14 | 安全：用Wireshark把DDoS攻击照出原形

2022-02-21 杨胜辉

《网络排查案例课》 课程介绍 >

讲述：杨胜辉
时长 24:43 大小 22.64M

你好，我是胜辉。

在过去几节课里，我们集中学习了 TCP 传输相关的知识，无论是第 8 讲的 MTU、第 9 讲和第 10 讲的传输速度、第 11 讲的拥塞控制，还是第 12 和 13 讲的各种重传，我们可以说是把 TCP 传输相关的核心概念全部过了一遍。一方面学习了 RFC 规范和具体的 Linux 实现，一方面也通过案例，把这些知识灵活运用了起来。想必现在的你，再去处理 TCP 传输问题的时候，已经强大很多了。

不过，上面的种种，其实还是在协议规范这个大框架内的讨论和学习，默认前提就是通信两端是遵照了 TCP 规定而展开工作的，都可谓是谦谦君子，道德楷模。

但是，有明就有暗。不遵照 TCP 规范，甚至寻找漏洞、发起攻击，这种“小人”乃至“强盗”的行为，也并非少见，比如我们熟知的 **DDoS 攻击**。

那么这节课，我们就不来学习怎么做君子了，当然，也不是教你做“小人”，而是我们要**了解“小人”可能有的各种伎俩，通过 Wireshark 把这种种攻击行为照个彻底，认个清楚**。这样，以后你如果遇到这类情况，就心里有数，也有对策了。

NTP 反射攻击案例

我在公有云做技术工作的时候，发现游戏类业务遭到 DDoS 的情况比较多见。有一次，一个游戏客户就发现，他们的游戏服务器无法登录，被玩家投诉，可以说是十万火急。客户的工程师做了 tcpdump 抓包，然后赶紧把抓包文件传给我们一起分析。

> 补充：抓包示例文件已经上传至Gitee，建议 Wireshark 打开抓包文件，结合文稿一起学习。

从抓包文件概览来看，确实是不正常。按惯例，我们看一下 Expert Information：

Severity	Summary	Group	Protocol	Count
Warning	Connection reset (RST)	Sequence	TCP	1
Note	This frame is a (suspected) retransmission	Sequence	TCP	4
Chat	Connection establish request (SYN): server port 443	Sequence	TCP	10
Chat	Connection finish (FIN)	Sequence	TCP	2

游戏业务一般也是基于 TCP，但这里并没多少 TCP 相关的信息。那我们看下具体报文呢？

No.	Time	Source	Destination	SrcPort	DstPort	TTL	TCP Seglen	Info
1	0.000000	c-75-69-167-78…	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1
2	0.000011	c-75-69-167-78…	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1
3	0.000011	22.195.150.241	30.189.233.153			57		NTP Version 2, private, Response, MON_GETLIST_1
4	0.000011	234.213.237.115	30.189.233.153			43		NTP Version 2, private, Response, MON_GETLIST_1
5	0.000010	22.195.150.241	30.189.233.153			57		NTP Version 2, private, Response, MON_GETLIST_1
6	0.000011	210-71-233-86.…	30.189.233.153			50		NTP Version 2, private, Response, MON_GETLIST_1
7	0.000010	43.117.219.15	30.189.233.153			53		NTP Version 2, private, Response, MON_GETLIST_1
8	0.000011	218.92.107.50	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1
9	0.000010	c-75-69-167-78…	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1
10	0.000020	87.69.239.177	30.189.233.153			47		NTP Version 2, private, Response, MON_GETLIST_1
11	0.000011	255.111.107.14	30.189.233.153			44		NTP Version 2, private, Response, MON_GETLIST_1
12	0.000010	218.92.47.146	30.189.233.153			55		NTP Version 2, private, Response, MON_GETLIST_1
13	0.000011	c-75-69-167-78…	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1
14	0.000010	c-75-69-167-78…	30.189.233.153			51		NTP Version 2, private, Response, MON_GETLIST_1

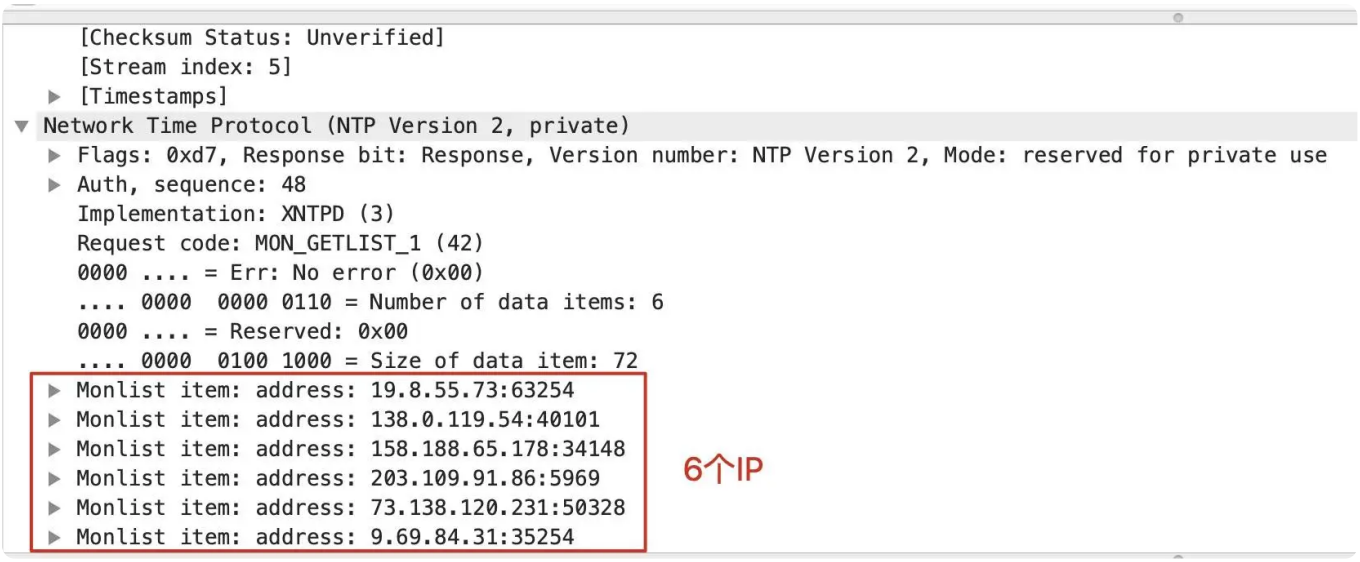
```
▶ Frame 51: 486 bytes on wire (3888 bits), 486 bytes captured (3888 bits)
▶ Ethernet II, Src: 00:aa:bb:cc:dd:ee, Dst: 00:11:22:33:44:55
▶ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 1940
▼ Internet Protocol Version 4, Src: 26.109.249.118 (26.109.249.118), Dst: 30.189.233.153 (30.189.233.153)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 468
    Identification: 0x0000 (0)
  ▼ Flags: 0x4000, Don't fragment
      0... .... .... .... = Reserved bit: Not set
      .1.. .... .... .... = Don't fragment: Set
      ..0. .... .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 49
    Protocol: UDP (17)
```

全部都是这种 NTP Version 2, Private, Response, MON_GETLIST_1 的报文。客户根本没有对外提供什么 NTP 服务，这些报文是怎么来的呢，这是攻击吗？

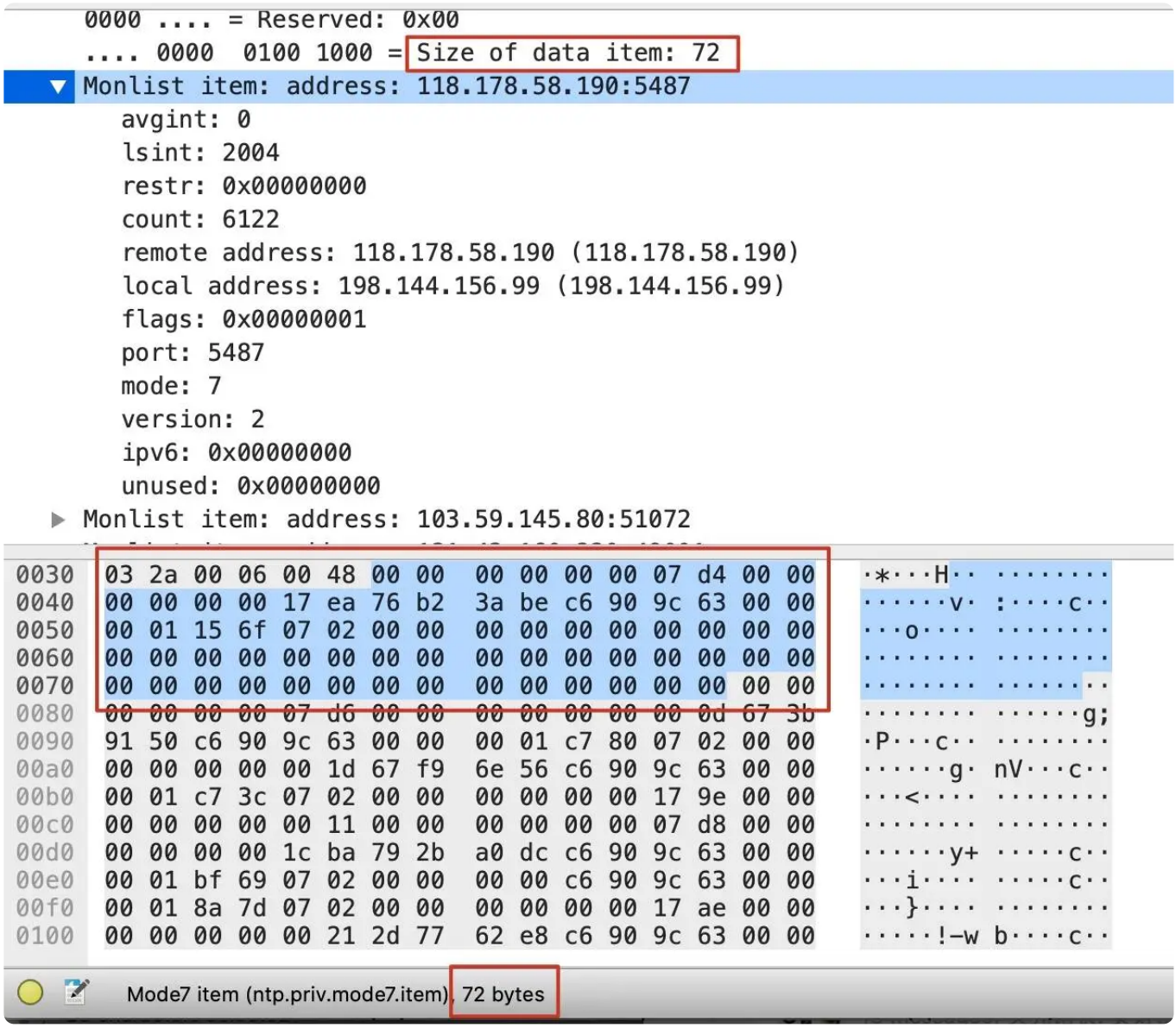
确实，这就是 DDoS 攻击的一种类型，叫 **NTP 反射放大攻击**。NTP 全称是 Network Time Protocol，它的作用是通过网络服务来同步时间。其中有一项功能叫 Monlist，它在一些比较老的设备上是默认启用的，会返回与 NTP 服务器进行过时间同步的最后 600 个客户端的 IP。

响应包按照每 6 个 IP 进行分割，最多有 100 个响应包。这样的话，一个简单的 NTP Monlist 响应，就可能是请求的 200 多倍。想象一下，如果请求是 1Gbps，那这次反射攻击就可以达到 200Gbps 以上，实在惊人！

我们还是用 Wireshark，展开 UDP 部分，看一看这种 Monlist 响应报文的细节：



每个 Monlist item 占用 72 字节：



选中一个 Monlist item 后，我们可以通过 3 种不同的方式找到它的长度：

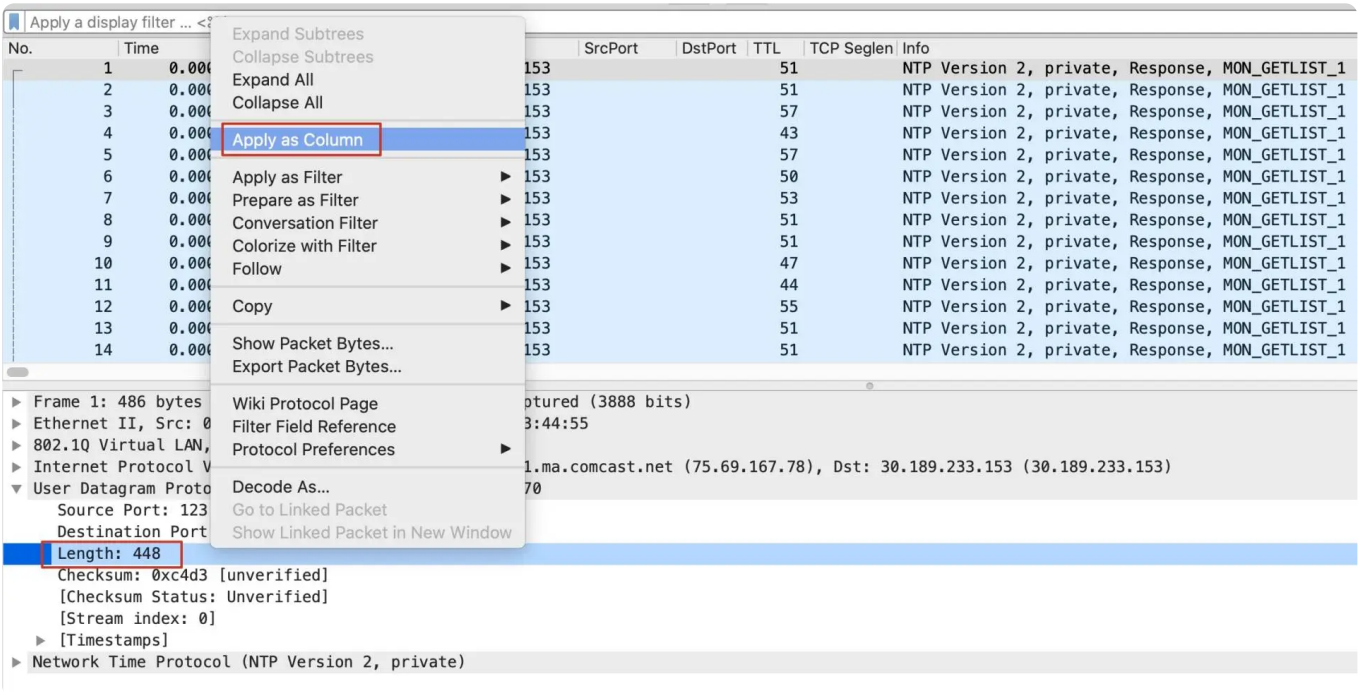
它有个字段 Size of data item 的值是 72，这是协议本身提供的元数据；

在下方的字节码部分，数一下有底色的字节数，也是 72 个；

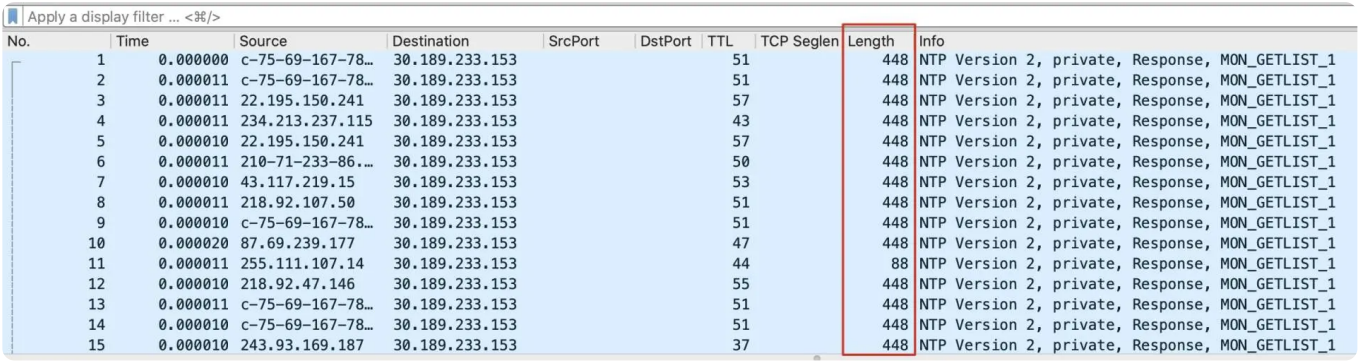
窗口底部对我们选中的字段有提示字节数，这里也是 72 bytes。

这些方法，对于你平时用 Wireshark 解读抓包文件，特别是需要核对字段的具体信息时，是挺有用的。

不过，现在 Wireshark 窗口里解读 UDP 报文长度不是很直观，这是因为 **Wireshark 默认没有显示 UDP 长度的列，但我们可以自己添加**。在 UDP 详情里选中 Length，然后右单击，选中 Aplly as Column：



然后就能在主窗口里看到 UDP 报文长度了，这个列的默认名称是 Length。当然你也可以把它改为 UDP Length 等你觉得更合适的名称。



从图上看，这些 UDP 报文的长度都不大，只有 448 字节，这是为什么呢？我们知道 MTU 一般是 1500 字节，去掉 IP 头 20 字节和 UDP 头 8 字节，最多还有 1500-20-8=1472 字节，远大于 448 字节，为什么不用足这 1472 字节呢？

其实，这里就涉及 UDP 的一个概念：**UDP 报文的载荷最好不要大于 512 字节。**

这个限制来自于 IPv4 协议。在 IPv4 的协议规范RFC791里建议，虽然 IP 报文的长度字段是 2 个字节，最大可以到 65535，但是由于网络不允许传输这么大的报文，所以 IPv4 规范建议，报文长度应该控制在相对小的范围内，这个范围是 576 字节，相应的 UDP 载荷在 512 字节以内：

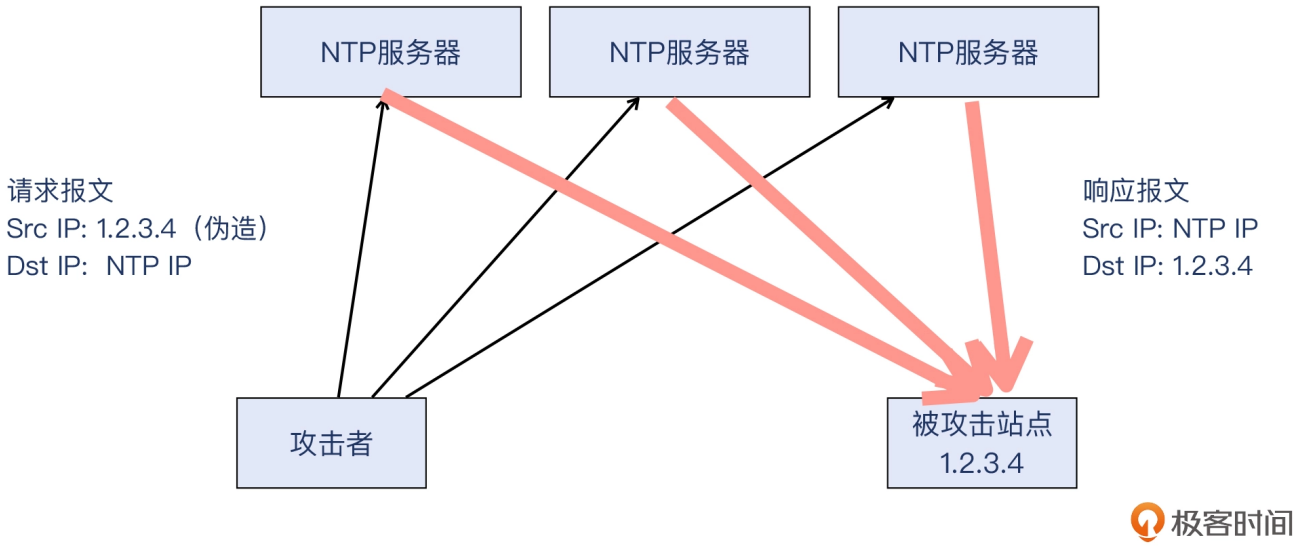
复制代码

```
The number 576 is selected to allow a reasonable sized data block to
    be transmitted in addition to the required header information.  For
    example, this size allows a data block of 512 octets plus 64 header
    octets to fit in a datagram.  The maximal internet header is 60
    octets, and a typical internet header is 20 octets, allowing a
    margin for headers of higher level protocols.
```

很多应用程序都做了这部分逻辑的处理，也就是控制 UDP 载荷在 512 字节以内，比如这次的 NTP Monlist 的长度就是 NTP 协议实现，而不是内核 UDP 实现的。另外像 DNS 解析，如果数据量超过 512 字节，也是会自动切换为 TCP 模式的，根本原因也是这个很早以前的规定。

那么检查了载荷，我们很快会发现新的问题：“这里怎么只有 NTP 回复，没有 NTP 请求呢？”

其实，这正是前面说的 1Gbps 能放大为 200 多 Gbps 的原因。它的背后，就是反射攻击的核心技巧：**它利用 IP 协议“不对源 IP 做验证”的不足，构造一个 IP 报文，其源 IP 为被攻击站点的 IP，使得 NTP 服务器回复的报文也被发往被攻击站点**。大量的响应报文就被引到了被攻击站点这里。而且这个过程中，NTP 服务器被利用了还不知道。我们看个示意图：



如果我们面临这种攻击，该怎么办呢？

假如这些被利用的 NTP 服务器是我们的，那么需要升级版本，避免自己成为“帮凶”。

如果我们是单纯的被攻击者，那就需要上一些手段了，我会在这次课程的后半段讲到。这次的游戏客户，就是上了高防后，扛住了这次攻击。

我们再来看一个例子。

SSDP 反射型攻击案例

你可能听说过“肉机”这个词，这也是国内技术圈发明的一个有意思的词汇，它指的是被黑客掌握了系统权限的主机。作为傀儡，这些成千上万的“肉机”可以被黑客集中调动起来发起攻击行为。假如一个黑客组织掌握了 1 万台“肉机”，那么，只要每台“肉机”发起哪怕只有 1Mbps 的攻击流量，乘以 1 万，就是 10 个 Gbps 的流量，不可小视。

但是，要拿到这么多“肉机”却并非易事。于是，聪明的黑客又想到了另外一种思路：借力打力。

借什么力呢？借协议的“**响应是请求的很多倍**”这个力。显然，前面介绍的 NTP 反射攻击就是这样的。所以这种攻击，英文里叫 reflection attack with amplification。amplication 就是放大的意思。在这种模式下，只需要量很少的“肉机”，就可以发起巨大的攻击流量。

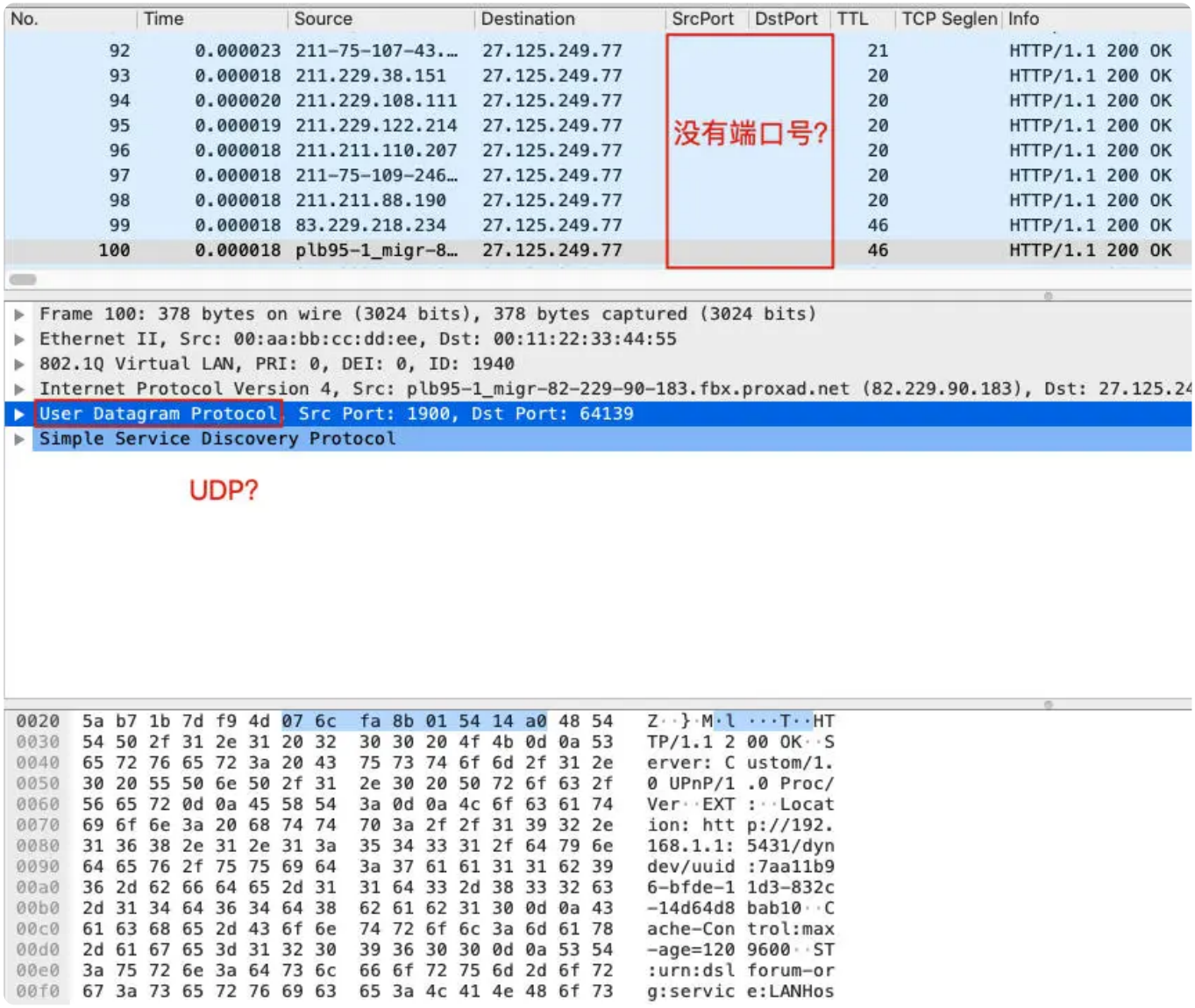
下面是另外一个客户的案例。当时他们遭受了一波攻击，也正好做了抓包。我们看一下抓包文件的 Expert Information：

Severity	Summary	Group	Protocol	Count
Error	Malformed Packet (Exception occurred)	Malformed	IAPP	1
Warning	Bad checksum [should be 0x7eb1]	Checksum	ICMP	1
Chat	HTTP/1.1 200 OK\r\n	Sequence	SSDP	51257

> 补充：抓包示例文件已经上传至Gitee，建议 Wireshark 打开抓包文件，结合文稿一起学习。

你是不是也觉得比较奇怪？这里没有 TCP 握手报文，上来就是 HTTP/1.1 200 OK，这 HTTP 有点“自来熟”啊。既然这里有 51257 个 HTTP 200 响应报文，那按常理也至少有几十个 TCP 连接，而这里连一个 SYN 和 FIN 都没有。

那就让我们直接看看这些 HTTP 200 具体是怎样的：



奇怪，这里 srcPort 和 dstPort 居然都是空白的？不过视线移到下方，很快就找到答案了：原来是用了 UDP 协议。我们上面的 srcPort 和 dstPort 列是指 TCP 的端口号，难怪是空白。

但更奇怪的问题来了：这里的 HTTP 竟然用了 UDP 作为传输协议？HTTP 一般是用 TCP 协议的，那这里用 UDP 又是怎么回事呢？是不是感觉网络协议到处都可能有意想不到的情况？

其实，这就是有名的 **SSDP 反射放大攻击**。SSDP 是在 UDP 这个传输层协议上，用 HTTP 协议格式传送信息。2014 年，人们发现 SSDP 可以被攻击者利用。启用了 SSDP 协议的主要是一些家用路由器，在它们的 UPnP 软件中有一个漏洞，这个漏洞被攻击者利用后，这些路由器会从端口 1900 返回响应报文。那么显然，这些响应报文的目的地址，是被攻击站点的 IP，而不是攻击发起者自己的 IP 了。

具体的攻击过程，跟前面 NTP 反射攻击里面的图差不多，这里就不重复了。当时的应对方法也是上了高防系统，顶住了这次攻击。

> 补充：有趣的是，著名的网络服务公司 Cloudflare 把 SSDP 戏称为 **S**tupidly **S**imple **DDoS P**rotocol。你可以在这里看到 Cloudflare 对 SSDP 攻击的更多解释。

如果你也担心自己家的路由器也中招了的话，可以自测一下。访问 https://badupnp.benjojo.co.uk/这个站点，它会对你的出口 IP（家用路由器的出口 IP）进行探测，看看是否有 1900 端口可以被利用。如果没有漏洞，网页会提醒你“All good! It looks like you are not listening on UPnP on WAN”。

了解了这次攻击的类型，接下来我们再来看一下这次抓包的概览。这里我们要学习一个新的命令：**capinfos**。

capinfos 这个命令，是 Wireshark 自带的工具集中的一个小工具，也就是你安装完 Wireshark 就有它了。我一般在命令行里用它来查看抓包文件的时长、总包量等信息。我们直接运行 **capinfos 文件名**，输出如下：

复制代码

```
$ capinfos SSDP_attack_example.pcap
File name:           SSDP_attack_example.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 65535 bytes
Number of packets:   100 k
File size:           36 MB
Data size:           34 MB
Capture duration:    1.916902 seconds
First packet time:   2016-05-08 10:25:02.721642
Last packet time:    2016-05-08 10:25:04.638544
Data byte rate:      18 MBps
Data bit rate:       145 Mbps
Average packet size: 348.38 bytes
Average packet rate: 52 kpackets/s
SHA256:              8fa365f01c62023576623116410a6ca289915db4717e4805120009a57
RIPEMD160:           5ab83442a5178b5f34bee1009a58ddb351bd292b
SHA1:                52a2c52644bc2753f9a11b7bf71eb1144f2c9a30
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                     Encapsulation = Ethernet (1 - ether)
```

```
Capture length = 65535
Time precision = microseconds (6)
Time ticks per second = 1000000
Number of stat entries = 0
Number of packets = 100000
```

上面的信息比较多，我们可以重点关注下面这几个信息：

复制代码

```
Number of packets:   100 k
Capture duration:    1.916902 seconds
Average packet rate: 52 kpackets/s
```

可以看到，这次抓包一共抓取了 10 万个报文，耗时只有 1.9 秒，平均包率为 52kpackets/s，也就是每秒 5 万 2 千个包。而且都是 SSDP 协议响应报文，所以是 DDoS 没错。

在平时，你想了解一个抓包文件的包率、时长、平均报文大小等信息的时候，都可以用 capinfos 命令来快速获得。

回顾完两个具体的案例，□想必你对于 DDoS 有了感性的认识了，接下来我们就来系统地认识一下 DDoS。

到底什么是 DDoS 攻击？

DDoS 跟 DOS 有着密切的联系。DOS（Denial of Service），就是服务拒绝，黑客通过各种手段，使得被攻击者无法正常提供服务。DOS 这种攻击早已经存在了，而 DDoS（Distributed DOS）就是它的升级版，**通过调动分布在各地的客户端发起攻击，使得被攻击站点无法正常服务**。

DDoS 属于“攻击”，但不是“入侵”。两者的区别是，攻击是破坏服务，入侵可能不破坏，但会窃取资料、劫持勒索等等。既然 DDoS 要破坏服务，那就需要破坏计算资源。那么，什么又是资源？

一般说的计算机资源还是 CPU、内存这些。旨在耗尽 CPU 和内存资源，这也是早期的攻击形式，也跟很多年前软件病毒肆虐的时候类似。当时的攻击和病毒，主要目的是让被攻

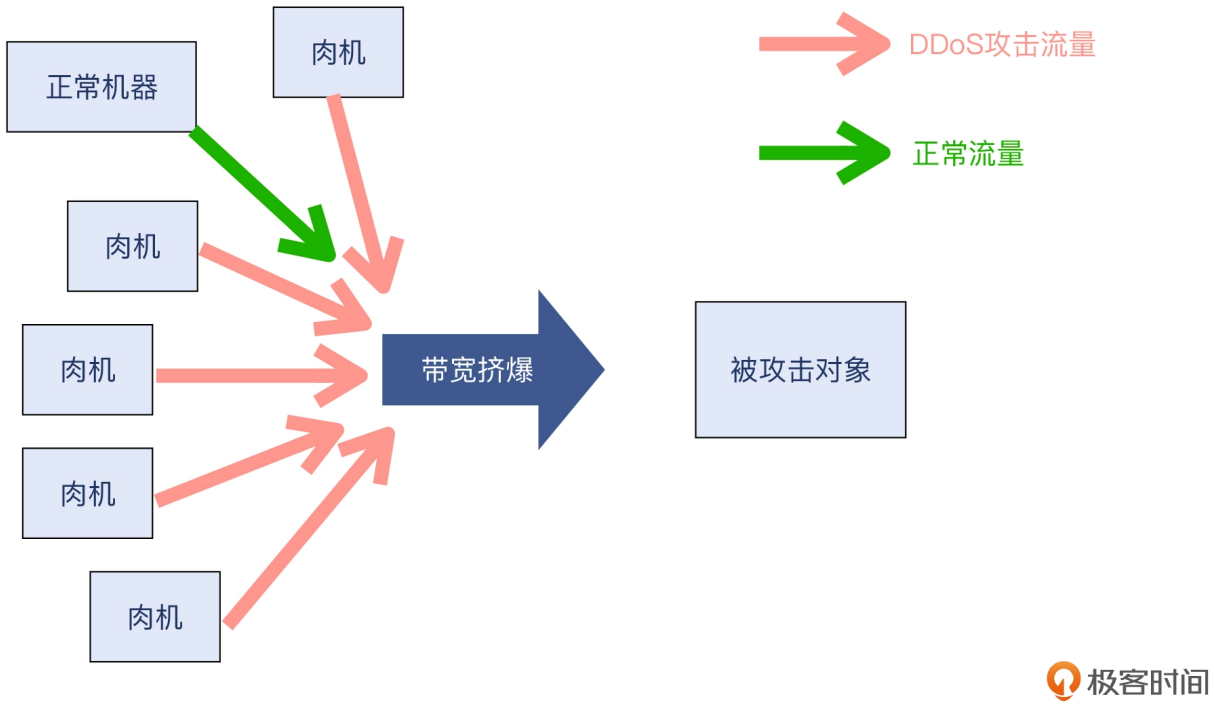
击站点本身失去服务能力。另外，早期的黑客大多也是极客，攻击是他们展现技术能力的一种方式。

不过，随着安全加固技术和意识的不断增强，攻破系统的成本越来越高，于是攻击者转换了方向。其实他们不需要想办法攻入对端，只要在前面的网络环节上搞破坏，同样可以达到让对方服务瘫痪的目的。这时，服务瘫痪的原因已经不是之前的服务本身不可用，而是变成：网络通道不可用了！

而这，就是 DDoS 的核心目标：**耗尽网络带宽**。

假设被攻击的站点的带宽为 1Gbps，那么攻击者只要让到达这个站点的流量超过 1Gbps，就可以让这个站点失去正常服务的能力。至于这些报文是否属于被攻击站点正在监听的有效流量，是没有关系的。它的目的很直接：把你家门口的路给堵死，让正常的流量没有机会进来。

我们看一下示意图：



理解了原理，那么技术性问题就来了：如何产生巨大的流量呢？

一种常见的实现方式就是反射型攻击。它的核心方法论是：利用一些协议的“**响应是请求的很多倍**”这样的特点，同时也利用“**IP 协议不验证源 IP**”的不足，达到把流量引到被攻

击站点上去的目的。

上面的 NTP 反射攻击和 SSDP 都是如此。除此以外，你有没有发现别的这种“响应报文是请求报文的很多倍”的情况呢？如果有，那么恭喜你，你也能找到反射攻击的方式了！

这可真的是“举一反三”。明白了反射型攻击的原理，你是不是好像也有机会自己创造出新的 DDoS 攻击手段了。当然，还有很多别的事情要搞定，但是核心思路你已经清楚了。现在的你，是不是对 DDoS 有了更加深刻的认识了呢？

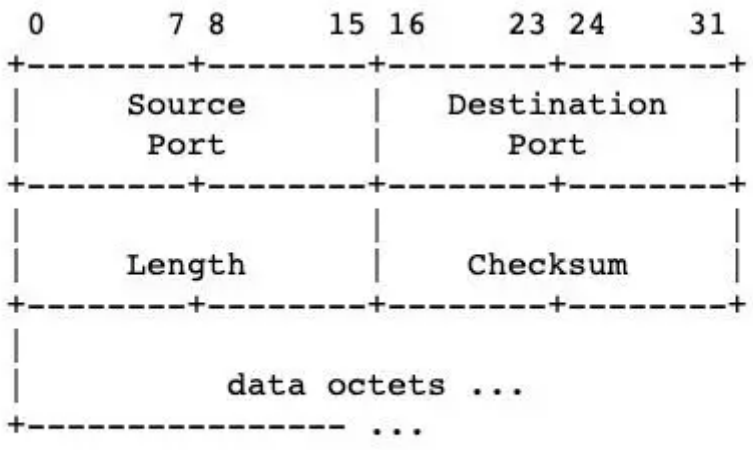
这里还有个细节。你有没有发现，前面介绍的 NTP 反射攻击是依托于 UDP 协议的，其他很多 DDoS 类型也是利用了 UDP。为什么都是 UDP 呢？让我们再来回答一下这个问题。

为什么 UDP 容易被用来做 DDoS 攻击？

TCP 当然也可能被 DDoS 所用，但是相对来说，如果用同样的成本，选择反射型攻击更加高效。而反射型攻击，主要基于 UDP，这是为什么呢？主要有两个原因。

UDP 报文简单易于构造

我们看一下 UDP 头部。RFC768定义了 UDP 头部的格式：



User Datagram Header Format

由上图可见，UDP 头部其实只有 8 个字节，分别是：

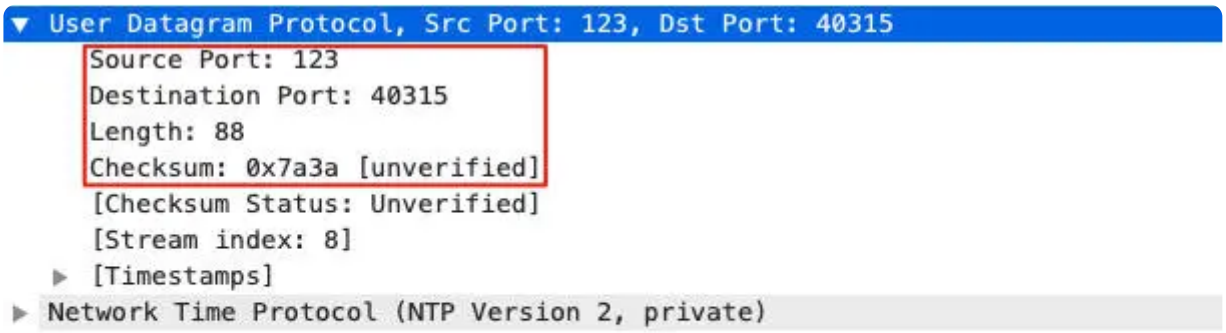
2 个字节的源端口号；

2 个字节的目的端口号；

2 个字节的报文长度；

2 个字节的校验和。

还是借助 Wireshark，我们更加近距离地看一个 UDP 报文：



而 TCP 头部就复杂多了，除了源目端口，还有序列号、确认号、各种标志位、各种 TCP 扩展选项等等。而因为 UDP 报文头部如此简单，这就减少了攻击者做伪造的难度，只要做好这几件事就好了：

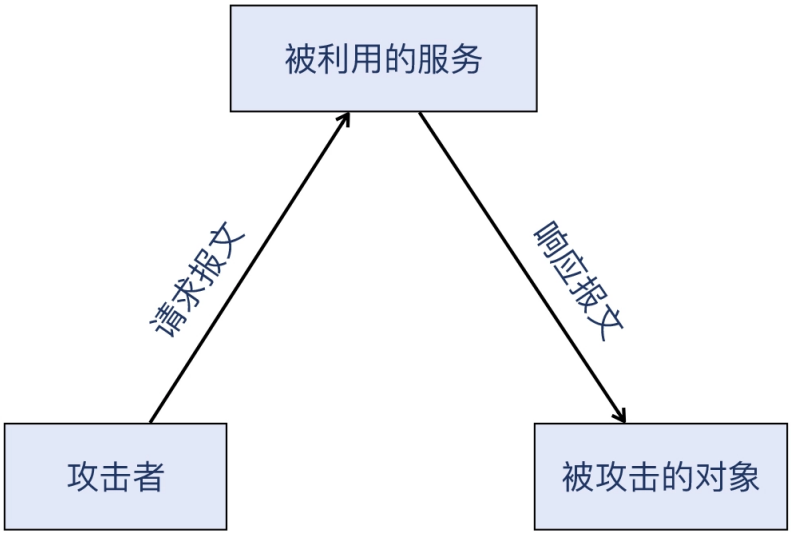
伪造一个源 IP；

找到 NTP 等有反射攻击漏洞的服务器；

向这些服务器发送构造好的虚假的 UDP 报文。

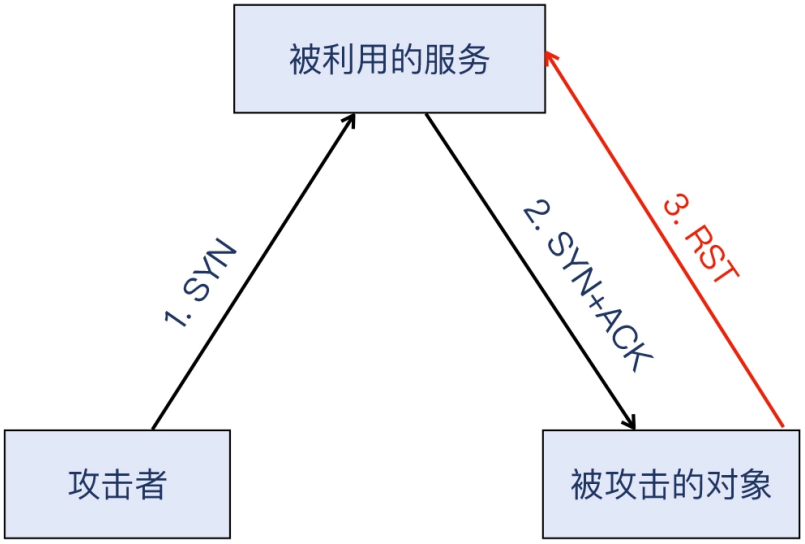
UDP 是无状态的

这可能是一个**更加关键**的原因。UDP 是无状态的，不需要握手。像 NTP 反射攻击、SSDP 反射攻击，都是只要“一问一答”即可，所以攻击者只需要伪造一个请求报文，那么后续的响应报文，自然就发送给了被攻击站点了。

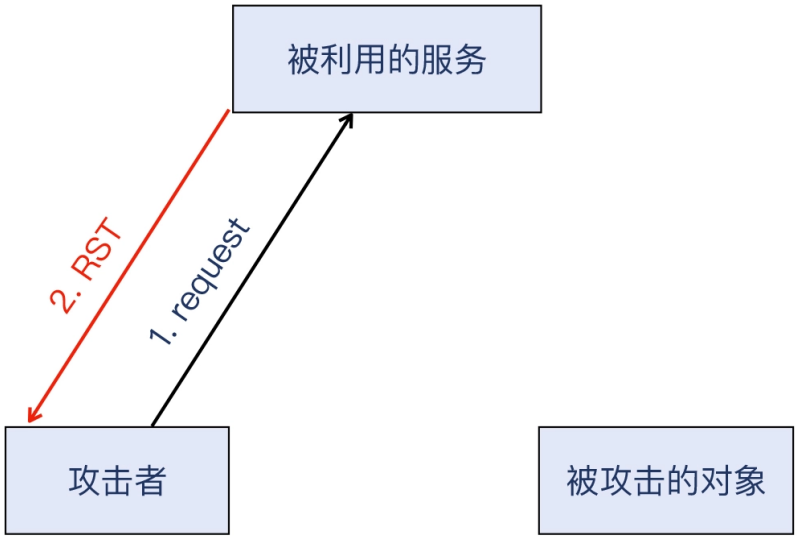


但是 TCP 就非常不同了。首先 TCP 需要三次握手，如果攻击者的 SYN 报文的源地址是伪造成被攻击站点的 IP，那么 SYN+ACK 报文就直接回复到那个站点的 IP 了，而不是攻击者。然后会发生什么呢？

被攻击站点收到一个莫名其妙的 SYN+ACK，就会被 RST 掉。这次 TCP 握手就这么结束了，攻击就没法继续了。



那跳过 TCP 握手，直接发送应用层请求（源地址还是伪造成被攻击站点的 IP）给反射服务呢？当然是直接被 RST，因为连握手都没做过呢。



极客时间

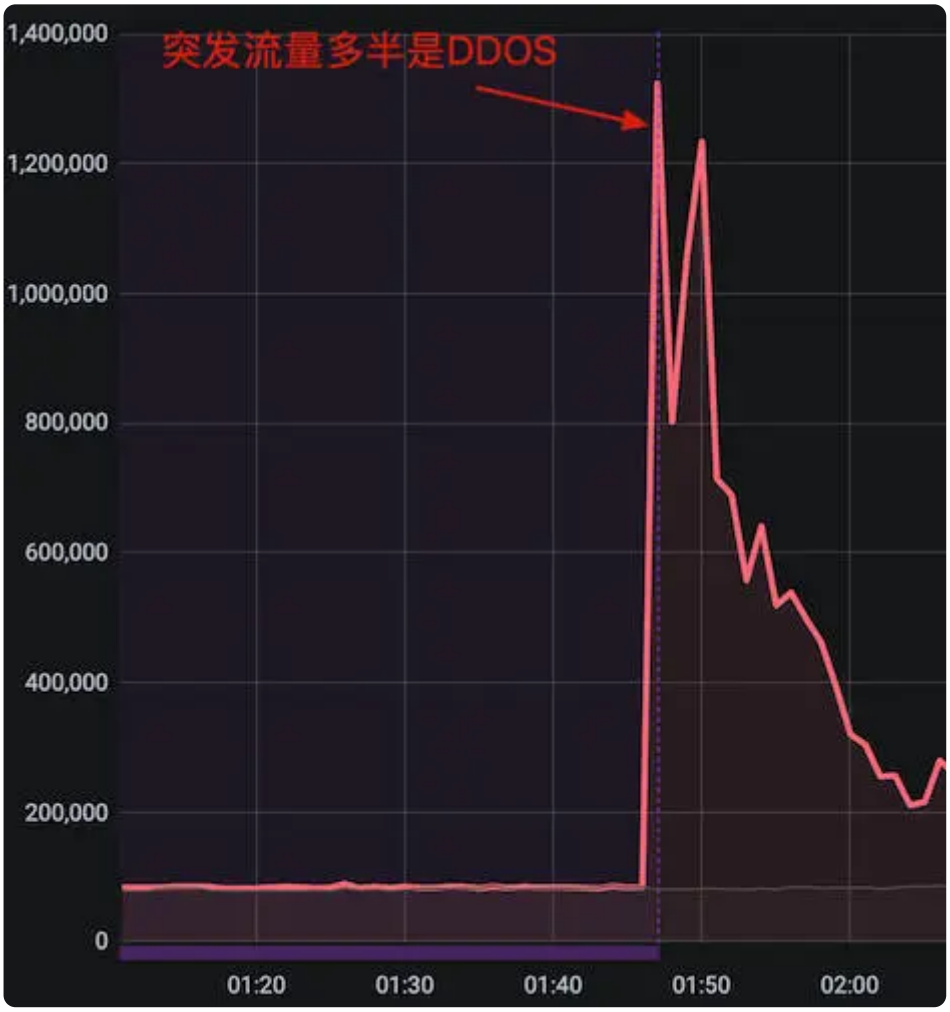
而且，即使通过了握手，后续通信双方还有对序列号和确认号进行校验等机制。虽然这些在技术上都可以实现，但难度大了很多，而选择 UDP，就不需要考虑这么多问题。

所以，**用 TCP 的话就是直接攻击**，而不是反射攻击了。比如 SYN 攻击、半连接攻击、全连接攻击、CC 攻击等等。从“性价比”上看，反射攻击的优势更大些。

如何对付 DDoS？

前面我们分析了 DDoS 中最为典型的反射放大攻击。以后如果我们发现服务异常，比如客户端的请求十分卡顿的话，就可以在服务端抓包，然后进行分析，就能快速定位是否是 DDoS 攻击了。

当然，还有一个更为简单直接的证据，就是你的公网接口带宽使用图，如果图上有明显的突增，甚至达到了接口带宽的上限，那也基本可以判定是遭遇 DDoS 了。

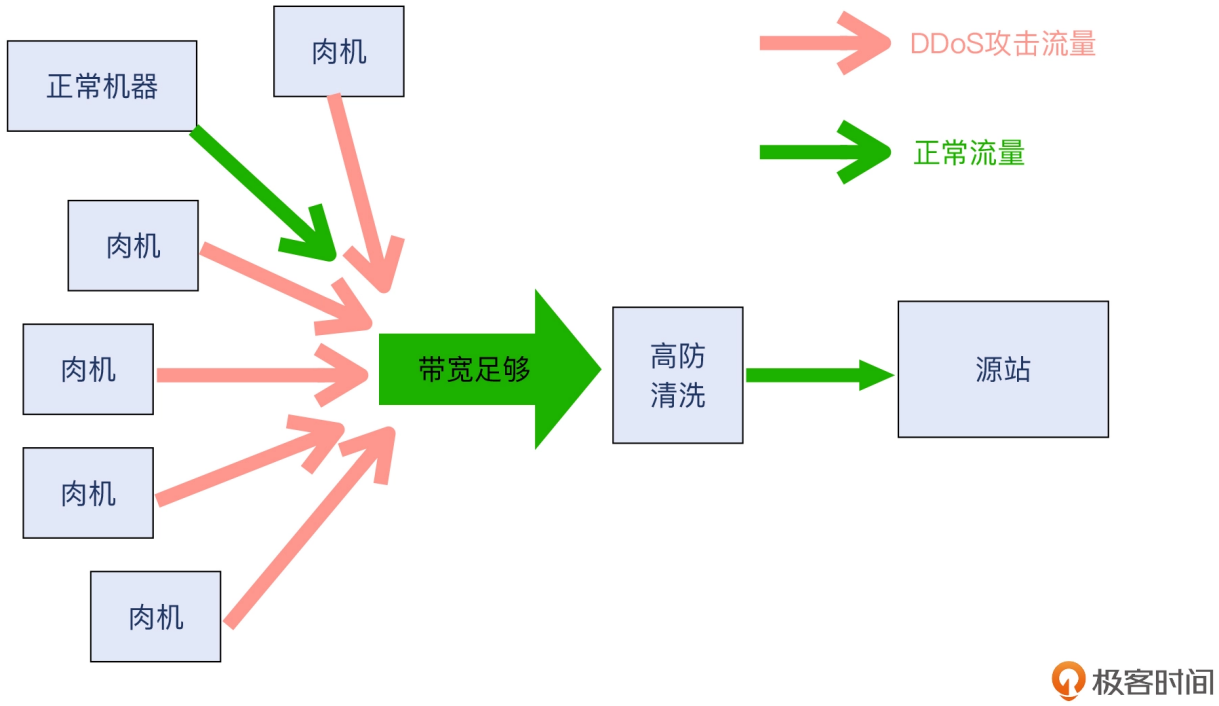


上面都是排查的手段，那接下来如何处理呢？一般来说，你自己单干是不行的，这里给你介绍几种应对策略，这样你以后就心里有数了。

高防

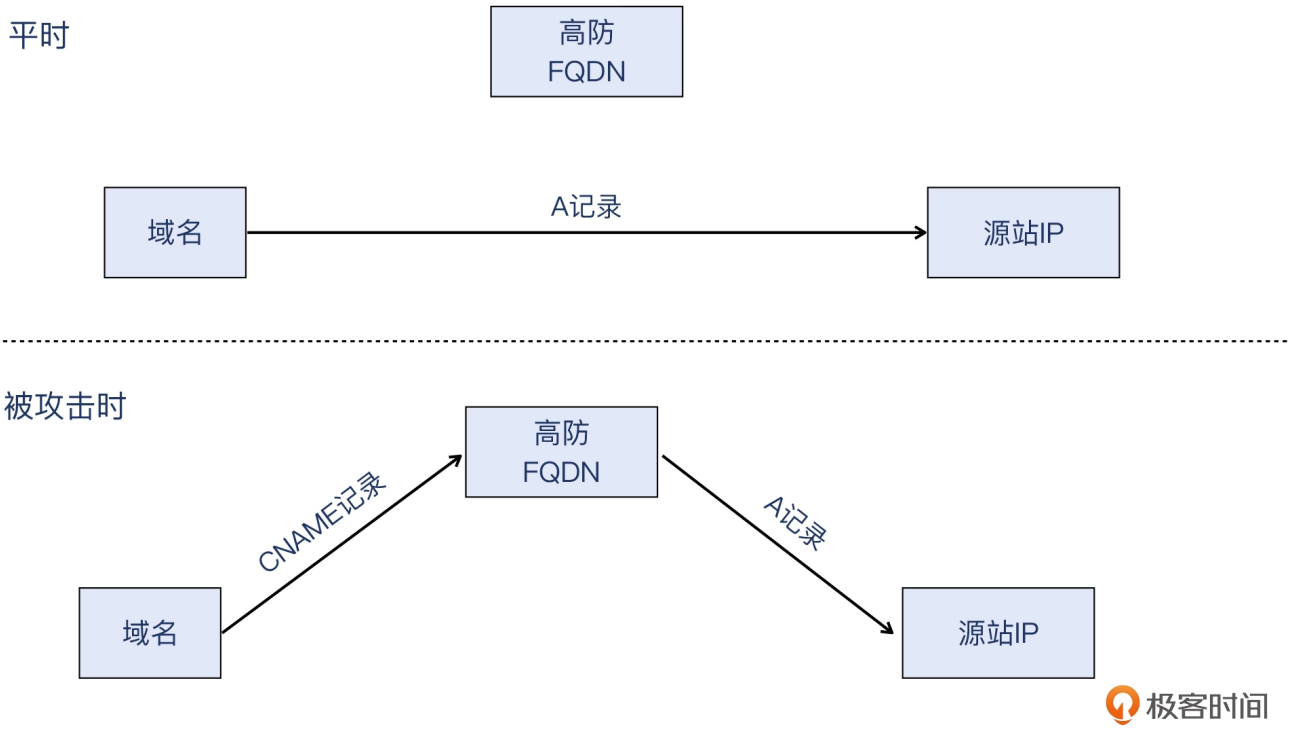
一般来说，如果你的服务架设在公有云上，那么可以考虑使用云商或者其他专业安全服务商的高防产品。

高防是需要放置在源站前面的一类安全防护和清洗系统。它利用了自身的足够大的带宽，以及强大的防护清洗集群，实现对流量清洗，最终把攻击流量拦截在外面，清洗过后的正常流量进入源站，得以被正常处理。示意图如下：



> 补充：这里的源站，就是被攻击的站点。

由于高防按时计费而且费用高昂，一般平时是不接入高防的。只有探测到被攻击时，才自动或者手动转入高防。这里的“接入高防”是什么意思呢？其实就是把站点域名指向高防的域名，这样就把流量先流向高防，再经清洗后回到源站。

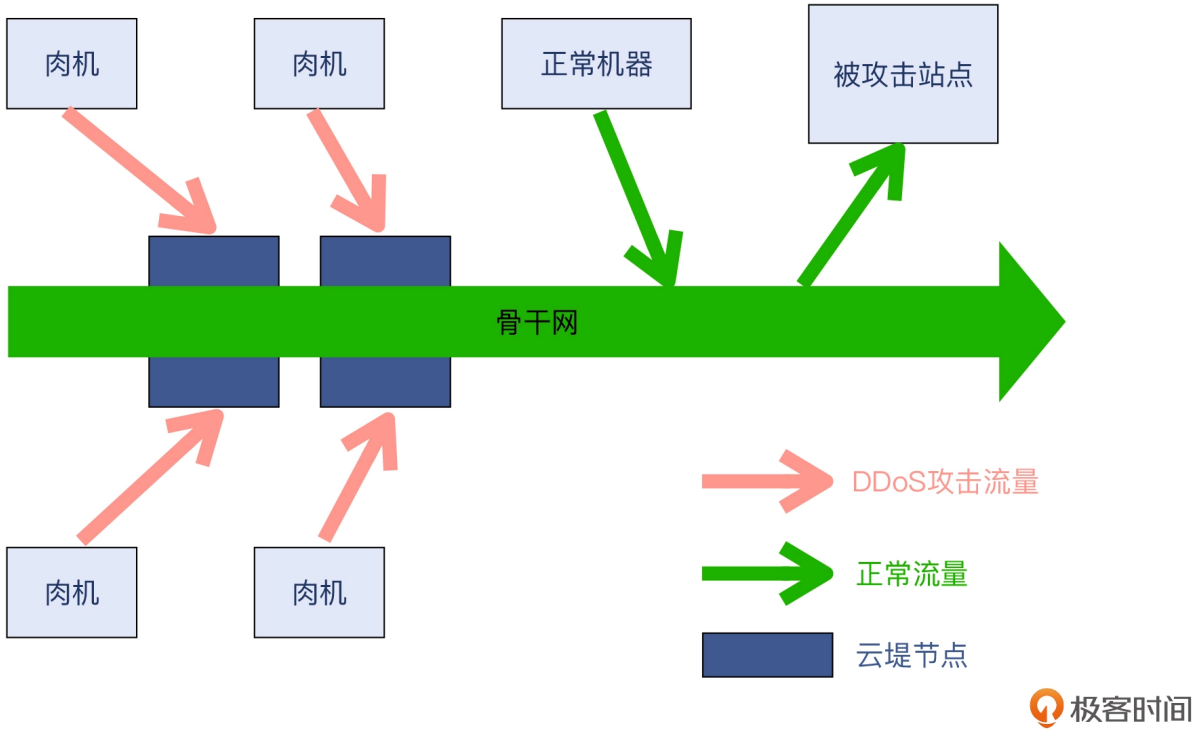


那如果攻击者不是通过域名解析，而是直接盯着 IP 做攻击的呢？也不难，你就把老的 IP 解绑，让攻击流量进入路由黑洞，然后绑定新的 IP。这时候，不要暴露新的 IP 的信息，它只能给高防回源用，不能让更多的人知道。

你有没有发现，从防护的生效点来说，高防这种方式是作用在**服务端这一侧**。那你可能会想到：如果我们能**在攻击的源头就做防护**，那是不是效果会更好呢？这就是另一类 DDoS 防护产品的设计思想，其中比较典型的产品是电信云堤。

云堤

云堤本身属于运营商自己的系统，而无论被攻击站点还是肉机，都依托于运营商的公网线路才能进入因特网，所以云堤**具有“地理”上的天然优势**。它可以作用在肉机的攻击流量进入骨干网之前，所以很可能这些攻击都没有机会走到被攻击站点的跟前了，相当于“扼杀在摇篮里”。我们看一下示意图：



anycast 和多 POP

我们知道了 DDoS 的本质是挤占网络带宽，那么对付它的核心策略就是：

用更大的带宽来接纳，先解决正常流量被挤出网络的问题。

在接纳后进行清洗，把正常流量识别出来，发回给源站，让业务继续进行。

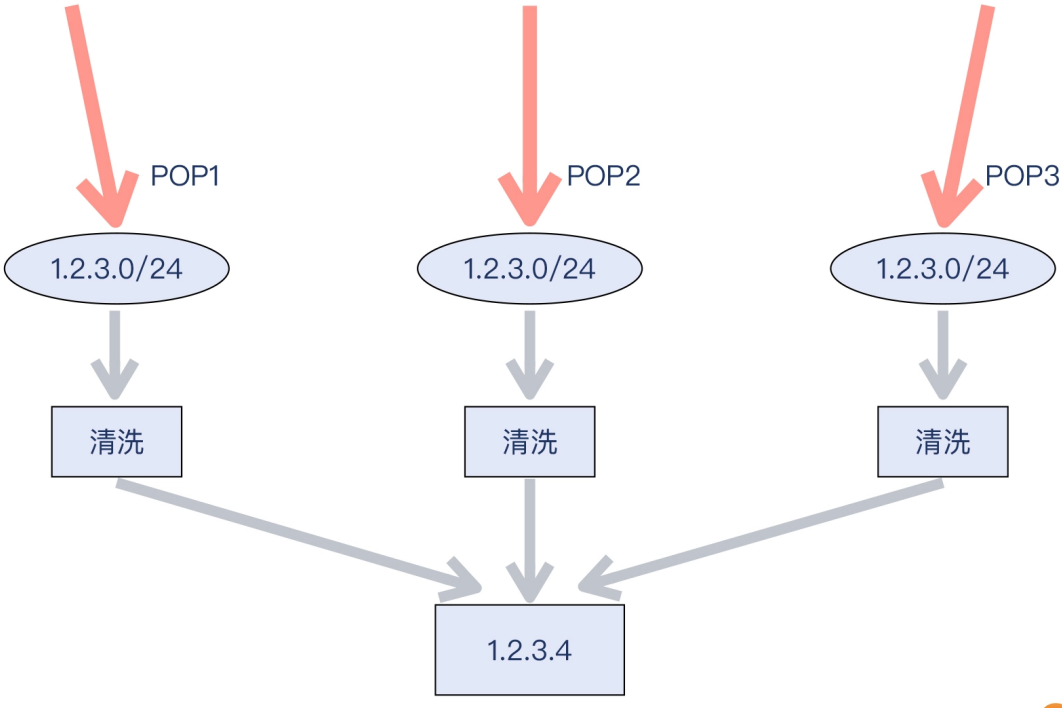
前面介绍了高防和云堤，两者分别在被攻击站点的近端和远端起到了作用，也都是商业服务。那么，另外一种方式是自己搞定，这也是一些**自身规模比较大的网站会部署的架构**，它就是 anycast 和多 POP。

anycast 是网络术语，是指**多个地点宣告同一个网段或者同一个 IP 地址的行为**。比如，最典型的电信的 DNS 服务地址 114.114.114.114，还有谷歌的 DNS 服务 8.8.8.8，就是在全国乃至全球各处做了 anycast 的 IP 地址。与这个词类似的，还有 unicast 和 multicast，分别是指单播和多播。

比较大型的网站都会在各地部署 POP 点（也就是多 POP），然后这些 POP 点会宣告相同的 IP 段。一旦有 DDoS 攻击，因为它的目标 IP 是属于 anycast 网段的，所以会被因特网的路由策略，相对均匀地分布到这些 POP。

假如你有 20 个 POP 点宣告同一个网段，那么你就有机会把 DDoS 攻击化整为零，平均每个 POP 点承受 1/20 的攻击流量，大大降低了危害性。在攻击流量不高的时候，仅依靠自己的多个 POP 就可以吸收掉这些攻击流量，然后用自己的设备进行清洗就可以了。

这一点上看，anycast+ 多个 POP+ 自有的清洗设备，这一整套做好以后，相当于自己建设了一个中小型的高防系统。我画了个示意图供你参考：



> 补充：这里的 anycast 一般是作用在网段级别。而在单个 IP 级别的 anycast 应用还较局限，目前主要还是主要应用在基于 UDP 的服务，比如 DNS 服务上。基于 anycast 的 HTTP 是比较前沿的领域，目前有少数公司已经开始实践，相信在不久的将来，应该会看到越来越多的公司应用 HTTP over anycast。

CDN

与前一点类似，CDN 也是通过“多点分布”来达到防护或者缓解 DDoS 攻击的目的。而且 CDN 服务商一般也会采用 anycast 等策略混合使用，使得其防护 DDoS 的能力更加出色。

小结

这节课，我们通过 NTP 反射攻击和 SSDP 反射攻击这两个典型的 DDoS 案例的学习，了解了反射放大攻击的特点，它主要利用了以下三点：

IP 协议不对源 IP 进行校验，所以可以伪造源 IP，把它设定为被攻击站点的 IP，这样就可以把响应流量引向被攻击站点。

UDP 协议是无连接的，可以直接进行应用层的一问一答，这就使得 IP 欺骗可以奏效。

某些服务具有“**响应报文的大小是请求报文的很多倍**”的特点，使攻击行为达到了“四两拨千斤”的攻击效果。

我们也系统性地分析了 DDoS 的核心方法，也就是用“**耗尽网络带宽**”的方式，让被攻击站点无法正常提供服务。在排查方面，当我们发现服务异常时，在服务端做抓包分析，可以快速定位是否有 DDoS 攻击。也可以直接根据带宽使用图，关注到突发的巨型流量时也可以直接判定是 DDoS 攻击。

另外，我们还了解了应对 DDoS 攻击的策略，包括：

使用高防产品，可以防护非常巨大的攻击流量。

如果对防护效果有更高的需求，可以使用运营商的**云堤类的产品。**

如果自身条件足够，可以部署**多 POP 和 anycast**，平均吸收攻击流量。

也可以**上 CDN**，让 CDN 天然的分布式布局减轻 DDoS 的影响。

在技术细节方面，你也可以记住这个新的命令 **capinfos**，用它可以快速获取到抓包文件的整体信息，包括抓包时长、总报文量、平均报文大小等信息。关于如何在 Wireshark 里解读出报文字段的长度，你也要知道至少下面这两种方法：

选中你要解读的报文字段，然后在下面的字节码部分，数一下有底色的字节个数。

还是选中你要解读的报文字段，在底边栏里也有对应的字节数的显示。

最后，你要知道这一点：**UDP 载荷最好不要超过 512 字节**，这也是 IPv4 协议规范的建议，像 NTP 和 DNS 这些基于 UDP 的协议都实现了这个规范。

思考题

给你留两道思考题：

“肉机”发出 100Mbps 的攻击流量，到达被攻击站点的时候，仍然是 100Mbps 吗？为什么呢？

为什么 CDN 可以达到缓解 DDoS 的效果呢？

欢迎你把答案和思考写到留言区，我们一起讨论，进步。

上一篇 13 | 重传的再认识：没有任何丢包却也一直重传？

下一篇 春节特别放送（一）| 书单推荐

精选留言 (4)

写留言

那时刻 民

2022-02-21

这次理解了为什么dns协议为啥既采用UDP，又采用TCP的原因了。

作者回复: 恩～

1

Chao

2022-02-23

CDN通过 GSLB 将流量都分散到不同的POP节点去了。 如果堵住GSLB，应该就另说了吧

作者回复: 正确，CDN分散吸收了流量。

GSLB本质上是DNS服务，本身不接受HTTP流量，只接受DNS请求，然后通过回复不同的解析结果，来控制流量的配比和分布。DNS是分布式同步机制，是各个运营商帮你"分担"的，所以GSLB被"堵住"的概率应该是不太大的~

共 2 条评论 >

Realm

2022-02-21

1 假如利用反射攻击，有可能攻击流量翻倍；也有可能攻击流量打出来，被运营商给干掉了一部分，结果没有100M；
2 cdn按照不同来源IP，把攻击流量稀释了，一定程度上可以缓解ddos；

作者回复: 你答的很全面，区分了反射攻击和直接攻击👍确实如你所说，如果是直接攻击，因为流量会因为链路上存在的各种限制和瓶颈（这也是为什么tcp需要做拥塞控制），流量会降低不少。有说法是真正到达被攻击站点的流量只有最初发起时的1/10。当然这是指大规模发起攻击的时候。如果只是单台发起100Mbps，衰减没有这么厉害，但是比出发时低是肯定的。
第二个答案也正确：）

那时刻 民

2022-02-21

"肉机"发出 100Mbps 的攻击流量，到达被攻击站点的时候，仍然是 100Mbps 吗？为什么

呢？
答：到达被攻击点的时候；可能高于100Mbps，可能因为中间节点mtu导致的。
为什么 CDN 可以达到缓解 DDoS 的效果呢？
答：因为CDN有多个边缘节点，这些边缘节点的IP不同，ddos在这些边缘节点会被消散...
展开

作者回复: 第一个答案我没看明白，为什么中间节点mtu会导致流量增大呢？
第二个答案对的，cdn节点多，攻击流量来自全国乃至全球，就相应的被很多cdn节点给平均消化掉了，到达每个cdn节点的流量并不很高，节点一般可以搞定。即使几个节点挂了，因为多数节点依然是好的，对业务影响小很多

共 2 条评论