Please COM again

Clément Rouault (hakril)

But de la rump

COM du point de vue d'un reverser

- Quels outils/techniques pour RE du COM
- Cas concret:
  - Win32/Aibatook
  - Malware RE à ESET en 2014
  - Cible les clients de banques japonaises
  - Utilise COM pour communiquer avec Internet Explorer
- Les techniques sont présentées avec quelques années de recul
  - statique
  - dynamique

Component Object Model

- COM: système orienté-objet pour créer des composant logiciels binaires pouvant interagir.

- COM Client
  - Il voit juste des interfaces
    - Une liste de fonction
      - Avec des getter/setter au passage
    - Un Interface ID (IID)
  - On va juste voir des vtable partout
  - Aucune info sur l'implémentation
- COM Server
  - COM class
    - Implémentation de 1 ou plusieurs interface
  - Un Class ID (CLSID)
  - Le serveur peut être un peu partout
    - Une DLL (In-process)
    - Un autre process de la machine (Out-of-proces)
    - Un autre process d'une autre machine

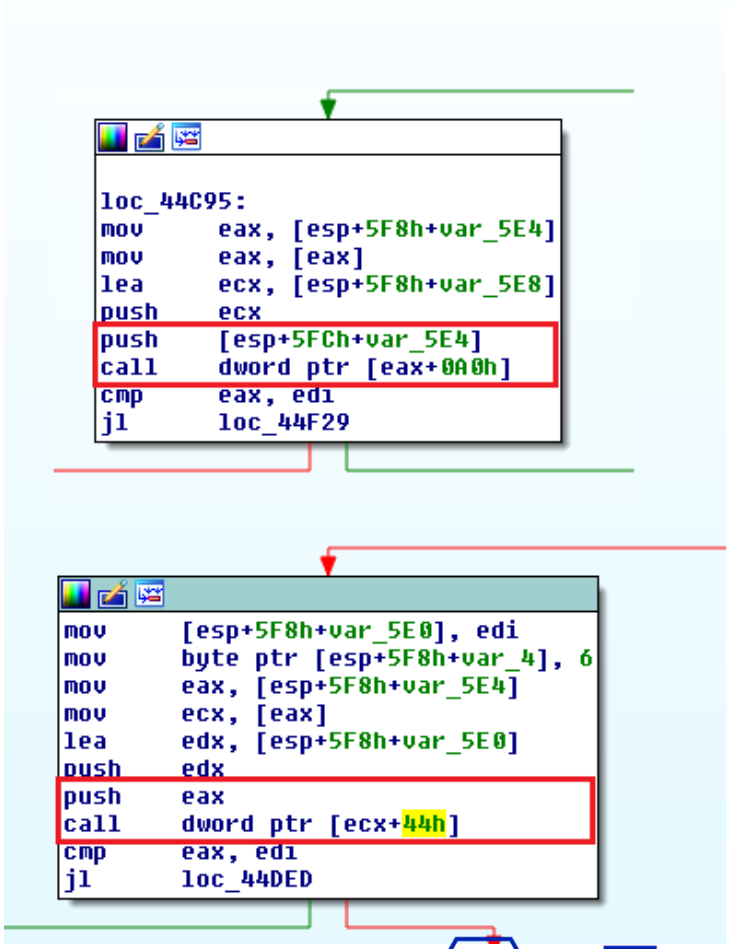
Exemple d'interface

```
typedef struct ICallFrameEventsVtbl
{
    BEGIN_INTERFACE

    HRESULT ( STDMETHODCALLTYPE *QueryInterface )(
        ICallFrameEvents * This,
        /* [in] */ REFIID riid,
        /* [annotation][iid_is][out] */
        _COM_Outptr_  void **ppvObject);

    ULONG ( STDMETHODCALLTYPE *AddRef )(
        ICallFrameEvents * This);

    ULONG ( STDMETHODCALLTYPE *Release )(
        ICallFrameEvents * This);

    HRESULT ( STDMETHODCALLTYPE *OnCall )(
        ICallFrameEvents * This,
        /* [in] */ ICallFrame *pFrame);

    END_INTERFACE
} ICallFrameEventsVtbl;

interface ICallFrameEvents
{
    CONST_VTBL struct ICallFrameEventsVtbl *lpVtbl;
};
```

ExaTrack

RE du COM: Aibatook

- Premier vrai malware en stage
  - Toujours cru que RE sur Windows c'était lire des API
- On pourrait step-into sur les calls voir la dest
  - Inter-process
  - On tombe sur une fonction de sérialisation
  - Fail
- Il faut donc
  - Trouver l'interface
  - Chopper la def de l'interface
  - Faire la struct de la Vtable
  - Et remplacer les call par la vtable
- C'est chiant et j'ai pas envie de le faire 15 fois.



ExaTrack

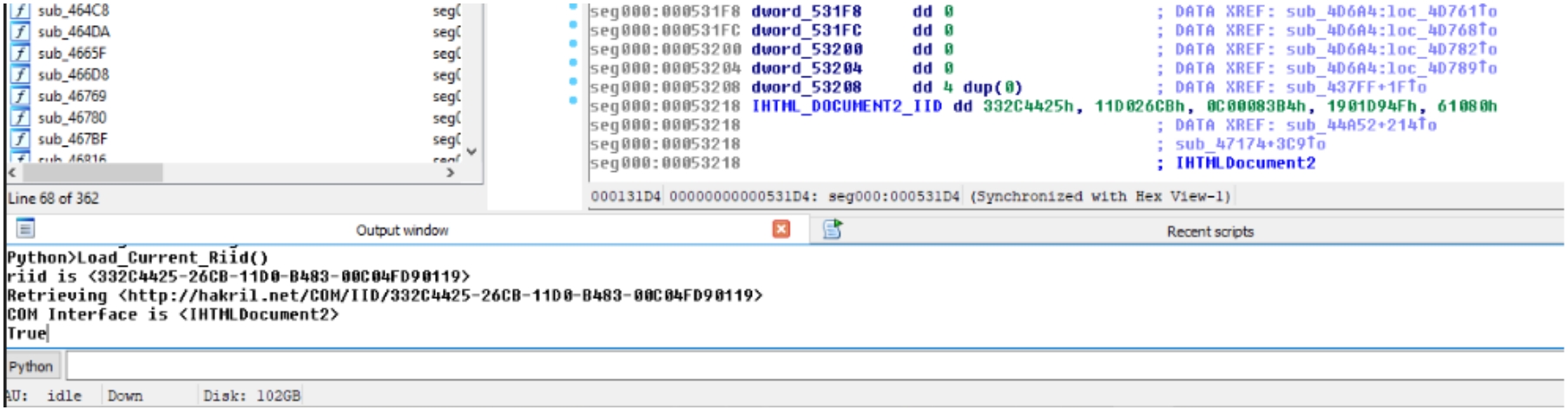
Automatisation

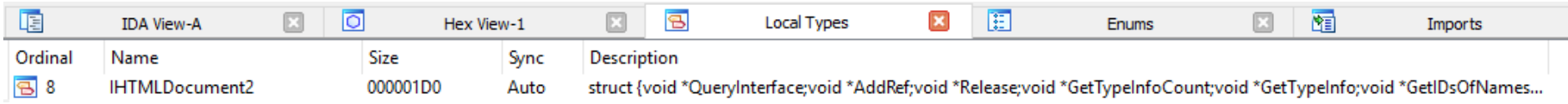
- J’ai pas envie de le faire 15 fois
  - Donc je vais le faire 6000 fois
- « Parser » les headers windows
  - C’est du code généré: regexp
  - Extraire toutes les VTABLE / IID possible
  - Générer des structs C parsables par IDA
  - Rendre ça accessible par nom / IID
    - Sur l’interweb !

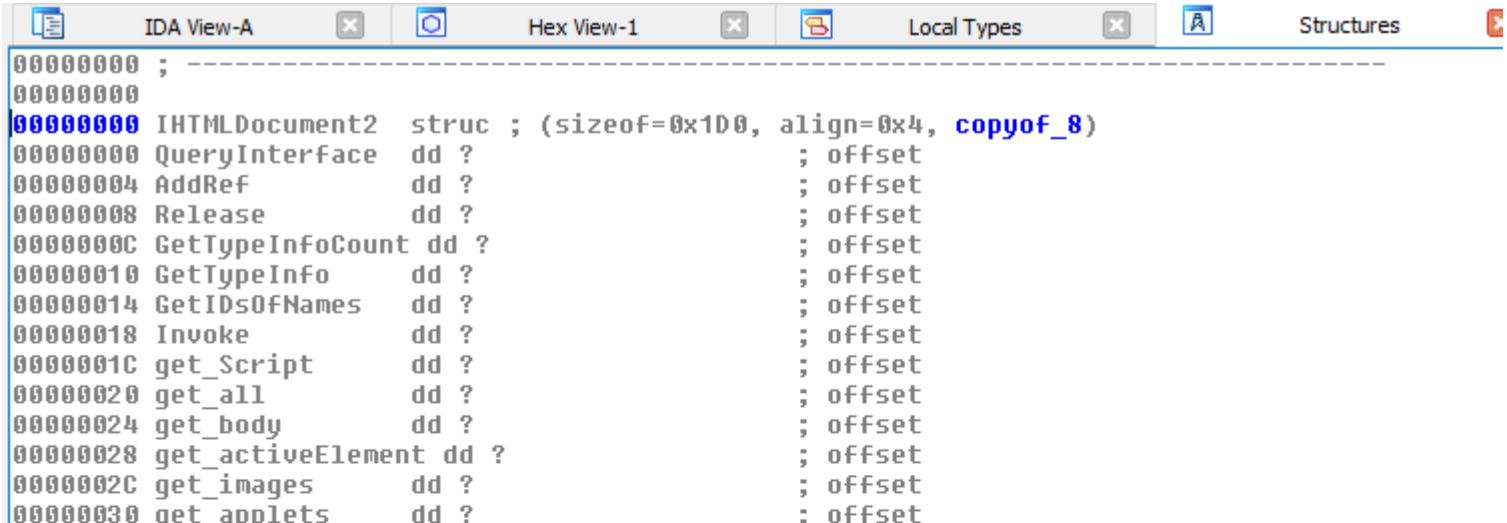
Résultat


```
// Name: IHTMLDocument2
// IID: 332C4425-26CB-11D0-B483-00C04FD90119
// FILE: c:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include\Mshtmlc.h

struct IHTMLDocument2 {
void* QueryInterface; /* 0 */
void* AddRef; /* 1 */
void* Release; /* 2 */
void* GetTypeInfoCount; /* 3 */
void* GetTypeInfo; /* 4 */
void* GetIDsOfNames; /* 5 */
void* Invoke; /* 6 */
void* get_Script; /* 7 */
void* get_all; /* 8 */
void* get_body; /* 9 */
void* get_activeElement; /* 10 */
void* get_images; /* 11 */
void* get_applets; /* 12 */
void* get_links; /* 13 */
void* get_forms; /* 14 */
void* get_anchors; /* 15 */
void* put_title; /* 16 */
void* get_title; /* 17 */
void* get_scripts; /* 18 */
void* put_designMode; /* 19 */
void* get_designMode; /* 20 */
void* get_selection; /* 21 */
void* get_readyState; /* 22 */
void* get_frames; /* 23 */
```

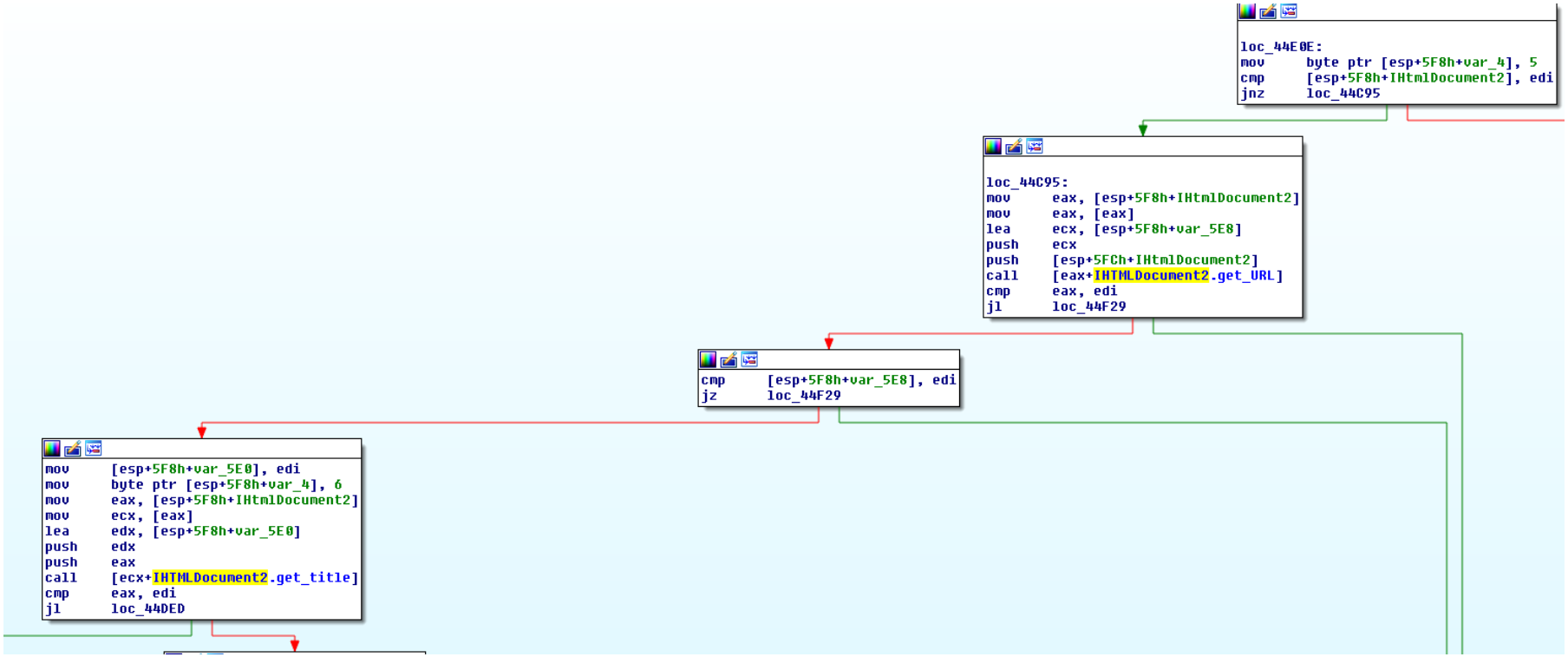
ExaTrack


sub_464C8
sub_464DA
sub_4665F
sub_466D8
sub_46769
sub_46780
sub_467BF
Line 68 of 362
seg000:000531F8 dword_531F8 dd 0 ; DATA XREF: sub_4D6A4:loc_4D761↑o
seg000:000531FC dword_531FC dd 0 ; DATA XREF: sub_4D6A4:loc_4D768↑o
seg000:00053200 dword_53200 dd 0 ; DATA XREF: sub_4D6A4:loc_4D782↑o
seg000:00053204 dword_53204 dd 0 ; DATA XREF: sub_4D6A4:loc_4D789↑o
seg000:00053208 dword_53208 dd 4 dup(0) ; DATA XREF: sub_437FF+1F↑o
seg000:00053218 IHTML_DOCUMENT2_IID dd 332C4425h, 11D026CBh, 0C00083B4h, 1901D94Fh, 61080h
seg000:00053218 ; DATA XREF: sub_44A52+214↑o
seg000:00053218 ; sub_47174+3C9↑o
seg000:00053218 ; IHTMLDocument2
000131D4 00000000000531D4: seg000:000531D4 (Synchronized with Hex View-1)
Output window
Recent scripts
Python>Load_Current_Riid()
riid is <332C4425-26CB-11D0-B483-00C04FD90119>
Retrieving <http://hakril.net/COM/IID/332C4425-26CB-11D0-B483-00C04FD90119>
COM Interface is <IHTMLDocument2>
True
Python
AU: idle Down Disk: 102GB


IDA View-A
Hex View-1
Local Types
Enums
Imports
Ordinal Name Size Sync Description
8 IHTMLDocument2 000001D0 Auto struct {void *QueryInterface;void *AddRef;void *Release;void *GetTypeInfoCount;void *GetTypeInfo;void *GetIDsOfNames...


IDA View-A
Hex View-1
Local Types
Structures
00000000 ; ---
00000000
00000000 IHTMLDocument2 struc ; (sizeof=0x1D0, align=0x4, copyof_8)
00000000 QueryInterface dd ? ; offset
00000004 AddRef dd ? ; offset
00000008 Release dd ? ; offset
0000000C GetTypeInfoCount dd ? ; offset
00000010 GetTypeInfo dd ? ; offset
00000014 GetIDsOfNames dd ? ; offset
00000018 Invoke dd ? ; offset
0000001C get_Script dd ? ; offset
00000020 get_all dd ? ; offset
00000024 get_body dd ? ; offset
00000028 get_activeElement dd ? ; offset
0000002C get_images dd ? ; offset
00000030 get_applets dd ? ; offset

Utilisation des vtable



ExaTrack

Analyse Dynamique

- En gros Aibatook:
  - Récupère un *IHTMLDocument2* sur Internet explorer
    - Ca lui permet d'explorer la page / titre / URL
    - Ca lui permet de check les formulaires
    - Ca lui permet de modifier le HTML
  - Il va attendre que l'utilisateur soit sur la bonne page
    - En checkant URL / Titre
    - Il va injecter son « payload HTML » qui a pour but de récup les credentials du client
- Notre but: trigger l'injection sans internet

CoGetInterceptor

- J'ai découvert ca grâce a **Pavel Yosifovich (article)**
  - Il a fait un article sur le sujet
- Permet de créer un interceptor
  - Object COM qui simule une interface X
  - Offre les mêmes fonctions que X
  - Tous les call sont redirigés vers un Sink
    - Un Object COM de Callback
- Le Sink
  - Implemente ICallFrameEvents:
    - Callback -> « OnCall »
  - On récupère un object « frame »
    - Offre de bonne informations d'introspection
- Prérequis:
  - Avoir les bibliothèques de types de l'interface a intercepter
  - Etre dans le contexte du client COM

CoGetInterceptor - POC

```
import windows # https://github.com/hakril/PythonForWindows/
import windows.generated_def as gdef


# Custom Python ICallFrameEvents implementation
class MySink(windows.com.COMImplementation):
    IMPLEMENT = gdef.ICallFrameEvents

    def OnCall(self, this, frame):
        ifname, methodname = gdef.PWSTR(), gdef.PWSTR()
        frame.GetNames(ifname, methodname)
        print("Hello from <{0}.{1}>".format(ifname.value ,methodname.value))
        frame.SetReturnValue(1234)
        return 0
```

ExaTrack

CoGetInterceptor POC

```
import windows # https://github.com/hakril/PythonForWindows/
import windows.generated_def as gdef
from windows import winproxy

windows.com.init()
# Create an interceptor for the firewall (INetFwPolicy2)
interceptor = gdef.ICallInterceptor()
winproxy.CoGetInterceptor(gdef.INetFwPolicy2.IID, None, interceptor.IID, interceptor)

class MySink(windows.com.COMImplementation):
    # …

# Create and register our ICallFrameEvents sink
xsink = MySink()
interceptor.RegisterSink(xsink)
# Create the INetFwPolicy2 interceptor interface
fakefirewall = gdef.INetFwPolicy2()
interceptor.QueryInterface(fakefirewall.IID, fakefirewall)
# fakefirewall is a INetFwPolicy2 redirecting to the sink
enabled = gdef.VARIANT_BOOL()
res = fakefirewall.get_FirewallEnabled(2, enabled)
print("return value = {0}".format(res))
print("firewall enabled = {0}".format(enabled))
```

```
λ python minpoc.py
Hello from <INetFwPolicy2.FirewallEnabled>
return value = 1234
firewall enabled = VARIANT_BOOL(False)
```

ExaTrack

Application à Aibatook

- Aibatook récupère un *IHTMLDocument2*
  - En utilisant *ObjectFromLresult*
  - Qui retourne l'interface
  - Notre but: changer ce pointer vers un Interceptor
- On a besoin
  - D'injecter du code
    - Pour au moins créer l'interceptor
  - De changer le control flow
    - Sauf que ca va être galère de debug le process
    - Gérer le thread injecté spécialement depuis le debugger c'est chiant

Application à Aibatook

- Solution
  - On injecte du Python
  - On utilise les Vectored Exception Handler (VEH)
    - Permet d'intercepter toutes les exceptions du process
      - Accès au contexte au moment de l'exception
      - On peut faire continuer l'exécution
    - On peut build un debugger de son propre process basé sur les VEH
      - On va donc injecter du Python et un LocalDebugger

Injecter du Python

```
import windows
import windows.generated_def as gdef

target =
r"C:\Users\IEUser\Desktop\c5ffed550addfa27dc1adbc58f3f99fa9a5bc9e8"
aibatook = windows.utils.create_process(target,
dwCreationFlags=gdef.CREATE_SUSPENDED)
# Update sys.path with our script path
aibatook.execute_python(r"import sys;
sys.path.append(r'C:\Users\IEUser\Desktop\FollowAibatook')")
aibatook.execute_python("import windows;
windows.utils.create_console()")
# import our payload script in the context of aibatook
aibatook.execute_python("import payload")
print("Resuming threads")
for t in aibatook.threads: t.resume()
```

- Avec ça on exécute *payload.py* dans le contexte d'aibatook !

ExaTrack

LocalDebugger

```
# payload.py
# Code before setup an IHTMLDocument2 interceptor called 'fakevtable'
REAL_VTB = None

class ReplaceInterfaceBP(windows.debug.FunctionBP):
    TARGET = windows.winproxy.ObjectFromLresult # The function we put the BP on
    def trigger(self, dbg, exc):
        fargs = self.arguments(dbg)
        self.retparam = fargs["ppvObject"].value # PVOID*
        self.break_on_ret(dbg, exc)

    def ret_trigger(self, dbg, exc):
        global REAL_VTB
        print("RET ObjectFromLresult HELLO")
        p = windows.current_process
        REAL_VTB = windows.current_process.read_ptr(self.retparam)
        windows.current_process.write_ptr(self.retparam, fakevtable.value)

d = windows.debug.LocalDebugger()
d.add_bp(ReplaceInterfaceBP())
```

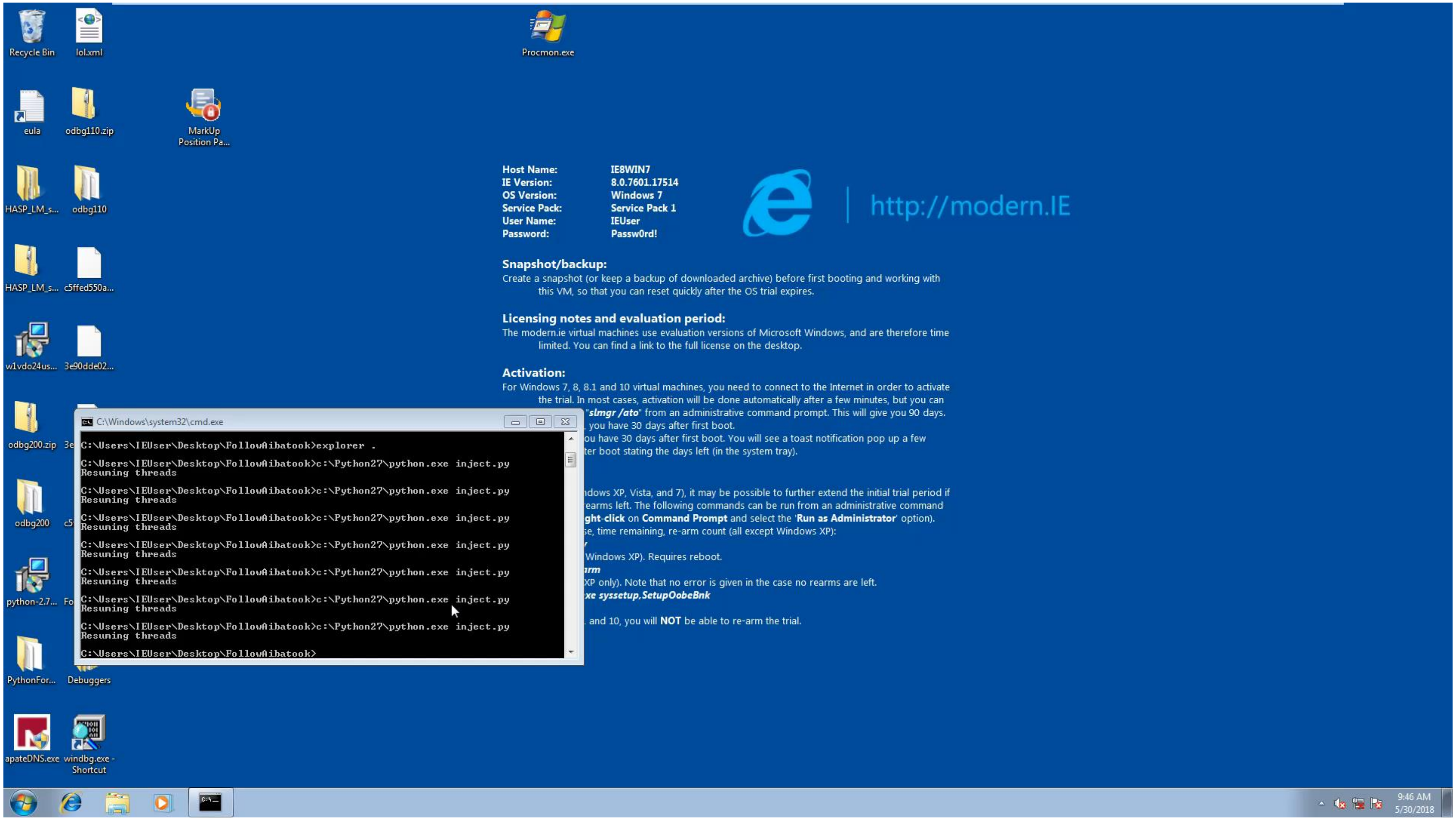
- On utilise VEH + un BP pour changer l'interface retournée par *ObjectFromLresult*
- Aibatook utilise maintenant notre intercepteur

ExaTrack

Le sink

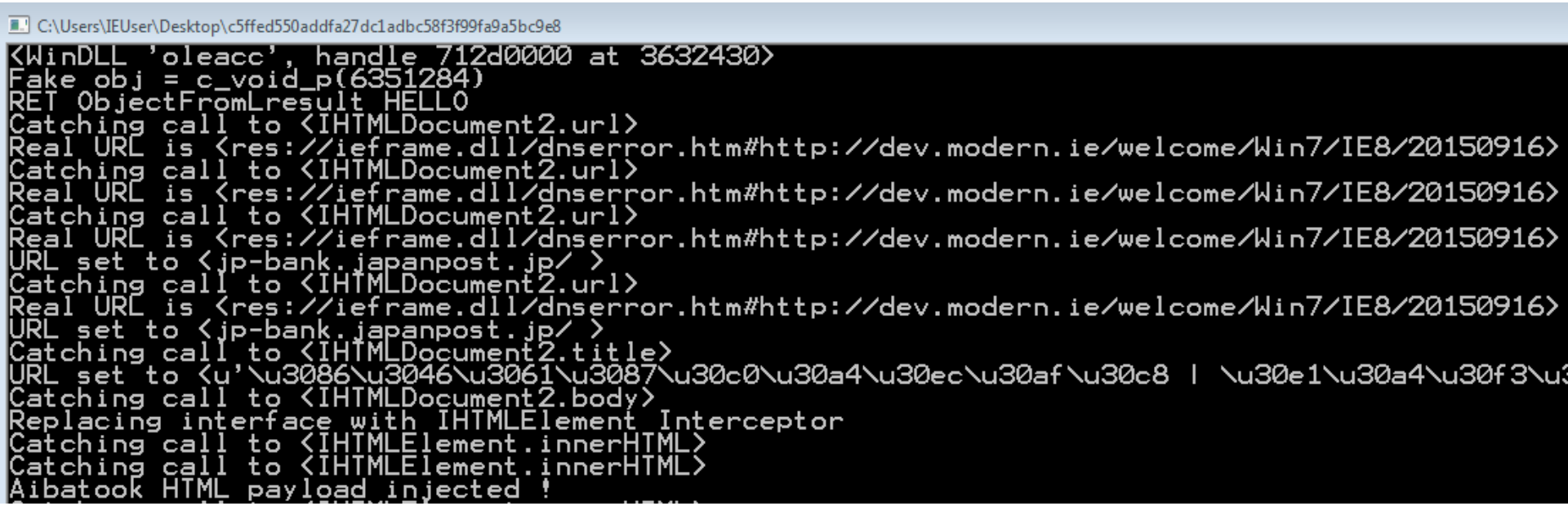
```
class MySink(windows.com.COMImplementation):
    IMPLEMENT = gdef.ICallFrameEvents

    def OnCall(self, this, frame):
        # Dispatch

    def ihtmldocument2_url(self, this, frame):
        frame.Invoke(REAL_VTB)
        param0 = windows.com.ImprovedVariant()
        frame.GetParam(0, param0)
        url = param0._VARIANT_NAME_3.pbstrVal[0]
        print("Real URL is <{0}>".format(url))
        self.count += 1
        if self.count >= 3:
            fakeurl = "jp-bank.japanpost.jp/\x00"
            x = ctypes.c_wchar_p(fakeurl)
            param0._VARIANT_NAME_3.pbstrVal[0] = SysAllocString(x)
            print("URL set to <{0}>".format(fakeurl))
        return
```

Démo



ExaTrack

Démo Explication



ExaTrack

Questions ?

- Clément Rouault (@hakril)
- https://github.com/hakril/pythonforwindows
- https://exatrack.com

Thank you

Please COM again.

ExaTrack