CLOUDFLARE®

A Submission from Cloudflare, Inc., in response to

“Promoting Stakeholder Action Against Botnets and Other Automated Threats”

A Notice by the National Telecommunications and Information Administration on 06/13/2017

Cloudflare is a Internet security company working to build a better Internet, including limiting the damage done by botnets. We protect more than six million web properties around the world from Distributed Denial of Service (DDos) attacks. With over 115 data centers in more than 50 countries, we serve approximately 10% of global Internet requests, which is more web traffic than Twitter, Amazon, Apple, Instagram, Bing, & Wikipedia combined. Every week, the average Internet user travels through our network more than 500 times. In a typical day, the Cloudflare network of data centers blocks more than 400 million DDoS attacks. This gives us a unique perspective on where botnets are located and how they are being used to launch DDoS attacks.

Since Cloudflare was founded eight years ago, we have worked to make Internet security easy and inexpensive so that anyone with a website can use leading-edge technologies to ensure their visitors have a safe, secure, and reliable way to connect to the Internet. In addition to standard web security services, Cloudflare offers emergency protection for those in the midst of a DDoS attack, giving all users the ability to mitigate attacks and keep themselves online.

We believe the power of innovation and cooperation can help address most of the security issues we face today and will face tomorrow. As such, Cloudflare engineers work closely with standards bodies, technical organizations working on network issues and cybersecurity, and other Internet companies to identify and address cyber vulnerabilities and limit cyberattacks.

While governments have played a key role in the development of the Internet, most of the solutions to cyber threats come from the private sector and the technical community. Rather than resorting to regulation or imposing one-size-fits-all standards, governments should help foster the kind of innovation and competition that has enabled the phenomenal growth of the Internet for more than twenty-five years. New technologies, adoption of best practices by businesses and other organizations, smart procurement, awareness raising, and greater transparency are the key to thwarting botnets--not new legislation or regulation.

The key question is “How can a distributed, multi-pronged approach to botnets convince the people running them and using them to find a new line of work--preferably a legal one?”

Evolution of Botnet Attacks

When botnets first emerged as a problem more than fifteen years ago, almost all the traffic was coming from compromised personal computers. Today, the situation is much more complicated. Cloudflare data indicates that almost every major attack exploits vulnerable corporate servers. Furthermore, cybercriminals have found ways to exploit cloud computing services to launch attacks.

In a typical DDoS attack, a malicious party tries to make a website or web service unavailable by overwhelming it with requests from compromised machines (called “bots”) from around the world. If the volume of attacks from these bots is large enough, the web server may become overloaded and unable to provide services to legitimate clients. One way attackers generate sufficient traffic for successful DDoS attacks is to collect and control a set of compromised machines or devices, known as a botnet.

To increase the size of DDoS attacks even further, attackers use reflection or amplification. In an amplification attack, the attacker uses a spoofed IP source address to submit a query that will prompt a large reply back to the target IP address. These attacks therefore require both an internet protocol vulnerable to reflection and servers or internet devices that support the vulnerable protocol. Using these tools, it is very easy for a botnet to magnify the amount of data delivered to the target twenty times or more. Reflection attacks are relatively common; Cloudflare’s system detected a reflection attack on average every 40 minutes, during a recent six-month period. Although Cloudflare has publicly identified the protocols most often used for attack (see Appendix A), attackers constantly look for new protocols that can be used for reflection.

At the same time, cybercriminals are learning to harness the power of simpler devices like connected cameras. In September, 2016, Mirai software was used to infect more than 100,000 devices and unleash one of the largest DDoS attacks up to that time. Early the following month, Cloudflare identified multiple large attacks coming from Internet of Things (IoT) devices, like CCTV cameras, and described the attacks as the new trend. (See Appendix B.) Later the same month, Mirai was used to disable DNS service provider Dyn, which in turn had worldwide impact and resulted in popular websites like GitHub, Twitter, Reddit, Netflix, and Airbnb being unavailable for hours. (See Appendix C.). The attack on Dyn highlighted both the need to protect Internet of Things (IoT) devices better and the need to defend against traffic coming from infected IoT devices. It also showed how an attack on one part of the Internet infrastructure can affect dozens of the most popular websites and billions of users.

Towards A Comprehensive Strategy for Addressing Botnet Attacks
To stop these types of attacks, attackers must be convinced they will be ineffective in achieving the desired goal of taking down the website. Based on the extensive number and type of attacks we've observed, Cloudflare suggests a holistic approach to stopping these attacks, which needs to include four components: (1) Blocking DDoS traffic, (2) Preventing reflection attacks, (3) Fixing or isolating the number of infected computers and devices that comprise botnets, and (4) Taking control of the servers used to command and control botnets.

The first and most immediate priority is to block traffic that is coming from the millions of PCs, laptops, servers, and other devices that have been infected with malware and are being used for attacks. The first defenses developed to block traffic from DDoS attacks were hardware-based. However, over the last five years, cloud-based techniques for identifying and blocking traffic from botnets have taken over this role and have proven to be more effective, versatile, and cost-effective. This has been particularly helpful in less developed countries, where the scarcity of IT skills makes it hard to defend devices.

Companies like Cloudflare have developed very cost-effective means of using cloud technology to detect and block malicious traffic during such incidents. Cloudflare's security services operate at the edge of the network, allowing us to adapt and mitigate threats as they are identified and use the capacity of our network of data centers. Our enterprise-class DDoS protection network, for example, has a capacity of more than 10 terabits per second. Operating at the edge also allows the use of the collective intelligence of an entire network to identify and block new threats.

But the problem is that too many web site owners remain vulnerable to attack because they have not taken advantage of any of the available solutions, some of which can be installed in less than five minutes.

The second component of a strategy to address botnets is to prevent reflection attacks by limiting the number of servers that are not properly configured. Much of this can be accomplished by transparency and sharing information about improperly secured networks. For example, network operators have largely made Smurf attacks -- the original amplification attacks that date back to the late 1990s -- a thing of the past simply by reconfiguring their systems in response to the threat. Similarly, in 2014, after experiencing a large DDoS reflection attack with nearly 400 Gbps of traffic, Cloudflare published a list of networks originating these attacks. (See Appendix D.) Within less than two weeks, more than 75 percent of the vulnerable servers that had been identified were no longer vulnerable. Projects to identify servers vulnerable to reflection have an important role to play in this space, but the need for education and efforts to encourage all levels of government and the private sector to take similar steps remains.

The third component of a strategy addresses the most difficult challenge: How to fix or isolate the tens of millions of devices that have been infected with malware and turned into sources of DDoS traffic. Many of these are older devices with outdated or pirated software, making them easy targets (and hard to fix). Since botnets can be found in almost every country--on almost every ISP--a broad-based approach is needed. And still, it is clear that it will not be possible to fix every infected machine. The private sector can help address this problem by crafting new approaches and technologies, such as remote remediation, isolation or disabling of devices.

The fourth and final component of a strategy for limiting the damage done by botnets (and the profits they generate) is to find and shutdown the servers used to control them and the services used to collect money from extortion schemes. The private sector plays an important role in helping to identify command and control servers and prevent their use. Since such command and control servers may be located in several countries, however, such efforts require extensive international cooperation.

Promising Technologies and Techniques

New technology can help implement all components of this strategy. Companies continue innovating to develop new ways of addressing the threat.

In order to block DDoS and prevent reflection attacks, for example, network operators use a number of tools that help monitor networks, detect harmful traffic, and reduce the effectiveness of attacks. If adopted more widely, these types of tools and techniques could make the entire infrastructure of the Internet more resistant to DDoS attacks. Examples include:

1) Tools like NetFlow / IP-FIX, which enable network providers to detect DDoS traffic and ascertain where it is coming from. At Cloudflare we have had a positive experience with these technologies, as discussed in the blog post (Appendix E) by Marek Majkowski. Unfortunately, these technologies are rarely deployed by the internet providers. This makes it very hard to assess the true source of reflection attacks.
2) Tools for blocking unusual spikes in traffic. One example is BGP Flowspec, which is defined in RFC 5575 and provides a versatile, granular options for filtering out DDoS traffic.
3) Shrinking the packet size of DNSSEC to reduce the effectiveness of DNS reflection attacks. This technique was described in a blog post (Appendix F) by Dani Grant published in March 2017.

There are also technological solutions that can help address the growing threat posed by botnets of IoT devices--and to protect IoT devices from attacks. Cloudflare, for example, recently announced Orbit, which allows IoT manufacturers to secure IoT devices. This cloud-based service enables IoT device manufacturers to have a Virtual Private Network (VPN) for the devices they sell. Rather than relying on the devices being always secure, the manufacturers can use the cloud as a layer of protection to deflect exploits against a device that may not have been patched. This helps ensure the IoT devices that cannot or have not been patched are not taken over to launch DDoS traffic.

These are just some examples of the existing tools and techniques for mitigating DDoS attacks. Important research is being done in corporations and academia to develop new techniques using technologies like artificial intelligence and machine learning. One area of work that is particularly important (but often underappreciated) is efforts to make anti-DDoS tools easier to deploy and use.

So we have the tools and techniques needed to address DDoS attacks--and more are being developed. Most experts agree that 80-90% of all cyberattacks could be avoided if all computer users simply practiced effective cyber hygiene and used easily-available tools. The case is similar for Distributed Denial of Service attacks and botnets.

The Power of Private Sector Solutions

In many ways, botnets are at a point in their development that is similar to that of spam fifteen years ago. At that time, the volume of spam was growing at an alarming rate and defenses against it were still being developed. In response to the growing cost and annoyance caused by spam, the United States Congress passed the CAN-SPAM Act that empowered the Federal Trade Commission and law enforcement agencies to bring charges against spammers. While some spammers were convicted, most spam is difficult to trace and often comes from outside the United States. Despite the CAN-SPAM Act, the amount of spam sent continued to grow so that by 2010, 80 percent all email messages (around 200 billion messages a day) were spam.

While legislation was not effective in preventing spam, nowadays a range of tools for blocking spam prevent the vast majority of spam from reaching their target. ISPs prevent spammers from sending spam. Corporate networks screen out spam, particularly messages with malware attached. And email inbox software keeps spam out of inboxes. Spammers cannot make money if the email messages they send are never seen or opened. In addition, many organizations have security software to block employees from clicking on URLs known to be associated with spam scams. And governments have cracked down on the banks processing payments associated with the phony products and services spammers are advertising. This kind of multi-pronged effort has made it much less profitable to send spam than it was ten years ago.

Of course, cybercriminals have moved onto other more lucrative forms of online crime. The rise of DDoS attacks is due in part to organized crime groups that use botnets to knock websites offline and then demand extortion money from the website owner. Ransomware is another new source of income of cybercriminals.

Transparency and Motivation

How can companies be encouraged to do more to ensure they are defended against botnet attacks (and not inadvertently facilitating them)? One thing is greater transparency. Data breach notification requirements that ensure companies inform their customers when their personal data may have been compromised have motivated companies to use encryption and other means to better protect the data they collect.

Similarly, better information on which companies have suffered from DDoS attacks (and why) could be very helpful in protecting best practices. Knowing the damage done to victims can inspire other companies to install adequate anti-DDoS technologies.

Another place where transparency could motivate best practices is in DNS amplification attacks. ISPs and companies like Cloudflare have data on which web servers are not properly configured and are being used for amplify DDoS attacks. Notifying the owners of these servers (and publicizing those who do not correct the problem) could highlight the extent of the problem and encourage better cyber hygiene.

It would also be very useful to compile and publish data and analysis on where DDoS traffic is coming from, whose IT systems are being used for DNS reflection attacks, and how many organizations are being adversely affected by DDoS attacks. Clearly, any effort to collect and analyze such data would need to take into account the privacy of network users. One of the biggest obstacles to identifying individual systems that are compromised is that once they are identified and their owners notified, there will need to be some way to respond to resulting requests for help in fixing vulnerabilities.

Even if network providers and network security companies do not wish to share data on where specific bots are and which sites they are attacking, by aggregating and analyzing such data, it would be possible put a spotlight on those countries that have a particular large percentage of infected computers and poorly-configured servers (and limited deployment of anti-DDoS services). Such “report cards” could help motivate both government and private-sector efforts in those countries--and benefit website owners around the world.

Governance and Stakeholders

For more than thirty years, there has been no global coordinating body guiding the evolution of the Internet. Indeed, in some cases, different groups compete to develop and promote different standards or guidelines to address specific challenges and opportunities. This is a feature, not a bug. The idea that different ideas can “bubble up” and get tested in the marketplace of ideas is why the Internet and the Web have been able to evolve so quickly and grow so fast--and meet the varying needs of different users and communities.

The evolution of the Internet has been guided by decentralized, bottom-up processes. These include: (1) Standards-setting organizations such as the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), (2) The Internet Corporation for Assigned Names and Numbers (ICANN) and the Regional Internet Registries (RIRs), (3) NANOG and other network operators’ groups, (4) Groups responsible for the security infrastructure of the Internet (e.g. certificate authorities), (5) Computer Emergency Readiness Teams (CERTs) that respond to cyberattacks and other major incidents, analyze threats, and exchange critical cybersecurity information, and (6) A number of other organizations (e.g. the Internet Society and the Internet Governance Forum).

The story is no different in the fight against botnets. Companies are competing to provide new, cheaper, easier-to-use techniques for blocking DDoS traffic and prevent DDoS attacks in the first place. Various companies and governments are collecting and analyzing data on where threats and vulnerabilities are and how they are changing. There is even a healthy competition between intergovernmental bodies trying to track down users of botnets and help national law enforcement agencies arrest them. In most technical areas, existing, private-sector-led bodies are doing a good job of fostering new, innovative ways to make the Internet more secure, more robust and more resilient, and more efficient. However, as mentioned above, there are many areas where more work is needed to encourage owners and users of IT systems to use available tools and techniques to make their systems--and the Internet--more secure. Governments have a particularly important role to play in ensuring that their own systems and services are resistant to DDoS attacks.

The Role of Government

Securing Government Systems
With relatively little effort and expense, Federal agencies could be excellent role models in adopting best practices to protect their IT systems from attack.

In addition to mandating a process to promote action against botnet attacks, Executive Order 13800 requires agency heads to report on--and be held accountable for--managing the cybersecurity risk to their networks. DDoS attacks must be considered part of that risk. Given the availability of effective tools against DDoS attacks, every agency head should have a strategy to mitigate the risks of an attack and should be held accountable for any failure to protect against them.

The government must also ensure that its systems are not contributing to the problem. As mentioned above, Cloudflare monitors cyberattacks and often detects DNS reflection attacks, which use IP spoofing. One of the most egregious example of this was the web site run by the Consumer Product Safety Commission (CPSC). As explained in a Cloudflare blog last year (Appendix G), at one point, almost every major DDoS attack using DNS amplification that Cloudflare detected was taking advantage of the CPSC server. Clearly, the CPSC is just one agency; every agency needs to take steps to ensure that they are not inadvertently “aiding and abetting” botnet users. Federal agencies could work with the companies that have the data needed to detect and fix servers that are improperly configured.

Promoting Transparency
One of the most important and most effective measures for motivating businesses and other organizations to invest more effort and money in securing their digital assets has been requirements that companies affected by data breaches report them. Similarly, Securities and Exchange Commission regulations that require public companies to report large cyberattacks, have highlighted the growth of such incidents and exposed companies that had not devoted enough attention to cyber threats.

Currently, companies are not required to report large botnet attacks and the damage they can cause. Public disclosure of such attacks would do a great deal to demonstrate the growing threat they pose, the types of attacks, and how they are changing.

Educating Users
The fact that many organizations are not using the many tools that are already available to prevent malware from infecting their machines or to block DDoS traffic from reaching them means that there is a clear need to do more to make computer users and website owners aware of how they can protect themselves. Thus, recently-introduced legislation or other mechanisms to help small business cope with cybersecurity challenges could be helpful. In addition, new efforts by cybersecurity trade associations and other business groups are helping highlight the threat posed by botnets, particularly since last year’s Mirai IoT botnet attacks. In addition, it would be helpful if Internet Services Providers redoubled their efforts to inform their customers on what they can do to prevent or mitigate DDoS attacks.

Law enforcement
As mentioned above, shutting down the command and control servers that manage botnets needs to be part of any strategy to reduce the threat of DDoS attacks. This is where law enforcement--coordinating with industry initiatives like Microsoft’s Digital Crimes Unit-- has a clear role.

Advising policy makers
The Department of State, the Department of Commerce, the Department of Homeland Security, and other agencies could help foreign governments raise awareness about

DDoS attacks and best practices for mitigating them. Similar efforts at the state and local level could would also be useful.

Conclusion

Cloudflare commends the NTIA (and NIST) for engaging the full range of stakeholders in their efforts to address the threat posed by botnets. We hope that the recent executive order on cybersecurity will build even more momentum behind efforts to convince cybercriminals that “DDoS doesn’t pay,” and look forward to contributing to future workshops and discussions on this and related topics.

APPENDICES

Also available online at:

Appendix A
https://blog.cloudflare.com/reflections-on-reflections/https://blog.cloudflare.com/reflections-on-reflections/

Appendix B
https://blog.cloudflare.com/say-cheese-a-snapshot-of-the-massive-ddos-attacks-coming-from-iot-cameras/

Appendix C
https://blog.cloudflare.com/how-the-dyn-outage-affected-cloudflare/

Appendix D
https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/

Appendix E
https://blog.cloudflare.com/ssdp-100gbps/

Appendix F
https://blog.cloudflare.com/a-deep-dive-into-dns-packet-sizes-why-smaller-packet-sizes-keep-the-internet-safe/

Appendix G
https://blog.cloudflare.com/how-the-consumer-product-safety-commission-is-inadvertently-behind-the-internets-largest-ddos-attacks/

APPENDIX A

Reflections on reflection (attacks)

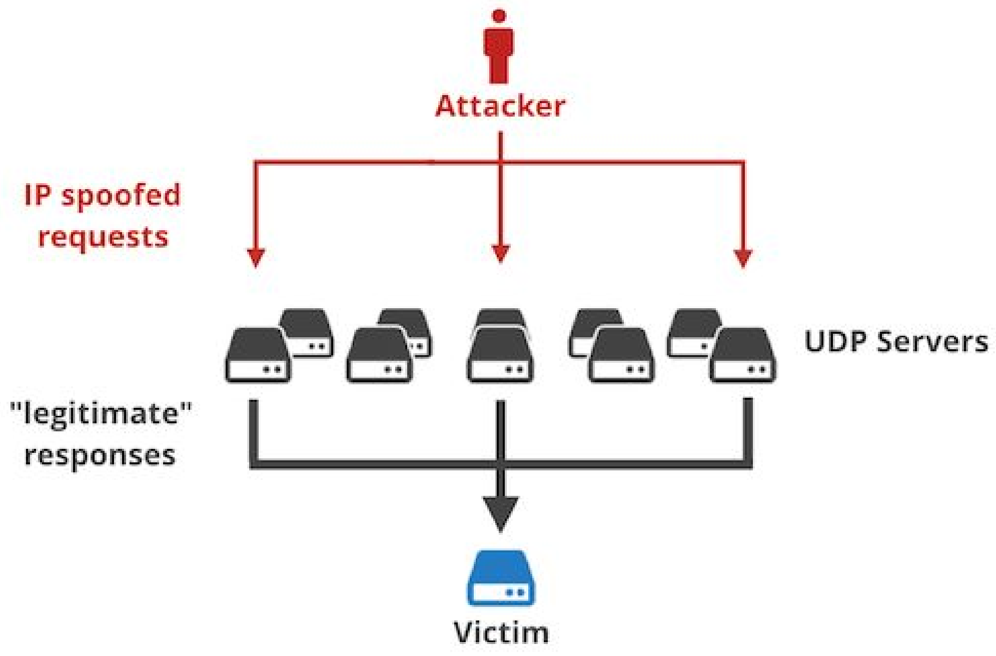
24 May 2017 by Marek Majkowski.

Recently Akamai published an article about CLDAP reflection attacks. This got us thinking. We saw attacks from Connectionless LDAP servers back in November 2016 but totally ignored them because our systems were automatically dropping the attack traffic without any impact.

CC BY 2.0 image by RageZ

We decided to take a second look through our logs and share some statistics about reflection attacks we see regularly. In this blog post, I'll describe popular reflection attacks, explain how to defend against them and why Cloudflare and our customers are immune to most of them.

A recipe for reflection



Let's start with a brief reminder on how reflection attacks (often called "amplification attacks") work.

To bake a reflection attack, the villain needs four ingredients:

- A server capable of performing IP address spoofing.
- A protocol vulnerable to reflection/amplification. Any badly designed UDP-based request-response protocol will do.
- A list of "reflectors": servers that support the vulnerable protocol.
- A victim IP address.

The general idea:

- The villain sends fake UDP requests.
- The source IP address in these packets is spoofed: the attacker sticks the victim's IP address in the source IP address field, not their own IP address as they normally would.
- Each packet is destined to a random reflector server.
- The spoofed packets traverse the Internet and eventually are delivered to the reflector server.
- The reflector server receives the fake packet. It looks at it carefully and thinks: "Oh, what a nice request from *the victim*! I must be polite and respond!". It sends the response in good faith.
- The response, though, is directed to the victim.

The victim will end up receiving a large volume of response packets it never had requested. With a large enough attack the victim may end up with congested network and an interrupt storm.

The responses delivered to victim might be larger than the spoofed requests (hence *amplification*). A carefully mounted attack may amplify the villain's traffic. In the past we've documented a 300Gbps attack generated with an estimated 27Gbps of spoofing capacity.

Popular reflections

During the last six months our DDoS mitigation system "Gatebot" detected 6,329 simple reflection attacks (that's one every 40 minutes). Here is the list by popularity of different attack vectors. An attack is defined as a large flood of packets identified by a tuple: (Protocol, Source Port, Target IP). Basically - a flood of packets with the same source port to a single target. This notation is pretty accurate - during normal Cloudflare operation, incoming packets rarely share a source port number!

Count	Proto	Src port	
3774	udp	123	NTP
1692	udp	1900	SSDP
438	udp	0	IP fragmentation
253	udp	53	DNS
42	udp	27015	SRCDS
20	udp	19	Chargen
19	udp	20800	Call Of Duty
16	udp	161	SNMP
12	udp	389	CLDAP
11	udp	111	Sunrpc
10	udp	137	Netbios
6	tcp	80	HTTP
5	udp	27005	SRCDS
2	udp	520	RIP

Source port 123/udp NTP

By far the most popular reflection attack vector remains NTP. We have blogged about NTP in the past:

- Understanding and mitigating NTP-based DDoS attacks
- Technical Details Behind a 400Gbps NTP Amplification DDoS Attack
- Good News: Vulnerable NTP Servers Closing Down

Over the last six months we've seen 3,374 unique NTP amplification attacks. Most of them were short. The average attack duration was 11 minutes, with the longest lasting 22 hours (1,300 minutes). Here's a histogram showing the distribution of NTP attack duration:

```
Minutes min:1.00 avg:10.51 max:1297.00 dev:35.02 count:3774
Minutes:
 value |-------------------------------------------------- count
     0 |                                                   2
     1 |                                                 * 53
     2 |                         ************************* 942
     4 |************************************************** 1848
     8 |                                   *************** 580
    16 |                                             ***** 221
    32 |                                                 * 72
    64 |                                                   35
   128 |                                                   11
   256 |                                                   7
   512 |                                                   2
  1024 |                                                   1
```

Most of the attacks used a small number of reflectors - we've recorded an average of 1.5k unique IPs per attack. The largest attack used an estimated 12.3k reflector servers.

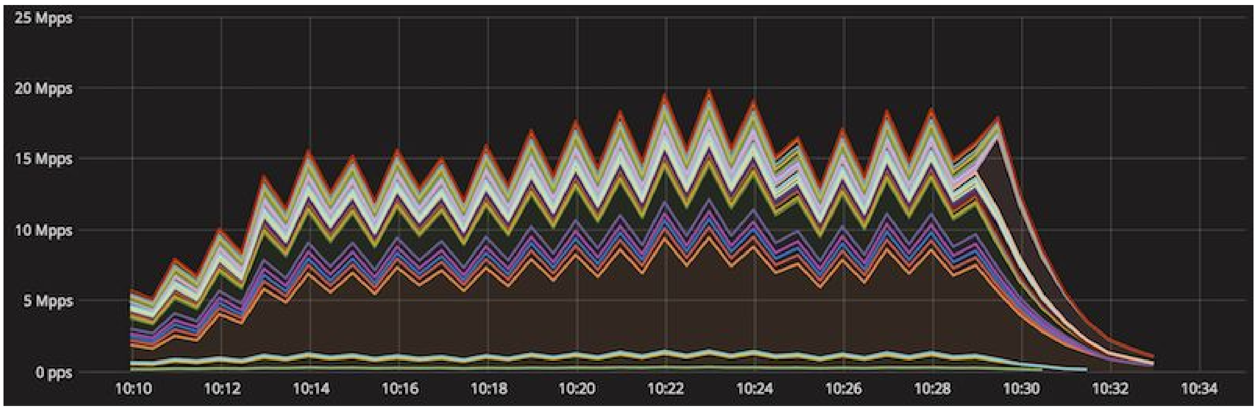
```
Unique IPs min:5.00 avg:1552.84 max:12338.00 dev:1416.03 count:3774
Unique IPs:
 value |-------------------------------------------------- count
    16 |                                                   0
    32 |                                                   1
    64 |                                                   8
   128 |                                             ***** 111
   256 |                         ************************* 553
   512 | ************************************************* 1084
  1024 |************************************************** 1093
  2048 |                   ******************************* 685
  4096 |                                        ********** 220
  8192 |                                                   13
```

The peak attack bandwidth was on average 5.76Gbps and max of 64Gbps:

```
Peak bandwidth in Gbps min:0.06 avg:5.76 max:64.41 dev:6.39 count:3774
Peak bandwidth in Gbps:
 value |-------------------------------------------------- count
     0 |                                            ****** 187
```

```
   1 |                        ******************** 603
   2 |************************************************* 1388
   4 |                 ***************************** 818
   8 |                          ***************** 526
  16 |                                    ******* 212
  32 |                                        * 39
  64 |                                          1
```

This stacked chart shows the geographical distribution of the largest NTP attack we've seen in the last six months. You can see the packets per second number directed to each datacenter. One our datacenters (San Jose to be precise) received about a third of the total attack volume, while the remaining packets were distributed roughly evenly across other datacenters.



The attack lasted 20 minutes, used 527 reflector NTP servers and generated about 20Mpps / 64Gbps at peak.

Dividing these numbers we can estimate that a single packet in that attack had on average size of 400 bytes. In fact, in NTP attacks the great majority of packets have a length of precisely 468 bytes (less often 516). Here's a snippet from tcpdump:

```
$ tcpdump -n -r 3164b6fac836774c.pcap -v -c 5 -K
11:38:06.075262 IP -(tos 0x20, ttl 60, id 0, offset 0, proto UDP (17), length 468)
   216.152.174.70.123 > x.x.x.x.47787:  [|ntp]
11:38:06.077141 IP -(tos 0x0, ttl 56, id 0, offset 0, proto UDP (17), length 468)
   190.151.163.1.123 > x.x.x.x.44540:  [|ntp]
11:38:06.082631 IP -(tos 0xc0, ttl 60, id 0, offset 0, proto UDP (17), length 468)
   69.57.241.60.123 > x.x.x.x.47787:  [|ntp]
11:38:06.095971 IP -(tos 0x0, ttl 60, id 0, offset 0, proto UDP (17), length 468)
   126.219.94.77.123 > x.x.x.x.21784:  [|ntp]
11:38:06.113935 IP -(tos 0x0, ttl 59, id 0, offset 0, proto UDP (17), length 516)
   69.57.241.60.123 > x.x.x.x.9285:  [|ntp]
```

Source port 1900/udp SSDP

The second most popular reflection attack was SSDP, with a count of 1,692 unique events. These attacks were using much larger fleets of reflector servers. On average we've seen around 100k reflectors used in each attack, with the largest attack using 1.23M reflector IPs. Here's the histogram of number of unique IPs used in SSDP attacks:

```
Unique IPs min:15.00 avg:98272.02 max:1234617.00 dev:162699.90 count:1691
Unique IPs:
  value |-------------------------------------------------- count
    256 |                          0
    512 |                          4
   1024 |                 **************** 98
   2048 |             ************************ 152
   4096 |          ***************************** 178
   8192 |            ************************* 158
  16384 |           **************************** 176
  32768 |     *************************************** 243
  65536 |************************************************** 306
 131072 |       ************************************ 225
 262144 |                 *************** 95
 524288 |                     ******* 47
1048576 |                        * 7
```

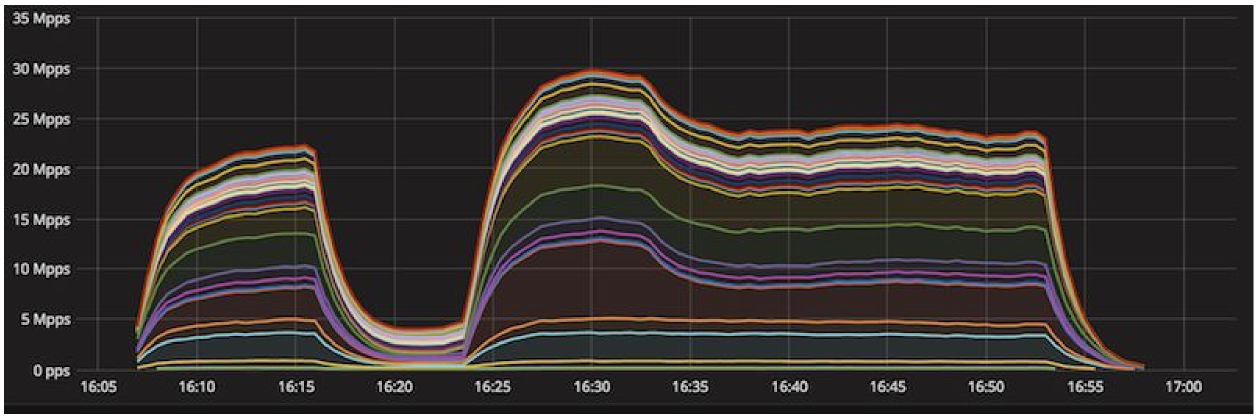
The attacks were also longer, with 24 minutes average duration:

```
$ cat 1900-minutes| ~/bin/mmhistogram -t "Minutes"
Minutes min:2.00 avg:23.69 max:1139.00 dev:57.65 count:1692
Minutes:
 value |-------------------------------------------------- count
     0 |                          0
     1 |                          10
     2 |                ***************** 188
     4 |         ******************************** 354
     8 |************************************************** 544
    16 |         ******************************* 342
    32 |                 *************** 168
    64 |                       **** 48
   128 |                        * 19
   256 |                        * 16
   512 |                          1
  1024 |                          2
```

Interestingly the bandwidth doesn't follow a normal distribution. The average SSDP attack was 12Gbps and the largest just shy of 80Gbps:

```
$ cat 1900-Gbps| ~/bin/mmhistogram -t "Bandwidth in Gbps"
Bandwidth in Gbps min:0.41 avg:11.95 max:78.03 dev:13.32 count:1692
Bandwidth in Gbps:
 value |-------------------------------------------------- count
     0 |          ******************************* 331
     1 |              ********************* 232
     2 |              ********************** 235
     4 |                 *************** 165
     8 |                      ****** 65
    16 |************************************************** 533
    32 |                   *********** 118
    64 |                        * 13
```

Let's take a closer look at the largest (80Gbps) attack we've recorded. Here's a stacked chart showing packets per second going to each datacenter. This attack was using 940k reflector IPs, generated 30Mpps. The datacenters receiving the largest proportion of the traffic were San Jose, Los Angeles and Moscow.



The average packet size was 300 bytes. Here's how the attack looked on the wire:

```
$ tcpdump -n -r 4ca985a2211f8c88.pcap -K -c 7
10:24:34.030339 IP - 219.121.108.27.1900 > x.x.x.x.25255: UDP, length 301
10:24:34.406943 IP - 208.102.119.37.1900 > x.x.x.x.37081: UDP, length 331
10:24:34.454707 IP - 82.190.96.126.1900 > x.x.x.x.25255: UDP, length 299
10:24:34.460455 IP - 77.49.122.27.1900 > x.x.x.x.25255: UDP, length 289
10:24:34.491559 IP - 212.171.247.139.1900 > x.x.x.x.25255: UDP, length 323
10:24:34.494385 IP - 111.1.86.109.1900 > x.x.x.x.37081: UDP, length 320
10:24:34.495474 IP - 112.2.47.110.1900 > x.x.x.x.37081: UDP, length 288
```

Source port 0/udp IP fragmentation

Sometimes we see reflection attacks showing UDP source and destination port numbers set to zero. This is usually a side effect of attacks where the reflecting servers responded with large fragmented packets. Only the first IP fragment contains a UDP header, preventing subsequent fragments from being reported properly. From a router point of view this looks like a UDP packet without UDP header. A confused router reports a packet from source port 0, going to port 0!

This is a tcpdump-like view:

```
$ tcpdump -n -r 4651d0ec9e6fdc8e.pcap -c 8
02:05:03.408800 IP - 190.88.35.82.0 > x.x.x.x.0: UDP, length 1167
02:05:03.522186 IP - 95.111.126.202.0 > x.x.x.x.0: UDP, length 1448
02:05:03.525476 IP - 78.90.250.3.0 > x.x.x.x.0: UDP, length 839
02:05:03.550516 IP - 203.247.133.133.0 > x.x.x.x.0: UDP, length 1472
02:05:03.571970 IP - 54.158.14.127.0 > x.x.x.x.0: UDP, length 1328
02:05:03.734834 IP - 1.21.56.71.0 > x.x.x.x.0: UDP, length 1250
02:05:03.745220 IP - 195.4.131.174.0 > x.x.x.x.0: UDP, length 1472
02:05:03.766862 IP - 157.7.137.101.0 > x.x.x.x.0: UDP, length 1122
```

An avid reader will notice - the source IPs above are open DNS resolvers! Indeed, from our experience most of the attacks categorized as fragmentation are actually a side effect of DNS amplifications.

Source port 53/udp DNS

Over the last six months we've seen 253 DNS amplifications. On average an attack used 7100 DNS reflector servers and lasted 24 minutes. Average bandwidth was around 3.4Gbps with largest attack using 12Gbps.

This is a simplification though. As mentioned above multiple DNS attacks were registered by our systems as two distinct vectors. One was categorized as source port 53, and another as source port 0. This happened when the DNS server flooded us with DNS responses larger than max packet size, usually about 1,460 bytes. It's easy to see if that was the case by inspecting the DNS attack packet lengths. Here's an example:

```
DNS attack packet lengths min:44.00 avg:1458.94 max:1500.00 dev:208.14 count:40000
DNS attack packet lengths:
 value |-------------------------------------------------- count
     8 |                                                   0
    16 |                                                   0
    32 |                                                   129
    64 |                                                   479
   128 |                                                   84
   256 |                                                   164
   512 |                                                   268
  1024 |************************************************** 38876
```

The great majority of the received DNS packets were indeed close to the max packet size. This suggests the DNS responses were large and were split into multiple fragmented packets. Let's see the packet size distribution for accompanying source port 0 attack:

```
$ tcpdump -n -r 4651d0ec9e6fdc8e.pcap \
    | grep length \
    | sed -s 's#.*length \([0-9]\+\).*#\1#g' \
    | ~/bin/mmhistogram -t "Port 0 packet length" -l -b 100
Port 0 packet length min:0.00 avg:1264.81 max:1472.00 dev:228.08 count:40000
Port 0 packet length:
 value |-------------------------------------------------- count
     0 |                                                   348
   100 |                                                   7
   200 |                                                   17
   300 |                                                   11
   400 |                                                   17
   500 |                                                   56
   600 |                                                   3
   700 |                                                ** 919
   800 |                                                 * 520
   900 |                                                 * 400
  1000 |                                          ******** 3083
  1100 |             ************************************* 12986
  1200 |                                             ***** 1791
 1300 |                                        ***** 2057
 1400 |************************************************ 17785
```

About half of the fragments were large, close to the max packet length in size, and rest were just shy of 1,200 bytes. This makes sense: a typical max DNS response is capped at 4,096 bytes. 4,096 bytes would be seen on the wire as one DNS packet fragment with an IP header, one max length packet fragment and one fragment of around 1,100 bytes: `4,096 = 1,460+1,460+1,060`

For the record, the particular attack illustrated here used about 17k reflector server IPs, lasted 64 minutes, generated about 6Gbps on the source port 53 strand and 11Gbps of source port 0 fragments.

We have blogged about DNS reflection attacks in the past:

- How to Launch a 65Gbps DDoS, and How to Stop One
- Deep Inside a DNS Amplification DDoS Attack
- How the Consumer Product Safety Commission is (Inadvertently) Behind the Internet's Largest DDoS Attacks

Other protocols

We've seen amplification using other protocols such as:

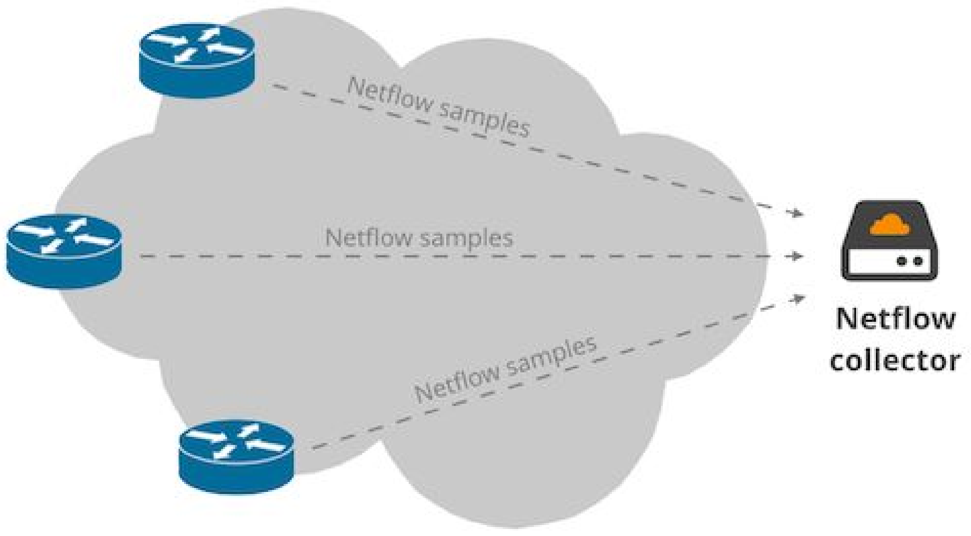
- port 19 - Chargen
- port 27015 - SRCDS
- port 20800 - Call Of Duty

...and many other obscure protocols. These attacks were usually small and not notable. We didn't see enough of then to provide meaningful statistics but the attacks were automatically mitigated.

Poor observability

Unfortunately we're not able to report on the contents of the attack traffic. This is notable for the NTP and DNS amplifications - without case by case investigations we can't report what responses were actually being delivered to us.

This is because all these attacks stopped at the network layer. Routers are heavily optimized to perform packet forwarding and have a limited capacity of extracting raw packets. Basically there is no "tcpdump" there.



We track these attacks with netflow, and we observe them hit our routers firewall. The tcpdump snippets shown above were actually fake, reconstructed artificially from netflow data.

Trivial to mitigate

With properly configured firewall and sufficient network capacity (which isn't always easy to come by unless you are the size of Cloudflare) it's trivial to block the reflection attacks. But note that we've seen reflection attacks up to 80Gbps so you do need sufficient capacity.

Properly configuring a firewall is not rocket science: default DROP can get you quite far. In other cases you might want to configure rate limiting rules. This is a snippet from our JunOS config:

```
term RATELIMIT-SSDP-UPNP {
    from {
        destination-prefix-list {
            ANYCAST;
        }
        next-header udp;
        source-port 1900;
    }
    then {
        policer SA-POLICER;
        count ACCEPT-SSDP-UPNP;
        next term;
    }
}
```

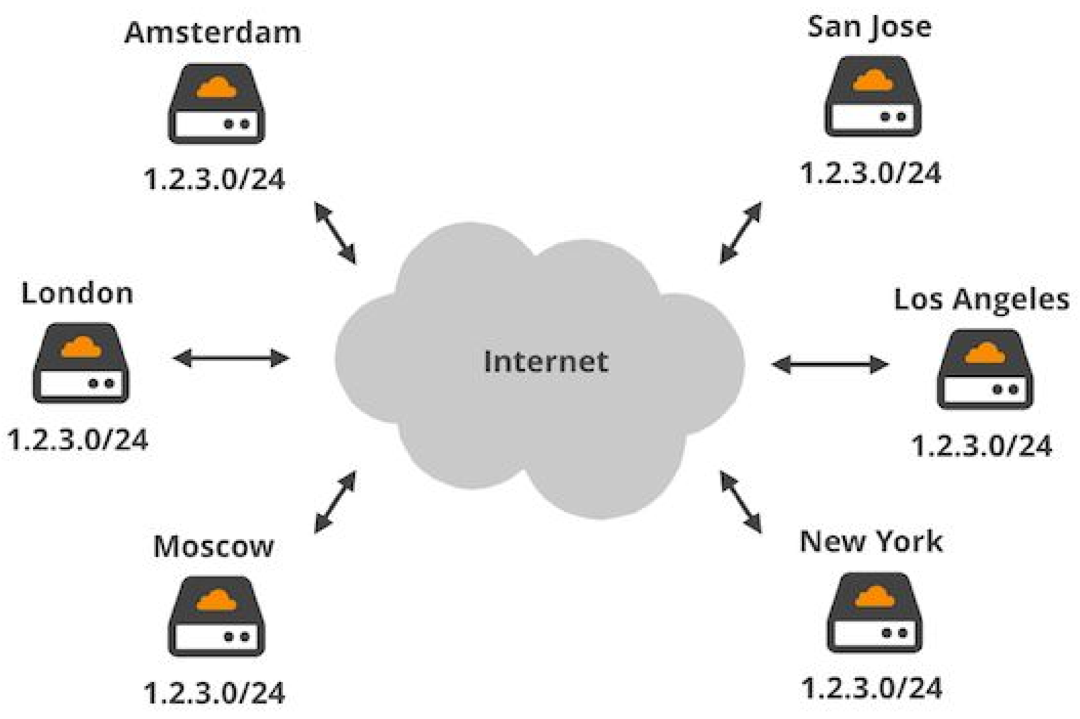
But properly configuring firewall requires some Internet hygiene. You should avoid using the same IP for inbound and outbound traffic. For example, filtering a potential NTP DDoS will be harder if you can't just block inbound port 123 indiscriminately. If your server requires NTP, make sure it exits to the Internet over non-server IP address!

Capacity game

While having sufficient network capacity is necessary, you don't need to be a Tier 1 to survive amplification DDoS. The median attack size we've received was just 3.35Gbps, average 7Gbps, Only 195 attacks out of 6,353 attacks recorded - 3% - were larger than 30Gbps.

```
All attacks in Gbps: min:0.04 avg:7.07 med:3.35 max:78.03 dev:9.06 count:6329
All attacks in Gbps:
 value |-------------------------------------------------- count
     0 |                                  **************** 658
     1 |                         ************************* 1012
     2 |************************************************** 1947
     4 |                    ****************************** 1176
     8 |                                  **************** 641
    16 |                               ******************* 748
    32 |                                              **** 157
    64 |                                                   14
```

But not all Cloudflare datacenters have equal sized network connections to the Internet. So how can we manage?



Cloudflare was architected to withstand large attacks. We are able to spread the traffic on two layers:

- Our public network uses Anycast. For certain attack types - like amplification - this allows us to split the attack across multiple datacenters avoiding a single choke point.
- Additionally we use ECMP internally to spread a traffic destined to single IP address across multiple physical servers.

In the examples above, I showed a couple of amplification attacks getting nicely distributed across dozens of datacenters across the globe. In the shown attacks, if our router firewall failed, our physical servers wouldn't receive more than 500kpps of attack data. A well tuned `iptables` firewall should be able to cope with such a volume without a special kernel offload help.

Inter-AS Flowspec for the rest

Withstanding reflection attacks requires sufficient network capacity. Internet citizens not having fat network cables should use a good Internet Service Provider supporting flowspec.

Flowspec can be thought of as a protocol enabling firewall rules to be transmitted over a BGP session. In theory flowspec allows BGP routers on different Autonomous Systems to share firewall rules. The rule can be set up on the attacked router and distributed to the ISP network with the BGP magic. This will stop the packets closer to the source and effectively relieve network congestion.

Unfortunately, due to performance and security concerns only a handful of large ISP's allow inter-AS flowspec rules. Still - it's worth a try. Check if your ISP is willing to accept flowspec from your BGP router!

At Cloudflare we maintain an intra-AS flowspec infrastructure, and we have plenty of war stories about it.

Summary

In this blog post we've given details of three popular reflection attack vectors: NTP, SSDP and DNS. We discussed how the Cloudflare Anycast network helps us avoid a single choke point. In most cases dealing with reflection attacks is not rocket science though sufficient network capacity is needed and simple firewall rules are usually enough to cope.

The types of DDoS attacks we see from other vectors (such as IoT botnets) are another matter. They tend to be much larger and require specialized, automatic DDoS mitigation. And, of course, there are many DDoS attacks that occur using techniques other than reflection and not just using UDP.

APPENDIX B

Say Cheese: a snapshot of the massive DDoS attacks coming from IoT cameras

11 Oct 2016 by Marek Majkowski.

Over the last few weeks we've seen DDoS attacks hitting our systems that show that attackers have switched to new, large methods of bringing down web applications. They appear to come from an IoT botnet (like Mirai and relations) which were responsible for the large attacks against Brian Krebs.

Our automatic DDoS mitigation systems have been handling these attacks, but we thought it would be interesting to publish some of the details of what we are seeing. In this article we'll share data on two attacks, which are perfect examples of the new trends in DDoS.

CC BY 2.0 image by E Magnuson

In the past we've written extensively about volumetric DDoS attacks and how to mitigate them. The attacks are distinguished by their heavy use of L7 (i.e. HTTP) attacks as opposed to the more familiar SYN floods, ACK floods, and NTP and DNS reflection attacks.

Many DDoS mitigation systems are tuned to handle volumetric L3/4 attacks; in this instance attackers have switched to L7 attacks in an attempt to knock web applications offline.

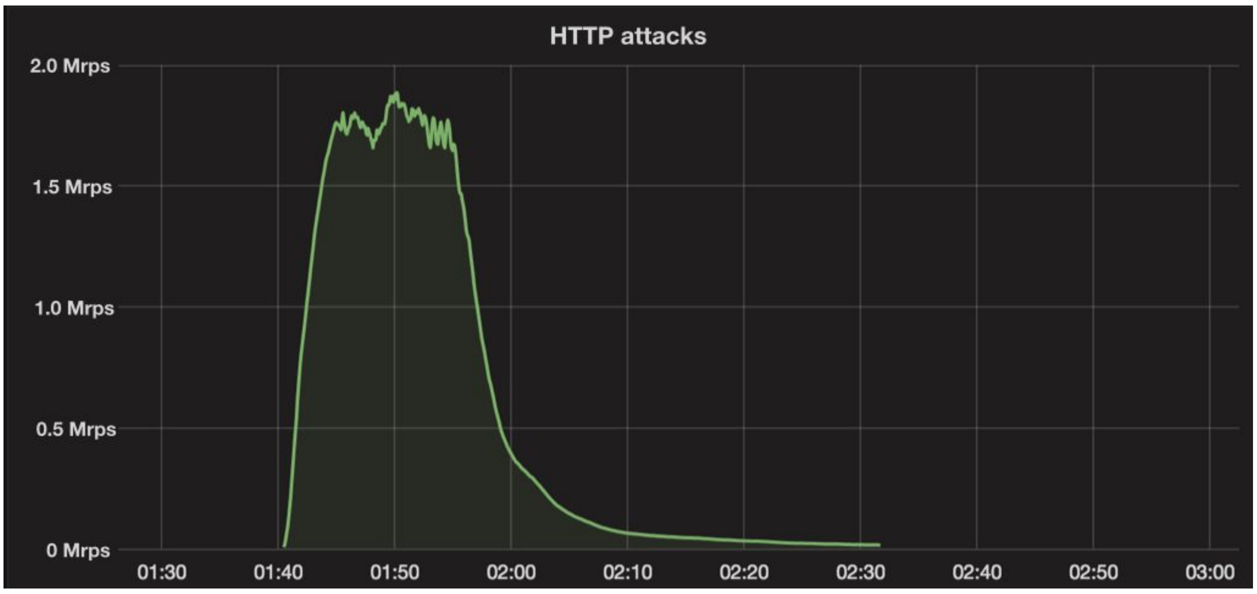
Seeing the move towards L7 DDoS attacks we put in place a new system that recognizes and blocks these attacks as they happen. The L7 mitigator recognizes attacks against a single host and distributes a fingerprint that protects all 4 million Cloudflare customers. We'll write more about it in the future.

HTTP Requests per second

Often when DDoS attacks are reported the size of the attack is reported in Gbps (or even Tbps), but there are many ways to measure the size of an attack.

For L7 HTTP-based attacks it also makes sense to measure requests per second. That's because, unlike volumetric L3/4 attacks, HTTP-based attacks eat up resources by making actual HTTP requests to the attacked server.

Recently we were hit by a couple of unusually large L7 attacks, crossing 1 million HTTP requests per second (1 Mrps). Here is one of them:

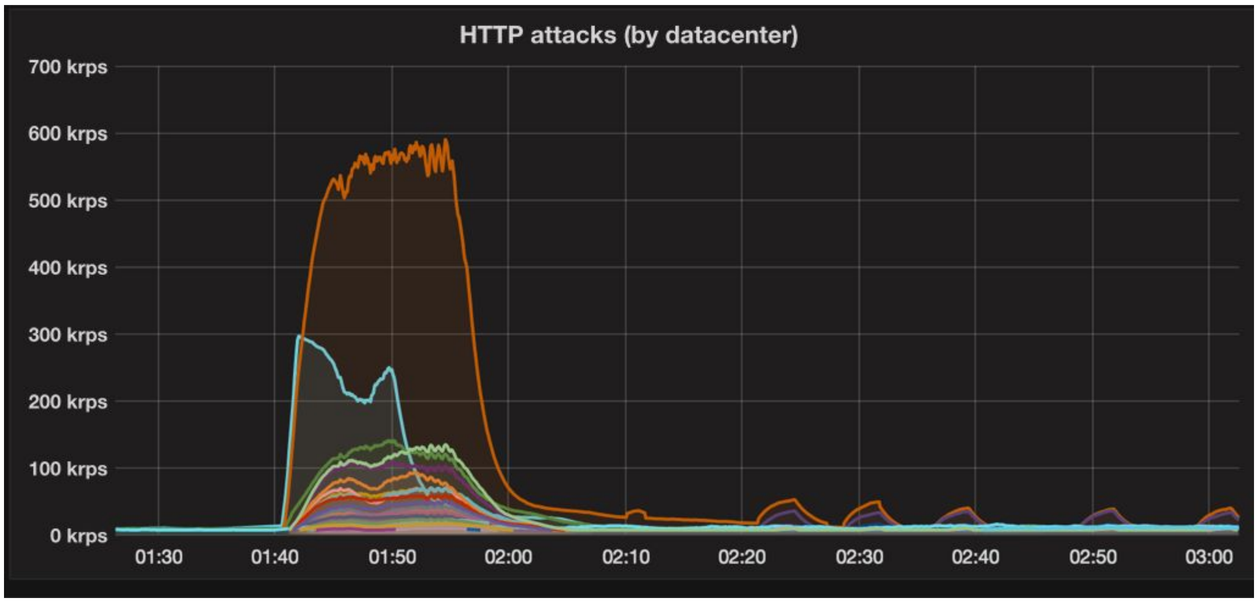


This attack continued for 15 minutes. Multiple recent attacks had >1 Mrps and lasted for minutes.

This particular attack peaked at 1.75 Mrps. It was composed of short HTTP requests (around 121 bytes per request), without anything unusual in the HTTP headers. The requests had a fixed `Cookie` header. We counted 52,467 unique IP addresses taking part in this attack.

Due to the Anycast nature of the Cloudflare network, the malicious traffic was spread across multiple Cloudflare cities and with 100 cities we are able to get a good picture of where the bots are located.

Here are the top affected datacenters:



This attack went largely to our Hong Kong and Prague datacenters. This is another common characteristic; most of the recent attacks looked similar.

Since the attack looks concentrated, we wondered if only a small number of AS numbers (networks) were the source of the attack. Unfortunately no, the IP addresses participating in the flood are evenly distributed. Out of 10,000 random requests we analyzed, we saw source IP addresses from over 300 AS numbers. These are the biggest sources:

```
  48 AS24086 ; Vietnam
 101 AS4134  ; China
 128 AS7552  ; Vietnam
 329 AS45899 ; Vietnam
2366 AS15895 ; Ukraine
```

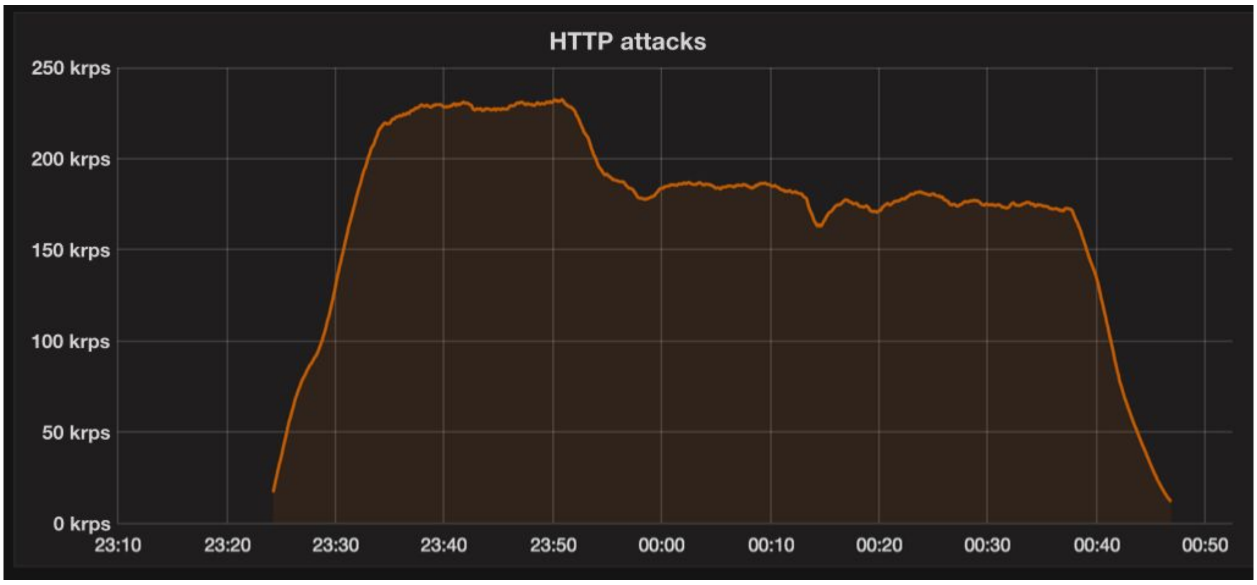
These attacks are a new trend, so it's not fair to blame the AS operators for not cleaning up devices participating in them. Having said that, the Ukrainian ISP and Vietnamese AS45899 seem to stand out. We'll get back to those in a moment.

Bandwidth

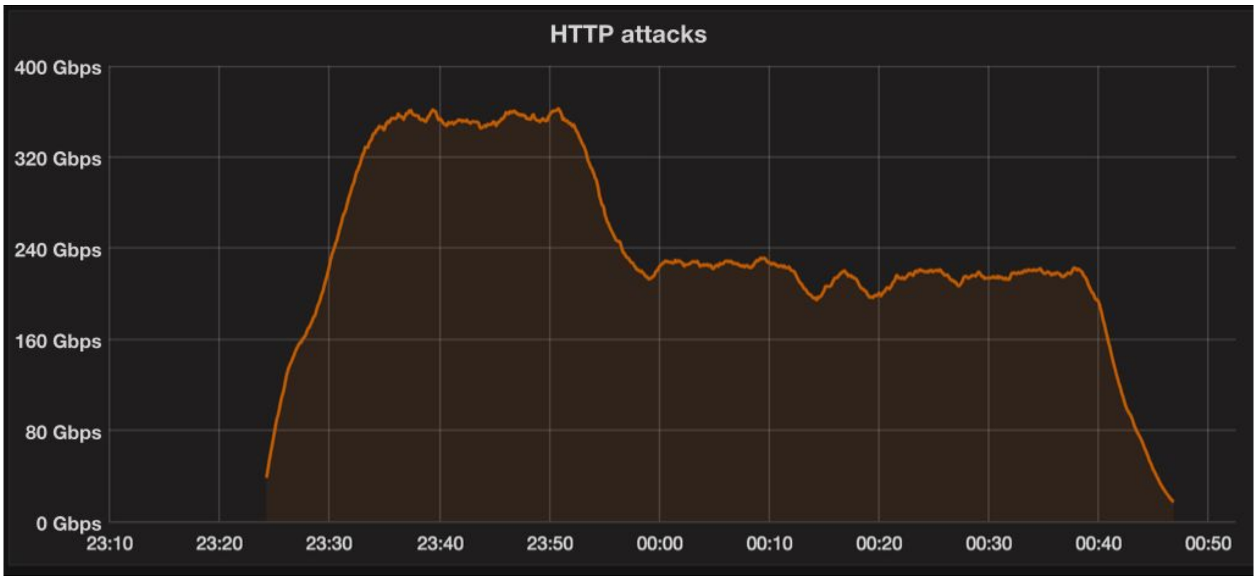
Although requests per second is a common metric for measuring these attacks, it's not the only one. We can also measure the bandwidth used in the attack.

By this count the attack mentioned above was pretty small (since we've got used to DDoS attacks being reported in 100s of Gbps). It peaked at roughly 2Gbps. But recall that these L7 attacks end up hitting a web server and are not simply volumetric: they use server resources.

However, we saw another attack that was unusual in that it was an L7 with similar bandwidth consumption to traditional L3/4 volumetric attacks. First, here's the requests per second graph:



This attack generated "only" 220k requests per second at peak. However, it generated significant inbound bandwidth:



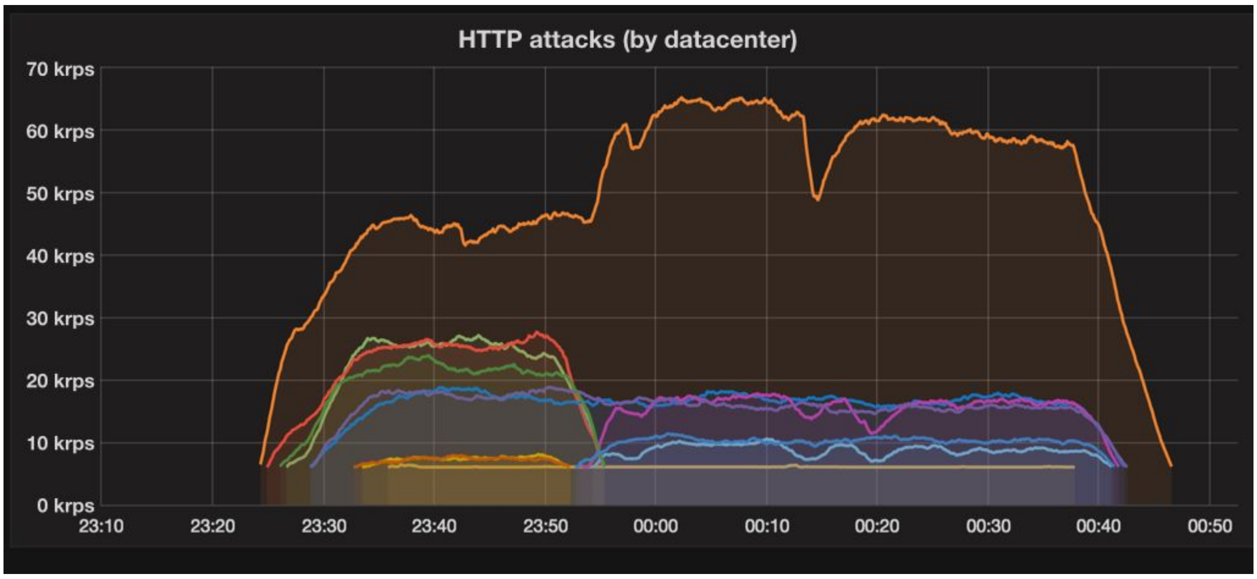
This attack topped out at 360Gbps per second of inbound HTTP traffic. It's pretty unusual for an HTTP attack to generate a substantial amount of network traffic. This attack was special, and was composed of HTTP requests like this:

```
GET /en HTTP/1.1
User-Agent: <some string>
Cookie: <some cookie>
Host: example.com
Connection: close
Content-Length: 800000

a[]=&b[]=&a[]=&b[]=&a[]=&b[]=&a[]=&b[]=&a[]=&b[]=&a[]=&b[]=...
```

It's the long payload sent after the request headers that allowed the attackers to generate substantial traffic. Since this attack we've seen similar events with varying parameters in the request body. Sometimes these attacks came as GET requests, sometimes as POST.

Additionally, this particular attack lasted roughly one hour, with 128,833 unique IP addresses. The datacenter distribution was different, with most of the attack concentrated on Frankfurt:



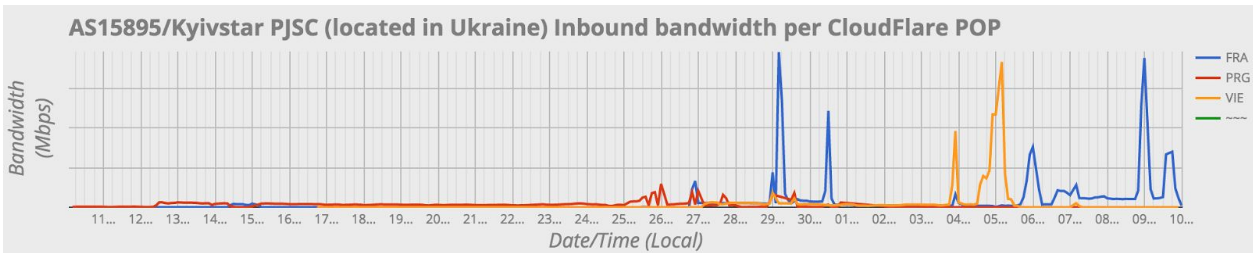
As the attack was composed of a very large number of bots, we expected the AS distribution to be fairly even. Indeed, in the 10,000 request sample we recorded a whopping 737 unique AS numbers. Here are the top sources:

```
 286 AS45899 ; Vietnam
  314 AS7552  ; Vietnam
  316 AS3462  ; Taiwan
  323 AS18403 ; Vietnam
 1510 AS15895 ; Ukraine
```

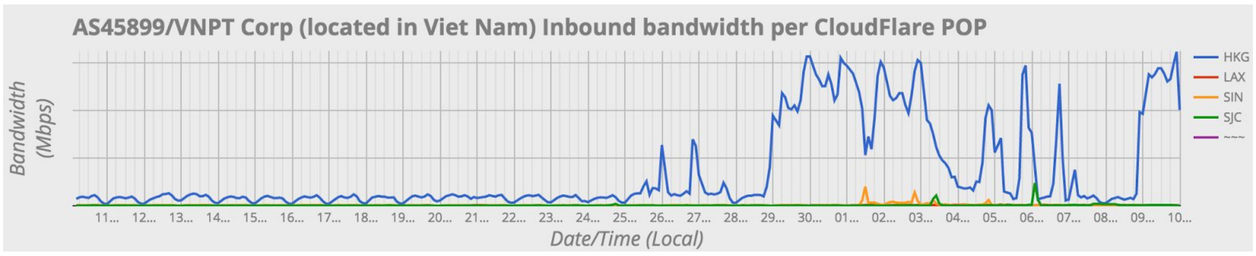
Once again, the Ukrainian ISP and couple of Vietnamese networks are the top hitters.

More on the sources

We wondered why AS15895 was so special. First, we investigated our traffic charts. Here is the inbound traffic we received from them over last 30 days:



The first significant attack was clearly seen as a spike on September 29 and reached 30Gbps. A very similar chart is visible for AS45899:



We can see some smaller attacks attempted around September 26. A couple of days later the attackers turned the throttle up hitting 7.5Gbps non-stop from this ASN. Other AS numbers we investigated reveal a similar story.

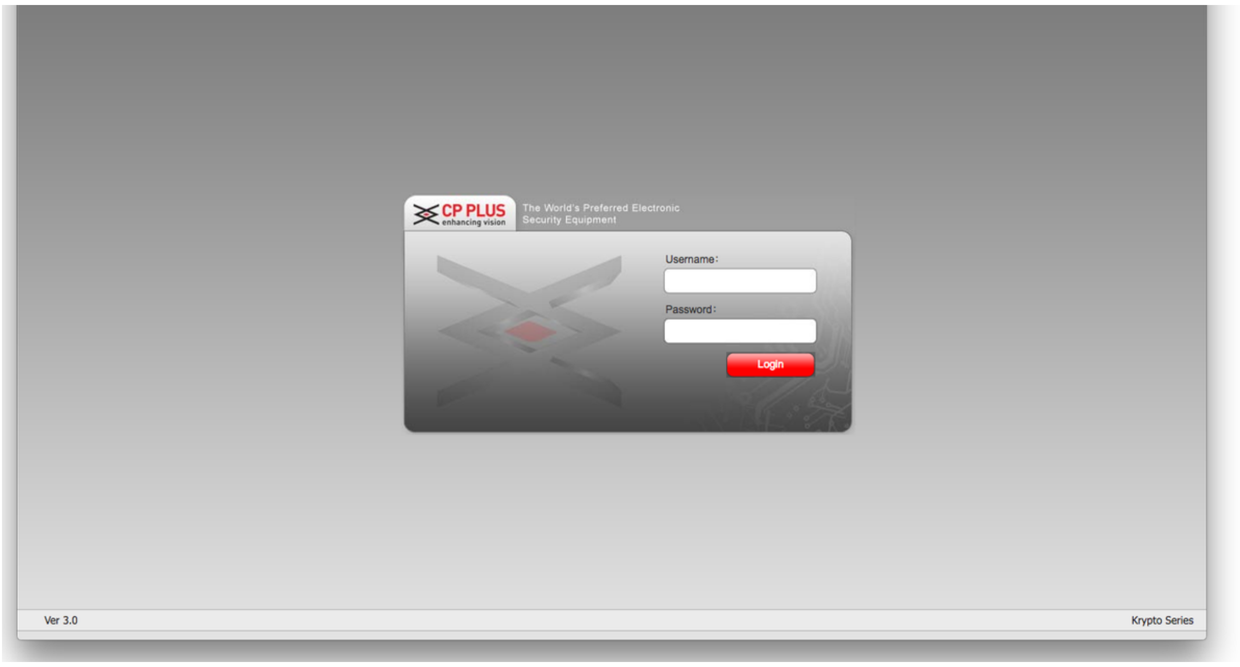
Devices

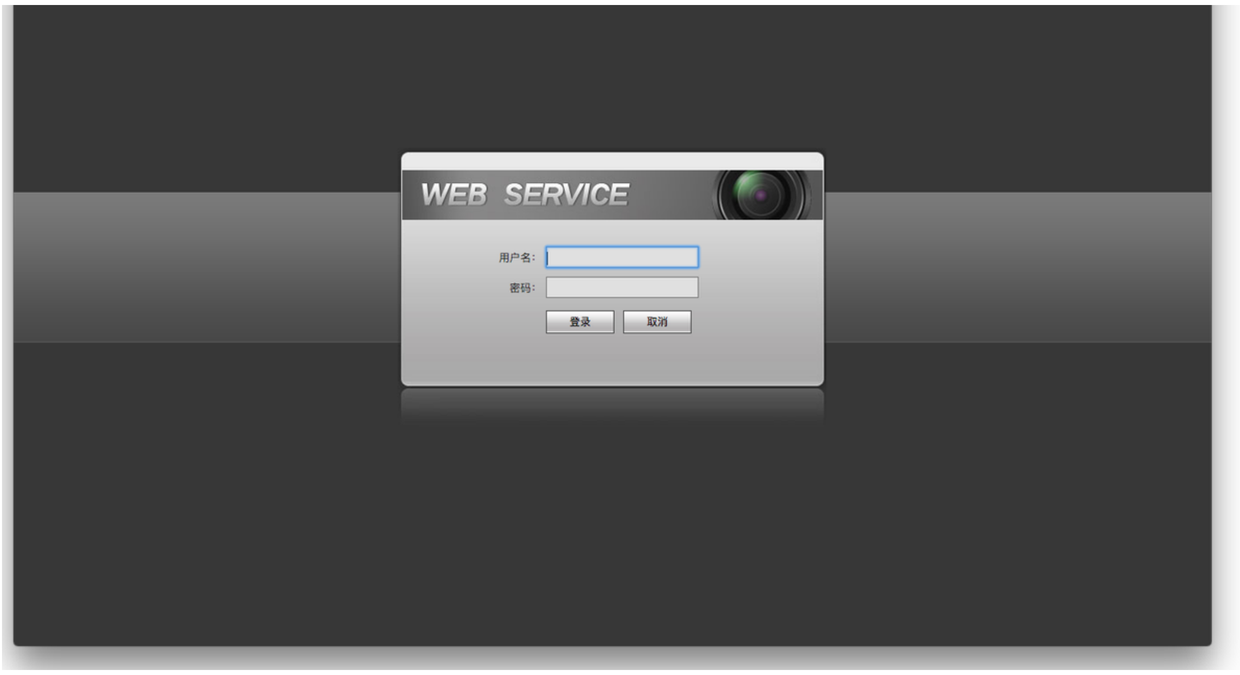
While it's not possible for us to investigate all the attacking devices, it is fair to say that these attacks came from Internet-of-Things (IoT) category of devices.

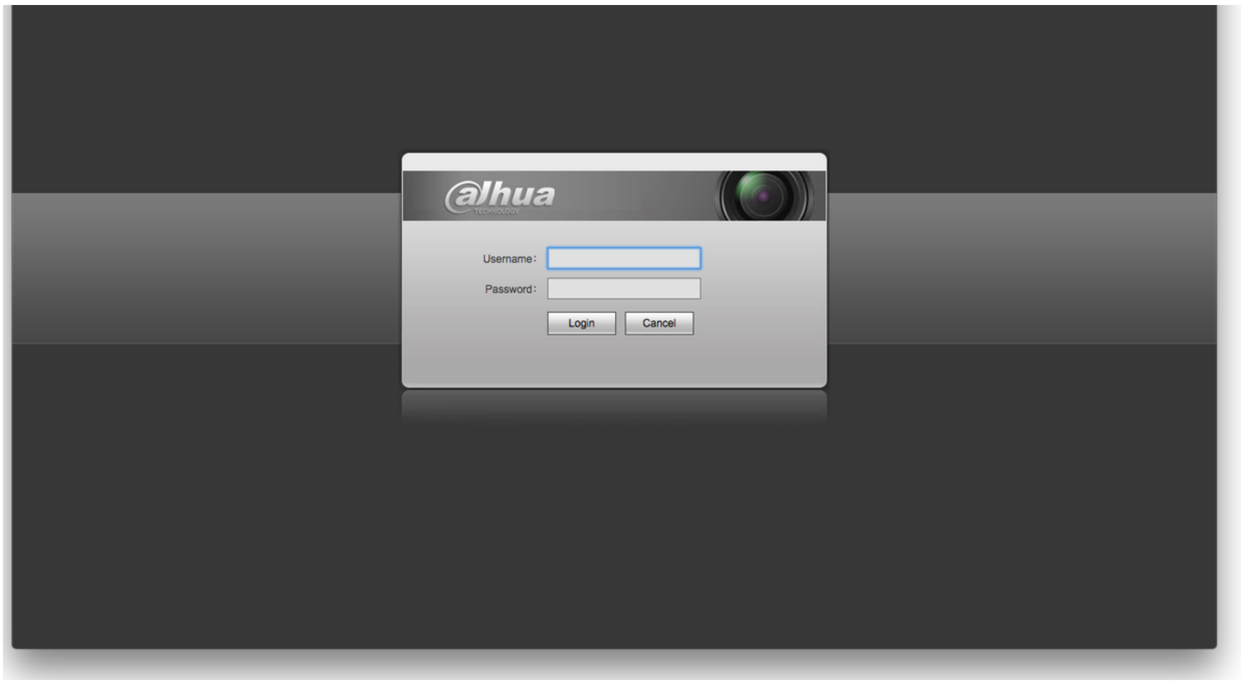
There are multiple hints confirming this theory.

First, all of the attacking devices have port 23 (telnet) open (closing connection immediately) or closed. Never filtered. This is a strong hint that the malware disabled the telnet port just after it installed itself.

Most of the hosts from the Vietnamese networks look like connected CCTV cameras. Multiple have open port 80 with presenting "NETSurveillance WEB" page.


CP PLUS
enhancing vision
The World's Preferred Electronic Security Equipment
Username:
Password:
Login
Ver 3.0
Krypto Series


WEB SERVICE
用户名:
密码:
登录
取消



The Ukrainian devices are a bit different though. Most have port 80 closed, making it harder to identify.

We had noticed one device with port 443 open serving a valid TLS cert issued by Western Digital, handling domain `device-xxxx.wd2go.com` suggesting it was a hard drive (Network Attached Storage to be precise).

未来: the future of DDoS

We plan to continue our investigation and collaborate with external researchers to find a permanent solution to this rising threat.

Although the most recent attacks have mostly involved Internet-connected cameras, there's no reason to think that they are likely the only source of future DDoS attacks. As more and more devices (fridges, fitness trackers, sleep monitors, ...) are added to the Internet they'll likely be unwilling participants in future attacks.

CC BY 2.0 image by CODE_n

Update:

Originally this article attributed the Mirai botnet for the shown attacks. We now believe that, for technical reasons, the large-bandwidth attack might not have come from a botnet running the leaked Mirai code.

APPENDIX C

How the Dyn outage affected Cloudflare

27 Oct 2016 by John Graham-Cumming.

Last Friday the popular DNS service Dyn suffered three waves of DDoS attacks that affected users first on the East Coast of the US, and later users worldwide. Popular websites, some of which are also Cloudflare customers, were inaccessible. Although Cloudflare was not attacked, joint Dyn/Cloudflare customers were affected.

Almost as soon as Dyn came under attack we noticed a sudden jump in DNS errors on our edge machines and alerted our SRE and support teams that Dyn was in trouble. Support was ready to help joint customers and we began looking in detail at the effect the Dyn outage was having on our systems.

An immediate concern internally was that since our DNS servers were unable to reach Dyn they would be consuming resources waiting on timeouts and retrying. The first question I asked the DNS team was: "*Are we seeing increased DNS response latency?*" rapidly followed by "*If this gets worse are we likely to?*". Happily, the response to both those questions (after the team analyzed the situation) was no.

However, that didn't mean we had nothing to do. Operating a large scale system like Cloudflare that deals with the continuously changing nature of the Internet means that there's always something to learn.

Back in July 2015 Dyn had an outage that also affected some of our customers and we changed our handling of so-called infrastructure DNS records in response to prevent a similar problem, from any provider, affecting Cloudflare.

Based on what we learned last Friday we are making some changes to our internal DNS infrastructure so that it performs better when a major provider is having problems or an outage (whether caused by DDoS or not). To understand those changes it's helpful to take a look at the role of DNS and what we saw on Friday.

A little bit about DNS

The Domain Name System (DNS) provides an address book service for the Internet. It is responsible for converting the friendly, human-readable domain names we type into our web browsers to IP addresses for websites. Let's walk through the life of an example web request to see where DNS plays a role.

We can start by entering a web address into our browser, https://www.cloudflare.com/. The browser translates this name into an IP address so it can contact the server that's hosting the page, it will do this using DNS. We can make these DNS queries ourselves using the dig command line tool to see what values are returned.

```
$ dig www.cloudflare.com A
...
;; QUESTION SECTION:
;www.cloudflare.com.       IN  A


;; ANSWER SECTION:
www.cloudflare.com.   10  IN  A   198.41.215.162
www.cloudflare.com.   10  IN  A   198.41.214.162
```

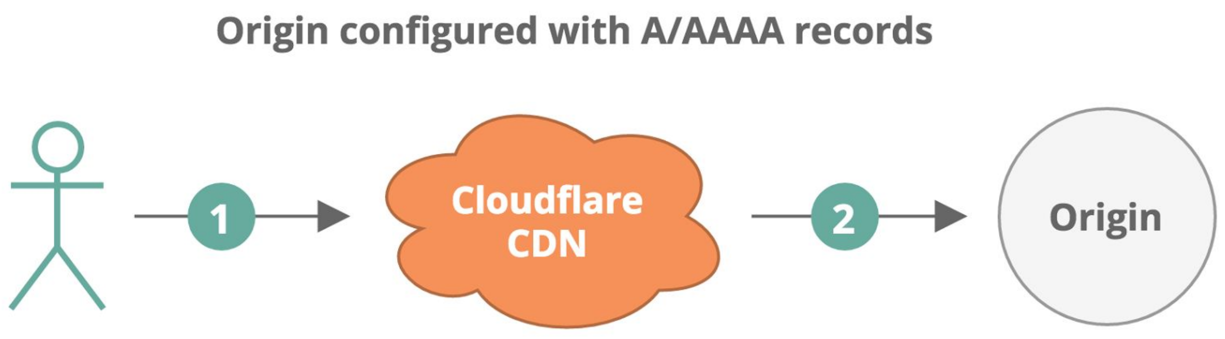
The DNS data model is split into two core concepts, names and records. The name here is www.cloudflare.com and the record type we have queried is A, which is used to store IPv4 addresses. There are other types of records for storing other types of data, e.g AAAA records for IPv6 addresses. We can see from the answer above that there are two IPv4 addresses for www.cloudflare.com; the browser picks one of these to use.

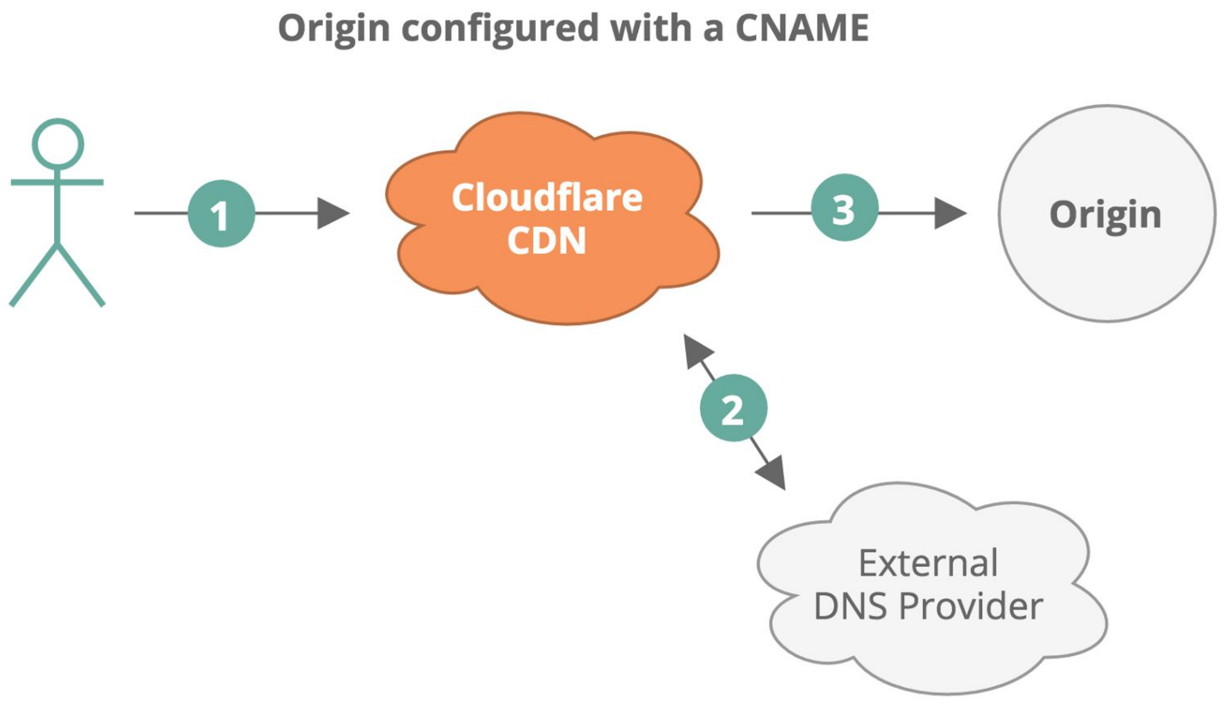
Records in the DNS also have an associated TTL which defines how long the data should be cached for, these records have a TTL of 10 seconds. This means the browser can store this information and skip making further DNS queries for the domain for the next 10 seconds.

For Cloudflare customers, the answer will contain our Anycast IPs instead of the origin ones (the IP addresses of the web hosting provider). The browser will then send requests to us, and we will serve content from our cache or proxy the request to the origin web server.

There are two common ways of configuring origins on Cloudflare. The first is to specify A and AAAA records, which explicitly provides us with the IP addresses of the origin. In this situation, our network knows ahead of time where it can contact the origin, so no further DNS resolution is required. For example, if `www.example.com` uses Cloudflare and has specified `2001:db8:5ca1:ab1e` as the IP address of the origin server in the Cloudflare control panel, we can connect directly to the origin server to retrieve resources.

The other is to use a CNAME, which is a pointer to another DNS name.





When our servers receive a request with the origin configured using a CNAME, we have to perform an external DNS lookup to resolve the target of the CNAME to IP addresses. This information is cached, based on the TTL defined on the CNAME record. In this case, our ability to serve content (that is not in the cache) entirely depends on an external DNS lookup to resolve the CNAME to IPs.

For example, suppose www.example.com had set up a CNAME in the Cloudflare control panel pointing to server11.myhostingprovider.biz it would be necessary to look up the IP address of server11.myhostingprovider.biz before contacting the origin server.

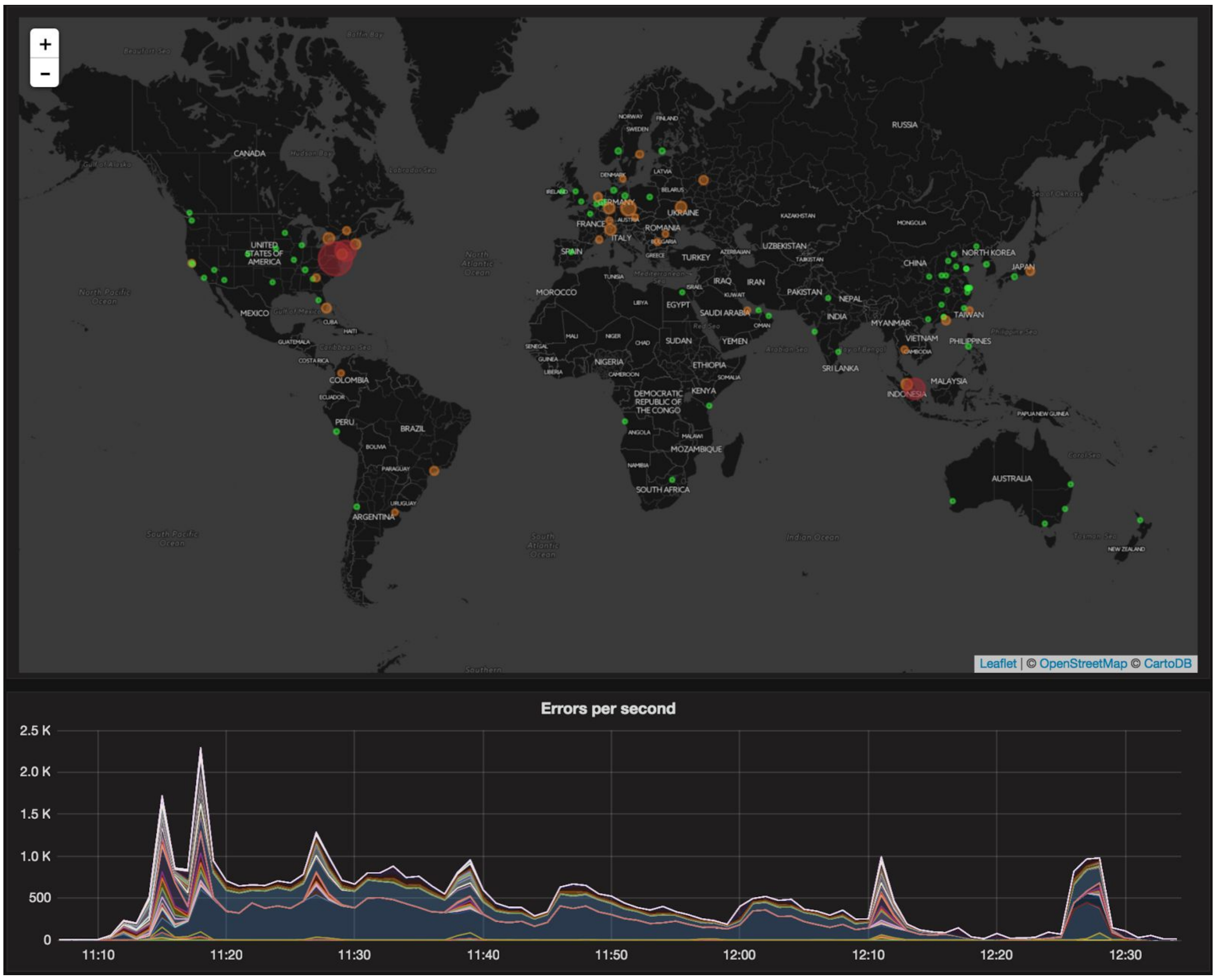
In many cases the target of a CNAME is handled by a third party DNS provider. If the third party provider is unable to answer our query, we are unable to resolve the domain to an origin IP and cannot serve the request.

What Friday’s Dyn outage looked like

As Dyn says in their discussion of the DDoS attack there were three distinct waves. For Cloudflare that manifested itself in two periods during which our internal DNS query error rate spiked.

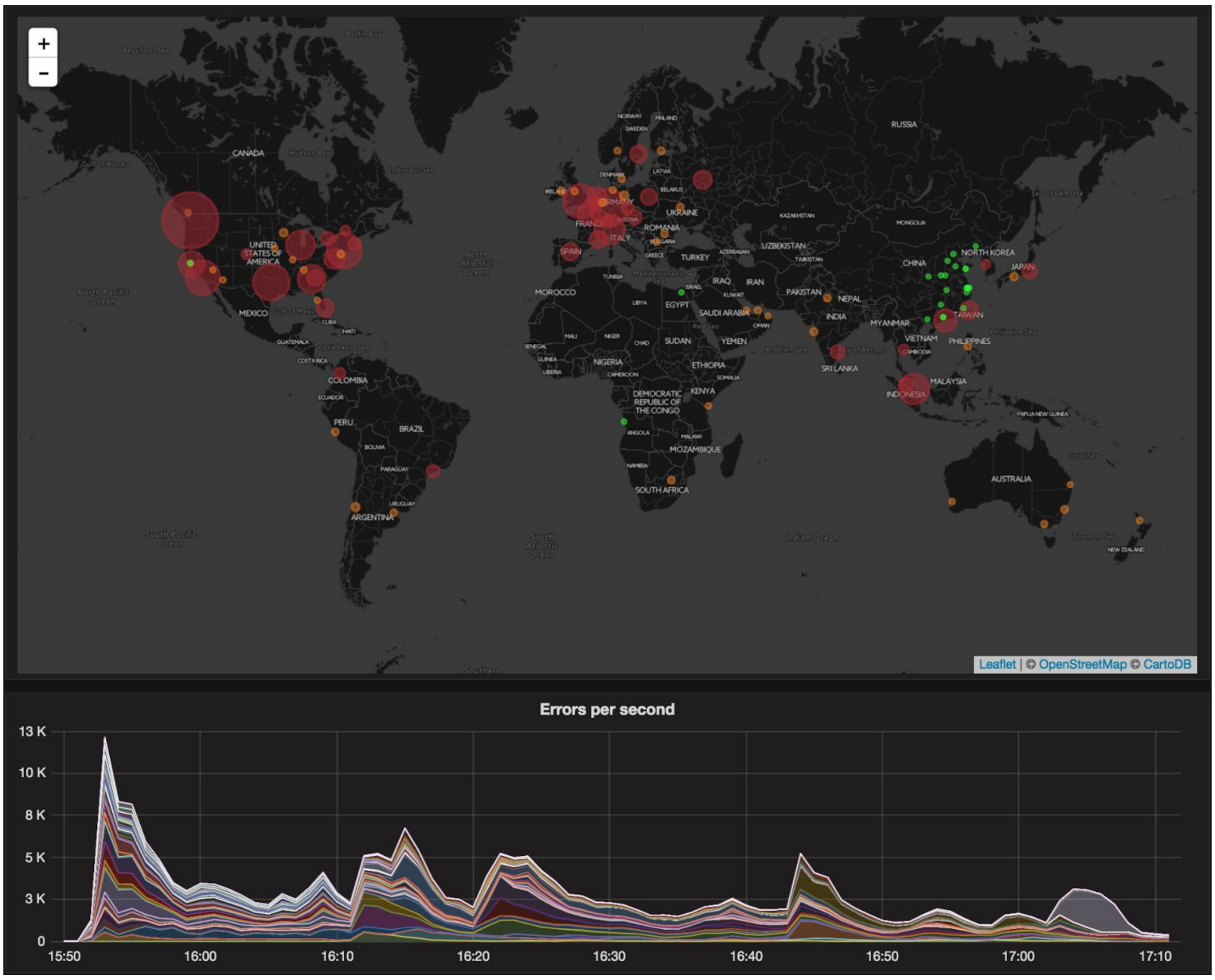
The first attack started at 1110 UTC and mostly affected DNS resolution on the US East Coast. This world map from our monitoring systems shows the Cloudflare data centers where the DNS error rate was spiking because of the Dyn outage.

The green dots on the map are Cloudflare data centers that were unaffected by the Dyn DDoS. The largest effect was on the US East Coast, although the attack had a knock-on effect in Singapore and some parts of Europe. This is most likely because the architecture of the Internet does not directly line up with geography. Locations that are physically disparate can sometimes appear ‘close’ on the Internet because of undersea cables or decisions on how to route traffic.



The chart shows the DNS error rate in each Cloudflare data center affected by the outage. It's possible to see the attack ramp up rapidly and then remained sustained until Dyn was able to tackle it.

Later in the day the attackers returned with greater force and had a worldwide impact. This map shows the Cloudflare data centers seeing errors because Dyn was inaccessible. As you can see almost the entire planet was affected (with the exception of our China locations; we'll return to why below).



Once again it's possible to see the attack ramping up at 1550 UTC and continuing for some time. Dyn reports that the attack was fully mitigated at 1700 UTC.

Media and Dyn reported a third wave of attacks later on Friday, but Dyn mitigated that wave immediately and so fast that it did not have any affect on Cloudflare protected websites and applications and did not show up in our systems.

Why China was unaffected

During the most intense period of attack on Dyn our locations in China were almost completely unaffected. That's because we handle DNS a little differently inside China.

To cope with sometimes fluctuating network conditions inside China our data centers are configured to keep DNS records for origin servers cached in our servers for longer than the rest of the world. This caching meant that even though Dyn was down and couldn't be reached from anywhere (including China) we still had cached DNS records for sites that used Dyn on our China servers. Thus we were able to reach origin servers and continue serving content. That shows up as green dots on the map above.

Unfortunately, there's a downside to hanging on to DNS records for a long time: if one of our customers changes their origin's DNS records we'll keep using the old DNS records and IP addresses. That could lead to downtime, or poor service.

The ideal system would recheck DNS records frequently so that changes are reflected quickly but in the event that the upstream DNS provider was unavailable (because of an attack or other outage) it would be able to use the DNS records it has cached.

Doing so is known as 'serve stale while revalidating'. Our upstream DNS resolvers will cache records checking frequently for changes. If the upstream DNS is unavailable we'll continue to serve from cache until it's possible to refresh the DNS records.

We are testing and rolling out that change now and expect this to greatly diminish the impact of events similar to the Dyn DDoS for all of our customers who use CNAME'd DNS records that rely on a third-party DNS provider.

Conclusion

The Internet is a shared space. Because companies, people, and institutions work together we have a global, connected network that allows us to work and play from almost anywhere. Cooperation means that we work together on standards and interoperability to keep the network running and evolving.

But the Internet is very complex and, as with many things, the devil is in the details and operating Internet infrastructure is a process of constant improvement. Although the Dyn DDoS felt scary to many people unfamiliar with how the Internet operates, such attacks result in a stronger network. Just as Cloudflare is making changes to its software and configuration, so are others across the net.

APPENDIX D

Technical Details Behind a 400Gbps NTP Amplification DDoS Attack

13 Feb 2014 by Matthew Prince.

On Monday we mitigated a large DDoS that targeted one of our customers. The attack peaked just shy of 400Gbps. We've seen a handful of other attacks at this scale, but this is the largest attack we've seen that uses NTP amplification. This style of attacks has grown dramatically over the last six months and poses a significant new threat to the web. Monday's attack serves as a good case study to examine how these attacks work.

NTP Amplification 101

Before diving into the particular details of this attack, it's important to understand the basic mechanics of how NTP amplification attacks work. This is a quick overview of how these attacks occur. John Graham-Cumming on our team previously wrote a detailed primer on NTP amplification attacks if you're interested in further technical details. If you're interested in amplification attacks, you may also find interesting our posts about DNS Amplification attacks. These attacks use a similar method but target open DNS resolvers rather than NTP servers.

An NTP amplification attack begins with a server controlled by an attacker on a network that allows source IP address spoofing (e.g., it does not follow BCP38). The attacker generates a large number of UDP packets spoofing the source IP address to make it appear the packets are coming from the intended target. These UDP packets are sent to Network Time Protocol servers (port 123) that support the MONLIST command.

I'd personally be curious to talk with whoever added MONLIST as a command to NTP servers. The command seems of such little practical use -- it returns a list of up to the last 600 IP addresses that last accessed the NTP server -- and yet it can do so much harm. If an NTP server has its list fully populated, the response to a MONLIST request will be 206-times larger than the request. In the attack, since the source IP address is spoofed and UDP does not require a handshake, the amplified response is sent to the intended target. An attacker with a 1Gbps connection can theoretically generate more than 200Gbps of DDoS traffic.

Not Just Theoretical

Monday's DDoS proved these attacks aren't just theoretical. To generate approximately 400Gbps of traffic, the attacker used 4,529 NTP servers running on 1,298 different networks. On average, each of these servers sent 87Mbps of traffic to the intended victim on CloudFlare's network. Remarkably, it is possible that the attacker used only a single server running on a network that allowed source IP address spoofing to initiate the requests.

While NTP servers that support MONLIST are less common than open DNS resolvers, they tend to run on beefier servers with fatter connections to the network. Combined with the high amplification factor, this allows a much smaller number of NTP servers to generate very large attacks. For comparison, the attack that targeted Spamhaus used 30,956 open DNS resolvers to generate a 300Gbps DDoS. On Monday, with 1/7th the number of vulnerable servers, the attacker was able to generate an attack that was 33% larger than the Spamhaus attack.

Globally Distributed Threat

We saw attack traffic hitting every one of CloudFlare's data centers. While we were generally able to mitigate the attack, it was large enough that it caused network congestion in parts of Europe. The map above shows the global distribution of the 4,529 NTP servers used in the attack. The chart below lists the AS Numbers and names of the top 24 networks we saw traffic from in the attack, as well as the number of exploited NTP servers running on each.

ASN	Network	Count
9808	CMNET-GD Guangdong Mobile Communication Co.Ltd.	136
4134	CHINANET-BACKBONE No.31,Jin-rong Street	116
16276	OVH OVH Systems	114
4837	CHINA169-BACKBONE CNCGROUP China169 Backbone	81
3320	DTAG Deutsche Telekom AG	69
39116	TELEHOUSE Telehouse Inter. Corp. of Europe Ltd	61
10796	SCRR-10796 - Time Warner Cable Internet LLC	53
6830	LGI-UPC Liberty Global Operations B.V.	48

6663	TTI-NET Euroweb Romania SA	46
9198	KAZTELECOM-AS JSC Kazakhtelecom	45
2497	IIJ Internet Initiative Japan Inc.	39
3269	ASN-IBSNAZ Telecom Italia S.p.a.	39
9371	SAKURA-C SAKURA Internet Inc.	39
12322	PROXAD Free SAS	37
20057	AT&T Wireless Service	37
30811	EPiServer AB	36
137	ASGARR GARR Italian academic and research network	34
209	ASN-QWEST-US NOVARTIS-DMZ-US	33
6315	XMISSION - XMission, L.C.	33
52967	NT Brasil Tecnologia Ltda. ME	32
4713	OCN NTT Communications Corporation	31
56041	CMNET-ZHEJIANG-AP China Mobile communications corporation	31
1659	ERX-TANET-ASN1 Tiawan Academic Network (TANet) Information Center	30
4538	ERX-CERNET-BKB China Education and Research Network Center	30

At this time, we've decided not to publish the full list of the IP addresses of the NTP servers involved in the attack out of concern that it could give even more attackers access to a powerful weapon. However, we have published a spreadsheet with the complete list of the networks with NTP servers that participated in the attack. While the per server amplification makes these attacks troubling, the smaller number of servers and networks involved gives us some hope that we can make a dent in getting them cleaned up. We are reaching out to network operators whose resources were used in the attack to encourage them to restrict access to their NTP servers and disable the MONLIST command.

Somewhat ironically, the large French hosting provider OVH was one of the largest sources of our attack and also a victim of a large scale NTP amplification attack around the same time.

Time to Clean Up the Problem

If you're a network administrator and on Monday you saw network graphs like the one in the Tweet below then you are running a vulnerable NTP server:
https://twitter.com/NetworkNub/status/433629994351214592

You can check whether there are open NTP servers that support the MONLIST command running on your network by visiting the Open NTP Project. Even if you don't think you're running an NTP server, you should check your network because you may be running one inadvertently. For example, some firmware on Supermicro's IPMI controllers shipped with a MONLIST-enabled NTP server on by default. More details on NTP attacks and instructions on how to disable the MONLIST command can be found on the Internet Storm Center's NTP attack advisory.

NTP and all other UDP-based amplification attacks rely on source IP address spoofing. If attackers weren't able to spoof the source IP address then they would only be able to DDoS themselves. If you're running a network then you should ensure that you are following BCP38 and preventing packets with spoofed source addresses from leaving your network. You can test whether your network currently follows BCP38 using tools from MIT's the Spoofer Project. If you're running a naughty network that allows source IP address spoofing, you can easily implement BCP38 by following the instructions listed at BCP38.info.

Finally, if you think NTP is bad, just wait for what's next. SNMP has a theoretical 650x amplification factor. We've already begun to see evidence attackers have begun to experiment with using it as a DDoS vector. Buckle up.

APPENDIX E

Stupidly Simple DDoS Protocol (SSDP) generates 100 Gbps DDoS

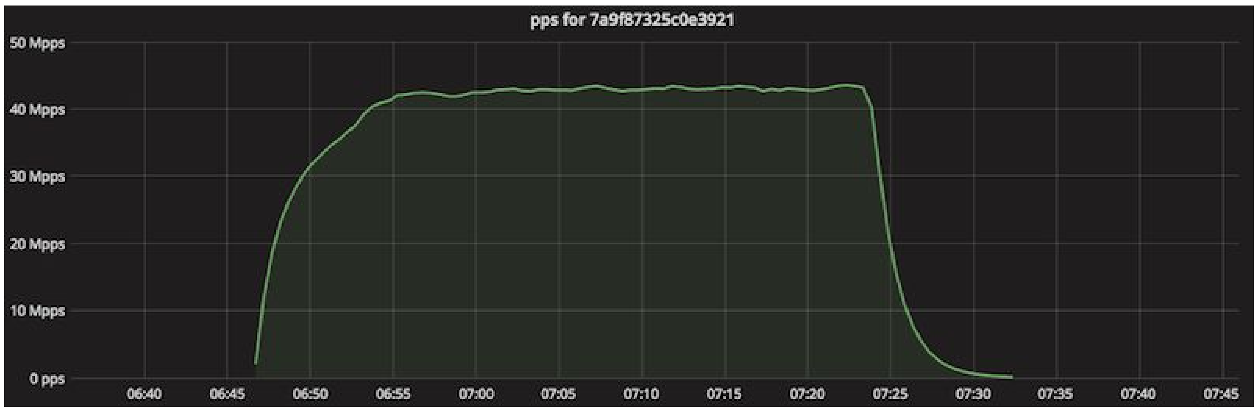
28 Jun 2017 by Marek Majkowski.

Last month we shared statistics on some popular reflection attacks. Back then the average SSDP attack size was ~12 Gbps and largest SSDP reflection we recorded was:

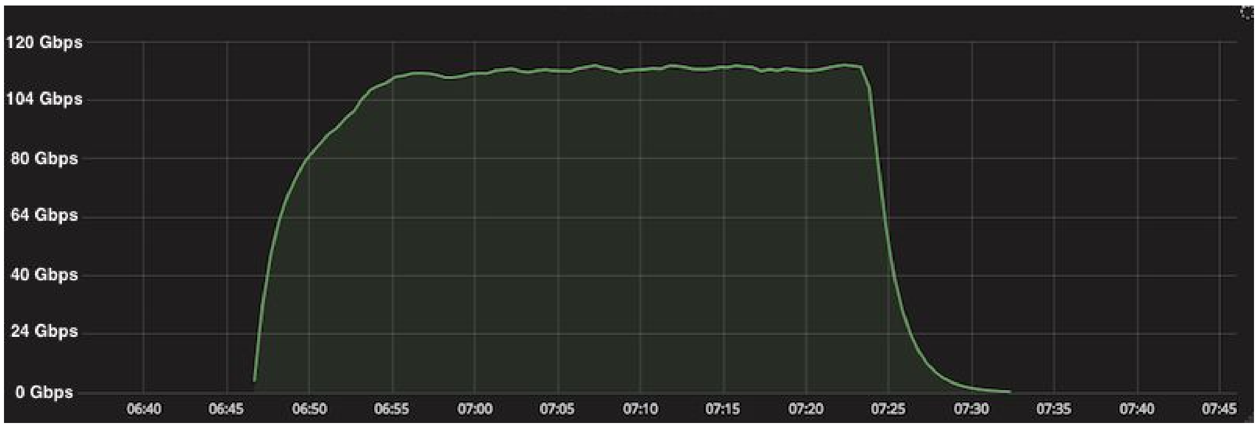
- 30 Mpps (millions of packets per second)
- 80 Gbps (billions of bits per second)
- using 940k reflector IPs

This changed a couple of days ago when we noticed an unusually large SSDP amplification. It's worth deeper investigation since it crossed the symbolic threshold of 100 Gbps.

The packets per second chart during the attack looked like this:



The bandwidth usage:



This packet flood lasted 38 minutes. According to our sampled netflow data it utilized 930k reflector servers. We estimate that the during 38 minutes of the attack each reflector sent 112k packets to Cloudflare.

The reflector servers are across the globe, with a large presence in Argentina, Russia and China. Here are the unique IPs per country:

```
$ cat ips-nf-ct.txt|uniq|cut -f 2|sort|uniq -c|sort -nr|head
 439126 CN
 135783 RU
  74825 AR
  51222 US
  41353 TW
  32850 CA
  19558 MY
  18962 CO
  14234 BR
  10824 KR
  10334 UA
   9103 IT
   ...
```

The reflector IP distribution across ASNs is typical. It pretty much follows the world's largest residential ISPs:

```
$ cat ips-nf-asn.txt |uniq|cut -f 2|sort|uniq -c|sort -nr|head
 318405 4837   # CN China Unicom
  84781 4134   # CN China Telecom
  72301 22927  # AR Telefonica de Argentina
  23823 3462   # TW Chunghwa Telecom
  19518 6327   # CA Shaw Communications Inc.
  19464 4788   # MY TM Net
  18809 3816   # CO Colombia Telecomunicaciones
  11328 28573  # BR Claro SA
   7070 10796  # US Time Warner Cable Internet
   6840 8402   # RU OJSC "Vimpelcom"
   6604 3269   # IT Telecom Italia
   6377 12768  # RU JSC "ER-Telecom Holding"
   ...
```

What's SSDP anyway?

The attack was composed of UDP packets with source port 1900. This port is used by the SSDP and is used by the UPnP protocols. UPnP is one of the zero-configuration networkingprotocols. Most likely your home devices support it, allowing them to be easily discovered by your computer or phone. When a new device (like your laptop) joins the network, it can query the local network for specific devices, like internet gateways, audio systems, TVs, or printers. Read more on how UPnP compares to Bonjour.

UPnP is poorly standardised, but here's a snippet from the spec about the M-SEARCHframe - the main method for discovery:

When a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. It does this by multicasting on the reserved address and port (239.255.255.250:1900) a search message with a pattern, or target, equal to a type or identifier for a device or service.

Responses to M-SEARCH frame:

To be found by a network search, a device shall send a unicast UDP response to the source IP address and port that sent the request to the multicast address. Devices respond if the ST header field of the M-SEARCH request is "ssdp:all", "upnp:rootdevice", "uuid:" followed by a UUID that exactly matches the one advertised by the device, or if the M-SEARCH request matches a device type or service type supported by the device.

This works in practice. For example, my Chrome browser regularly asks for a Smart TV I guess:

```
$ sudo tcpdump -ni eth0 udp and port 1900 -A
IP 192.168.1.124.53044 > 239.255.255.250.1900: UDP, length 175
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 1
ST: urn:dial-multiscreen-org:service:dial:1
USER-AGENT: Google Chrome/58.0.3029.110 Windows
```

This frame is sent to a multicast IP address. Other devices listening on that address and supporting this specific ST (search-target) multiscreen type are supposed to answer.

Apart from queries for specific device types, there are two "generic" ST query types:

- upnp:rootdevice: search for root devices
- ssdp:all: search for all UPnP devices and services

To emulate these queries you can run this python script (based on this work):

```
#!/usr/bin/env python2
import socket
import sys

dst = "239.255.255.250"
if len(sys.argv) > 1:
    dst = sys.argv[1]
st = "upnp:rootdevice"
if len(sys.argv) > 2:
    st = sys.argv[2]
```

```
msg = [
    'M-SEARCH * HTTP/1.1',
    'Host:239.255.255.250:1900',
    'ST:%s' % (st,),
    'Man:"ssdp:discover"',
    'MX:1',
    '']

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
s.settimeout(10)
s.sendto('\r\n'.join(msg), (dst, 1900) )

while True:
    try:
        data, addr = s.recvfrom(32*1024)
    except socket.timeout:
        break
    print "[+] %s\n%s" % (addr, data)
```

On my home network two devices show up:

```
$ python ssdp-query.py
[+] ('192.168.1.71', 1026)
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = 60
EXT:
LOCATION: http://192.168.1.71:5200/Printer.xml
SERVER: Network Printer Server UPnP/1.0 OS 1.29.00.44 06-17-2009
ST: upnp:rootdevice
USN: uuid:Samsung-Printer-1_0-mrgutenberg::upnp:rootdevice

[+] ('192.168.1.70', 36319)
HTTP/1.1 200 OK
Location: http://192.168.1.70:49154/MediaRenderer/desc.xml
Cache-Control: max-age=1800
Content-Length: 0
Server: Linux/3.2 UPnP/1.0 Network_Module/1.0 (RX-S601D)
EXT:
ST: upnp:rootdevice
USN: uuid:9ab0c000-f668-11de-9976-000adedd7411::upnp:rootdevice
```

The firewall

Now that we understand the basics of SSDP, understanding the reflection attack should be easy. You see, there are in fact two ways of delivering the `M-SEARCH` frame:

- what we presented, over the multicast address
- directly to a UPnP/SSDP enabled host on a normal unicast address

The latter method works. We can specifically target my printer IP address:

```
$ python ssdp-query.py 192.168.1.71
[+] ('192.168.1.71', 1026)
```

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = 60
EXT:
LOCATION: http://192.168.1.71:5200/Printer.xml
SERVER: Network Printer Server UPnP/1.0 OS 1.29.00.44 06-17-2009
ST: upnp:rootdevice
USN: uuid:Samsung-Printer-1_0-mrgutenberg::upnp:rootdevice
```

Now the problem is easily seen: the SSDP protocol does not check whether the querying party is in the same network as the device. It will happily respond to an `M-SEARCH`delivered over the public Internet. All it takes is a tiny misconfiguration in a firewall - port 1900 UDP open to the world - and a perfect target for UDP amplification will be available.

Given a misconfigured target our script will happily work over the internet:

```
$ python ssdp-query.py 100.42.x.x
[+] ('100.42.x.x', 1900)
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=120
ST: upnp:rootdevice
USN: uuid:3e55ade9-c344-4baa-841b-826bda77dcb2::upnp:rootdevice
EXT:
SERVER: TBS/R2 UPnP/1.0 MiniUPnPd/1.2
LOCATION: http://192.168.2.1:40464/rootDesc.xml
```

The amplification

The real damage is done by the `ssdp:all` `ST` type though. These responses are *much*larger:

```
$ python ssdp-query.py 100.42.x.x ssdp:all
[+] ('100.42.x.x', 1900)
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=120
ST: upnp:rootdevice
USN: uuid:3e55ade9-c344-4baa-841b-826bda77dcb2::upnp:rootdevice
EXT:
SERVER: TBS/R2 UPnP/1.0 MiniUPnPd/1.2
LOCATION: http://192.168.2.1:40464/rootDesc.xml

[+] ('100.42.x.x', 1900)
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=120
ST: urn:schemas-upnp-org:device:InternetGatewayDevice:1
USN:
uuid:3e55ade9-c344-4baa-841b-826bda77dcb2::urn:schemas-upnp-org:device:InternetGatewa
yDevice:1
EXT:
SERVER: TBS/R2 UPnP/1.0 MiniUPnPd/1.2
LOCATION: http://192.168.2.1:40464/rootDesc.xml

... 6 more response packets....
```

In this particular case, a single SSDP M-SEARCH packet triggered 8 response packets. tcpdump view:

```
$ sudo tcpdump -ni en7 host 100.42.x.x -ttttt
 00:00:00.000000 IP 192.168.1.200.61794 > 100.42.x.x.1900: UDP, length 88
 00:00:00.197481 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 227
 00:00:00.199634 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 299
 00:00:00.202938 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 295
 00:00:00.208425 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 275
 00:00:00.209496 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 307
 00:00:00.212795 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 289
 00:00:00.215522 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 291
 00:00:00.219190 IP 100.42.x.x.1900 > 192.168.1.200.61794: UDP, length 291
```

That target exposes 8x packet count amplification and 26x bandwidth amplification. Sadly, this is typical for SSDP.

IP Spoofing

The final step for the attack is to fool the vulnerable servers to flood the target IP - not the attacker. For that the attacker needs to spoof the source IP address on their queries.

We probed the reflector IPs used in the shown 100 Gbps+ attack. We found that out of the 920k reflector IPs, only 350k (38%) still respond to SSDP probes.

Out of the reflectors that responded, each sent on average 7 packets:

```
$ cat results-first-run.txt|cut -f 1|sort|uniq -c|sed -s 's#^ \+##g'|cut -d " " -f 1|
~/mmhistogram -t "Response packets per IP" -p
Response packets per IP min:1.00 avg:6.99 med=8.00 max:186.00 dev:4.44 count:350337
Response packets per IP:
 value |-------------------------------------------------- count
     0 |                    ****************************** 23.29%
     1 |                                              ****  3.30%
     2 |                                                **  2.29%
     4 |************************************************** 38.73%
     8 |            ************************************** 29.51%
    16 |                                               ***  2.88%
    32 |                                                    0.01%
    64 |                                                    0.00%
   128 |                                                    0.00%
```

The response packets had 321 bytes (+/- 29 bytes) on average. Our request packets had 110 bytes.

According to our measurements with the ssdp:all M-SEARCH attacker would be able to achieve:

- 7x packet number amplification
- 20x bandwidth amplification

We can estimate the 43 Mpps/112 Gbps attack was generated with roughly:

- 6.1 Mpps of spoofing capacity
- 5.6 Gbps of spoofed bandwidth

In other words: a single well connected 10 Gbps server able to perform IP spoofing can deliver a significant SSDP attack.

More on the SSDP servers

Since we probed the vulnerable SSDP servers, here are the most common `Server` header values we received:

```
104833 Linux/2.4.22-1.2115.nptl UPnP/1.0 miniupnpd/1.0
  77329 System/1.0 UPnP/1.0 IGD/1.0
  66639 TBS/R2 UPnP/1.0 MiniUPnPd/1.2
  12863 Ubuntu/7.10 UPnP/1.0 miniupnpd/1.0
  11544 ASUSTeK UPnP/1.0 MiniUPnPd/1.4
  10827 miniupnpd/1.0 UPnP/1.0
   8070 Linux UPnP/1.0 Huawei-ATP-IGD
   7941 TBS/R2 UPnP/1.0 MiniUPnPd/1.4
   7546 Net-OS 5.xx UPnP/1.0
   6043 LINUX-2.6 UPnP/1.0 MiniUPnPd/1.5
   5482 Ubuntu/lucid UPnP/1.0 MiniUPnPd/1.4
   4720 AirTies/ASP 1.0 UPnP/1.0 miniupnpd/1.0
   4667 Linux/2.6.30.9, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
   3334 Fedora/10 UPnP/1.0 MiniUPnPd/1.4
   2814  1.0
   2044 miniupnpd/1.5 UPnP/1.0
   1330 1
   1325 Linux/2.6.21.5, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
    843 Allegro-Software-RomUpnp/4.07 UPnP/1.0 IGD/1.00
    776 Upnp/1.0 UPnP/1.0 IGD/1.00
    675 Unspecified, UPnP/1.0, Unspecified
    648 WNR2000v5 UPnP/1.0 miniupnpd/1.0
    562 MIPS LINUX/2.4 UPnP/1.0 miniupnpd/1.0
    518 Fedora/8 UPnP/1.0 miniupnpd/1.0
    372 Tenda UPnP/1.0 miniupnpd/1.0
    346 Ubuntu/10.10 UPnP/1.0 miniupnpd/1.0
    330 MF60/1.0 UPnP/1.0 miniupnpd/1.0
    ...
```

The most common `ST` header values we saw:

```
298497 upnp:rootdevice
 158442 urn:schemas-upnp-org:device:InternetGatewayDevice:1
 151642 urn:schemas-upnp-org:device:WANDevice:1
 148593 urn:schemas-upnp-org:device:WANConnectionDevice:1
 147461 urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1
 146970 urn:schemas-upnp-org:service:WANIPConnection:1
 145602 urn:schemas-upnp-org:service:Layer3Forwarding:1
```

```
113453 urn:schemas-upnp-org:service:WANPPPConnection:1
100961 urn:schemas-upnp-org:device:InternetGatewayDevice:
100180 urn:schemas-upnp-org:device:WANDevice:
 99017 urn:schemas-upnp-org:service:WANCommonInterfaceConfig:
 98112 urn:schemas-upnp-org:device:WANConnectionDevice:
 97246 urn:schemas-upnp-org:service:WANPPPConnection:
 96259 urn:schemas-upnp-org:service:WANIPConnection:
 93987 urn:schemas-upnp-org:service:Layer3Forwarding:
 91108 urn:schemas-wifialliance-org:device:WFADevice:
 90818 urn:schemas-wifialliance-org:service:WFAWLANConfig:
 35511 uuid:IGD{8c80f73f-4ba0-45fa-835d-042505d052be}000000000000
  9822 urn:schemas-upnp-org:service:WANEthernetLinkConfig:1
  7737 uuid:WAN{84807575-251b-4c02-954b-e8e2ba7216a9}000000000000
  6063 urn:schemas-microsoft-com:service:OSInfo:1
   ...
```

The vulnerable IPs are seem to be mostly unprotected home routers.

Open SSDP is a vulnerability

It's not a novelty that allowing UDP port 1900 traffic from the Internet to your home printer or such is not a good idea. This problem has been known since at least January 2013:

- "Security Flaws in Universal Plug and Play: Unplug, Don't Play"

Authors of SSDP clearly didn't give any thought to UDP amplification potential. There are a number of obvious recommendations about future use of SSDP protocol:

- The authors of SSDP should answer if there is any real world use of unicast `M-SEARCH`queries. From what I understand `M-SEARCH` only makes practical sense as a multicast query in local area network.
- Unicast `M-SEARCH` support should be either deprecated or at least rate limited, in similar way to DNS Response Rate Limit techniques.
- `M-SEARCH` responses should be only delivered to local network. Responses routed over the network make little sense and open described vulnerability.

In the meantime we recommend:

- Network administrators should ensure inbound UDP port 1900 is blocked on firewall.
- Internet service providers should never allow IP spoofing to be performed on their network. IP spoofing is the true root cause of the issue. See the infamous BCP38.
- Internet service providers should allow their customers to use BGP flowspec to rate limit inbound UDP source port 1900 traffic, to relieve congestion during large SSDP attacks.

- Internet providers should internally collect netflow protocol samples. The netflow is needed to identify the true source of the attack. With netflow it's trivial to answer questions like: "Which of my customers sent 6.4Mpps of traffic to port 1900?". Due to privacy concerns we recommend collecting netflow samples with largest possible sampling value: 1 in 64k packets. This will be sufficient to track DDoS attacks while preserving decent privacy of single customer connections.
- Developers should not roll out their own UDP protocols without careful consideration of UDP amplification problems. UPnP should be properly standardized and scrutinized.
- End users are encouraged to use the script scan their network for UPnP enabled devices. Consider if these devices should be allowed to access to the internet.

Furthermore, we prepared on online checking website. Click if you want to know if your public IP address has a vulnerable SSDP service:

- https://badupnp.benjojo.co.uk/

Sadly, the most unprotected routers we saw in the described attack were from China, Russia and Argentina, places not historically known for the most agile internet service providers.

Summary

Cloudflare customers are fully protected from SSDP and other L3 amplification attacks. These attacks are nicely deflected by Cloudflare anycast infrastructure and require no special action. Unfortunately the raising of SSDP attack sizes might be a tough problem for other Internet citizens. We should encourage our ISPs to stop IP spoofing within their network, support BGP flowspec and configure in netflow collection.

This article is a joint work of Marek Majkowski and Ben Cartwright-Cox.

APPENDIX F

A Deep Dive Into DNS Packet Sizes: Why Smaller Packet Sizes Keep The Internet Safe

04 Mar 2016 by Dani Grant.

Yesterday we wrote about the 400 gigabit per second attacks we see on our network.
One way that attackers DDoS websites is by repeatedly doing DNS lookups that have small queries,

but large answers. The attackers spoof their IP address so that the DNS answers are sent to the server they are attacking, this is called a reflection attack.

Domains with DNSSEC, because of the size of some responses, are usually ripe for this type of abuse, and many DNS providers struggle to combat DNSSEC-based DDoS attacks. Just last month, Akamai published a report on attacks using DNS lookups against their DNSSEC-signed .gov domains to DDoS other domains. They say they have seen 400 of these attacks since November.

To prevent any domain on CloudFlare being abused for a DNS amplification attack in this way, we took precautions to make sure most DNS answers we send fit in a 512 byte UDP packet, even when the zone is signed with DNSSEC. To do this, we had to be creative in our DNSSEC implementation. We chose a rarely-used-for-DNSSEC signature algorithm and even deprecated a DNS record type along the way.

Elliptic Curves: Keeping It Tight

Dutch mathematician Arjen Lenstra famously talks about cryptography in terms of energy. (We've covered him once before on our blog). He takes the amount of energy required to break a cryptographic algorithm and compares that with how much water that energy could boil. To break a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water. On the other hand, to break a 228-bit elliptic curve key requires the amount of energy needed to boil all the water on the earth.

With elliptic curve cryptography in the ECDSA signature algorithm, we can use smaller keys with the same level of security as a larger RSA key. Our elliptic curve keys are 256 bits long, equivalent in strength to a 3100 bit RSA key (most RSA keys are only 1024 or 2048 bits). You can compare below two signed DNSKEY sets, an RSA implementation against our ECDSA one. Ours is one quarter of the size of the matching RSA keys and signature.

As a side benefit, ECDSA is lightning fast, and our engineer Vlad Krasnov actually helped make it even faster. By implementing ECDSA natively in assembler, he was able to speed up signing by 21x. His optimizations are now part of the standard Go crypto library as of Go version 1.6. It now only takes us a split of a second, 0.0001 of a second, to sign records for a DNS answer.

Deprecating ANY: The Obituary Of A DNS Record Type

In Akamai's security report, the authors draw the conclusion that DNSSEC is the only cause of the large answers used for DDoS attacks, but the other cause of the large answers is that the attackers use ANY queries to maximize the amplification factor. ANY queries are a built-in debugging tool, meant to return every DNS record that exist for a name. Unfortunately, they are instead more often used for launching large DDoS attacks.

In September, we stopped answering ANY queries and published an Internet Draft to begin the process of making ANY deprecation an Internet standard. We did this carefully, and worked closely with the few remaining software vendors who use ANY to ensure that we wouldn't affect their production systems.

An ANY query for DNSSEC-enabled cloudflare.com returns an answer that is 231 bytes. The alleged domain in Akamai's paper, for comparison, returns an ANY query almost 18 times larger, at a whopping 4016 bytes.

ECDSA + ANY

By keeping our packet size small enough to fit in a 512 byte UDP packet, we keep the domains on us safe from being the amplification factor of a DDoS attack. If you are interested in using DNSSEC with CloudFlare, here are some easy steps to get you setup. If you are interested in working on technical challenges like these, we'd love to hear from you.

How the Consumer Product Safety Commission is (Inadvertently) Behind the Internet's Largest DDoS Attacks

25 Aug 2016 by Matthew Prince.

inShare51

The mission of the United State's Government's Consumer Product Safety Commission(CPSC) is to protect consumers from injury by products. It's ironic then that the CPSC is playing an unwitting role in most of the largest DDoS attacks seen on the Internet. To understand how, you need to understand a bit about how you launch a high volume DDoS.

Logo of the Consumer Product Safety Commission

Amplification

DDoS attacks are inherently about an attacker sending more traffic to a victim than the victim can handle. The challenge for an attacker is to find a way to generate a large amount of traffic. Launching a DDoS attack is a criminal act, so an attacker can't simply go sign up for large transit contracts. Instead, attackers find ways to leverage other people's resources.

One of the most effective strategies is known as an amplification attack. In these attacks, an attacker can amplify their resources by reflecting them off other resources online that magnify the level of traffic. The most popular amplification vector is known as DNS reflection.

DNS Reflection

We've written about DNS reflection attacks in detail before. The basics are that an attacker generates DNS requests from a network that allows source IP address spoofing. The attacker forges the victim's IP as the source of the DNS requests. The attacker sends these requests to DNS resolvers that aren't locked down and respond to requests from any network. The DNS resolvers receive the requests and then send the responses back to the spoofed IP of the victim.

This technique "amplifies" an attack because a small DNS request will generate a larger response. How much larger depends on the size of the DNS record that's being queried. And here's where the Consumer Product Safety Commission comes in.

#1 Among DDoS Attackers

You see, the CPSC's DNS server will respond with a very large DNS record when sent a very small DNS query. This DNS query is only 65 bytes:

```
dig ANY cpsc.gov +notcp +bufsize=65535 @X.X.X.X
```

Substitute X.X.X.X for one of the 13 million currently open DNS resolvers and you'll get back a response that is 4,426 bytes. To generate 68Gbps of traffic to a victim requires only 1Gbps of requests with a spoofed source IP address to open resolvers for the CPSC.gov DNS record. That's an amplification factor of 68x.

Now, lots of people have large DNS records, but the CPSC.gov record is very large and particularly popular among attackers. Based on both data about the large volumetric attacks we see hitting CloudFlare, as well as data from various resolver honey pots we run, approximately 94% of the DNS reflection attack requests we see are for CPSC.gov. (The next most popular are httrack.com which is used by 2.9% of attacker requests and isc.org which is used by 0.5%.)

At one level there's nothing the CPSC can do to prevent attackers from using their DNS record to launch attacks. CPSC's resources aren't being queried directly by the attackers so they can't block the requests themselves. However, they could construct their DNS record more carefully to make it smaller and therefore less attractive to attackers.

To see how, we can walk through the CPSC DNS record entry by entry and make recommendations on how to reduce its size.

The CPSC's Ginormous Zone

Here's the CPSC.gov's entire 4,426-byte DNS response which you get from running the ANY DNS query above against an open resolver (scroll to the left in the grey boxes below to see the full DNS record):

```
cpsc.gov.       18894   IN  SOA auth00.ns.uu.net. hostmaster.uu.net. 994622 1800 600
1728000 21600
cpsc.gov.       18894   IN  A   63.74.109.2
cpsc.gov.       18894   IN  AAAA    2600:803:240::2
cpsc.gov.       18894   IN  NS  auth61.ns.uu.net.
cpsc.gov.       18894   IN  NS  auth00.ns.uu.net.
cpsc.gov.       18894   IN  MX  5 stagg.cpsc.gov.
cpsc.gov.       18894   IN  MX  5 hormel.cpsc.gov.
cpsc.gov.       18894   IN  TXT "v=spf1 ip4:63.74.109.6 ip4:63.74.109.10 ip4:63.74.109.20 mx
a:list.cpsc.gov -all"
cpsc.gov.       18894   IN  DNSKEY  256 3 7
AwEAAbpeSszphwwkOIJn1ha6DE/W3YRXFR2vsMi0RKhq5x9t487UJc0c
eamz5TZj6KV5/tzL8/Qr2jnTaQmpWtJHbnF0kqpxeZIR+wzaNbMtEH30
UF5BDv9Bya0W9I+40dS48996kedhEvL6KwmMelB7FH6QPd0ixyhp0+ci
5vew9lzTESEsJ2X2uJrCqo3UacsHyYIzaTSXPpfwizQCql4VySq6+im1
74QaYw/FU4aADAv3R2KQvsR/uI0a7oOihxDDAvtYG7SZvotW3ASZFscd
4B6Yd84RMZC3yGdGtyrSD6tZsiJZoXhLQkkf0kOTWCjPvD8oPm+yYiGI Cm64eM3kflU=
cpsc.gov.       18894   IN  DNSKEY  256 3 7
AwEAAXQUfP1/CdN5/YYXxWdePx3dWhhY7RmzxEfGXSZ0ea5BZoOXTLHd
giWlm9ORnZ5hC+kaDRoYjgZXnkZOkhQhCDBwm2O2IOGBjBLMMtbm9hNK
b2WGE8WC/E3j56YfepaMSzhxICu1xgY8JeYhmfpc3C5Z9Mm2oPm91cwU
WZZY8i5f2F04tNBBymXTfu0mytCvp/dNxUjM45svY+SNRJltgcy07qBj
T/GHDglEc6iJdBtvik3Nd4RfFfI+ftG8xSxfna3Nv4BVdYxPkE4us3ti
0dv/Ejwl9kuoXhT7/Ydpdze/boWmIuwjn3a66Afg7CHtmYyW6InLz57r tzUTGUdStgc=
cpsc.gov.       18894   IN  DNSKEY  257 3 7
AwEAAZztz17cVspxUk8egfYEFLuyPXVETlPdT2PAuy+cZTk3afTS7cda
Tnsk43AIqgnCkTvHE9m4gVuOhNmFjPIABPkfmaCtOzyqVmLjxb36JMxJ
TnhBPBYjWY0HrBdEGCGG7eZzY4ll9kAMPIxE1OmMl9iM0dQSzamITEWN
89oPptHnlbjz8k7nQO3xyzXreamjhIW/2iIJhM+CdHe2CgMhPtf8b4QR
8CuIBMH07gvsTKljuvQLiS1ThQYYpmLgriiWjnQFum2FJe6J7x8joDAq
YCzbQUdGSyJPp6FYibaG70Y62fIf9DNgHRMH/3c79DW9RmwzFggjfKLf y4hOgRbsVFc=
cpsc.gov.       18894   IN  DNSKEY  257 3 7
AwEAAX5Tor9V7TnhfUMAL67reT+IFYd+4ciQv/UnvZbNgj7DgDuJPpcl
Owh6ypAldCYgTXkF2Qt+an9WVp+Khsp2wRCCOhvGIUR9sOGdzxumDUCT
Uru2dxHAqIn1QYSjuT8huMDDyBJmnoA4AY1Te86mcE1Jwpo+S9KoB23Z
JgnMedU+6i8Qm9cdGLNM7nqEXhgKgmKc/387UFdh25jltsg0d2gOK//q
k2HfLdDqv8XlrlacfMsSXniVwK7E6mtqcfbF518M2bl6UFJWXuxp+cU8
0WdmGiQfxmLvm62a2aVs9IzR6qGg0Ce5bxbx68v6gYTgIOUBm8ERytZ3 T2jzc0QOKQc=
cpsc.gov.       18894   IN  NSEC3PARAM 1 0 12 AABBCCDD
cpsc.gov.       18894   IN  RRSIG   NS 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. e4NrJJq0j7ZcHb/UAm0E34nuPfnbgCW6FARJM/5QKwgYDqZpCixaVQgL
s9KGLR/eYQsp+BdPQibdLD3Ef2ICqQmBl0qRIyxh5Sbg7XiXnObJXgtR
+vJuhVjzLtYfAGGicAm3MfYzcxk2/Usr8rwr/EakzhMLEr03Tshj4uwm
WaRQ97M3R6dyTDvJ35X0m76KeWAyIS/Z2WmbGzNWUn8qpgerD8d1CIrP
R7S5psgTMFIApD5I6mfJjuvZ5B+RRs7VhjY9tn6IKKoBLMqHxZAtwf1n
td9Ho8anT3LJrrB1b/NZsRWhVdGLDlohDKzM9qqa6wQkoHg+/zrRrapZ 5NiUlQ==
cpsc.gov.       18894   IN  RRSIG   MX 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. jAYmQYaSMOkU7VfwTqIb9CHO3nBBjkiYQ4zOoRIaD8u0ASMaweXhy/H2
75kxrhvwYs9VR0RXOlTal69uV9vN74cffu4lIfi9mIfSSVvo3JZz6dDM
bpgmpdXRd6DEFVpmrHfTUK1sbKGB5qo3Yxiz38VzcZr7hclApmC/Bf0/
```

```
/luymk7BeCvtGjLVamRaaHKD76f/FA+0dkLvNOZITqrj36wZwOb1JTv8
XAl61dVxJl3R3z8TCtWm8+Za5nTzt3oNjad3Hf4g9r+88ugCvCcOibSU
ujUH1sdc3/Uvr7ZEGT+51a6H6BkoctWwAimkN6CwbM0M8GhqwNiISXov 5QKO+Q==
cpsc.gov.        18894   IN  RRSIG   TXT 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. CYoU8McQOKcy82SKfTXPfT8AnheHQmtdf4m0se/Ht7q64ll6iflSkh3O
VGRmrQlgiMdAKaPIRMU7liGftMUntYKDu7RivVG18JokePDYHH9vYjss
nQf92Reu08dOwm/Ica9cM1naGL1RDzvBbDkQkucyjJe2fvd00Npj+vBG
H5kyLULmbkfD2PVL6c5F+eEhyGCF/IKPveag1ymkwBMGRM+Za+LYOeLY
SkB1zwL5GodnHj/uaZfkGxT5ulNOnukMx536719pjf7ig/oMdaf8MySo
hGQCjseS5ON6dRYuAzkOVDwdc5/Qw2jWqQlQUUH213AAYKz6eZXZtLAt +AflUQ==
cpsc.gov.        18894   IN  RRSIG   A 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. DOLz8p1/pyr3LByKjHKZEcrgQwDqkB+Gg3nlcqSbFp8ChIenAjBGBvXy
24u7atI55ZxTqF6dtQEo3+D9a30piWeDm5U2tOg/VTHtwFFfi2h5DwQx
WNxh6/71h7uzoidiBr4cKEkZSjB4F+bRviqJomxSdGPA9ZrQFZv4RcCb
B8InJJZwyu9gfT767nt3yOq+Os/nlr19vyicmt5MiG9WQpxBn/ZhUG2u
DYf6WkZVVo2i3nrD9PmpOjeDuj4g0DKAlhyPzPM5L8CKNDJSxGAdAHy+
auOJUFRPan3TGQqmgaqaEs/8HLiRr9QSgafuRJCO0HNW97p91zXX2krI AtDHgw==
cpsc.gov.        18894   IN  RRSIG   AAAA 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. ANnB5Mw+iU6b2TD/3ULnt2LR138Vvd80BaYuOX765TNxpsoPosTxbls1
n6810qVvkHxR+uFafmrHA9PHOwJavEgmwSk4TyKbiTdu5fY0+wnR0tXr
Fl8ARAALvmunyZ49aaEQUa0TUgKyNQIKoNktnfiLFhcFpsfQD678FGm3
IKtFnFpj3HvM3Y3vkfKORnqHEyT37O8EK5cIbK4YZ3UE/AJwqBEqUt7x
xca51gRK3IW0/MBRoargRbg1zgVxhNCnnxWk59uMEER0wePsQK9XAMJ1
9QjpXlPFW+0426pFqbUTQyymwTjHFu2G8yzJas5g2r+oQShPTR80K6WG rmNbCA==
cpsc.gov.        18894   IN  RRSIG   SOA 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. I7umZbpbUnh24bvphYP9K70imTyDFxc5QgOEDVVCSkmtVMhRaOU0eiSO
6b/O3ExcMhL0xn+D+/Xs1R7z9zPsKrzvGCfqotkxu8L1mRpAmjgr21WH
mEzXftcx+IbSrOCwOa7YuPyBGqJW3f8kb/UHcI7ykqk08xvglD2I77PV
TgI59oYsZyufRhfChcFxEyhBY2LZ4o/UO5Tw8Lg5HH/9OGtA0FHum1To
v/CeLBOaATrZNygZINKmHlsP6Mm8j83jM4bRNxLGPs8VjpDbKVAHtROq
aXR0Zy81zNs0eWfs4vThdOf56nCUnA/HiwlVNgmbO219CCMKQxoBBuY+ FYcWfA==
cpsc.gov.        18894   IN  RRSIG   DNSKEY 7 2 21600 20160824030507 20160817020507
53799 cpsc.gov. HlTGZb9hSEwtxLayRdqxBkS+1jZIx/2FbsGaNJREfijFf7HTX/gpmXil
g2DcVSvXwh6J473YKKpvdIe0LOfgIGA9NWaO5M3FVeKbwRk4OENdKTEG
Vwm/uB8Av0aLsxD8YRvprPKxi7YxVDta7/cRMX1vP4ULQHrAbrJ5xlbu
SzrlwK5b1BDoil3qfPNVbpHg2NMuqyp/hbFzUfWzvRNbLG3PJZfUEppl
QTzInHnBONKmS0oRW4oUj8la2Zu6BGIOjB/0IktrLyxRKHJ2KNZDa1+o
0pKr68/gY2j7EfJwWrrohieG5LHfisCdTgaqs1zZbYgv+CAAl9hvq6vJ 5xrFxA==
cpsc.gov.        18894   IN  RRSIG   DNSKEY 7 2 21600 20160824030507 20160817020507
58273 cpsc.gov. DlWHCC0re6XY3+MzRDYpJ2nIo2bWczGkJpxvA1yXhUivz8Qlv5mK1mtV
3YTislUzE9qWQTDOMoatcmQHVqZiHZG9d7w8/5XzBv3YuAqJNvUKs5QF
7YFBfF/acQCI40/pEyDhwNn5NbCJ3kpJBo7KSHBJpo9GNMnVQMmcK6bo
UCeXEeI9Z1cnu/9XiD8jVSyVEMSs0KdjDmKEXNXgChVUWCr6+oWn8g/T
1SCUHF5mWn7r/ZgGPAyBwzoJMA/f9qdEBighiOz/DNe5dSKHAr/3Cf78
AJIFvEPe+Yn+KsRo14WLzzfuNT2NZp/nQ6TIHVLC04uDMXeimjQiHMJZ 8QDQzw==
cpsc.gov.        18894   IN  RRSIG   NSEC3PARAM 7 2 21600 20160824030507
20160817020507 53799 cpsc.gov.
ink6+sysDZdHK+1Ntfllouw2I7nAvqiV+9uPQI7O0MFdGMSIZJ6Dli7v
z81S+yTp3diXZv+m4j0wTFSGtcN3Zte14jC+Kl+SJ2SRmfXedgcAXcZW
QrPYgDps2C4jxxytxePKxHuDKLuSsxtUFmonOWKJuC2p7UqGDhsa5AHS
XB8SoUX5ezgWjrSZgUmA7sWDE8szlvOtv1DtJKR62RIXQEOckd7z2HKh
oQe5fZx1C1eoR+Ppgi1eNiC+Dqh3Q4FYcCzvSfY/rtZ88dyg8DKj0tV1
RMXYDR8+x18qSqAK9NlXLldumVTKBcC3rzTUKd0PLj5/sdbyaYo6X2mn RpHQXg==
cpsc.gov.        893 IN  RRSIG   DS 8 2 3600 20160824041014 20160817041014 64415 gov.
```

```
c78BIiQZ0L8C19OaNWlRtmr1/2cnL9yap0tcC4WPgKTtzluj4esJaN7u
PZkKgaQME0BuImPs94s/IKy0JmWrobApCWPVjhtHq0E4m8UWsBbaORJX
Ihg8JuS80rHFdPU6Y4u6Gu2ShTmE5eM+0wE1L5KKjt2co+Wa7AN4W2Kv Udo=
cpsc.gov.       893 IN  DS  58273 7 1 75F81A67F1D76F017B02689E9586FCF3A68F655F
cpsc.gov.       893 IN  DS  58273 7 2
A04919AA51B4395014E2753430A6DD6AF7585CE2C5C2E693E9ADD4DE C1365DC9
cpsc.gov.       893 IN  DS  27793 7 2
C078BB7D7A46B0654A7D83755BA6FAB6AA56DAC4CB4D94B6A3562B67 BA7C285C
cpsc.gov.       893 IN  DS  27793 7 1 E148C892A62ABF2B9157293DA4EE270D9880C238
```

You may get the records back in a different order, but you should see something similar if you run the same ANY query.

Walking through the DNS zone file line-by-line you can see the mistakes that the CPSC has made that bloat its record and learn a bit about DNS and, in particular, DNSSEC.

The Good

The Good, The Bad, and The Ugly artwork by Billy Perkins

Let's start with the good. Here are the first eight records, which are pretty standard and can't be optimized much.

```
cpsc.gov.       18894   IN  SOA auth00.ns.uu.net. hostmaster.uu.net. 994622 1800 600
1728000 21600
cpsc.gov.       18894   IN  A   63.74.109.2
cpsc.gov.       18894   IN  AAAA    2600:803:240::2
cpsc.gov.       18894   IN  NS  auth61.ns.uu.net.
cpsc.gov.       18894   IN  NS  auth00.ns.uu.net.
cpsc.gov.       18894   IN  MX  5 stagg.cpsc.gov.
cpsc.gov.       18894   IN  MX  5 hormel.cpsc.gov.
cpsc.gov.       18894   IN  TXT "v=spf1 ip4:63.74.109.6 ip4:63.74.109.10 ip4:63.74.109.20 mx
a:list.cpsc.gov -all"
```

These first eight records are bread and butter DNS: SOA, A, AAAA, NS, MX, and TXT. If you want a quick refresher on these basic DNS records, read on. Otherwise, you can skip to the next section ("The Bad") to see where the DNSSEC mess begins.

The first record is the SOA record. SOA stands for Start of Authority. Walking through the record left-to-right, the number 18894 — which you'll see appears next to all records — is the TTL (Time to Live) at the moment for the particular resolver I queried. That number represents the seconds until the resolver needs to fetch another result from authoritative

DNS server. If I queried the same resolver 10 seconds later, the number would be 18884 — 10 seconds less than it had been before. If you query a different resolver you'll likely get another number that is somewhere between 21600 (the max TTL the CPSC.gov has specified) and 0. When the number reaches 0, the resolver will query the authoritative name server, update its cache, and reset the TTL to 21600 — the maximum TTL specified for the record.

```
cpsc.gov.       18894   IN  SOA auth00.ns.uu.net. hostmaster.uu.net. 994622 1800 600
1728000 21600
```

Continuing on the SOA record, next up is "IN" — which also appears on every DNS record in the CPSC zone. IN stands for "Internet Class." DNS predated the Internet, so there are other classes that are possible, but IN is the only class you'll see in common use today. After "IN" comes "SOA" which designates this record type.

The next entries are the contents of the SOA record. The first entry specifies the primary authoritative name server for the domain ("auth00.ns.uu.net."). The second is the email address of the name server's administrator ("hostmaster.uu.net." means hostmaster@uu.net is the name server's administrator's email). "994622" is the zone serial number, which other name servers use to make sure they're in sync with the primary authoritative name server. "1800" is the refresh interval, specifying the number of seconds before non-primary name servers should check to see if the zone has changed. "600" is the retry interval, specifying the number of seconds to wait if a refresh failed. "1728000" is the number of seconds that non-primary name servers can serve a zone if they've failed to reach the primary name server. In practice with modern DNS setups, none of these numbers end up having much of an impact.

The last number in the SOA record, however, is important. "21600" is the length of time in seconds a negative result should be cached by recursive resolvers. For example, if you query "12345.cpsc.gov" — a record that doesn't exist — then the resolver you query will store the fact that the record doesn't exist for 21600 seconds.

SOA records look complicated but, in practice, only the last number — specifying the number of seconds that a negative result should be cached — has an impact on modern DNS implementations.

The other records in this group of 8 are less cryptic than SOA so we can quickly work through how they work.

```
cpsc.gov.       18894   IN  A   63.74.109.2
cpsc.gov.       18894   IN  AAAA   2600:803:240::2
cpsc.gov.       18894   IN  NS  auth61.ns.uu.net.
cpsc.gov.       18894   IN  NS  auth00.ns.uu.net.
```

The A and AAAA records are "anchor" records which specify the IP addresses responsible for the domain. A records are for IPv4 addresses — in this case 63.74.109.2 — and AAAA records are for IPv6 addresses — in this case 2600:803:240::2. The two NS entries define the name servers responsible for the domain ("auth00.ns.uu.net" which is primary as specified by the SOA record, and "auth61.ns.uu.net" which is secondary).

```
cpsc.gov.      18894   IN  MX  5 stagg.cpsc.gov.
cpsc.gov.      18894   IN  MX  5 hormel.cpsc.gov.
```

The two MX records specify the mail servers responsible for email sent to the domain ("hormel.cpsc.gov" and "stagg.cpsc.gov" — someone at the CPSC seems to have a sense of humor as Hormel is the maker of the pork product spam and "Stagg" is a brand of chili also made by Hormel). The number "5" in both MX records is the weight of that mail server. The lower the number the more it will be preferred. Since these are equally weighted, email will be load balanced equally between the two.

```
cpsc.gov.      18894   IN  TXT "v=spf1 ip4:63.74.109.6 ip4:63.74.109.10 ip4:63.74.109.20 mx
a:list.cpsc.gov -all"
```

Not much from the above could be optimized. The next record is special kind of TXT record known as a SPF record. It specifies what domains or IPs are allowed to send email on behalf of CPSC.gov. In this case, three IP addresses (63.74.109.6, 63.74.109.10 & 63.74.109.20), the MX records (Hormel & Stagg), and the A record for the subdomain (lists.cpsc.gov). The "-all" at the end means mail sent from other domains or IPs not specified here should fail. Alternatives could be "~all" which would soft-fail mail sent from other domains/IPs, or "+all" which would allow mail to be sent from any domain/IP — which would effectively render the SPF record useless.

There's a tiny optimization that CPSC.gov could make by listing one CIDR (63.74.109.0/24) rather than each of the three IPs. By doing so they'd save 15 bytes:

```
CPSC.gov.      18894   IN  TXT "v=spf1 ip4:63.74.109.0/24 mx a:list.cpsc.gov -all"
```

However, as you'll see, that's hardly worth the effort compared with other optimizations they could make.

The Bad

The Good, The Bad, and The Ugly artwork by Billy Perkins

The misconfiguration of the CPSC.gov DNS largely involves how they've setup DNSSEC. DNSSEC is the mechanism to sign DNS records and help prevent cache poisoning. Each record in the DNS zone needs to be signed. To accomplish this you need RRSIG records and a simple change in their configuration could cut the size of the DDoS attacks the CPSC is indirectly responsible for nearly in half.

The signatures for each record are included as RRSIG records. Here's an example of the RRSIG record signing CPSC.gov's SOA record (again, scroll to the left in the box to see the whole record):

```
cpsc.gov.       18894   IN  RRSIG   SOA 7 2 21600 20160824030507 20160817020507 53799
cpsc.gov. I7umZbpbUnh24bvphYP9K70imTyDFxc5QgOEDVVCSkmtVMhRaOU0eiSO
6b/O3ExcMhL0xn+D+/Xs1R7z9zPsKrzvGCfqotkxu8L1mRpAmjgr21WH
mEzXftcx+IbSrOCwOa7YuPyBGqJW3f8kb/UHcI7ykqk08xvglD2I77PV
TgI59oYsZyufRhfChcFxEyhBY2LZ4o/UO5Tw8Lg5HH/9OGtA0FHum1To
v/CeLBOaATrZNygZINKmHlsP6Mm8j83jM4bRNxLGPs8VjpDbKVAHtROq
aXR0Zy81zNs0eWfs4vThdOf56nCUnA/HiwlVNgmbO219CCMKQxoBBuY+ FYcWfA==
```

Let's walk through this RRSIG record. Just like with the previous records "18894" is the TTL in seconds, "IN" stands for "Internet Class", and RRSIG is the type of DNS record. "SOA" here specifies the type of record this signature applies to. That's easy. However, after that, things get cryptic (both literally and metaphorically).

The "7" here specifies the signature algorithm being used. By the DNSSEC specification, the 7th zone signing algorithm is: RSASHA1-NSEC3-SHA1.

Here's a list of the 10 current algorithms (represented by 14 numbers, just to be confusing) that can be used for signing public DNS records along with the number — or range of numbers if the signatures can be of variable lengths — of bytes each signature generates:

1. RSA/MD5 / 128–512 bytes (deprecated)
2. [Not used for RRSIG records]
3. DSA/SHA1 / 41 bytes
4. [Reserved]
5. RSA/SHA-1 / 128–512 bytes
6. DSA-NSEC3-SHA1 / 328 bits
7. RSASHA1-NSEC3-SHA1 / 128–512 bytes
8. RSA/SHA-256 / 128–512 bytes
9. [Reserved]
10. RSA/SHA-512 / 128–512 bytes
11. [Reserved]
12. GOST R 34.10-2001 / 64 bytes
13. ECDSA Curve P-256 with SHA-256 / 64 bytes
14. ECDSA Curve P-384 with SHA-384 / 96 bytes

Algorithm 7 allows the signer to specify a key length between 128 and 512 bytes. The CPSC specifies a key length of 256 bytes, which, today, for RSA is considered secure. Unfortunately, that means they have a 256-byte signature for every RRSIG record. Compare that with Algorithm 13 which, for a higher level of security, would produce only a 64-byte signature — one fourth the size per RRSIG record.

Returning to the RRSIG record, the "2" (after the "7") represents the number of "labels" in the domain being signed. In this case, the domain is cpsc.gov, which is 2 labels deep. If it were www.cpsc.gov then it would be 3 labels. a.b.cpsc.gov would be 4 labels.

"21600" represents the TTL that was used when the signature was originally calculated. Since the TTL could be different for each resolver's response, the TTL value needs to be set to a default to ensure that the signature is calculated consistently.

The next two numbers are timestamps. The first (20160824030507) represents the time this signature will expire (24 August 2016 at 03:05:07 UTC). The second (20160817020507) represents the time this signature became valid (17 August 2016 at 02:05:07 UTC).

"53799" is a checksum on the DNSKEY that was used to sign this record. This checksum is used to more quickly find the DNSKEY that was used to sign this particular record when more than one key is returned. (More on DNSKEYs in the next section.)

After the checksum is "cpsc.gov." which is the root domain. This root domain is included regardless of the number of subdomains (e.g., cpsc.gov, www.cpsc.gov, and a.b.cpsc.gov would all have "cpsc.gov." listed as the root domain). This specifies where to find the DNSKEY records that could have signed this record.

Next up is the signature itself. In this case, the signature is encoded in base64 and 256 bytes long. The base64 encoding means that the 256 bytes turns into 344 on the wire.

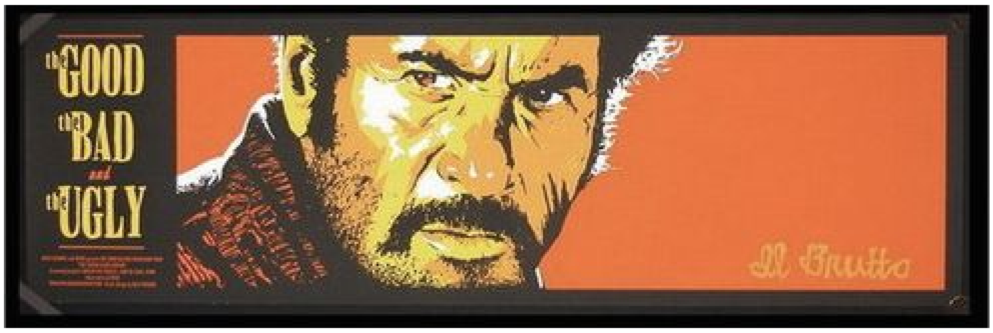
Stepping back, the easy optimization here for the RRSIG record would be to pick a better signature algorithm. By choosing Algorithm 13, rather than Algorithm 7, the CPSC could save 192 bytes per RRSIG record. We've written previously about why we use Algorithm 13 for CloudFlare's DNSSEC implementation.

Doing so here would add up because there are ten RRSIG records in the CPSC's zone in order to sign each of the other records (e.g., A, AAAA, NS, MX, etc...).

Just by switching to a better encryption algorithm the CPSC could increase the security of their DNS zone and save 1,920 bytes per DNS response. That small change alone would cut the maximum possible DNS reflection attack possible with the CPSC's zone by 44 percent.

While that's the biggest savings, choosing a less optimal outcome can't be considered an actual mistake. Unfortunately, the CPSC's zone has some ugly mistakes as well.

The Ugly



The Good, The Bad, and The Ugly artwork by Billy Perkins

The RRSIG records discussed in the previous section need to be verified with a public key. These public keys are stored as DNSKEY records. For the CPSC, there are four DNSKEYs that each look like this:

```
cpsc.gov.       18894   IN  DNSKEY  256 3 7
AwEAAbpeSszphwwkOIJn1ha6DE/W3YRXFR2vsMi0RKhq5x9t487UJc0c
eamz5TZj6KV5/tzL8/Qr2jnTaQmpWtJHbnF0kqpxeZIR+wzaNbMtEH30
UF5BDv9Bya0W9I+40dS48996kedhEvL6KwmMelB7FH6QPd0ixyhp0+ci
5vew9lzTESEsJ2X2uJrCqo3UacsHyYIzaTSXPpfwizQCql4VySq6+im1
74QaYw/FU4aADAv3R2KQvsR/uI0a7oOihxDDAvtYG7SZvotW3ASZFscd
4B6Yd84RMZC3yGdGtyrSD6tZsiJZoXhLQkkf0kOTWCjPvD8oPm+yYiGI Cm64eM3kflU=
```

Walking through the record they're similarly formatted to RRSIG records. Once again, 18894 is the TTL on the record, the record type is DNSKEY and that's followed by three numbers: 256, 3 and 7. These are the flags field, protocol, and algorithm.

The flags field is 16-bits wide representing a number of potential boolean flags. Only two bits of this 16-bit field are currently assigned for use with the rest reserved for the future. A value of 256 (or 0000000010000000 or only the 9th flag set to true) indicates that this DNSKEY holds a key that can be used to verify RRSIGs. The protocol requires a 16-bit number so there's no optimization possible here.

The protocol field is 3. That is the only valid value for this field in a DNSKEY record. As the RFC says:

```
The Protocol Field MUST have value 3, and the DNSKEY RR MUST be
treated as invalid during signature verification if it is found
to be some value other than 3.
```

Again, that seems a bit arbitrary but it's what the protocol requires so there's no optimization.

Finally, the algorithm field is 7 which indicates RSASHA1-NSEC3-SHA1. So, it should be an RSA public key formatted according to RFC3110.

Passing it through a base64 decoder and hexdump we can take a look inside:

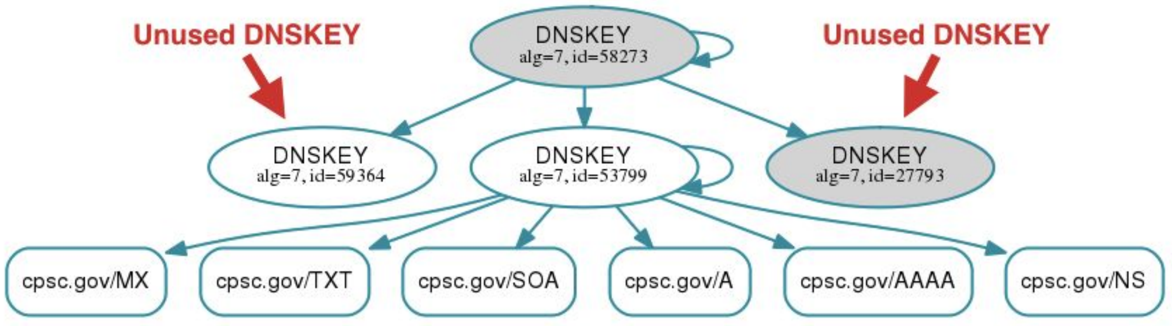
```
00000000  03 01 00 01 ba 5e 4a cc  e9 87 0c 24 38 82 67 d6  |.....^J....$8.g.|
00000010  16 ba 0c 4f d6 dd 84 57  15 1d af b0 c8 b4 44 a8  |...O...W......D.|
00000020  6a e7 1f 6d e3 ce d4 25  cd 1c 79 a9 b3 e5 36 63  |j..m...%..y...6c|
00000030  e8 a5 79 fe dc cb f3 f4  2b da 39 d3 69 09 a9 5a  |..y.....+.9.i..Z|
00000040  d2 47 6e 71 74 92 aa 71  79 92 11 fb 0c da 35 b3  |.Gnqt..qy.....5.|
00000050  2d 10 7d f4 50 5e 41 0e  ff 41 c9 ad 16 f4 8f b8  |-.}.P^A..A......|
00000060  d1 d4 b8 f3 df 7a 91 e7  61 12 f2 fa 2b 09 8c 7a  |.....z..a...+..z|
00000070  50 7b 14 7e 90 3d dd 22  c7 28 69 d3 e7 22 e6 f7  |P{.~.=.".(i.."..|
00000080  b0 f6 5c d3 11 21 2c 27  65 f6 b8 9a c2 aa 8d d4  |..\..!,'e.......|
00000090  69 cb 07 c9 82 33 69 34  97 3e 97 f0 8b 34 02 aa  |i....3i4.>...4..|
000000a0  5e 15 c9 2a ba fa 29 b5  ef 84 1a 63 0f c5 53 86  |^..*..)....c..S.|
000000b0  80 0c 0b f7 47 62 90 be  c4 7f b8 8d 1a ee 83 a2  |....Gb..........|
000000c0  87 10 c3 02 fb 58 1b b4  99 be 8b 56 dc 04 99 16  |.....X.....V....|
000000d0  c7 1d e0 1e 98 77 ce 11  31 90 b7 c8 67 46 b7 2a  |.....w..1...gF.*|
000000e0  d2 0f ab 59 b2 22 59 a1  78 4b 42 49 1f d2 43 93  |...Y."Y.xKBI..C.|
000000f0  58 28 cf bc 3f 28 3e 6f  b2 62 21 88 0a 6e b8 78  |X(..?(>o.b!..n.x|
00000100  cd e4 7e 55                                       |..~U|
```

Notice that the first four bytes are 03 01 00 01. That indicates an exponent of 65537 (03 means three bytes of exponent, 010001 in hex is 65537) which is very common for the RSA algorithm. The rest of the record consists of 256 bytes (2,048 bits). These 2,048 bits are the modulus part of the RSA key. Thus this is a 2,048 bit key.

Like with RRSIG, the CPSC could have better security and a shorter record if they specified a more efficient algorithm for their DNSKEY records. Using Algorithm 13 would save 192 bytes per DNSKEY, or 768 bytes across the 4 records.

But there's the rub: the CPSC doesn't need to have 4 DNSKEY records at all. Except during a key rollover, there should generally only be 2 DNSKEY records. In the CPSC's case, one of the DNSKEYs (id 59364) isn't signing anything. Another of the DNSKEYs (id 27793) is redundant. In other words, the CPSC includes 552 bytes of completely worthless data in every DNS record. Just eliminating these two useless records would cut the maximum size of a DNS Amplification DDoS attack using the CPSC's zone by almost 13 percent.

You can visualize the chain of how DNSSEC records are used to sign each other by using a tool provided by DNSVIZ. The diagram below shows how the two unused DNSKEYs don't actually sign any records.



A Better CPSC Zone

There are a handful of other minor optimizations. Add them all up and the CPSC Zone could be much smaller. At CloudFlare, we've worked to optimize the size and performance of our zone records. We used our automated zone generator on the CPSC and got the following zone file:

```
cpsc.gov.       281 IN  A   104.27.142.66
cpsc.gov.       281 IN  A   104.27.143.66
cpsc.gov.       281 IN  RRSIG   A 13 2 300 20160402121755 20160331101755 35273 cpsc.gov.
xYJVr9iNMm50wgoGbMaG76QNyZk2aU2STmzxlUGe09G9LyEqnlDW6YLK
pyAeMqHODKuxt83dKDUeB0bgvSPCyQ==
cpsc.gov.       278 IN  AAAA    2400:cb00:2048:1::681b:8e42
cpsc.gov.       278 IN  AAAA    2400:cb00:2048:1::681b:8f42
cpsc.gov.       278 IN  RRSIG   AAAA 13 2 300 20160402121752 20160331101752 35273
cpsc.gov. LHmzdf6AqsrBVT5ejyoRt2oruXuaHkavhvoNHiqr2OXALnquouOfHsdq
qRlHgQ1mkWRbeJ3nwIwDnWbInbIhkg==
cpsc.gov.       272 IN  MX  5 stagg.cpsc.gov.
cpsc.gov.       272 IN  MX  5 hormel.cpsc.gov.
cpsc.gov.       272 IN  RRSIG   MX 13 2 300 20160402121746 20160331101746 35273
cpsc.gov. jmAvZk3bMxcqBECvvnhC7Xn2gtdvJnrRqj3xypzQ4Kona3tzs8H5+IBp
a9X+TbcIGs1zgIny9UjybVUws+YTqw==
cpsc.gov.       86391   IN  SOA abby.ns.cloudflare.com. dns.cloudflare.com. 2021076894 10000
2400 604800 3600
cpsc.gov.       86391   IN  RRSIG   SOA 13 2 86400 20160402121805 20160331101805 35273
cpsc.gov. SrUvxJjlRvbAhFUXlEF6pbilXKLN/gzYcsyi4yYRR5o89k73nMv5mYZr
vIi+uaPZe8Qht6nuNWRHhrYxQiUpLw==
cpsc.gov.       86367   IN  NS  lars.ns.cloudflare.com.
cpsc.gov.       86367   IN  NS  abby.ns.cloudflare.com.
cpsc.gov.       86367   IN  RRSIG   NS 13 2 86400 20160402121741 20160331101741 35273
cpsc.gov. QO3FN7XVRMTewKuJGlbS1AglYnQrsbEfiWS65Mp+wT3D6n973Be9XzcR
nvel4fPLvQTcIe/h5kpbXjut3nfhFQ==
cpsc.gov.       3505    IN  DNSKEY  257 3 13
mdsswUyr3DPW132mOi8V9xESWE8jTo0dxCjjnopKl+GqJxpVXckHAeF+
KkxLbxILfDLUT0rAK9iUzy1L53eKGQ==
cpsc.gov.       3505    IN  DNSKEY  256 3 13
koPbw9wmYZ7ggcjnQ6ayHyhHaDNMYELKTqT+qRGrZpWSccr/lBcrm10Z
1PuQHB3Azhii+sb0PYFkH1ruxLhe5g==
cpsc.gov.       3505    IN  RRSIG   DNSKEY 13 2 3600 20160426183927 20160226183927 2371
cpsc.gov. o0c75Fa7KwogAAEpYIoWuTcsV1fPQ4sfuER+WOtxkOPb0Fe8PdF2yBWa
3gRLA/N2pmdCVrAAeIqdR0NDfX6PCw==
cpsc.gov.       47  IN  TXT "v=spf1 ip4:63.74.109.6 ip4:63.74.109.10 ip4:63.74.109.20 mx
a:list.cpsc.gov -all"
cpsc.gov.       47  IN  RRSIG   TXT 13 2 300 20160402121801 20160331101801 35273
cpsc.gov. LXjZYeejRtkCFOVqqUpupc3I/4s7Ypua/b/xE3wIC+7a4NRjM7+UA90s
gDdINQ8ZPF3SLcCqw2nfc3ex2II8Kg==
cpsc.gov.       3324    IN  NSEC    \000.cpsc.gov. A NS SOA WKS HINFO MX TXT AAAA LOC SRV
CERT SSHFP IPSECKEY RRSIG NSEC DNSKEY TLSA HIP CDS CDNSKEY OPENPGPKEY SPF
cpsc.gov.       3324    IN  RRSIG   NSEC 13 2 3600 20160402121810 20160331101810 35273
cpsc.gov. 3/1I/SQrwvdffqawg6qG3zfetA1ou5SQioVv0kzMmHbYWJooQv9QEEFX
L9RwdWxly0Hly/Aq+UMWzcQ8D2UXZg==
```

That is 1,389 bytes and close to as optimized as you could manually create. The actual response may be a bit larger because resolvers can add a 320-byte DS record. Even so, the optimized CPSC zone would be about one third the size without compromising any functionality and actually increasing security.

But we can do better than that. Unlike the CPSC's current DNS provider (Verizon/UU.net), CloudFlare has anti-DNS reflection protections in place. Specifically, we automatically upgrade from UDP to TCP when a DNS response is particularly large (generally, over 512 bytes). Since TCP requires a handshake, it prevents source IP address spoofing which is necessary for a DNS amplification attack.

In addition, we rate limit unknown resolvers. Again, this helps ensure that our infrastructure can't be abused to amplify attacks.

Finally, across our DNS infrastructure we have deprecated ANY queries and have proposed to the IETF to restrict ANY queries to only authorized parties. By neutering ANY, we've significantly reduced the maximum size of responses even for zone files that need to be large due to a large number of records.

Denouement

DNSSEC is complicated and it's easy to get wrong. Unfortunately, getting your DNSSEC configuration wrong creates a real potential harm to the rest of the Internet by making your domain's zone file into a potential weapon to be abused by attackers. That's why, in CloudFlare's implementation, we spent a significant amount of time to optimize our automatic DNSSEC configuration to be as efficient as possible while still being easy and free for all our customers.

As Clint Eastwood says in the final scene of the movie "The Good, the Bad, and the Ugly": "You see in this world there's two kinds of people, my friend, those with loaded guns and those who dig. You dig." Pardon the puns but... we hope the CPSC will fix their zone and unload the giant gun attackers have used to launch some of the biggest DDoS attacks on the Internet. When they do, it'll be a relief to "dig" their domain and get back a reasonably sized answer.

If the instructions above aren't enough on their own, we'd be happy to work with the CPSC to help them get their DNS record under control and ensure they are no longer inadvertently facilitating the Internet's largest DDoS attacks. Of course, when the CPSC does get its DNS issues fixed, undoubtedly attackers will find another misconfigured and bloated zone they can abuse to launch DNS amplification DDoS attacks. And, when that happens, we'll be happy to work with them too — until the whole Internet has the well-configured DNS foundation it needs to be safe and secure.