抵御模仿人类行为 DDoS 的软件防火墙①

袁 志

(广州大学 华软软件学院, 广州 510990)

摘 要: 模仿人类行为的 HTTP 洪水是一种分布式拒绝服务攻击. 提出一种抵御方法, 它包括三个关键点: 使用会话号标示请求者身份, 通过分析单位时间的请求消息序列发现傀儡主机, 通过丢弃或修改傀儡主机的请求消息中断其攻击. 基于该方法实现了一种软件防火墙, 它包括统计模块和转发模块, 统计模块用于发现傀儡主机, 转发模块用于丢弃或修改傀儡主机的请求消息. 防火墙部署在网站服务器上, 管理员根据网站特征设置运行参数, 能以较小的代价使服务器从 HTTP 洪水中脱困.

关键词: 网站安全; HTTP 洪水; 分布式拒绝访问攻击; 用户识别; 防火墙

Software Firewall Against DDoS Mimicking Human Behavior

YUAN Zhi

(South China Institute of Software Engineering, Guangzhou University, Guangzhou 510990, China)

Abstract: HTTP floods mimicking the human behavior is a kind of distributed denial of service attack. This paper presents a resist method, it includes three key points, uses session ID to identify users, discovers the puppet computer by analyzing the request sequence in unit time, interrupts the attacks by discarding or modifying the request message. A software firewall is implemented based on this method, it includes a statistics module and a forwarding module. The statistics module is used to discover the puppet computers. The forwarding module is used to discard or modify the request messages of puppet computers. The firewall is deployed on a web server, the administrator sets the running arguments according to the site characteristics, help rescuing the server from HTTP floods at a low cost.

Key words: Website security; HTTP Floods; distributed denial of service; user identification; fire wall

模仿正常行为的 HTTP 洪水是分布式拒绝服务攻击的方式之一. 攻击者按照正常业务的流程构造请求脚本, 然后利用攻击程序不断重放脚本. 攻击程序被部署在僵尸网络上, 借助傀儡主机自动发送的高频请求, 使服务器失去对正常用户的响应能力[1].

模仿正常行为的 HTTP 洪水的攻击目标主要是网站服务器, 它消耗服务器资源的途径有三种: 其一, 通过高频请求, 持续占据连接资源, 使正常用户的请求无法到达服务端; 其二, 不遵守缓存约定, 每次请求时都要求服务器重复发送所有数据, 对于包含大量的图片、动画等信息点的网页, 将大量消耗服务器的内存和带宽资源; 其三, 反复访问包含复杂业务逻辑的页面, 使服务器端频繁进行复杂计算和数据库读写操作, 从而大量消耗 CPU 资源和数据库资源. 由于一台傀儡主机上可以运行攻击程序的多个副本, 所以僵尸网络规模不需要太大就可以致瘫一个中小型网站.

1 相关工作

抵御模仿正常行为的 HTTP 洪水攻击的关键在于发现傀儡主机并阻断其请求. 发现傀儡主机建立在识别请求者身份的基础上.

请求者身份识别的已知方法有三种. 一是用 IP 地址来标记请求者身份, 但是, 由于大量的攻击主机位于代理服务器和网关背后, 针对 IP 地址进行拦截, 可

① 收稿时间:2014-07-29;收到修改稿时间:2014-09-22

能损失一个较大区域内的正常用户, 这是网站经营者所不愿见到的. 二是用 MAC 地址来标示请求者身份, 但是要获得客户端的 MAC 地址, 需要客户端允许运行 ActiveX 组件, 显然是难以做到的. 三是通过 Cookie 进行身份识别, 但是这种方法的前提是要求客户端支持 Cookie, 攻击者可以轻易绕过这样的陷阱, 因此也不具有通用性[2].

识别所有请求者身份之后, 要从中发现傀儡主机. 当前区分傀儡主机与人的方法可归类为三种: 超链接诱饵法[3]、语义模型法和时间特征法[4]. 超链接诱饵法和语义模型法可以检测遍历式攻击, 但难以检测模仿正常行为的 HTTP 洪水攻击. 文献[4]指出, 正常用户在网页上的停留时间服从于一定的规律, 因此在单位时间内发起的请求数是有限的, 对具有相同身份标示的请求序列进行时间特征分析, 可以发现傀儡主机. 由于单位页面包含多个信息点, 因此用户发送的请求消息数并不等于浏览页面的次数, 大多数时候前者远远大于后者. 为此, 应用时间特征法, 需要找到可信的计算方法, 为傀儡主机设定特征参数.

在阻断自动请求的问题上, 以图形难题为基础的验证码可以阻断机器自动访问[5], 但是如果只在登录页面使用验证码, 攻击者就可以在人工验证之后再发起自动攻击, 而如果在网站上过多地使用验证码, 又将对正常用户造成困扰. 除了验证码方法外, 其余方法都需要网站开发者设计相关的统计分析和拦截模块[6], 对大多数网站而言难以实现, 技术上的要求也造成抵御洪水攻击的现实困难. 为此, 不但需要有效的方法, 也要提供简易可行的技术方案.

2 抵御方法

目前, 大多数网站都通过验证码方法来对机器自动访问进行第一层阻断. 客户端持续请求的前提是它所发出的请求消息持有合法的访问凭证. 基于这一现实, 本文提出的抵御方法包括三个部分, 第一是用会话号标示请求者身份, 第二是使用时间特征法发现傀儡主机, 第三是修改傀儡主机的请求消息, 用于中断洪水攻击.

2.1 标示请求者身份

客户端首次访问动态网站, 即获得一个会话号. 请求者通过身份验证后, 在后续的请求-应答序列中, 用会话号作为访问凭证. 客户端发起的每一次请求, 其请求消息的头部都携带有会话号, 支持 Cookie 的客户端, 会话号被存储在 Cookie 段中, 不支持 Cookie 的客户端, 会话号存储在请求行中. 因此, 会话号可以用来标示一个客户端的一次连续请求. 在不同的动态网站技术(如 PHP,JSP,ASP,ASP.NET 等)中, 会话号有不同的标记名(比如 ASPSessionId, JSessionID), 这是可以预知的.

2.2 发现傀儡主机

网站包含多个页面, 一个页面包含一个或多个信息点(比如 HTML、图片、样式表、动画等). 在客户端的一次访问中, 每一个信息点都会导致一个请求-应答对. 由于正常用户在浏览页面时有一定的停留时间, 在正常的用户环境和预知的用户规模下, 单位时间内, 请求消息数的峰值可以预知. 标准浏览器(如 IE, FireFox 等)会缓存部分已经获得的信息点, 在下一次对同一页面发起访问时, 只要求服务端传送更改过的信息点. 服务端接到请求后, 对于没有更新的信息点, 它将生成一个“304”响应报文, 提示浏览器该信息点没有更新, 对于客户端以前没有访问过的信息点或者更新了的信息点, 则生成一个“200”响应报文, 报文中包含信息点内容. 对单位时间内具有相同会话号的请求序列进行分析, 可以用两种方法结合起来发现傀儡主机.

第一种方法, 看请求序列是否遵守缓存机制, 如果一个请求序列包含对图片、样式表、动画等信息点的大量重复请求, 则可以判断该请求序列来自于攻击程序. 设可缓存的信息点类型集为 C, 如 C={jpg,gif, css,swf,png}, 单个网页上含有该类型信息点数最大值为 MaxQ,单位时间内用户强制刷新请求的最大可能次数为 MaxK, 请求序列对 C 集的重复请求数峰值为 MaxR, 则 MaxR≤MaxK×MaxQ. 这种规律只有在客户端是傀儡主机时会被打破. 统计单位时间内具有相同会话号的请求消息对 C 集的重复请求数 R, 如果 R>MaxK×MaxQ,则可以认为该请求序列来自于傀儡主机.

第二种方法, 看请求消息数是否超过预设的峰值. 设单位时间内一个用户的点击数峰值为 MaxC, 网站中信息点最多的网页包含的信息点数为 MaxI, 一个用户发起的请求消息数峰值为 MaxP, 则 MaxP≤MaxC×MaxI. 这种规律只有在客户端是傀儡主机时会被打破. 统计单位时间内具有相同会话号的请

求消息数 P, 如果 P>MaxC×MaxI, 则可以认为该请求序列来自于傀儡主机.

2.3 中断洪水攻击

对请求消息进行分析, 如果发现消息中的会话号是一个傀儡会话号, 有两种模式来阻断其攻击. 第一种模式, 直接丢弃请求消息, 这样请求就无法到达服务端. 第二种模式, 对消息进行修改,将消息中的会话号修改为一个错误值, 将连接状态修改为 Close, 然后转发给网站. 网站收到请求消息后, 就会发现消息中持有的访问凭证不合法, 从而自动转入到验证页面, 同时, 由于连接状态修改为 Close, 服务器将会关闭连接. 攻击程序可以不断地修改请求消息中的会话号, 来隐瞒自己的傀儡身份, 但如此一来, 它也会不断丧失有效的访问凭证, 不得不反复进行身份验证.

由于验证页面所需的服务器资源很小, 实际应用中偏向于采用第二种模式. 通过处理, 作为洪水攻击源的傀儡主机要想维持攻击, 将不得不频繁进行身份验证, 最终迫使攻击者放弃攻击, 每次身份验证所需的时间间隔, 也可以使服务器从高消耗中脱身. 在实际应用中, 正常客户连续不断访问页面的情况也许存在, 但其请求数要超过阈值是困难的, 这是因为抵御方法中设置的参数留有很大的余地. 即便出现极端情况, 用户重新进行一次身份验证即可.

3 防火墙的设计与实现

在 Windows 操作系统下按照本文方法实现了一个软件防火墙. 防火墙使用 VC 开发, 用到了 Winpcap 开发包和 Winsock2 组件. 防火墙包含两个模块: 统计模块和转发模块, 此外提供一个运行界面, 用来设置防火墙参数和查看异常会话. 防火墙部署在网站服务器上, 对每一个传入的数据包进行分析处理, 防火墙的工作原理如图 1.

图 1 中, ①是未发现傀儡会话时的请求消息传输路径, ②和③是发现了傀儡主机时的请求消息传输路径, 其中②是正常用户请求消息的传输路径, ③是傀儡主机请求消息的传输路径.

3.1 统计模块

统计模块基于 Winpcap 开发, 该模块监听服务器上接收的数据帧, 从中过滤出 HTTP 请求消息, 维护一个“会话号-请求时间-请求行”集合, 用以保存 5 分钟的请求序列. 经过分析发现傀儡主机的会话号, 将傀儡会话号存放在一个数据文件中. 工作过程如下:

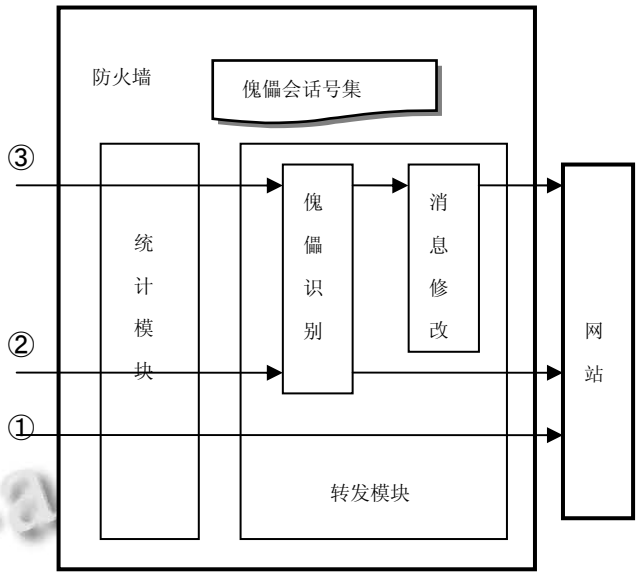

图 1　防火墙工作原理图

① 读取一帧数据, 对其进行解码分析, 看其是否满足“网络层协议为 IP,传输层协议为 TCP,端口为网站端口, 应用层数据第一行含有‘GET’或‘POST’以及‘HTTP’字串”, 如果不满足则结束本次处理, 否则执行②;

② 从应用层数据包中读取 Request-Line 段和 Cookie 字段, 获得请求会话号, 在“会话号-请求时间-请求行”集合中增加一个新成员. 对新成员赋值, 会话号←消息中的会话号, 请求时间←当前时间, 请求行←消息中的请求行.

③ 对“会话号-请求时间-请求行”集合进行一次统计, 提取 P> MaxC×MaxI 或者 R>MaxK×MaxQ 的会话号, 将其计入傀儡会话号集.

④ 删除“会话号-请求时间-请求行”集合中与当前时间相差大于 5 分钟的成员, 结束本次处理.

3.2 转发模块

转发模块在傀儡会话号集的配合下工作, 对每一个传入的 TCP 数据包进行处理. 该模块监视傀儡会话号数据文件, 当文件中存有数据时, 才启动傀儡请求匹配, 当匹配成功时才修改请求消息. 转发模块基于 Winsock2 开发. Winsock2 提供了 API(应用程序编程接口)、SPI(服务提供者接口)和 ws2_32.dll. 应用程序通过调用 API 函数来实现数据收发, API 通过 SPI 来调用

底层传输服务, 基于 SPI 可以使应用程序在调用 API 函数时自动调用自定义的传输服务模块. 设计一个动态链接库 httpModule.dll, 用来封装自定义的传输服务, 其中主要功能包括在 WSPRecv()和 WSPStartup()函数中. WSPRecv()函数中将收到的数据包与傀儡会话号进行匹配, 如果匹配成功, 则修改请求报文. WSPStartup() 函数建立 API 的 Recv() 函数与 httpModule.dll 中的 WSPRecv()函数的对应关系,并提供外部调用入口. 安装防火墙时, 调用 WS2_32.dll 中的 WSCInstallProvider()函数来安装 httpModule.dll, 安装完成后 httpModule.dll 将作为协议链的最后一层被调用, 在请求数据包抵达应用程序前工作. 当防火墙停用时, 调用 WS2_32.dll 中的 WSCDeinstallProvider()函数卸载 httpModule.dll. WSPRecv()函数的工作流程如下:

① 检查傀儡会话号集, 如果内容为空则结束本次处理; 否则执行②;

② 将请求数据消息中的会话号与“傀儡会话号集”进行匹配, 如果无匹配项则结束本次处理; 否则执行③;

③ 修改请求消息, 将会话号修改为一个错误值, 将连接状态字修改为“close”, 结束本次处理.

3.3 运行参数的设置

参数分为两类, 第一类是网站特征参数, 第二类是用户特征参数. 网站特征参数包括单页最大信息点数 MaxI 和单页最大缓存点数. 每个页面的信息点数可以通过页面的 HTML 文档分析获得. 比如一个 HTML 页面包含 9 个图片, 则共有 10 个信息点(包含 HTML 文件本身), 其中 9 个信息点是可以缓存的. 用户特征参数包括用户最高点击频率 MaxR 和强制刷新频率 MaxK, 都是经验值.

参数值由管理员根据网站的业务特点来设置, 应尽可能取大一些. 经验上, 一个正常用户刷新网页的时间间隔不可能低于 10 秒, 而强制刷新页面的时间间隔不可能低于 60 秒, 而单页最大信息点数可以依照最大的页面来设置. 为了防止错误识别, 设置一段统计时间, 默认为 5 分钟, 即: 统计时间内正常用户访问最大信息点网页不可能超过 30 次, 强制刷新时间不可能超过 5 次.

参数值用于统计模块, 由于统计模块在旁路监听用户请求, 因此参数值的设置对请求-应答不会造成延时, 而其计算量主要用于一段时间(如 5 分钟)的汇总, CPU 和内存的开销可以忽略.

4 实验

选择一个 ASP.NET 网站作为测试对象, 将防火墙部署在网站服务器上. 依据网站内容和客户习惯, 设置防火墙运行参数, 如表 1.

表 1 防火墙运行参数

网站特征参数			正常用户特征参数		请求统计时间跨度
单页最大信息点数	缓存信息点类型	单页最大缓存信息点数	用户最高点击频率	用户最高强制刷新频率	
10	jpg, gif, css, swf, png	9	10 秒/次	60 秒/次	5 分钟

根据表 1 的设定可以推算傀儡主机的特征, 5 分钟内如果一个客户端请求数大于 300 或者对缓存类型的信息点请求重复数超过 45, 则认为是傀儡主机.

为了模仿攻击行为, 先通过人工操作对网站进行一个序列访问, 访问序列包含了 5 个网页共 32 个信息点, 其中有 27 个信息点属于可缓存信息点. 记录下访问脚本并用程序来封装脚本, 程序每隔 15 秒钟重放一次脚本, 攻击程序部署在两台 PC 上. 攻击启动 1 分 20 秒后, 两台攻击主机上得到的响应都变为身份验证页面. 再过 5 分钟后, 查看防火墙记录, 异常会话数据如表 2.

表 2 防火墙记载的异常会话

会话号	请求数	缓存文件的重复请求数
nj5grsyehyn5x4zj3de3hdj2	631	252
mxf0ss55nav31l55z1nbib45	633	253

按照攻击程序的逻辑, 攻击主机在 1 分钟内重放脚本次数为 4 次, 因此发出请求数 P=4*32=128, 对缓存文件的重复请求数 R=27*3=81. 此时, 防火墙还不能识别攻击程序. 在 1 分 15 秒后, P=5*32=160, R=4*27=108, r>maxR,, 因此被防火墙认定是傀儡主机并加以拦截是符合预期的. 表 2 中, 显示的是“会话号-请求时间-请求行”集合 5 分钟的统计数据, 也符合攻

击程序的逻辑.

重复实验, 不断地调低攻击程序的重放频率, 发现当频率高于 75 秒/次, 防火墙都能准确识别出傀儡主机, 当频率低于 75 秒/次时傀儡主机变得不可识别. 对其进行分析, 当频率等于 75 秒/次, 由于每次重放都发送 27 个缓存信息点请求, 因此, 5 分钟内发送的请求消息数小于 90, 已经不符合防火墙预设的傀儡主机特征. 进一步实验发现, 即便攻击程序对缓存信息点不重复请求, 当他的重放频率高于 30 秒/次, 防火墙同样可以发现傀儡主机.

5 结语

本文分析了模仿正常行为的 HTTP 洪水攻击的特点, 提出了一种抵御方法, 抵御方法的关键部分有三点, 一是使用会话号来识别请求者, 二是统计分析请求消息来发现傀儡主机, 三是通过丢弃或修改傀儡主机的请求消息, 阻断其持续攻击. 依照该方法实现了一种软件防火墙, 防火墙能够按需装卸且应用简易, 部署在网站服务器上, 能以较小代价使服务器从 HTTP 洪水攻击中脱身.

参考文献

1 孙长华,刘斌.分布式拒绝服务攻击研究新进展综述.电子学报,2009,37(7):1562–1569.

2 肖惠,王立华.Web 日志挖掘中的用户识别算.计算机系统应用,2011,20(5):223–226.

3 Gavrilis D, Chatzis I, Dermatas E. Flash crowd detection using decoy hyperlinks, 2007 IEEE International Conference on Networking, Sensing and Control. April 2007. 466–470.

4 Oikonomou G, Mirkovic J. Modeling Human Behavior for Defense Against Flash-Crowd Attacks. Proc. of IEEE International Conference on Communications. June 2009. 14–18.

5 Kandula S, Katabi D, Jacob M, Berger A. Surviving Organized DDos Attacks That Mimic Flash Crowds. USENIX Symposium on Network Systems Design and Implementation, May 2005.

6 袁志.一种抵御 HTTP 洪水攻击的方法.计算机应用与软件,2012,7:271–273.