

DDoS in Lab 中期报告

刘德欣，马赢超，张煌昭，张一舟，张元玮

摘要—DDoS（分布式拒绝服务）作为一种网络攻击手段，其特点是原理简单，攻击方式较为简单，但难以防御，一旦成功后果往往十分严重。近年来如何有效地防御和进行 DDoS 攻击，已经成为网络安全领域的重要问题。本次实验，在局域网实验室环境下，组员分作相互独立的两部分，分别扮演防御方和攻击方，对一网络服务器进行防御和 DDoS 攻击。目前已经完成防御方的云端的待攻击 Web 服务器和性能监控 Web 服务器的架设，和攻击方的几种基础 DDoS 攻击的实现。之后将完成防御方的过滤防御，和攻击方的混合攻击，UDP 流量方法等较为高级的攻击方式。

I. 小组成员

本次大作业小组，“起飞”（“Taking-Off”）小组，的成员包括（以下按姓名字母序排序）：刘德欣，马赢超，张煌昭，张一舟，张元玮。其中张一舟为小组组长。小组成员信息见脚注，GitHub 组见 TakingOffPKU¹。

小组分作攻击组和防御组。攻击组成员包括张煌昭和张一舟，负责实现各类 DDoS 攻击方式，并对待攻击 Web 服务器发起 DoS 或 DDoS 攻击；防御组成员包括刘德欣，马赢超和张元玮，负责假设和维护待攻击 Web 服务器及其孪生的性能监控服务器，以及对攻击进行尽量的防御。

本次大作业所有源代码，实时更新于 GitHub 开源平台²。本次报告使用 Overleaf $L^A T_E X$ 在线平台编写³。

按字母序排序，组长为张一舟。

刘德欣，1500017704，元培学院

马赢超，1400015999，光华管理学院

张煌昭，1400017707，元培学院

张一舟（组长），1500012933，信息科学技术学院

张元玮，1400013399，信息科学技术学院

¹ GitHub 组请见 <https://github.com/TakingOffPKU>，项目源码请见 <https://github.com/TakingOffPKU/DDoS>

² 本次大作业源码可通过以下 git 命令获得，
git clone git@github.com:TakingOffPKU/DDoS.git

³ 本报告源码可通过以下 git 命令获得，
git clone https://git.overleaf.com/16486673jkcmqykmzrky

II. 防御组

防御组搭建和维护待攻击的 Web 服务器，搭建和维护用于监控流量和性能的另一监控 Web 服务器，对攻击组发起的 DoS/DDoS 攻击进行防御。目前已经完成了 Web 服务器和监控服务器的搭建，和对 SYN Flood 和 HTTP Flood 的防御。

A. 服务器搭建

通过 Apache HTTP Server 2 的默认配置来模拟大多数网站在没有进行特殊的安全配置时面对的默认环境。因为大多数网站都采用 Apache 和 PHP 7.0 架构来搭载 Web 服务，许多重要的 Web 服务和协议都通常在 Apache 上运行，例如：Docker，ownCloud，Aria2 RPC 等 Web 服务，和 WebDAV，CalDAV，CardDAV 等 Web 协议。

除了提供正常的 Web 服务，通过另外一个端口运行的 Linux Dash 仪表板可以随时查看当前服务器的性能状态。通过调整处理器优先级来始终让 Linux Dash 仪表板返回优先的结果

最终 80 端口上的站点 target 是一个使用 HTML 和 PHP 7.0 协议的示例站点，其中包含图片、视觉元素和多发请求；81 端口上的站点 dashboard 是 Linux Dash，可以查看的信息包括：CPU 使用情况，内存使用情况，磁盘使用情况，进程列表，端口使用情况，系统配置和配置文件。

使用浏览器访问监控服务器，获得服务器状态如图 II-A 所示。

B. DoS/DDoS 防御进度

在目标 Web 服务器的后台基础上，对几种常用的单一攻击手段进行简单防御，具体内容位于 server.py。目前完成了 SYN Flood 防御和 HTTP Flood 防御。

1) 对 SYN Flood 的防御：在 Server 端线程池的设计外，添加如下对连接请求的过滤：1) 建立白名单机制，将已经建立真实请求的 client 加入白名单，白名单

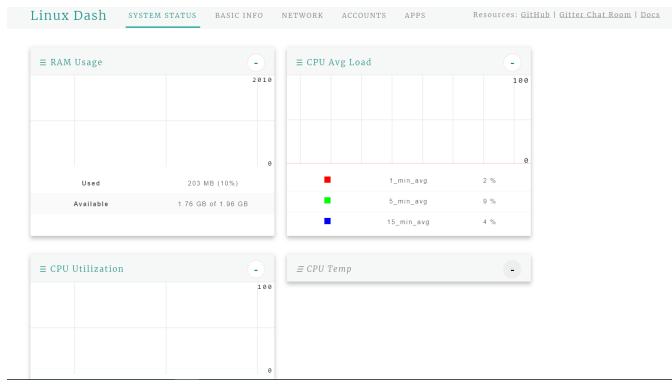


图 1. 通过 81 端口的监控服务器查看整个服务器状态，图中显示了内存利用情况和 CPU 符合情况。

设有一定的生存周期；2) 对不再白名单内的机制进行反向侦测，发送虚假 SYN+ACK 回应，检测后续是否有 ACK 包；如果收到认为是真实请求，加入白名单并给予正常响应；3) 对线程池大小进行限制为 1000，防止响应时提前耗尽资源（考虑到网站规模和服务器响应能力）

2) 对 *HTTP Flood* 的防御：使用和 SYN Flood 防御相同的白名单模式，主要的检测手段更替为对 HTTP 请求内容的检测。具体的内容如下。

检测 HTTP 请求头部是否有合法浏览器标识（目标网页只包含浏览器端的 GET、POST 请求，没有请求 api 的需要，认为所有请求必须俩字浏览器）。

检测 HTTP 请求内容是否符合为正常，如果请求的内容格式与网页设计内容有较大出入即过滤请求，不进行相应。该部分实现效率较低，使用正则表达式过滤判断请求是否合法同样需要耗费计算资源，但较响应网页内容依然有着较快的速度

III. 攻击组

攻击组对防御组搭建维护的待攻击的 Web 服务器发起 Dos/DDoS 攻击，目标为减缓服务器提供 Web 服务的速度，或使得服务器无法提供 Web 服务，甚至使得 Web 服务器直接宕机。目前已经完成了 SYN Flood, HTTP Flood 和慢速连接攻击。三种攻击都通过多线程并发的方式进行攻击，主线程生成攻击线程，攻击线程向服务器发送请求。

A. DoS/DDoS 进度

1) *SYN Flood*: SYN Flood 利用了 TCP 三次握手协议的漏洞进行攻击。TCP 的三次握手为分别为：客

户端发送 SYN，服务器端回复 SYN+ACK，客户端再回复服务器 ACK，三次握手完成后，TCP 连接建立。在这其中，服务器会在回复 SYN+ACK 时直接预分配本次 TCP 连接所需要的资源，预分配的资源只有在连接超时时，或者连接正常建立并在之后被关闭时被释放；然而服务器并不会在回复 SYN+ACK 时检查客户端的地址是否正确，这给 Syn flood 攻击带来了漏洞。攻击方利用漏洞 TCP 握手的 SYN+ACK 漏洞，向服务器频繁发送 SYN，这些 SYN 的源 IP 和端口都是被伪造的，其 IP 并不存在，因此服务器的 SYN+ACK 并不会得到 ACK 回应，进而预分配的资源只有在超时时被释放。攻击方频繁发送，长此以往服务器的资源被耗尽，攻击目的达成。

采用多线程的方法进行实现，每个攻击线程都只构造并发送一次 SYN，发送完成后死亡。该 SYN 的 IP 层的 IP 地址和 TCP 层的端口均为随机产生伪造的；主线程不断产生攻击线程直至达到动态饱和，达到 SYN Flood 的攻击效果。

实现详情请见 synflood.py，直接调用该文件，可以对云端部署的待攻击服务器发起 DoS SYN Flood 攻击，其它使用方法请见 “python synflood.py -h” 命令。

2) *HTTP Flood*: HTTP Flood 采用模仿用户行为，频繁发送无意义的 HTTP 请求的方式实现。通过频繁得发送 HTTP 请求（比如 GET 该 url 下某一文件或一不存在的文件），耗尽服务器资源。HTTP Flood 攻击的精妙之处在于模仿用户行为，使得防御方难以甚至无法分辨攻击行为。

采用多线程的方法进行实现，每个攻击线程都只构造并发送一次 HTTP GET 请求，发送完成后死亡。为了模仿用户行为，建立文件池，该文件池为有效该服务器下有效的文件名目录，初始化包含 index.html, index.php, index.htm 等常见的首页文件名。在生成 GET 请求时，以大概率从文件池取出某一文件，小概率随机生成文件名，若随机生成的文件可以得到服务器响应，则将其加入文件池中。概率大小的设定与服务器防御方的防御策略相关，需要进行调试。

实现详情请见 httpflood.py，直接调用该文件，可以对云端部署的待攻击服务器发起 DoS HTTP Flood 攻击，其它使用方法请见 “python httpflood.py -h” 命令。

3) 慢速连接：慢速连接攻击避免暴力发送请求从而规避流量过滤，通过利用协议漏洞和巧妙设计，对网

络服务进行攻击。其中 Slowloris 攻击针对 HTTP 协议漏洞进行慢速连接攻击。HTTP 协议中规定了客户端发送结束的标志，在收到这一标志前且没有超时时连接不会中断。攻击方发起攻击时，发送 HTTP 头为 Keep-Alive 的请求，使得连接不会断开，之后每隔几十秒或几分钟发送一小段关键数据至服务器端，最终耗尽服务器 TCP 资源。

在实现慢速链接时，通过发送 GET 请求来开始一个 HTTP 链接，HTTP 协议声明为 1.1，这样默认开启 Keep-alive。在程序中，构建了一个 User-Agent 表，每个线程随机的从表中选择一个作为自己的 User-Agent，这样分散分布，降低被侦测的可能性。在成功的发送了 HTTP 请求的头部之后。线程每隔一段时间发送一个关键值，但始终不使用 \r\n（结束标志）结束发送，直到连接超时被中断。

实现详情请见 slowloris.py，直接调用该文件，可以对云端部署的待攻击服务器发起 DoS Slowloris 攻击，其它使用方法请见“python slowloris.py -h”命令。

B. 效果

在初步的对无防御的 Web 服务器进行的 DoS 攻击试验中，HTTP Flood 可以达到使目标服务器完全无法提供 Web 服务的效果，SYN Flood 和 Slowloris 在 DoS 攻击环境下会使得访问目标服务器变慢，但依然可以使用。

从效率上看，HTTP Flood 效果最好，其次是 SlowLoris 和 SYN Flood，HTTP Flood 仅需要一台机器进行攻击即可让 web 服务器宕机，SYN Flood 和 SlowLoris 在单机上仅可以使 web 服务器变慢，需要分布式的攻击才可以有显著效果。

HTTP Flood 的攻击效果如动图所示（可能需要点击开始播放，若无法播放请更换至 Adobe Acrobat Reader DC，或查看附件 httpflood.gif）。靠上为未攻击的正常运行在 Web 服务器上的 DBChat 聊天服务，靠下为攻击开始后的 DBChat 聊天服务。HTTP Flood 攻击开始后，Web 服务无法继续进行。

IV. 后续工作

随着 Web 服务器防御手段的强化，之前的单一简单攻击手段变得不实用。因此，需要应用更加复杂的手段进行攻击。同理随着攻击行为的复杂化，Web 服务的防御收端也需要进行升级。

我们的后半段的工作，攻击方将围绕混合攻击等技术展开，而防守方则围绕过滤分流等技术展开。

A. 防御方：慢速连接防御

对 Slowloris 的防御主要通过服务器端设置来进行，通过设置 HTTP 连接最大时长（1min），或者不允许设置 HTTP 类型为 keep-alive，而禁止长久不使用的连接占用资源。

1 分钟时长的无效连接依然会占用一个线程池中的位置，考虑到我们的网页目前仅有静态内容，没有需要 POST 获取的长时连接需要，我们计划接下来将对连接中可能的虚假连接进行检测，在正常的 GET 请求结束后主动结束连接。

B. 防御方：综合防御手段

目前我们完成了对单一攻击类型的简单防御，但对实际中的大规模、多手段的攻击还无法防范。接下来的时间里里，我们计划构建出一个可以检测当前服务器被攻击情况的防御系统，以此实现对不同情形下遭受的攻击进行适当的防御。

检测当前服务器的网络连接数、使用带宽等信息，结合过滤器评估当前服务器是否受到攻击，根据攻击程

度的轻重开启不同过滤等级的缓冲池、白名单机制；检测当前 http 服务的连接情况，识别出“慢速攻击”的虚假连接，主动断开超时连接增加对综合攻击手段的整体防御，如对掺杂网络层、应用层的攻击进行过滤。

C. 攻击方：流量放大技术

流量攻击技术利用了一些服务器的服务回复通常比请求包更大的特点。攻击者选择一些基于 UDP 的服务器作为杠杆（比如 DNS），向服务器发送查询请求，查询请求里的源地址伪装为被攻击服务器的地址，这样被利用的服务器就会向被攻击服务器发送回复包，从而放大攻击流量。

该技术较为诱人，但在我们的实验中进行实现的难度相对较大。

D. 攻击方：混合攻击手段

普通的攻击手段容易被针对性的防御，比如 SYN Flood 容易被流量监测手段过滤，但是如果在 SYN Flood 的海量 SYN 中混入适当比例的 SYN+ACK，就可以伪造出看似合理的数据流，使得过滤手段失效。

由于 Web 服务器可能通过 IP 的历史行为来侦测是否为攻击者，因此在 HTTP Flood 时，需要混合入一些正常的请求（即真实存在的文件），来防止进入黑名单。

此外，还可以同时对应用层和网络层发动攻击，当网络层的大量资源被占用时，应用层也会受到影响，因此，混合两个层的攻击往往能互相放大各自的效果。

混合方式往往用作真实环境中的攻击，以期达到欺骗防御方采取错误防御手段的目的。例如将 SYN Flood 与 Slowloris 混合，防御方首先也是最容易检测到流量巨大的 SYN Flood，从而防御方针对其采取分流和过滤等手段；然而，真正的攻击手段在于 Slowloris 攻击，倘若防御方没有及时发现，那么 Slowloris 将隐藏在海量 SYN Flood 之下，完成攻击行为。