the victims were unavailable for more than half of the attack duration. Only 15% of the victims were completely unavailable during the entire attack time. In 32% of the cases, the victims were always available. This shows that most attacks indeed have a negative influence on the availability of the victim's services. Although most services are partially still available, these services have a flapping reachability state. We speculate that also these services are thus too unreliable at that point to be actually used by humans. For example, a web server needs to service dozens of TCP connections to serve a web site, and partially failing connections already have a quite negative impact.

Additionally, Figure 4 provides information where we consider the response time of victims ("unavailable+slow", dashed line). That is, next to being unavailable, we also add hosts to the fraction of affected target if their response times were significantly slower than in the non-attack phase. We define significantly slower if the time to complete the TCP handshake was greater or equal to the 90th percentile of the times of connection attempts during the non-attack phase. Figure 4 shows that for only 5% of the victims the TCP connections were not affected at any point in time — neither their availability nor their response time. 20% are either slow or unavailable throughout the whole attack. 60% of the victims are slow or unavailable in more than 40% of the measurements point.

HTTP-related observation	# of att.	% of att.
Incomplete read during request	5	2.0%
Connection reset	50	19.6%
Always 5xx error status codes	81	31.8%
Temporarily 5xx error status codes	48	18.8%
Switched from HTML to non-HTML	16	6.3%
Content length diff > 50%	81	31.8%
Content shorter than ≤ 64 bytes	15	5.9%
Content empty	33	12.9%

Table 4: HTTP observations for attacked victims.

5.3 HTTP

Even if a victim can still accept TCP connections, this does not completely reveal the availability of its services. In fact, a server may not be able to serve actual content to the accepted client. We therefore also evaluate if the most-commonly attacked service – HTTP – is available during the attack time for all 255 attacks against web servers.

Table 4 summarizes our HTTP-based observations. In 32% of the attacks, the victim always fails with a `5xx` status code. In further 19% of the attacks the victim only temporarily fails with this error. However, even if the request was handled with a normal status code (`2xx`), we observed that responses in many cases do not represent replies a normal service client would expect. For example, in 13% of the cases we received empty contents (HTTP bodies) and in 15 cases the content was very short (typically showing an error message). Moreover, in further 32% of the attacks, the served web content lexically differed more than 50% from the web site served during a non-attack phase.

5.4 Combined Results

So far we have only looked at each of the three different layers that we monitored separately. Clearly, if the attack did not influence DNS in any way, but the HTTP server fails to deliver content properly, the service is rendered unavailable. In this last evaluation, we therefore combine our observations to measure the overall service availability. That is, for each of the three layers (DNS, TCP, HTTP) we define

Category	effective attacks (number / %)	
DNS	48	11.6%
TCP	142	34.2%
HTTP	187	45.1%
Combined	**270**	**65.1%**

Table 5: Combined victim unavailability results.

states that indicate service unavailability.

For DNS, we consider all services that changed the IP addresses (e.g., to loopback or a load-balancing service), indicating that the victim could not handle the load of the attack itself. For TCP connections, we consider victims which were either unavailable or slow in at least 50% of our measurements. For HTTP, we consider victims that refused or reset the HTTP connection, delivered empty content, had erroneous status codes, or delivered content that deviated too much from the original content during a non-attack phase. When combining these observations, we can measure the fraction of victims that were overall affected by any means.

Table 5 summarizes our results for all the 415 aggregated victims. Looking at each layer separately, at most 45% of the victims were affected (as in the case of HTTP). However, when combining the results, it is clear that almost two thirds (65%) of the victims were severely affected by the DDoS attacks. This, however, also leaves 35% of victims that were apparently not really negatively influenced by the attacks. In future work, we plan to investigate the differences between those two victim groups, such as their available resources to withstand DDoS attacks.

5.5 Victim Survey

In an approach to understand the motivation behind the DDoS attacks, we have sent an email questionnaire to the DDoS victims. To automate this effort, we sent emails to `webmaster@<domain.tld>` of all victims, including a textual log of the attack details that we have observed. Unfortunately, the response rate of the survey was very low — only two DDoS victims have answered the survey. While our survey results are statistically not significant, we still describe our efforts to show the difficulties in contacting DDoS victims and understanding the attackers' motives.

Person A described that he was attacked multiple times in the past and all attacks were accompanied by extortion. As a defense, person A uses `iptables` to filter the botnet's IP addresses. Person B mentioned that he it was also not the first attack he observed. In contrast to the other victim, he was not blackmailed, though. He mentioned to filter bogus HTTP requests by inspecting the HTTP headers.

Next to these two responses, we received 259 email bounces showing the the `webmaster@<domain.tld>` address was not configured well. Our results unfortunately indicate that email is not the appropriate way to survey DDoS victims.

6. RELATED WORK

Specht/Lee [5] and Mirkovic/Reiher [3] gave a general overview and proposed a taxonomy of DDoS attacks. Closest to our work, Büscher and Holz monitored DirtJumper C&C servers [2]. They have described one of the DirtJumper variants, tracked about 2000 attacks and categorize the attack targets. We extend their work and measure the *impact* of DDoS attacks, i.e., we monitor how DDoS victims are affected by the attacks. In addition, we describe two DDoS botnet families in multiple variants, showing a high diversity in the DDoS botnet landscape.

Security vendors like Arbor Networks also reverse-engineer DDoS bots [1]. Their reports include a short discussion about the victims received when infiltrating botnets with reimplementations of bots. Whether they actively monitor the behavior of victims under attack, however, is unclear.

7. FUTURE WORK AND CONCLUSION

We have shown that botnet-driven DDoS attacks have indeed a high impact and negatively affected the availability about two thirds of the victims in our measurements. Our observations allow to break down what resources become scarce during an attack. For example, we have shown that most typically HTTP servers are the bottleneck, and not the resources the OS needs to reserve for handling further TCP connections. This allows to steer future research in the area of DDoS attack mitigation.

We also learned a few lessons in our early attempts to measure the impact of DDoS attacks. Measurement accuracies can for example be improved if the measurement frequencies are chosen in an improved way. We could not evaluate the impact of many short-lived attacks, as our measurement interval was static and too coarse-grained. A possible solution for future work is using a more fine-grained measurement interval at the beginning of the attack and then slowly decreasing the measurement frequencies. In addition, we did not yet correlate the attack types with the attack impact. This correlation may give further insights into the bottlenecks at the victims. Moreover, we plan to use multiple monitoring systems in the future to decrease the likelihood of geographical artefacts.

We also have not yet addressed the types of the DDoS targets, as done in prior work [2]. In future work we plan to analyze the targets (e.g., according to their businesses) and as such to identify the motivation behind DDoS attacks.

References

[1] Arbor Networks. `http://ddos.arbornetworks.com/`.

[2] A. Büscher and T. Holz. Tracking DDoS Attacks: Insights into the Business of Disrupting the Web. In *Proceedings of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, San Jose, CA, USA, April 2012.

[3] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 39–53, April 2004.

[4] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. C. Freiling, and N. Pohlmann. Sandnet: Network Traffic Analysis of Malicious Software. In *ACM EuroSys BADGERS*, 2011.

[5] S. M. Specht and R. B. Lee. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. In *Proceedings of the International Conference on Parallel and Distributed Computing (and Communications) Systems (ISCA PDCS)*, San Francisco, CA, September 2004.