[HW]Dropper

109550127 宋哲頤

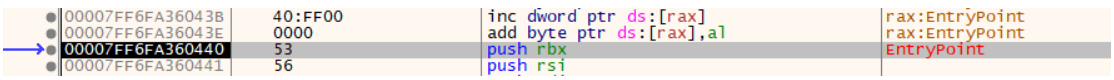
用 Detect it easy 發現 此檔案是有殼的(壓縮檔案用)

```
upx -d dropper_43741eb13c4a767e.exe
```

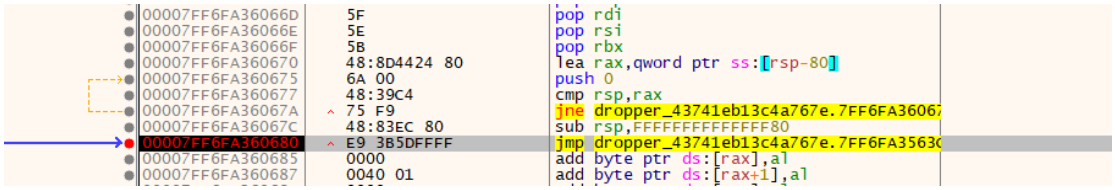
```
       File size         Ratio      Format      Name
--------------------   ------   -----------   -----------
     39424 <-     16896   42.86%    win64/pe     dropper_43741eb13c4a767e.exe

Unpacked 1 file.
```

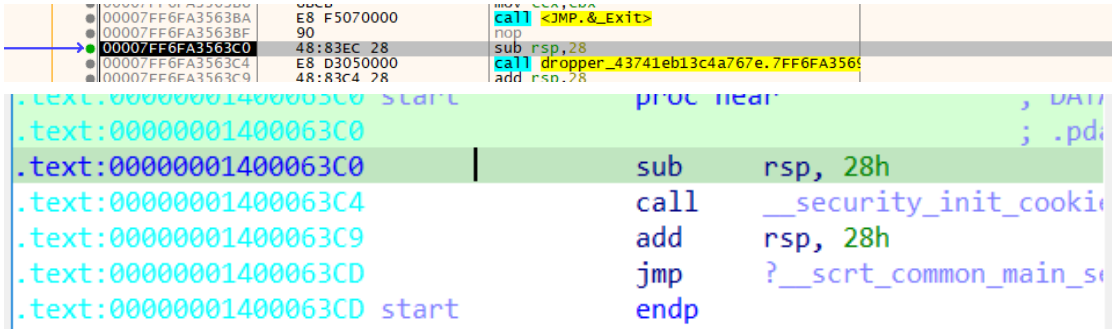
所以利用 upx unpack 功能來讓 ida pro 可以輕易分析裏面的 code，因為是有裝殼的關係，用 x64dbg 去解析發現 dropper 裏頭的內容，在 upx 還沒執行時，都是無法解析其組合語言的。於是觀察最一開始的指令，發現在 stack 上放了 rbx 的舊值



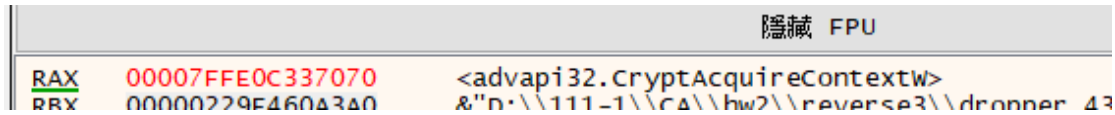
所以在 upx 快執行完時，應該會再去取得這份舊值，之後就會跳到正式的 dropper 來執行，所以就在放 rbx 舊值的地方，設置硬體存取中斷點，來執行程式，並找到要 JUMP 到 dropper 的地方



之後就順利找到 original entry point: 63C0



然後在 ida 細看時發現，sub_140001A00 函式，跟上課有提到的 getprocaddress，應該是要調用一些 windows 的 api，最後看 x64dbg 顯示的 rax 值應證了這件事，於是就逐一去尋找這支程式用了多少 api。



發現使用了不少，都跟加密有關。

```
BOOL CryptAcquireContextA(
  HCRYPTPROV *phProv,
  LPCSTR     szContainer,
  LPCSTR     szProvider,
  DWORD      dwProvType,
  DWORD      dwFlags
)

BOOL CryptCreateHash(
  HCRYPTPROV hProv,
  ALG_ID     Algid,
  HCRYPTKEY  hKey,
  DWORD      dwFlags,
  HCRYPTHASH *phHash
)

BOOL CryptHashData(
  HCRYPTHASH hHash,
  const BYTE *pbData,
  DWORD      dwDataLen,
  DWORD      dwFlags
)

BOOL CryptDeriveKey(
    HCRYPTPROV hProv,
    ALG_ID     Algid,
  HCRYPTHASH hBaseData,
    DWORD      dwFlags,
  HCRYPTKEY  *phKey
)

BOOL CryptDestroyHash(
  HCRYPTHASH hHash
)
```

```
v75[5] = -6;
sub_140001030(v75, 6i64, v14);
v43 = sub_1400036D0(v71, v75);
v44 = sub_1400036D0(v58, v76);
Sleep = (void (__fastcall *)(DWORD))sub_140001A00(v44, v43);
if ( !CryptoAcquireContextW(&phProv, 0i64, 0i64, 1i64, 0) )
{
  if ( GetLastError() != -2146893802 )
    return 0;
  if ( !CryptoAcquireContextW(&phProv, 0i64, 0i64, 1i64, 8) )
    return 0;
}
if ( !CryptCreatHash(phProv, 32772i64, 0i64, 0i64, &phHash) )
  return 0;
if ( !phHash )
  return 0;
if ( !CryptHashData(phHash, pbData, 1i64, 0i64) )
  return 0;
if ( !CryptDriveKey(phProv, 26625i64, phHash, 1i64, &phKey) )
  return 0;
CryptDestroyHash(phHash);
Sleep(2592000000i64);
LODWORD(Size) = 30;
Block = (BYTE *)malloc(0x1Eui64);
if ( !Block )
  return 0;
sub_140001260(Block, (unsigned int)Size);
sub_140001120(Block, (unsigned int)Size, &unk_14000B050, (unsigned int)Size);
if ( !CrpytEncrypt(phKey, 0i64, 1i64, 0i64, Block, (DWORD *)&Size, Size) )
  return 0;
if ( RegCreateKeyA(HKEY_CURRENT_USER, "CS_2022", &phkResult) )
  return 0;
if ( !RegSetValueExA(phkResult, "CS_2022", 0i64, 1i64, Block, Size) )
{
  RegCloseKey(phkResult);
  free(Block);
}
return 0;
```

在 IDA 中發現了跟 flag 有關的 code

```
  return 0;
sub_140001260(flag, (unsigned int)Size);
sub_140001120(flag, (unsigned int)Size, &unk_14000B050, (unsigned int)Size);
if ( !CryptEncrypt(phKey, 0i64, 1i64, 0i64, flag, (DWORD *)&Size, Size) )
  return 0;
if ( RegCreateKeyA(HKEY_CURRENT_USER, "CS_2022", &phkResult) )
  return 0;
if ( !RegSetValueExA(phkResult, "CS_2022", 0i64, 1i64, flag, Size) )
{
  RegCloseKey(phkResult);
  free(flag);
}
return 0;
```

但在 encrypt 前的 sleep 會導致程式永遠無法執行到加解密的步驟

```
  return 0;
CryptDestroyKey(phHash);
sleep(2592000000i64);
LODWORD(Size) = 30;
flag = (BYTE *)malloc(0x1Fui64);
```

所以我在 x64dbg 使用 nop 填充 dump 掉 sleep，之後執行，之後看到 IDA code

有呼叫 RegCreatKey

查詢 RegCreatKey 的 api，發現會幫我註冊登入檔

```
return 0;
f ( RegCreateKeyA(HKEY_CURRENT_USER, "CS_2022", &phkResult) )
return 0;
```

於是上登錄檔查看，找到了 flag。