DDoS in Lab 结题报告

刘德欣，马赢超，张煌昭，张一舟，张元玮

摘要—分布式拒绝服务（DDoS）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，分别扮演攻击方和防御方，对一网络服务器进行 DDoS 攻击和防御。我们的攻击组实现和提供了 DDoS 攻击中常见的 SYNFlood，HTTPFlood 和 Slowloris 攻击，及其混合攻击，并且实现了整合在僵尸网络（Botnet）端的攻击程序，以及配置和控制整个僵尸网络的控制程序；防御组搭建了一台 Web 服务器用于攻防测试，并在该服务器上实现了对上述三种攻击的防御。最终攻防双方在云端服务器部署 botnet 和 Web 服务器及其防御，并进行真实网络环境下的 DDoS 攻防实验。

I. 介绍

分布式拒绝服务（DDoS）是目前非常流行且破坏力最为强大的网络攻击方式之一，其原理简单，攻击方式较为简单但数目和流量巨大，难以进行自动防御，并且一旦攻击成功后果往往十分严重。2018 年 3 月 1 日凌晨 1 时 15 分，开源平台 GitHub 遭遇迄今为止规模最大的 DDoS 攻击，峰值流量高达 1.35Tbps。本小组为了对 DDoS 这一攻击方式有更为具体的认知，在实验室的安全环境下实现了 SYNFlood，HTTPFlood 和 Slowloris 三种典型的 DDoS 攻击和防御，并在真实网络环境中，部署少量僵尸网络（botnet），对本小组维护的 Web 服务器发起混合 DDoS 攻击进行实验。

实验分作 DDoS 攻击和 DDoS 防御两部分：攻击部分对上述三种 DDoS 攻击方式进行实现，并将其整合为混合攻击，同时完成 botnet 部署和控制分发；防御部分维护待攻击服务器，并针对上述三种 DDoS 攻击方式进行防御。整个实验分作 3 阶段，攻击与防御各独立进行（下划线部分为防御内容，其余部分为攻击内容）：

1) 实现经典的 SYNFlood，HTTPFlood 和 Slowloris 攻击，搭建待攻击 Web 服务器和性能监控服务器；2) 在待攻击服务器上对经典的 SYNFlood，HTTPFlood 和 Slowloris 攻击进行防御，优化三种攻击方式，对攻击进行伪装；3) 在真实网络环境下进行实验，将待攻击服务器及防御机制部署至云端服务器，利用若干台云端服务器和个人电脑组成 botnet，向待攻击服务器发起 DDoS 攻击。

本次大作业，我们的主要工作为两方面：1) 在实验室环境下对三种 DDoS 攻击和防御的原理进行研究和实现；2) 在真实网络环境中进行 DDoS 攻防实验，还原简单的 DDoS 攻防场景。通过本次大作业，我们对于网络安全，尤其是 DDoS 攻击，Web 服务器维护和防御等方面有了具体和较为深入的认识和了解。

II. 小组成员及分工

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 TakingOffPKU[1]。

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，实现了常见的 SYNFlood，HTTPFlood，Slowloris 等 DDoS 攻击方式，并整合完成完整的 DDoS 攻击系统，系统中攻击程序部署于各 botnet 主机（bot），受控于一个攻击控制程序；防御组成员包括刘德欣，马赢超和张元玮，负责架设和维护 Web 服务器及与其位于同一主机的性能监控服务器，Web 服务器用于 DDoS 攻防，性能监控服务器用于监控主机性能，并对攻击组实现的 SYNFlood，HTTPFlood 和 Slowloris 攻击进行防御。

小组成员的具体工作如下（下划线标注为防御组成员，没有标注则为攻击组成员）：刘德欣参与搭建性能监视服务器，并实现了对 HTTPFlood 和 Slowloris 攻击的防御；马赢超搭建待攻击的 Web 服务器，并参

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

与搭建其孪生的性能监视服务器；张煌昭实现经典的 SYNFlood 攻击和 HTTPFlood 攻击，整合三种攻击方式的混合攻击，并实现 botnet 中的 bot 攻击端和控制端；张一舟实现经典的 Slowloris 攻击，并对三种攻击方式进行伪装和改进；张元玮实现对 SYNFlood 防御的防火墙。

本次大作业所有源代码，开源于 GitHub 开源平台[2]。本次报告使用 Overeaf L^AT_EX 在线平台编写[3]。

III. 原理

本节对本次实验使用到的 DDoS 攻击和防御技术的原理进行说明：在第III-A节对 DoS 攻击和 DDoS 攻击的基本原理以及防御的基本思路进行介绍，之后再第III-B-III-E节对本次实验使用到的 SYNFlood，HTTPFlood，Slowloris 以及混合攻击的攻击原理和防御手段进行详细说明。

A. DoS 和 DDoS

DoS（Denial of Service，拒绝服务）攻击，是一种简单的，易于实现的，但破坏力很强的网络攻击方式。其基本思想为使用暴力手段或利用网络协议的漏洞，强占网络服务器资源，从而使得网络服务器无法提供正常的网络服务，甚至是宕机。DoS 攻击按其手段的暴力程度可以分作两类，一类为泛洪式（Flood）攻击，另一类为慢速连接攻击。下面对着两种攻击进行介绍。

Flood 攻击往往十分暴力，攻击者发送大量无意义的请求至被攻击服务器，这些请求一旦获取了服务器资源（比如获取了服务器的 TCP 连接，或占用了服务器的网络带宽）就会使得资源被无意义地耗用。Flodd 攻击者通过短时间的大量无意义请求，耗尽服务器资源，使得服务器无法向正常用户提供服务，甚至是由于资源耗尽而宕机。

慢速连接攻击更为巧妙，攻击者利用网络协议的漏洞，精确地、频次较低地发送请求。这一类攻击往往更加难以防范，而其一旦成功，将会耗尽服务器资源，同样会对服务器造成致命打击。

一般而言，越暴力的 DoS 攻击越需要攻击者更多的网络资源（比如网络带宽），同时防御方更容易进行过滤防御；协议层级越高，设计越精巧的 DoS 攻击，需要攻击者的网络资源越少，给防御方的防御难度和压力也越大。

由于 DoS 攻击本身所具备的暴力性，攻击方往往需要很多的网络资源，比如更大的网络带宽，更快的读写速度，更高的并发度等，而这些必需要求很难在同一台主机上得到满足。因此，DoS 攻击基本只能作为理论上的攻击方式，真实环境中的攻击往往是将 DoS 攻击分散在网络环境中的各个主机（即为 bot）中，以 DDoS 的方式进行。

DDoS（Distributed Denial of Service，分布式拒绝服务）攻击，将单主机的 DoS 攻击分配给若干个主机（bot），将其组织为僵尸网络（botnet），攻击者并不亲自发起攻击，而是利用其控制的 botnet 发起 DoS 攻击。在真实的网络攻击中，bot 往往是一些被入侵控制的个人主机，第三方服务器等。

由于 DoS 和 DDoS 攻击的原理相同，而差别在于攻击强度，其防御方式也基本相同，防御的基本手段即为**过滤**。对于 Flood 攻击，防御程序一旦发现流量异常升高，便需要在所有请求中识别无意义的攻击请求和正常用户的请求，使得正常用户的请求可以正常通过，而攻击请求直接被拒绝或分流至另一服务器进行特殊处理（比如建立黑名单，或追踪攻击者等）。而对于慢速连接攻击，并不会出现流量异常的情况，而是会占用 CPU 和内存等服务器资源，因此防御程序也需要监控服务器资源使用情况，在发现资源使用异常升高时，启用慢速连接过滤。

B. SYNFlood

SYNFlood 利用 TCP 协议三次握手的漏洞发起攻击。TCP 协议中三次握手分别为客户端发送 SYN，服务器端回复 SYN+ACK，客户端回复 ACK。该过程中的漏洞在于服务器资源分配的时机：当服务器端发送 SYN+ACK 后将预分配该 TCP 连接所需资源，该资源只有在连接失败或连接断开时才会释放，判断连接失败的条件往往是连接超时，通常被设置为几秒甚至几十秒；而当服务器端 TCP 资源不足时则将拒绝其余所有 SYN。

攻击方利用这一 TCP 协议资源分配的漏洞，伪造 SYN，该 SYN 中的源 IP 为伪造的不存在的 IP。由于该 IP 不存在，不可能回复 ACK，因此服务器预分配的资源在超时时间之内便无法被释放，长此以往，服务器资源被耗尽。

[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/16882473jndwghshvjjz

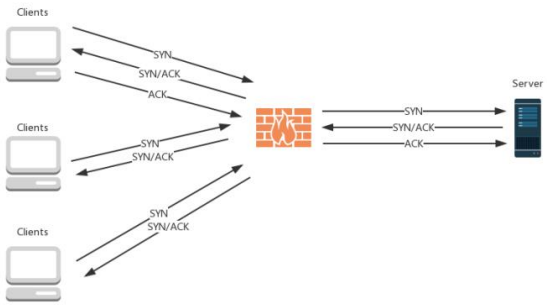


图 1. 使用防火墙防御 SYNFlood 攻击。防火墙作为中间媒介将用户与服务器之间的请求转发，以此缓解和消除服务器端的压力。

SYN Flood Defend 可以通过设立防火墙来防御掉所有 SYN Flood 攻击。具体流程展示图如图1所示。客户端和攻击端均给服务器发送含有 SYN 的 TCP 包，防火墙的功能是收到 SYN 之后自动返回 SYN+ACK，如果发送客户端回应 ACK，则客户端与防火墙建立连接。防火墙会把所有建立连接的客户端发送来的网络包转发给服务器，并把服务器给客户端返回的网络包转发给客户端。而攻击端由于采用的随机 IP 发送，不会对防火墙发出的 SYN+ACK 包做出任何反应。这样，防火墙便可以把 SYNFlood 攻击过滤。

在本小节的最后，需要补充说明的是，SYNFlood 目前已经可以直接被过滤。因此攻击者几乎不可能利用 SYNFlood 作为单一的攻击方式进行网络攻击，而是可以将 SYNFlood 与其它协议层级较高的攻击方式混合，利用 SYNFlood 的巨量流量误导防御方采取错误的防御措施，使得混杂在其中的其它攻击方式成功攻击。

C. HTTPFlood

HTTPFlood 模仿正常用户的网页行为在 HTTP 层发起攻击，即连续不断地向 Web 服务器发起 GET 等 HTTP 请求。网络安全厂商很难对正常用户的请求和攻击者伪造的攻击请求进行分辨，因此往往只能针对某个产品或某次攻击进行防御，而很难在不影响用户体验的前提下进行通用的防御。HTTP Flood 一旦成功，连锁反应很强，从 Web 前端服务器，到业务后端，甚至更后端的数据库服务器的压力都将增大，其中任何一台崩溃宕机，都将导致整个 Web 服务崩溃。

对 HTTPFlood 的防御往往都是对单次攻击特殊设计进行的。最简单的 HTTPFlood 会随机请求 Web 服务器下资源（不论是否真的存在该资源），因此可以统计各个用户的短期访问行为，若某用户短期内请求大量不存在的资源，则将其标记为攻击者，直接拒绝其任何请求；还可以进行流量过滤，将高频访问的用户标记为攻击者，拒绝其请求；通过缓存技术，减少单次 HTTP 请求处理时间来进行缓解。

D. Slowloris

Slowloris 是慢速连接中最为经典的一种攻击方式，目前已经具备多种变种。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。

攻击者发起攻击时，发送 HTTP 头为 Keep-Alive 的请求，使得连接不会断开，之后每隔一段时间（几秒甚至几分钟）发送一小段关键数据至服务器端，以此占用 HTTP 连接。长此以往，攻击者便可以耗尽服务器资源。

对 Slowloris 最简单的防御方式即为关闭服务器端 Keep-Alive 选项，但这会影响正常用户的访问和使用。一种妥协方式即为缩小 HTTP 传输的最大许可时间，使得攻击者每个攻击线程占用资源的时间变短，迫使攻击者使用更多的攻击线程来耗用服务器资源，从而可以进行流量过滤。

E. 混合攻击和防御

在第III-B节最后已经提及，目前单一的攻击方式已经很难完成网络攻击，利用各种攻击手段相互配合，才能完成一次完整的攻击行为。

此外，在第III-C节中也提到，高层协议上发动的 DDoS 攻击难以进行通用的防御，因而 DDoS 防御的策略和手段虽多，但实际上很难进行自动化选择。真实情况下，往往是防御程序发现攻击行为并通知管理员，管理员根据情况选择某一或某几个防御方案进行针对性防御。

综合以上两点，混合攻击和防御往往是攻击者和防御者的博弈：攻击者可以选择在 Flood 攻击中掺杂慢速连接攻击，利用 Flood 攻击的流量掩护慢速连接攻击（真正的攻击手段是慢速连接攻击），欺骗防御者采用流量过滤方案而没有对慢速连接进行防御，从而达到网络攻击的目的；防御者也可以像第III-D节中所述，缩小 HTTP 连接时长，迫使攻击者提高慢速连接攻击的攻击流量，从而可以使用流量过滤的手段进行防御。

表 I
BOTNET 控制端的控制命令

控制端命令	类型	功能	Bot 端回复	示例	示例回复
SYN	配置命令	配置 bot 端 SYNFlood 线程数	无	SYN 1000	无
HTTP	配置命令	配置 bot 端 HTTPFlood 线程数	无	HTTP 300	无
SLOW	配置命令	配置 bot 端 Slowloris 线程数	无	SLOW 300	无
IP	配置命令	配置 bot 端攻击目标 IP	无	IP 127.0.0.1	无
PORT	配置命令	配置 bot 端攻击目标端口号	无	PORT 80	无
ECHO	测试命令	与 bot 端进行 echo 测试	Echo 消息	ECHO Hi,bot!	Hi,bot!
STATUS	状态命令	查看 bot 端各类攻击运行情况	Bot 端状态	STATUS	234 31 657

混合 DDoS 攻击和防御，是目前常用的网络攻击手段和方法，但由于需要攻防双方丰富的经验才能真正实施，因此在本次实验中仅仅进行简单的实现。

IV. DDoS 攻击的实现

本节中对攻击组的 DDoS 实现进行介绍。攻击组所有程序均通过 Python3 实现，并需要 scapy 和 socket 包的环境支持。在第IV-A-IV-D节中对 SYNFlood，HTTPFlood，Slowloris 和混合攻击的实现，以及改进方法进行介绍。真实环境实验中的 bot 端和控制端的实现在第V节中进行说明。

A. *SYNFlood* 攻击实现

采用多线程的方法对经典的 SYNFlood 攻击进行实现，每个攻击线程都只构造并发送一次伪造的 SYN，发送完成后死亡，该 SYN 的 IP 层的 IP 地址和 TCP 层的端口利用 spacy 包进行随机伪造的。主线程不断产生攻击线程直至达到动态饱和，达到 SYNFlood 的攻击效果。

对 SYNFlood 攻击的改进如下：1) 发送伪造 SYN 后，等待一小段时间（零点几秒或几秒）后，再用同样的伪造 IP 发送伪造的 ACK，使得服务器给该伪造的 TCP 连接分配的资源得以保持；2) 利用伪造 IP 和攻击端本机 IP 混合，高频使用伪造 IP 用于进行攻击，低频使用本机 IP 用于探测服务器状态，并根据探测到的服务器连接状态，调整等待时间。

实现详情请见代码中 synflood.py 文件，需要 scapy 包的环境支持。

B. *HTTPFlood* 攻击实现

采用多线程的方法对经典的 HTTPFlood 攻击进行实现，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，GET 请求的文件为随机生成的文件名。主线程不断产生攻击线程直至达到动态饱和，达到 HTTPFlood 的攻击效果。

对 HTTPFlood 攻击的改进如下：1) 建立 Uesr-Agent 表，每个线程随机选择 User-Agent，分散以降低被侦测的可能；2) 建立文件池用于模仿用户行为，该文件池为该服务器下有效的文件名目录，初始包含 index.html，index.php，index.htm 等常见的首页文件名；3) 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 200 OK 响应，则将其加入文件池中，若文件池中取出的文件得不到 200 OK 响应，则说明服务器文件更新，该文件被删除，将其从文件池中删除。

实现详情请见代码中 httpflood.py 文件，需要 socket 包的环境支持。

C. *Slowloris* 攻击实现

采用多线程的方法对经典的 Slowloris 攻击进行实现，每个攻击线程通过发送 GET 请求来开始一个 HTTP 链接，HTTP 协议声明为 1.1，默认开启 Keep-Alive（HTTP 1.1 中默认开启 Keep-Alive 选项）。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。主线程不断产生攻击线程直至达到动态饱和，达到 Slowloris 的攻击效果。

对 Slowloris 攻击的改进如下：构建 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。

实现详情请见代码中 slowloris.py 文件，需要 socket 包的环境支持。

D. 混合攻击实现

以上三种攻击均实现为类，并且这些类均继承自 Threading 类（线程类），因此可以在主线程中按比例生

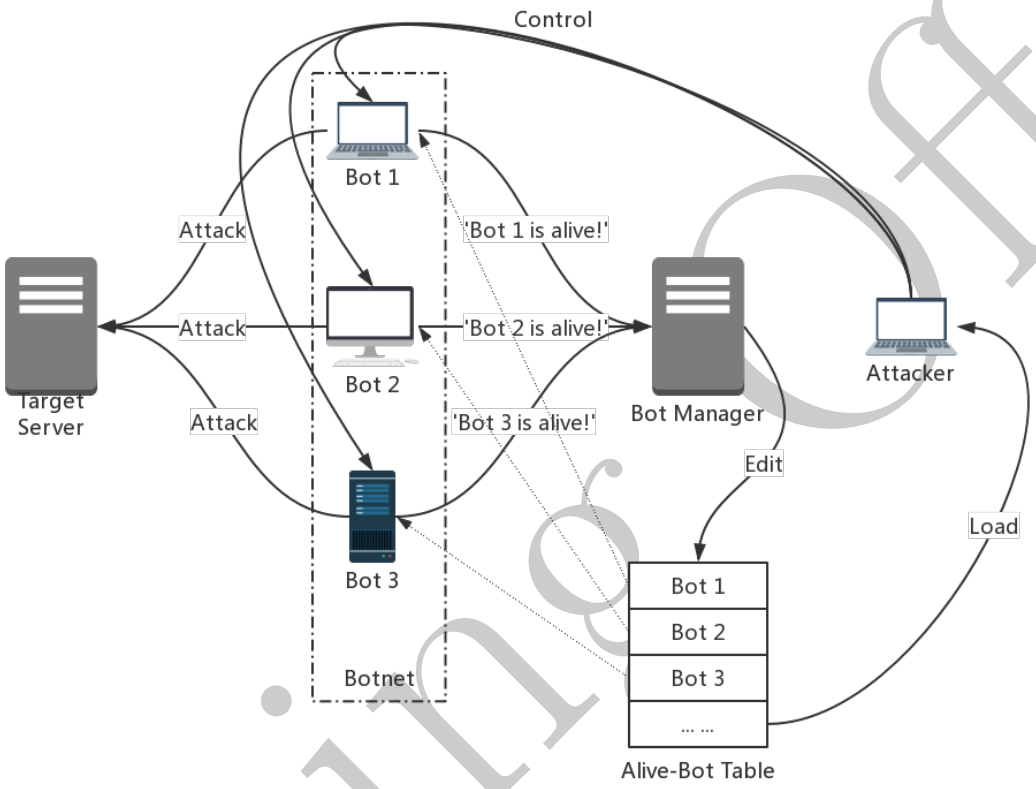


图 2. Botnet 示意图。各个 bot 定期向位于固定服务器的管理程序发送存活消息，管理程序根据存活消息修改存活 bot 表；可移动的控制程序从服务器加载存活 bot 表，控制 bot 发起 DDoS 攻击。

成三种类的线程即可实现简单的混合攻击。

混合攻击的实现在第V中的 bot 端实现进行详细介绍。

V. Botnet 的实现

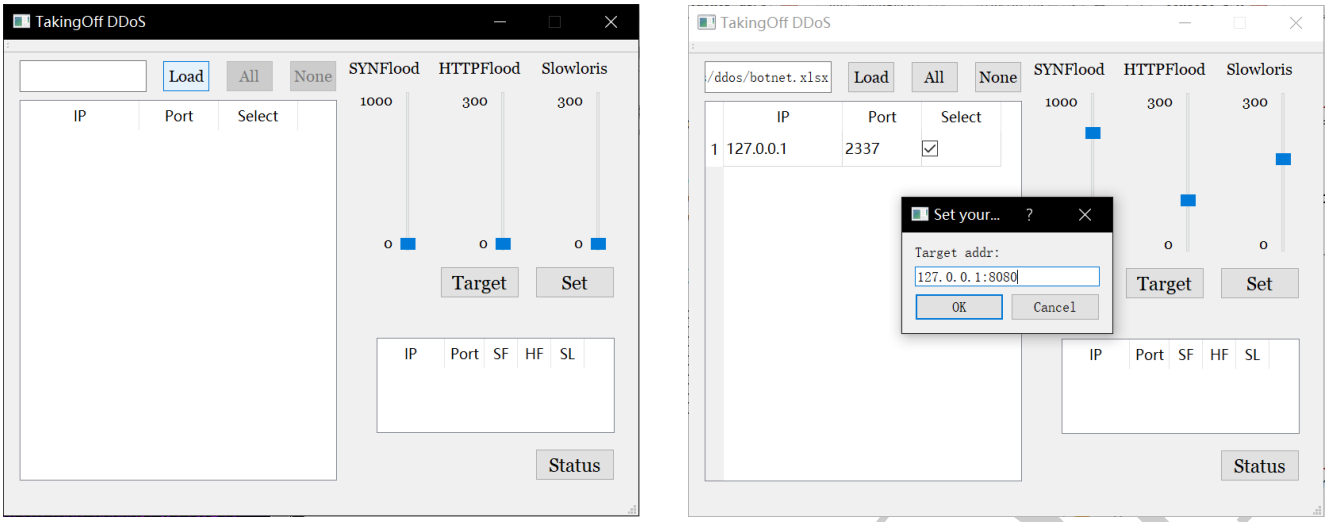
本节对于真实网络环境中的 botnet 实现进行介绍。第V-A节对 botnet 中的 bot 端，即攻击端，的实现进行介绍；第V-B节对控制端的实现进行介绍，并对控制端图形化界面（GIU）的实现和使用方法进行说明。

A. Bot 端

Bot 端为 botnet 中的组成节点，其可以为被控制的个人主机，被控制的第三方服务器等。攻击者通过控制 botnet 中的各个 bot 发起攻击，从而可以更好地隐藏自己，避免被发现甚至反攻击。

Bot 端本质上实现了混合攻击，bot 程序中具有配置信息，主线程，监控线程和报信线程。主线程根据配置信息生成 SYNFlood，HTTPFlood 和 Slowloris 线程进行攻击；监控线程则使用 UDP 协议监听本机某一端口，等待控制端进程发送命令并根据命令执行相应动作，命令如表I所示；报信线程定期向控制端发送存活消息，表明本 bot 程序依然在线，可以参与 DDoS 攻击。

控制命令包括配置命令，测试命令和状态命令：配置命令对 bot 端的各类攻击线程数和攻击目标地址进行设置，通过不同类别的攻击线程比例组合，实现不同的混合攻击；测试命令即为 echo 命令，通过重复消息



(a) 控制程序初始化界面

(b) 控制程序配置攻击目标地址

图 3. 带有 GUI 的 botnet 控制端的控制程序的界面。图3(a)为程序的初始化界面，点击 Load 按钮加载存活 bot 表后便可控制整个 botnet。图3(b)为设置 botnet 攻击目标地址的界面，直接输入 IP 地址和端口即可。

来测试 bot 端是否在线等；状态命令获取 bot 端各类攻击线程的运行状态，包括其配置的线程数目与真实在线的线程数目，通过比对二者即可获知攻击的运行状态是否良好。

Bot 程序长期运行于 Bot 端，并且定时（每几个小时）向控制端的发送存活消息。由于 Bot 程序初始化配置中攻击目标为 “127.0.0.1:8080”，各类攻击线程数目均为 0，因此只会在 Bot 端空转而不发起任何攻击；而一旦控制端进行攻击配置，该 Bot 便转入攻击模式，参与 DDoS 攻击。

Bot 端的实现详情请见代码 attack_bot.py 文件。

B. 控制端

Botnet 在没有控制端时，没有办法进行任何操作或执行任何行为。控制端如同 botnet 的大脑，负责配置和协调整个 botnet。

控制端分作两部分，一部分为部署在固定服务器上的管理程序，另一部分为可移动的控制程序，如图2所示。管理程序负责接收来自 Bot 的存活消息，并维护存活 bot 的 IP 和端口表；控制程序从服务器下载存活 bot 表，使用命令（如表I所示）对表中 bot 进行控制。如此设计，使得 bot 管理和 bot 控制分离。

管理程序的实现详情请见代码 bot_manage.py 文件，命令行界面的控制程序的实现详情请见代码 attacker.py 文件。

为了方便使用，为控制程序实现 GUI。该程序加载 Excel 格式的存活 bot 表，在加载时通过 ECHO 命令测试 bot 是否在线，之后可以勾选表中的 bot 进行配置或查看状态。控制程序的 GUI 界面如图3所示。使用时首先点击 Load 按钮选择 Excel 文件加载存活 bot 表（该表需要从管理程序所在的服务器进行下载），之后所有 bot 的信息显示在左部表格中，可以通过勾选的方式选择控制某个或某些 bot，也可以直接点击 All 或 None 按钮进行全选或全不选；滑动右侧滑块并点击 Set 按钮对选择了的 bot 进行攻击线程数配置，点击 Target 按钮对选择了的 bot 进行攻击目标配置；点击右下侧 Status 按钮获取选择了的 bot 的状态信息，显示于该按钮上方表格中。

VI. 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。

最终服务器的 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、

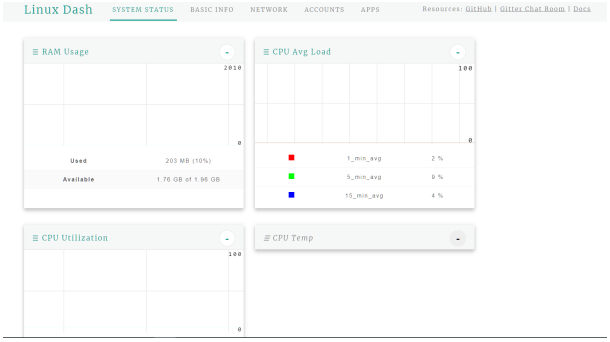


图 4. Linux Dash。通过 81 端口的监控服务器查看整个服务器状态，图中显示了 Web 服务器在未受到攻击的空闲状态下的内存利用情况和 CPU 负载情况。

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，dashboard 界面如图4所示。

在目标 Web 服务器的后台基础上，对 SYNFlood，HTTPFlood 和 Slowloris 攻击进行防御。

VII. DDoS 防御的实现

本节中对防御组的 DDoS 防御实现进行介绍，防御组所有程序均通过 Python3 实现。在第VII-A-VII-C节中对 SYNFlood，HTTPFlood 和 Slowloris 防御的实现进行介绍。

A. *SYNFlood* 防御实现

防火墙在 SYNFlood 防御的过程中相当于带过滤的代理服务器，因此将服务端口置于防火墙监听端口，之后防火墙另一端口连接服务器的监听端口。若监听端口给收到 SYN，则立即返回 SYN+ACK；当监听到已发送过 SYN 的 IP 地址再次发来的 ACK 包后，防火墙作为代理同用户端和服务器端建立连接，中转双方的网络通信。

实现过程中，因为有时发来的网络包的目的地址不一定为防火墙所在的外网 IP，有可能为内网 IP，所以直接监听以外网 IP 为目的地址的网络包会导致错误。开放所有 IP 地址，并监听端口才能正常地将所有发往防火墙的包抓取。此外，转发网络包需要修改目的 IP、目的端口、源 IP 和源端口等信息，校验和应该同时更改，否则网络包会因校验和不通过而被丢弃。

实现详情请见代码中 synflood_defend.py 文件，需要 scapy 包的环境支持。

B. *HTTPlood* 防御实现

HTTPFlood 防御的内容在 SYNFlood 防火墙之后，进一步过滤通过防火墙的 HTTP 请求。本次试验中识别 HTTPFlood 的主要手段有 4 种：1) 判断 HTTP Header 中的特定字段，主要进行鉴别的字段有 User-Agent，referer 和 cokkie，与存储的正常浏览器关键字进行比对，发现异常及将该请求的主机加入黑名单；2) 对每个发起请求的主机进行流量统计，设置访问阈值，一旦流量超过阈值即认为是异常流量源，将其置于黑名单；3) 统计请求返回情况，如果设置非正常文件请求的频次阈值（即出现 404 响应），若超过阈值，则将主机置于黑名单；4) 在网页中镶嵌 js 代码检测是否是正常请求，例如下面这段代码将访问 host 的 cookie 内容，如果没能得到 host 后续的回应（返回的 document.cookie 为空），则认为这一 HTTP 请求是来自浏览器之外的异常请求，将其置于黑名单。

```
<script language=" javascript " >
document.cookie = "yummy_cookie=choco";
document.cookie = "tasty_cookie=strawberry";
console.log(document.cookie);
</script>
```

HTTPFlood 防御具体实现位于 httpflood_defend.py 文件中，将在 synflood_defend.py 之后进行进一步鉴别。

本节最后，需要说明我们尝试过的其他方式——使用缓冲池手段提高服务器荷载能力，即使用另一台服务器回应静态内容，但是由于实验网页的内容较简单，没有长时间加载静态单元导致网页响应不及（而是由于服务器最大连接数限制），使用缓冲池的实验没有获得应有的效果。应指出的是，在真实场景下 HTTPFlood 防御首先需要通过这样增加 CDN 的方法缓解攻击流量，在应用层进行恶意流量的鉴别是十分困难的，甚至这样的鉴别耗费的资源会超过正常服务所需的资源（内存、处理器占用）。

C. *Slowloris* 防御实现

Slowloris 的攻击方式类似于 HTTPFlood，但是服务器对这两种 DDoS 攻击的防御手段却有很大差异。在 Slowloris 中，HTTP 请求会分为很多个分组发送，在这一过程中会占用大量时间，因此在接受完成 HTTP 后再对请求内容进行分析是无法被接受的。

因此，对 Slowloris 防御主要在服务器设置上进行，目前我们的采用的手段是监控已建立的连接，统计连接

时间和接受分组数目。我们的实验网页内容全部为静态，支持的 HTTP 请求类型也只有 GET，即认为网页请求时长很短，因此设置最大连接持续时间为 10s。

针对 Slowloris 会发送许多小短分组的特点，及时终结符合这一特性的连接：设置为接受分组内容长度小于 50 bytes 的分组数超过 30 个时终结连接，并且加入黑名单中。此外在一次 Slowloris 结束后，生成的 HTTP 请求也是含有很多乱码的非法请求，在鉴别之前时会返回 404，超出 HTTPFlood Defend 的限制时也会被终结（但在试验中并未出触发这一情形。）

VIII. 结果

使用本次实验实现的 DDoS 攻击，对一我们的服务器发起 DDoS 攻击，bot 共有两台，分别为一云端服务器，和一笔记本电脑，管理程序部署于同一台云服务器，控制程序运行于同一台笔记本电脑。分别使用 HTTPFlood 和 Slowloris 向待攻击 Web 服务器发起攻击，均可使得目标站点无法被访问，同时可以发现在攻击开始后，待攻击 Web 服务器的 CPU 负载和 CPU 使用率上升。说明 DDoS 攻击成功。攻击情况的演示请见HTTPFlood[4]和Slowloris[5]。

加入防御后，再次使用同样的攻击手段和攻击强度，发现性能监视界面展现的 CPU 利用率等指标相较没有防御而言，表现正常，服务器并没有超负荷运转，说明已经达到了预期的防御效果。此时继续加大攻击强度至 bot 所能支持的最大线程数，也未出现大范围内的浮动。

更多详细情况，请于本小组展示时了解；或访问本小组的 GitHub 组或本项目的 GitHub 主页，了解更多无法于报告中进行展示的结果。

[4]若超链接无法跳转，可以直接访问 https://github.com/TakingOffPKU/DDoS/blob/master/img/httpflood.gif
[5]若超链接无法跳转，可以直接访问 https://github.com/TakingOffPKU/DDoS/blob/master/img/slowloris.gif