

# 慢速 http 拒绝服务攻击及防御

Auth : Cryin'

Date : 2016.03.03

Link : <https://github.com/Cryin/Paper>

## 概述

HTTP-FLOOD 攻击是一种专门针对于 Web 的应用层 FLOOD 攻击,攻击者操纵网络上的肉鸡,对目标 Web 服务器进行海量 http request 攻击,直到服务器带宽被打满,造成了拒绝服务。

由于伪造的 http 请求和客户正常请求没有区别,对于没有流量清洗设备的用户来说,这无疑就是噩梦。

慢速 http 拒绝服务攻击则是 HTTP-FLOOD 攻击的其中一种。常见的慢速 DoS 攻击压力测试工具有 SlowHTTPTest、Slowloris 等

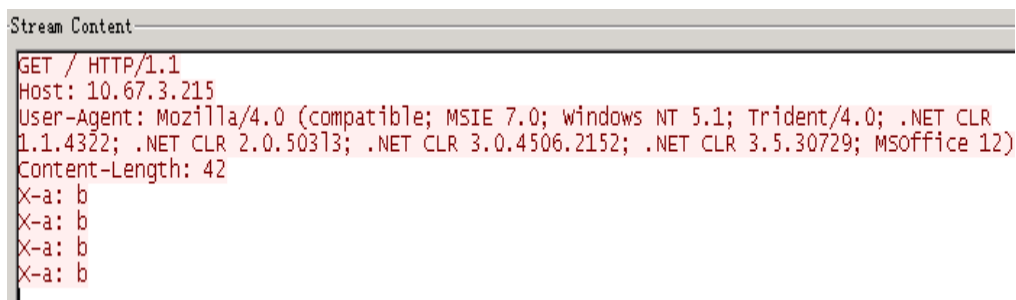
## 攻击原理

Web 应用在处理 HTTP 请求之前都要先接收完所有的 HTTP 头部,因为 HTTP 头部中包含了一些 Web 应用可能用到的重要的信息。攻击者利用这点,发起一个 HTTP 请求,一直不停的发送 HTTP 头部,消耗服务器的连接和内存资源。抓包数据可见,攻击客户端与服务器建立 TCP 连接后,每 40 秒才向服务器发送一个 HTTP 头部,而 Web 服务器再没接收到 2 个连续的\r\n 时,会认为客户端没有发送完头部,而持续的等等客户端发送数据。如果恶意攻击者客户端持续建立这样的连接,那么服务器上可用的连接将一点一点被占满,从而导致拒绝服务。这种攻击类型称为慢速 HTTP 拒绝服务攻击。

## 分类

慢速 HTTP 拒绝服务攻击经过不断的演变和发展,其主要分为以下几类:

- ◆ Slow headers: Web 应用在处理 HTTP 请求之前都要先接收完所有的 HTTP 头部,因为 HTTP 头部中包含了一些 Web 应用可能用到的重要的信息。攻击者利用这点,发起一个 HTTP 请求,一直不停的发送 HTTP 头部,消耗服务器的连接和内存资源。抓包数据可见,攻击客户端与服务器建立 TCP 连接后,每 30 秒才向服务器发送一个 HTTP 头部,而 Web 服务器再没接收到 2 个连续的\r\n 时,会认为客户端没有发送完头部,而持续的等等客户端发送数据。



```
Stream Content
GET / HTTP/1.1
Host: 10.67.3.215
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; windows NT 5.1; Trident/4.0; .NET CLR
1.1.4322; .NET CLR 2.0.50373; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MSoffice 12)
Content-Length: 42
X-a: b
X-a: b
X-a: b
X-a: b
```

- ◆ Slow body: 攻击者发送一个 HTTP POST 请求,该请求的 Content-Length 头部值很大,使得 Web 服务器或代理认为客户端要发送很大的数据。服务器会保持连接准备接收数

据，但攻击客户端每次只发送很少量的数据，使该连接一直保持存活，消耗服务器的连接和内存资源。抓包数据可见，攻击客户端与服务器建立 TCP 连接后，发送了完整的 HTTP 头部，POST 方法带有较大的 Content-Length，然后每 10s 发送一次随机的参数。服务器因为没有接收到相应 Content-Length 的 body，而持续的等待客户端发送数据。

```
Stream Content
POST /py HTTP/1.1
Host: 10.67.3.215:82
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2)
Referer: http://code.google.com/p/slowhttptest/
Content-Length: 4096
Connection: close
Content-Type: application/x-www-form-urlencoded
Accept: text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

foo=bar&FuH5WH1=P&Gn4CGUXP4Q24=i5Gkek7HN6yIHPNlG6rryHxI4BA&0uPaRgrwzmrSHZOUom=ky1xzRVYjIL
Z65owKJE68o1qua2En&JLHAh0k153YnBuQu1LwLa7xewXkFpi=yrnwOdxXXOJAP2P9rau2y&N5et7pFAeJn=SL00u
zp6Qzhvt8j8qud1n7ikmnl
```

- ◆ Slow read: 客户端与服务器建立连接并发送了一个 HTTP 请求，客户端发送完整的请求给服务器端，然后一直保持这个连接，以很低的速度读取 Response，比如很长一段时间客户端不读取任何数据，通过发送 Zero Window 到服务器，让服务器误以为客户端很忙，直到连接快超时前才读取一个字节，以消耗服务器的连接和内存资源。抓包数据可见，客户端把数据发给服务器后，服务器发送响应时，收到了客户端的 ZeroWindow 提示（表示自己没有缓冲区用于接收数据），服务器不得不持续的向客户端发出 ZeroWindowProbe 包，询问客户端是否可以接收数据。

```
Stream Content
GET /py/ HTTP/1.1
Host: 10.67.3.215:82
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:5.
Firefox/5.0.1
Referer: http://code.google.com/p/slowhttptest/

HTTP/1.1 200 OK
Date: Wed, 22 Apr 2015 16:30:50 GMT
Server: Apache/2.4.4 (win32) OpenSSL/0.9.8y PHP/5.4.16
X-Powered-By: PHP/5.4.16
Set-Cookie: PHPSESSID123=it9c2lk0ifouds8ic2kuau50n5; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check
Pragma: no-cache
Set-Cookie: PHPSESSID123=it9c2lk0ifouds8ic2kuau50n5; expires=w
GMT; path=/; httponly
Set-Cookie: NumVisitsPhp=1
```

## 易被攻击的 web 服务器

慢速 HTTP 拒绝服务攻击主要利用的是 thread-based 架构的服务器的特性，这种服务器会为每个新连接打开一个线程，它会等待接收完整 HTTP 头部才会释放连接。比如 Apache 会有一个超时时间来等待这种不完全连接（默认是 300s），但是一旦接收到客户端发来的数据，这个超时时间会被重置。正是因为这样，攻击者可以很容易保持住一个连接，因为攻击者只需要在即将超时之前发送一个字符，便可以延长超时时间。而客户端只需要很少的资源，便可以打开多个连接，进而占用服务器很多的资源。

经验证，Apache、httpd 采用 thread-based 架构，很容易遭受慢速攻击。而另外一种 event-based 架构的服务器，比如 nginx 和 lighttpd 则不容易遭受慢

速攻击。

## 防御措施

### ◆ Apache

- `mod_reqtimeout` : Apache 2.2.15 后, 该模块已经被默认包含, 用户可配置从一个客户端接收 HTTP 头部和 HTTPbody 的超时时间和最小速率。如果一个客户端不能在配置时间内发送万头部或 body 数据, 服务器会返回一个 408REQUEST TIME OUT 错误。配置文件如下:

```
< IfModule mod_reqtimeout.c >
RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500
< /IfModule >
```

- `mod_qos`: Apache 的一个服务质量控制模块, 用户可配置各种不同粒度的 HTTP 请求阈值, 配置文件如下:

```
< IfModule mod_qos.c >
/# handle connections from up to 100000 different IPs
QS_ClientEntries 100000
/# allow only 50 connections per IP
QS_SrvMaxConnPerIP 50
/# limit maximum number of active TCP connections limited to 256
MaxClients 256
/# disables keep-alive when 180 (70%) TCP connections are occupied
QS_SrvMaxConnClose 180
/# minimum request/response speed (deny slow clients blocking the server,
keeping connections open without requesting anything
QS_SrvMinDataRate 150 1200
< /IfModule >
```

### ◆ WebSphere

- 限制 HTTP 数据的大小

详细参考链接:

[http://www.ibm.com/developerworks/cn/websphere/techjournal/1210\\_lansche/1210\\_lansche.html#new-step32](http://www.ibm.com/developerworks/cn/websphere/techjournal/1210_lansche/1210_lansche.html#new-step32)

- 设置 keepalive 参数

1、更改http server的配置文件参数KeepAlive。

原因: 这个值说明是否保持客户与HTTP SERVER的连接, 如果设置为ON, 则请求数到达MaxKeepAliveRequests设定值时请求将排队, 导致响应变慢。

方法: 打开ibm http server安装目录, 打开文件夹conf, 打开文件httpd.conf, 查找KeepAlive值, 改ON为OFF, 其默认为ON

### ◆ Weblogic

- 在配置管理界面中的协议->一般信息下设置 **完成消息超时** 小于 200, wvvs 即不会爆出该漏洞, 如图:

AdminServer的设置

配置 协议 日志记录 调试 监视 控制 部署 服务 安全 注释

一般信息 HTTP jCOM IIOP 通道

保存

使用此页可以配置此服务器可以使用的各种通信协议的连接设置。此页上的所有设置均适用于使用服务器通道均可覆盖这些设置。

完成消息超时: 80

空闲连接超时: 65

- 在配置管理界面中的协议->HTTP 下设置 **POST 超时**、**持续时间**、**最大 POST 大小**等选项，可防止其它类型的慢速 HTTP 拒绝服务攻击。如图：

配置 协议 日志记录 调试 监视 控制 部署 服务 安全 注释

一般信息 HTTP jCOM IIOP 通道

保存

基于 Web 的客户机使用 HTTP (超文本传输协议) 与 WebLogic Server 进行通信。  
使用此页可以定义该服务器的 HTTP 设置。

默认 Web 应用程序上下文根:

POST 超时: 100

最大 POST 大小: -1

☒ 启用“保持活动”

持续时间: 100

HTTPS 持续时间: 100

#### ◆ IHS 服务器

- 请先安装最新补丁包，然后启用 mod\_reqtimeout 模块，在配置文件中加入：  
LoadModule reqtimeout\_module modules/mod\_reqtimeout.so  
为 mod\_reqtimeout 模块添加配置：  
<IfModule mod\_reqtimeout.c>  
RequestReadTimeout header=10-40,MinRate=500 body=10-40,MinRate=500  
</IfModule>

对于 HTTPS 站点，建议 header=20-40, MinRate=500。

参见: <http://www-01.ibm.com/support/docview.wss?uid=swg21652165>

#### ◆ Nginx

- 通过调整\$request\_method, 配置服务器接受 http 包的操作限制;
- 在保证业务不受影响的前提下, 调整 client\_max\_body\_size, client\_body\_buffer\_size, client\_header\_buffer\_size, large\_client\_header\_buffers, client\_body\_timeout, client\_header\_timeout 的值, 必要时可以适当增加;
- 对于会话或者相同的 ip 地址, 可以使用 HttpLimitReqModule and HttpLimitZoneModule 参数去限制请求量或者并发连接数;
- 根据 CPU 和负载的大小, 来配置 worker\_processes 和 worker\_connections 的值, 公式是: max\_clients = worker\_processes \* worker\_connections。

## 总结

传统的流量清洗设备针对 CC 攻击, 主要通过阈值的方式来进行防护, 某一个客户在一定的周期内, 请求访问量过大, 超过了阈值, 清洗设备通过返回验证码或者 JS 代码的方式。这种防护方式的依据是, 攻击者们使用肉鸡上的 DDoS 工具模拟大量 http request, 这种工具一般不会解析服务端返回数据, 更不会解析 JS 之类的代码。因此当清洗设备截获到 HTTP 请求时, 返回一段特殊 JavaScript 代码, 正常用户的浏览器会处理并正常跳转不影响使用, 而攻击程序会攻击到空处。

而对于慢速攻击来说, 通过返回验证码或者 JS 代码的方式依然能达到部分效果。但是根据慢速攻击的特征, 可以辅助以下几种防护方式:

1、周期内统计报文数量。一个 TCP 连接, HTTP 请求的报文中, 报文过多或者报文过少都是有问题的, 如果一个周期内报文数量非常少, 那么它就可能是慢速攻击; 如果一个周期内报文数量非常多, 那么它就可能是一个 CC 攻击。

2、限制 HTTP 请求头的最大许可时间。超过最大许可时间, 如果数据还没有传输完成, 那么它就有可能是一个慢速攻击。

## 参考

- [1] <http://blog.nsfocus.net/cc-attack-defense/>
- [2] <http://www.freebuf.com/tools/40413.html>