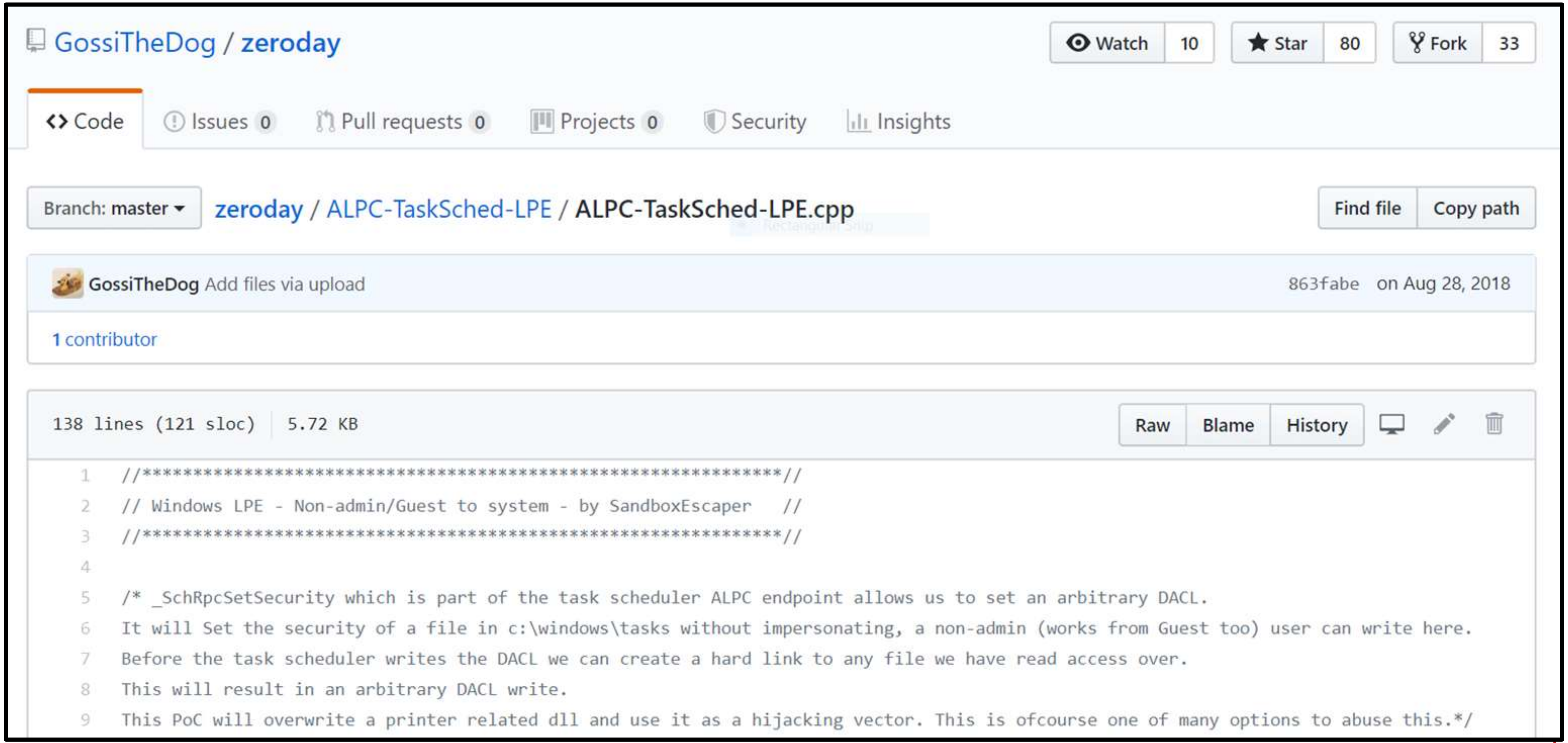
Reimplementing Local RPC in .NET

James Forshaw, Google Project Zero

Table of Contents

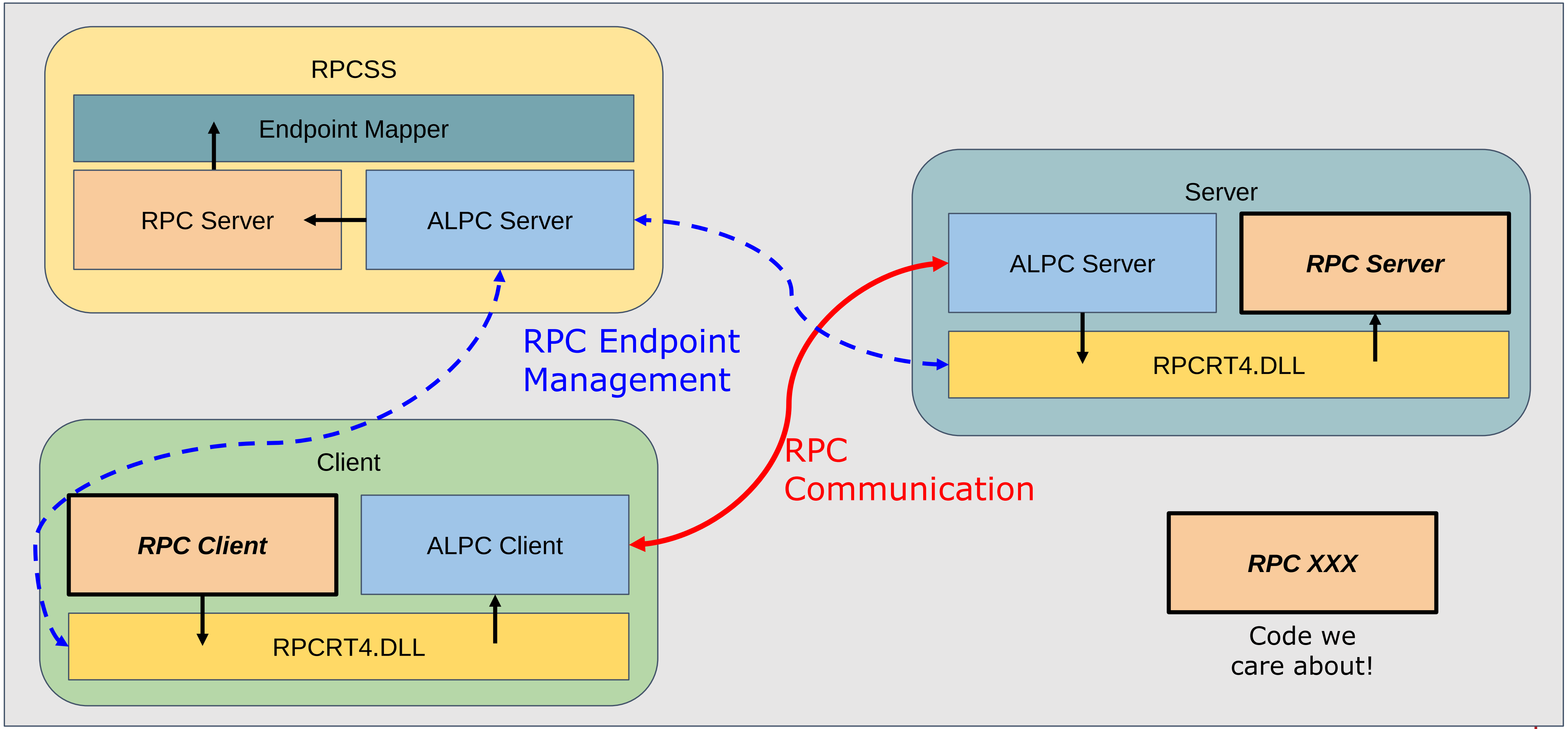
Why?



```
//******************************************************************//
// Windows LPE - Non-admin/Guest to system - by SandboxEscaper   //
//******************************************************************//

/* _SchRpcSetSecurity which is part of the task scheduler ALPC endpoint allows us to set an arbitrary DACL.
It will Set the security of a file in c:\windows\tasks without impersonating, a non-admin (works from Guest too) user can write here.
Before the task scheduler writes the DACL we can create a hard link to any file we have read access over.
This will result in an arbitrary DACL write.
This PoC will overwrite a printer related dll and use it as a hijacking vector. This is ofcourse one of many options to abuse this.*/
```

Windows RPC Programming Model

Architectural Overview



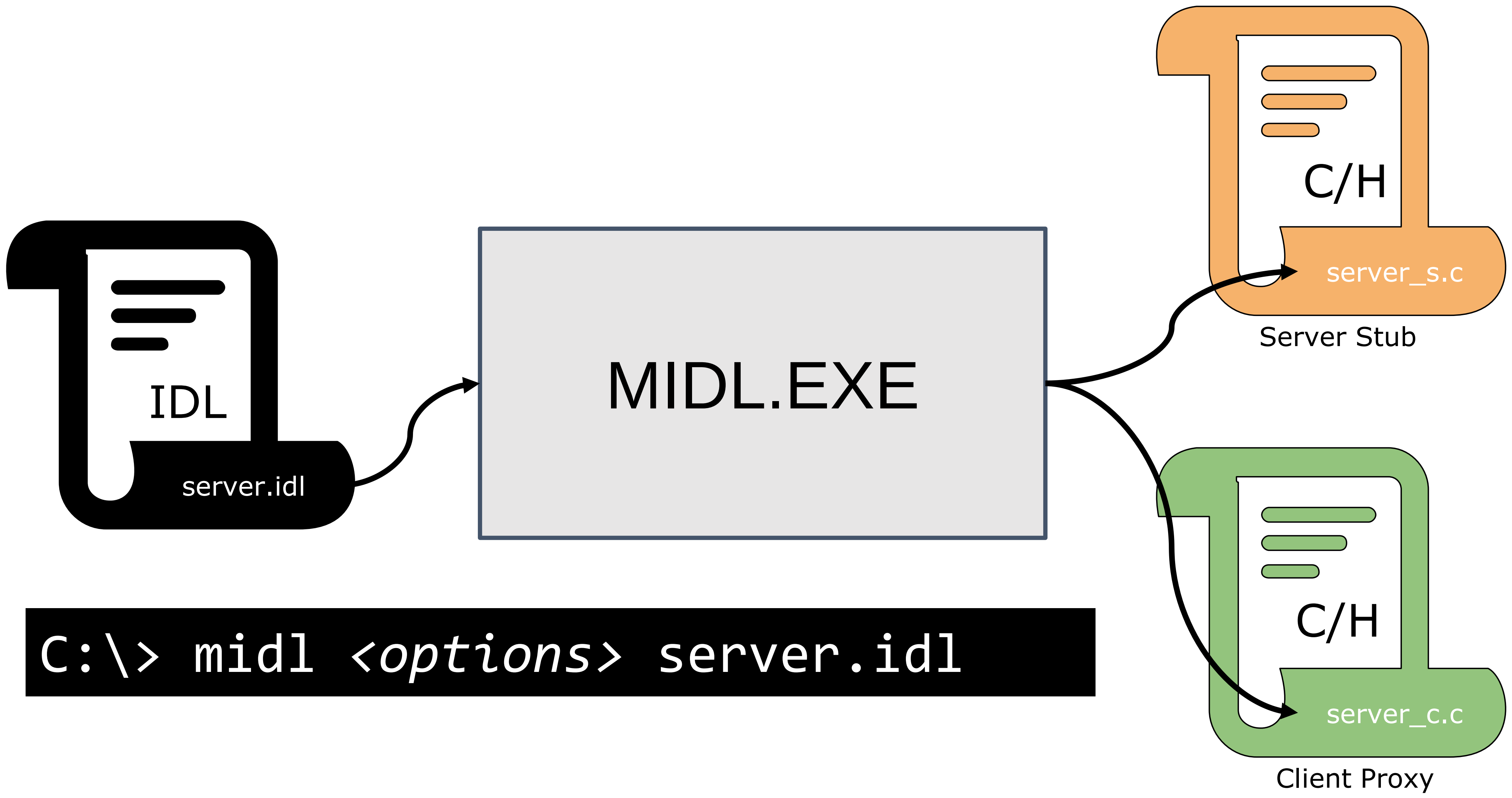
Interface Definition Language (IDL)

```
typedef struct _MYSTRUCT {
    DWORD a;
    LONGLONG b;
} MYSTRUCT;

[
    uuid(4870536E-23FA-4CD5-9637-3F1A1699D3DC),
    version(1.0),
]
interface RpcServer
{
    boolean Func1([in] handle_t hBinding, [string] const wchar_t* name);
    boolean Func2([in] handle_t hBinding, [out] MYSTRUCT* abc);
}
```

RPC Interface
GUID and Version

RPC functions

MIDL Compiler



```
C:\> midl <options> server.idl
```

Auto-generated Server Definition Information

```
struct RPC_SERVER_INTERFACE {
 unsigned int Length;
 RPC_SYNTAX_IDENTIFIER InterfaceId;
 RPC_SYNTAX_IDENTIFIER TransferSyntax;
 // ...
 void const* InterpreterInfo;
}
```

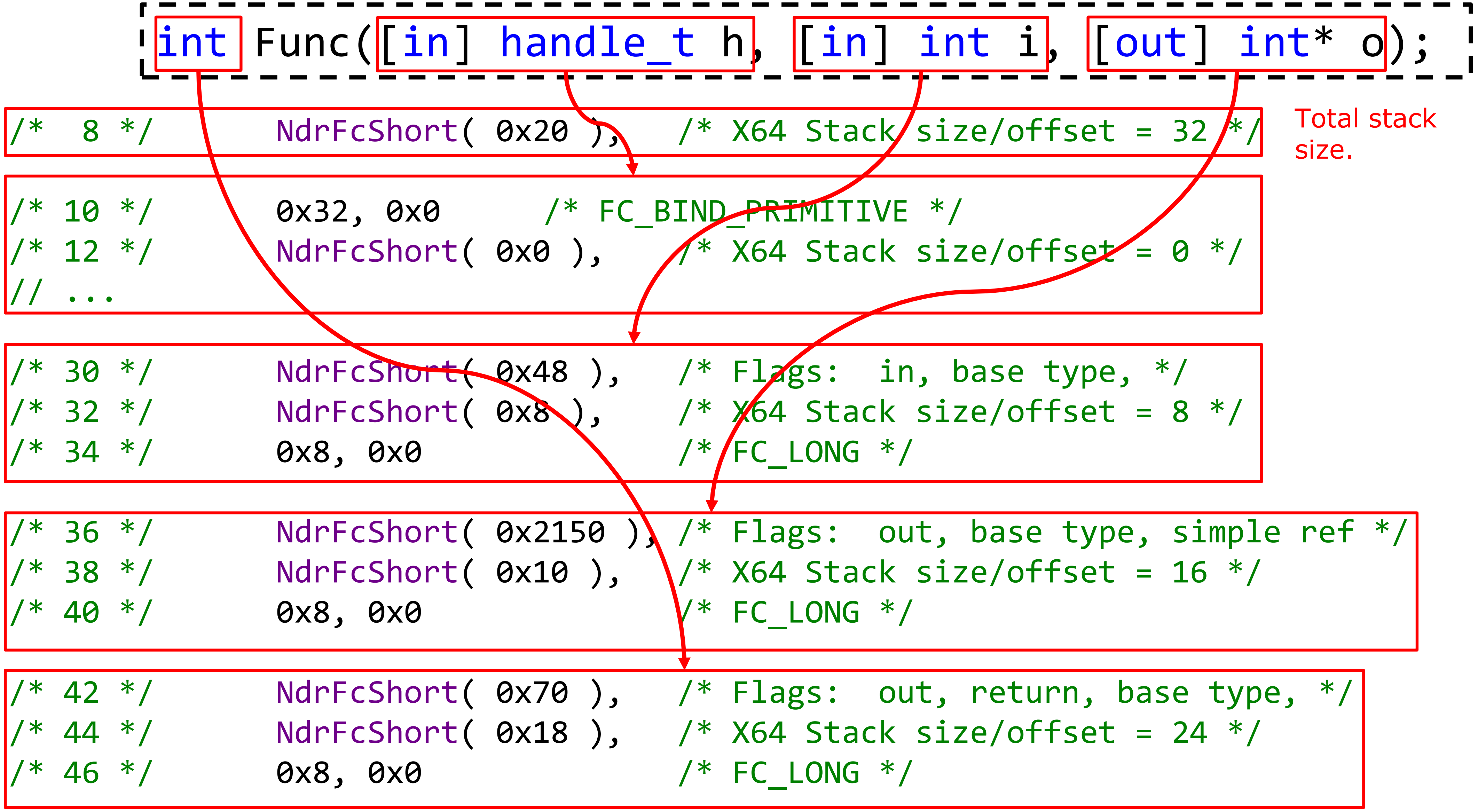
RPC Interface
GUID and Version

Transfer Syntax
DCE: 8A885D04-1CEB-11C9-9FE8-08002B104860
NDR64: 71710533-BEBA-4937-8319-B5DBEF9CCC36

```
struct MIDL_SERVER_INFO {
    // ...
    const SERVER_ROUTINE* DispatchTable;
    PFORMAT_STRING        ProcString;
    const unsigned short* FmtStringOffset;
    // ...
}
```

Server NDR
Format String

Example NDR Format String



```
int Func([in] handle_t h, [in] int i, [out] int* o);
```

```
/*  8 */        NdrFcShort( 0x20 ),    /* X64 Stack size/offset = 32 */
```

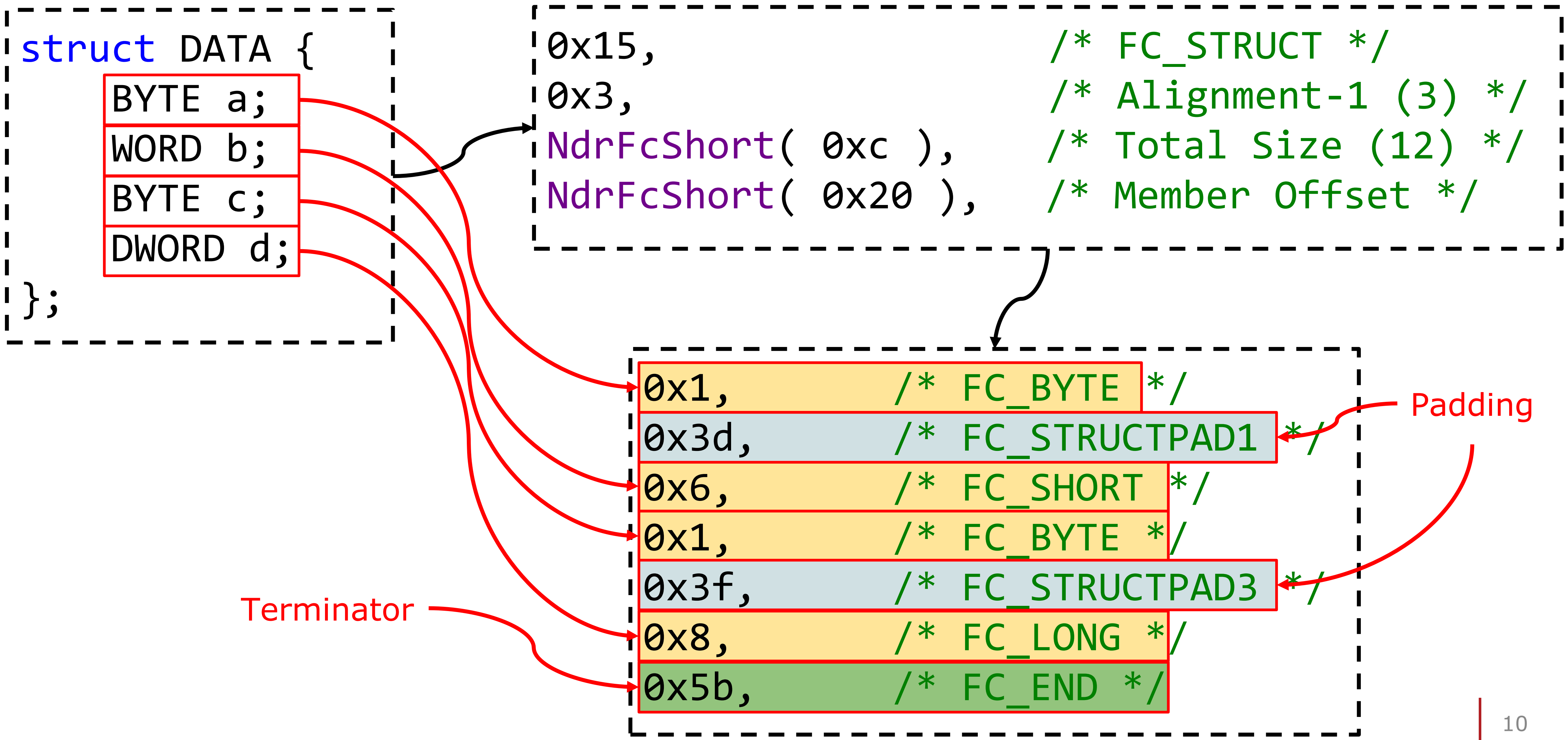
```
/* 10 */        0x32, 0x0       /* FC_BIND_PRIMITIVE */
/* 12 */        NdrFcShort( 0x0 ),     /* X64 Stack size/offset = 0 */
// ...
```

```
/* 30 */        NdrFcShort( 0x48 ),    /* Flags:  in, base type, */
/* 32 */        NdrFcShort( 0x8 ),     /* X64 Stack size/offset = 8 */
/* 34 */        0x8, 0x0               /* FC_LONG */
```

```
/* 36 */        NdrFcShort( 0x2150 ),  /* Flags:  out, base type, simple ref */
/* 38 */        NdrFcShort( 0x10 ),    /* X64 Stack size/offset = 16 */
/* 40 */        0x8, 0x0               /* FC_LONG */
```

```
/* 42 */        NdrFcShort( 0x70 ),    /* Flags:  out, return, base type, */
/* 44 */        NdrFcShort( 0x18 ),    /* X64 Stack size/offset = 24 */
/* 46 */        0x8, 0x0               /* FC_LONG */
```

Structure Marshalling



```
struct DATA {
    BYTE a;
    WORD b;
    BYTE c;
    DWORD d;
};
```

```
0x15,                    /* FC_STRUCT */
0x3,                     /* Alignment-1 (3) */
NdrFcShort( 0xc ),       /* Total Size (12) */
NdrFcShort( 0x20 ),      /* Member Offset */
```

```
0x1,        /* FC_BYTE */
0x3d,       /* FC_STRUCTPAD1 */
0x6,        /* FC_SHORT */
0x1,        /* FC_BYTE */
0x3f,       /* FC_STRUCTPAD3 */
0x8,        /* FC_LONG */
0x5b,       /* FC_END */
```

Client Implementation (32 bit)

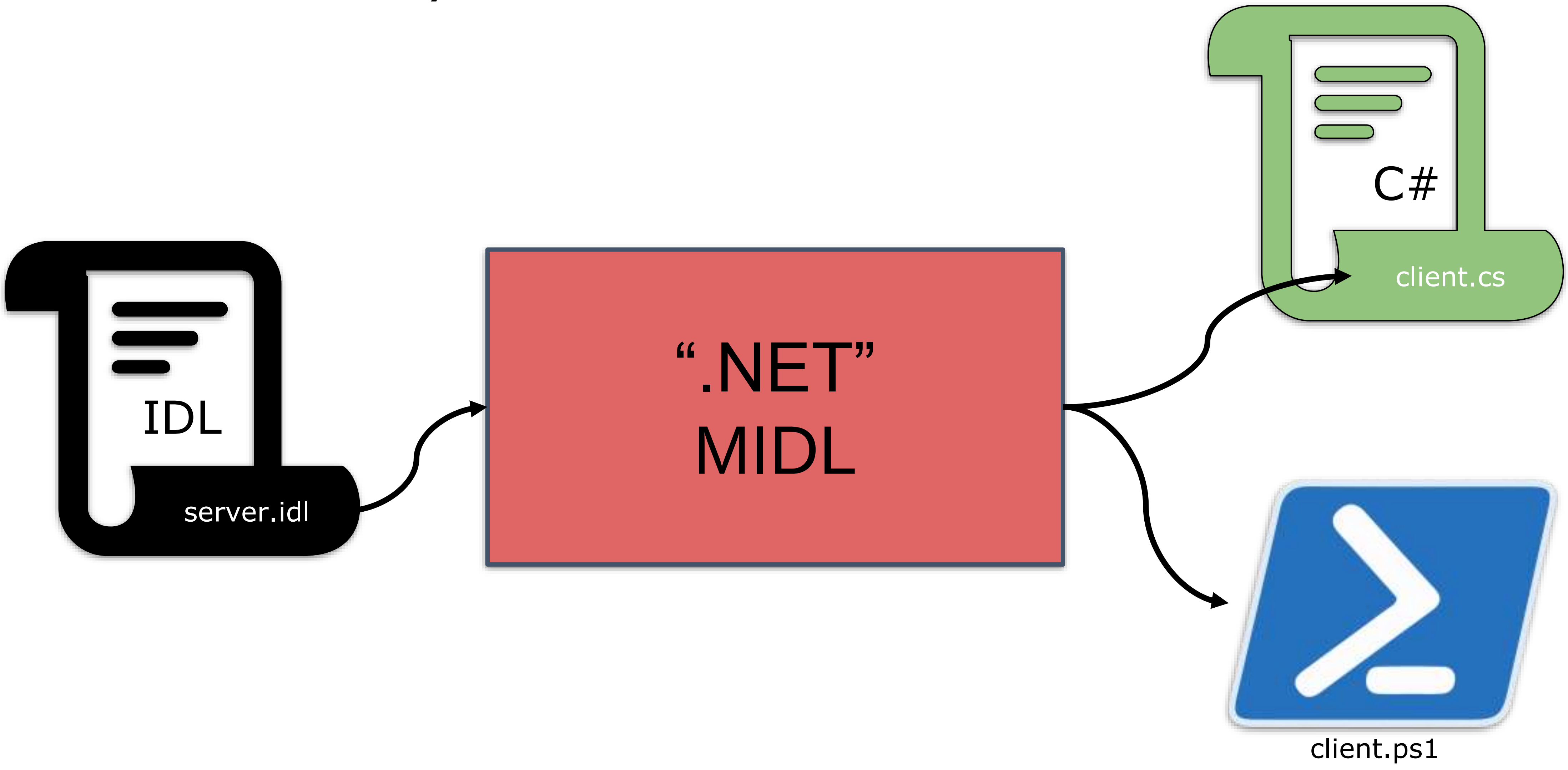
```
int Func(
    /* [in] */ handle_t h,
    /* [in] */ int i,
    /* [out] */ int *o)
{

    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
                  (PMIDL_STUB_DESC)&RpcServer_StubDesc,
                  (PFORMAT_STRING)&ProcFormatString.Format[0],
                  (unsigned char *)&h);
    return (int)_RetVal.Simple;
```

Implemented in RPCRT4.DLL

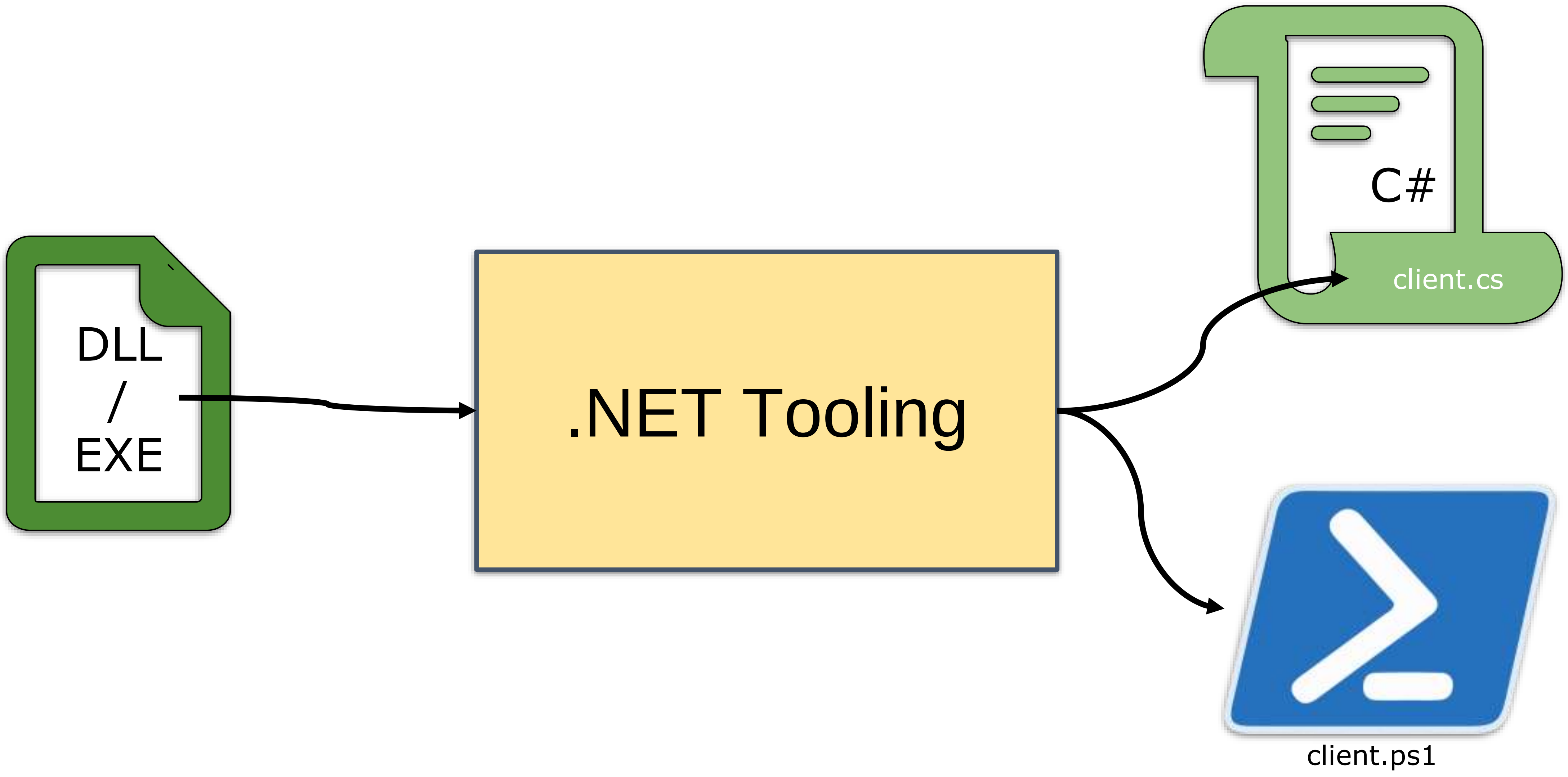
Pass pointer to first parameter on stack.

Implementing in .NET

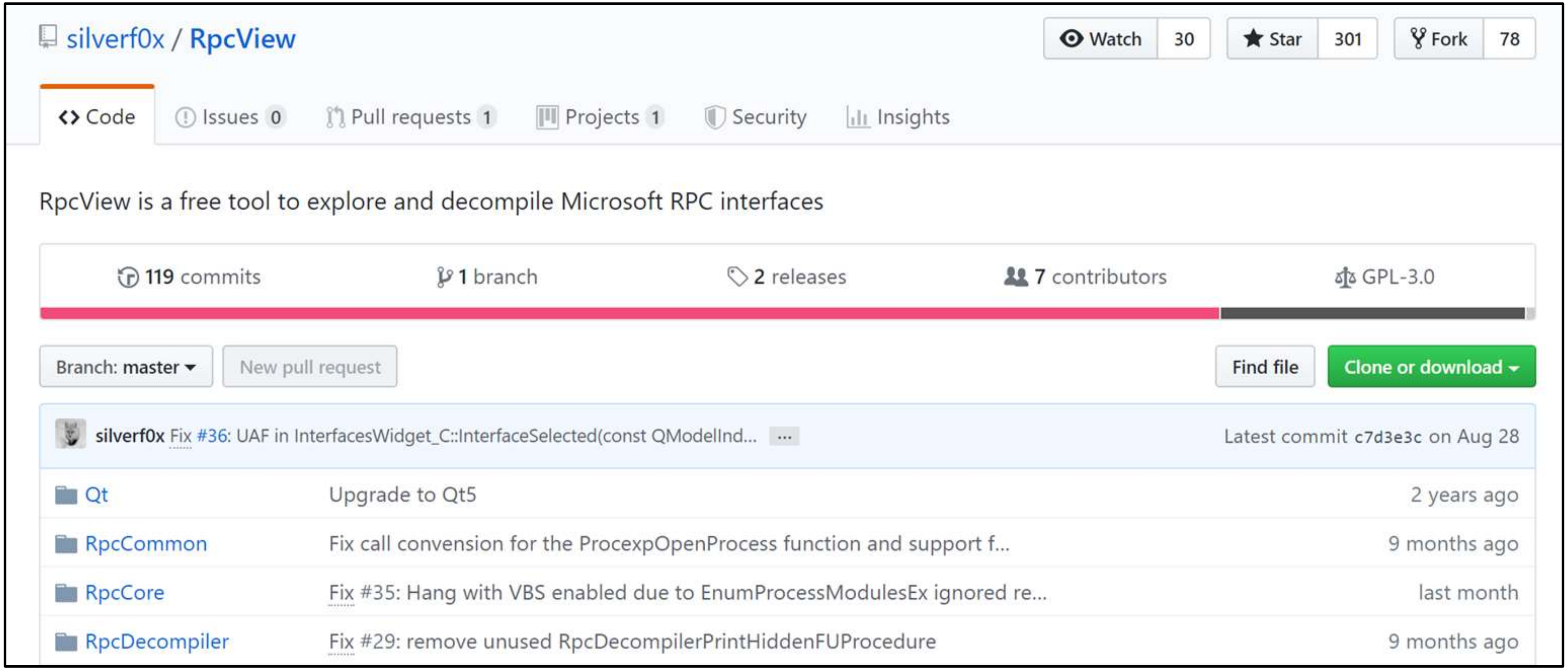
Parse IDL Directly



Final Choice - Parse From DLL/EXE



RpcView


silverf0x / RpcView

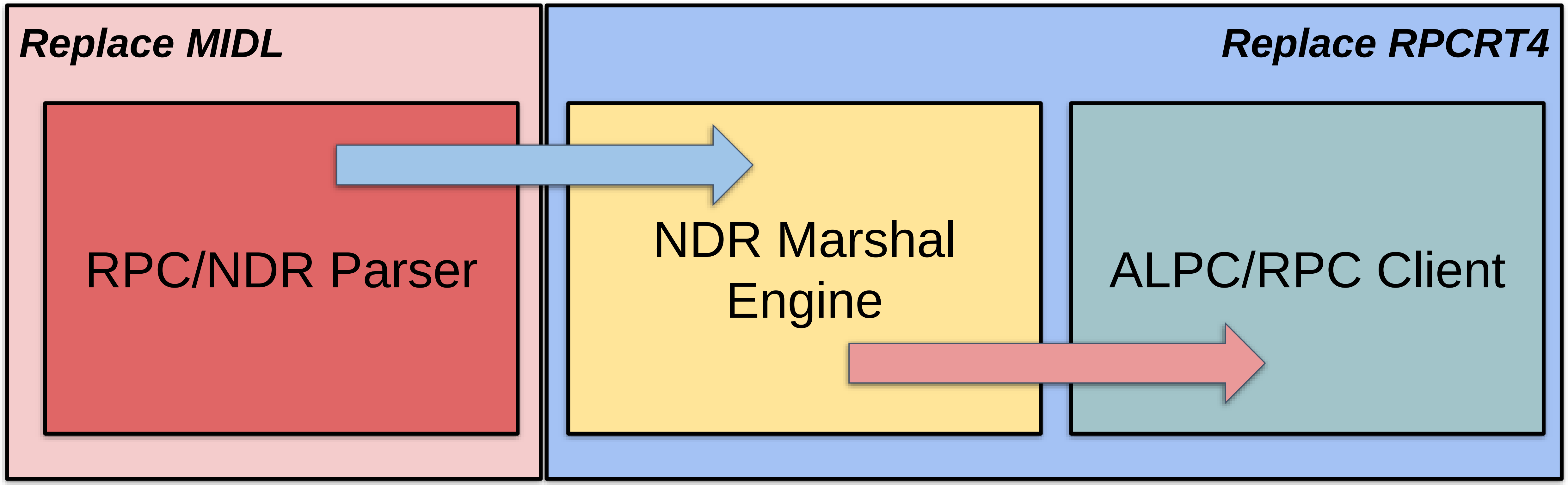
RpcView is a free tool to explore and decompile Microsoft RPC interfaces

silverf0x Fix #36: UAF in InterfacesWidget_C::InterfaceSelected(const QModelInd... … Latest commit c7d3e3c on Aug 28

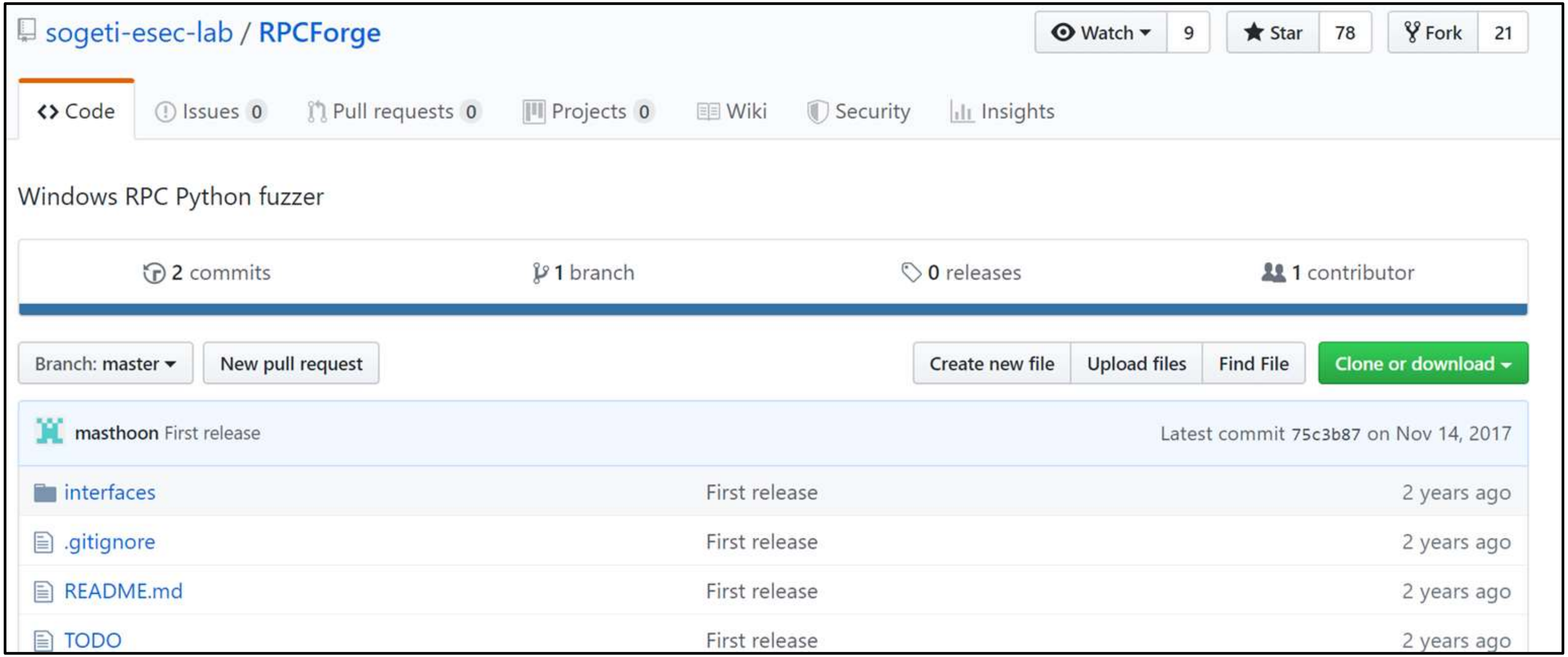
Qt	Upgrade to Qt5	2 years ago
RpcCommon	Fix call convension for the ProcexpOpenProcess function and support f...	9 months ago
RpcCore	Fix #35: Hang with VBS enabled due to EnumProcessModulesEx ignored re...	last month
RpcDecompiler	Fix #29: remove unused RpcDecompilerPrintHiddenFUProcedure	9 months ago

https://github.com/silverf0x/RpcView

Managed Implementation



RPCForge and PythonForWindows


sogeti-esec-lab / RPCForge

Windows RPC Python fuzzer

2 commits · 1 branch · 0 releases · 1 contributor

masthoon First release — Latest commit 75c3b87 on Nov 14, 2017

interfaces	First release	2 years ago
.gitignore	First release	2 years ago
README.md	First release	2 years ago
TODO	First release	2 years ago

https://github.com/sogeti-esec-lab/RPCForge

Implementing RPC/NDR Parser

Finding RPC Server Interfaces

```
struct RPC_SERVER_INTERFACE {
    unsigned int Length;
    RPC_SYNTAX_IDENTIFIER InterfaceId;
    RPC_SYNTAX_IDENTIFIER TransferSyntax;
    PRPC_DISPATCH_TABLE DispatchTable;
    unsigned int RpcProtseqEndpointCount;
    PRPC_PROTSEQ_ENDPOINT RpcProtseqEndpoint;
    void *DefaultManagerEpv;
    void const *InterpreterInfo;
}
```

Inline GUID value

Pointer to MIDL_SERVER_INFO

Brute Force Search.

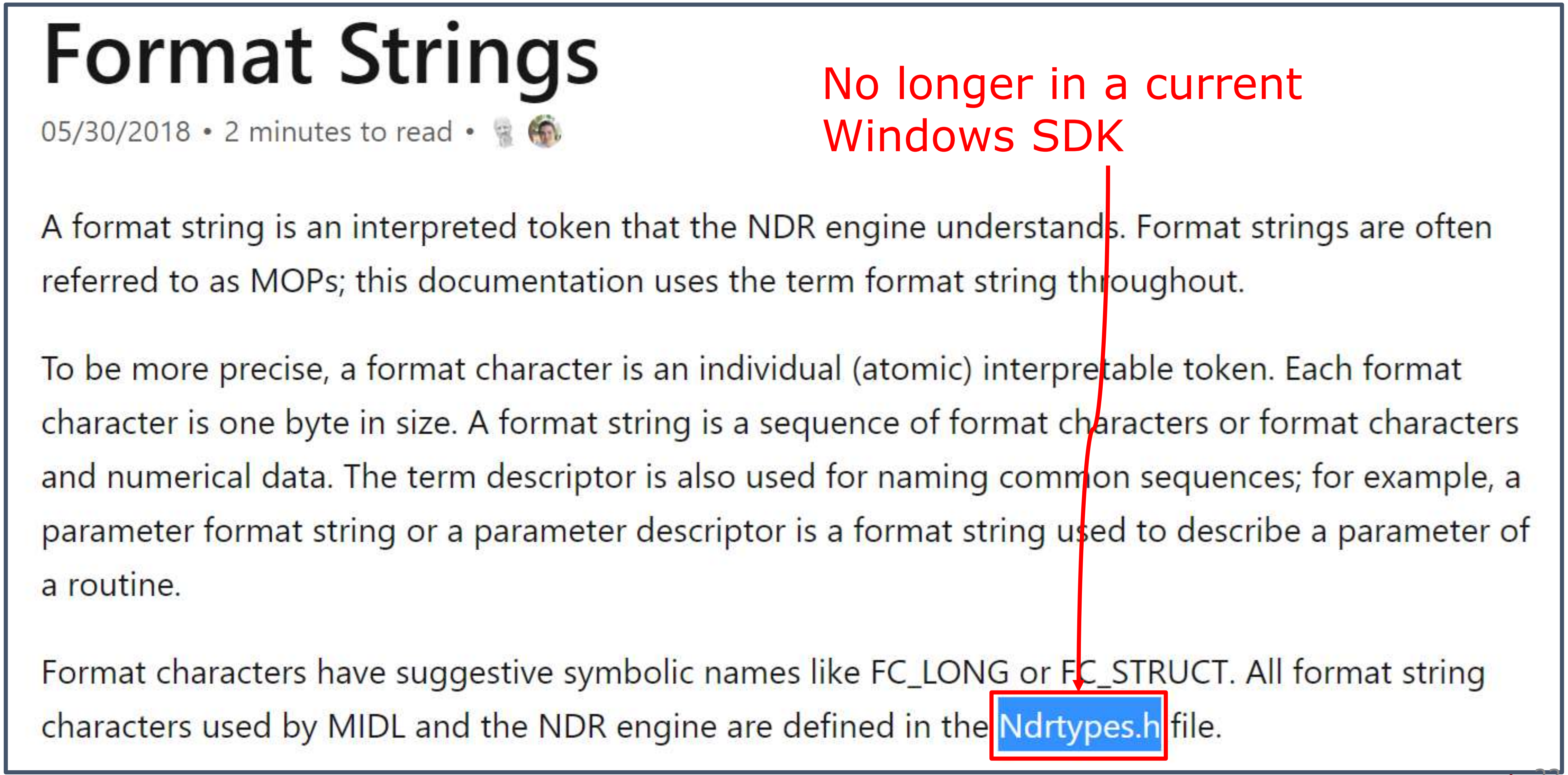
Find inline DCE transfer syntax GUID in image section data.

```
IEnumerable<long> FindRpcServerInterfaces(byte[] rdata) {
    foreach (int ofs in FindBytes(rdata, DCE_TransferSyntax)) {

      int size = Is64BitProcess ? 0x60 : 0x44;
      if (size != BitConverter.ToInt32(rdata, ofs - 24))
        continue;

      if (Is64BitProcess)
        yield return BitConverter.ToInt64(rdata, ofs + 20);
      else
        yield return BitConverter.ToInt32(rdata, ofs + 20);
    }
}
```

Sanity checks.

Read MIDL_SERVER_INFO pointer

Existing Documentation



Format Strings

05/30/2018 • 2 minutes to read •

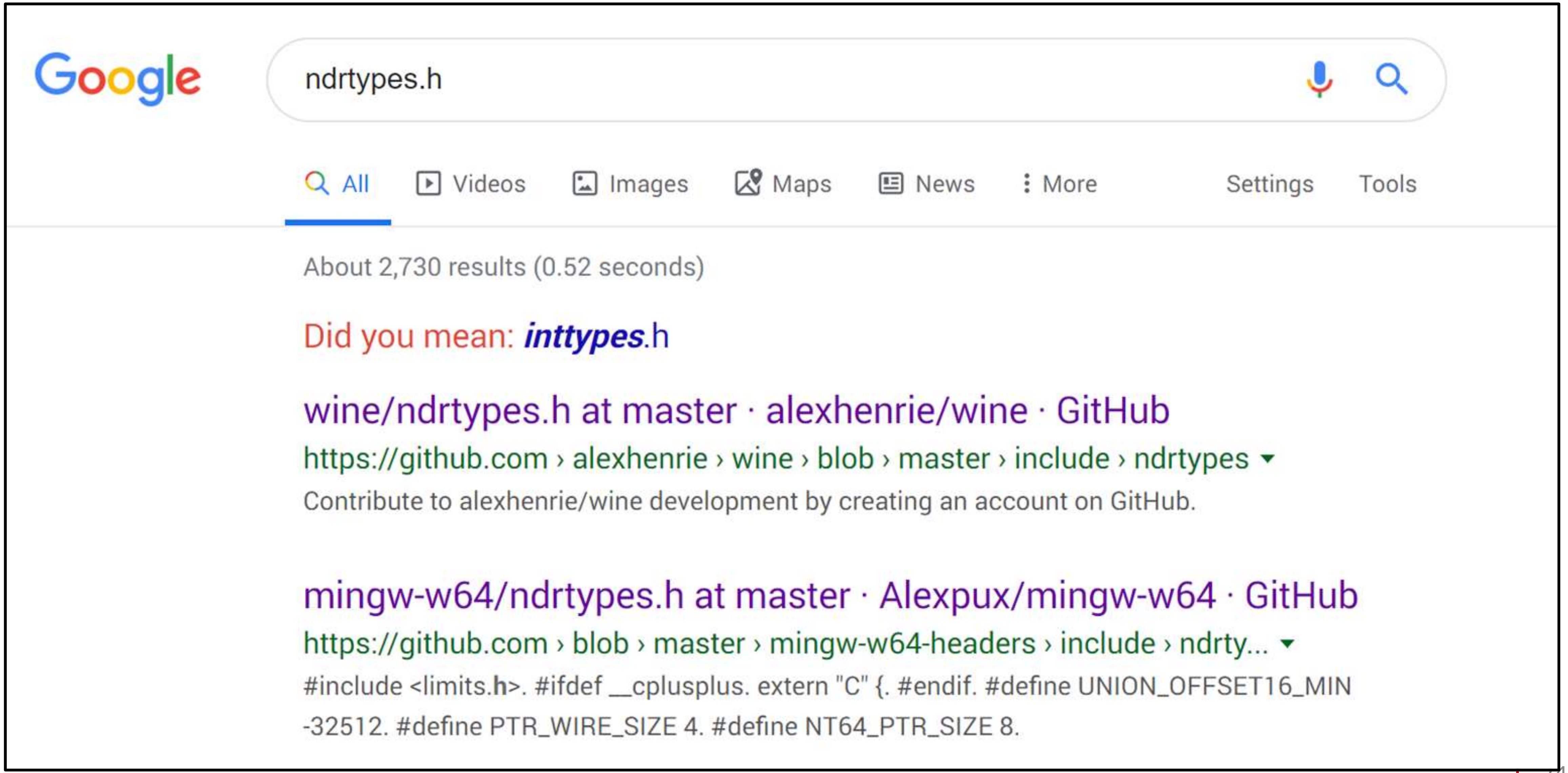
A format string is an interpreted token that the NDR engine understands. Format strings are often referred to as MOPs; this documentation uses the term format string throughout.

To be more precise, a format character is an individual (atomic) interpretable token. Each format character is one byte in size. A format string is a sequence of format characters or format characters and numerical data. The term descriptor is also used for naming common sequences; for example, a parameter format string or a parameter descriptor is a format string used to describe a parameter of a routine.

Format characters have suggestive symbolic names like FC_LONG or FC_STRUCT. All format string characters used by MIDL and the NDR engine are defined in the Ndrtypes.h file.

No longer in a current Windows SDK

Searching for the File


ndrtypes.h

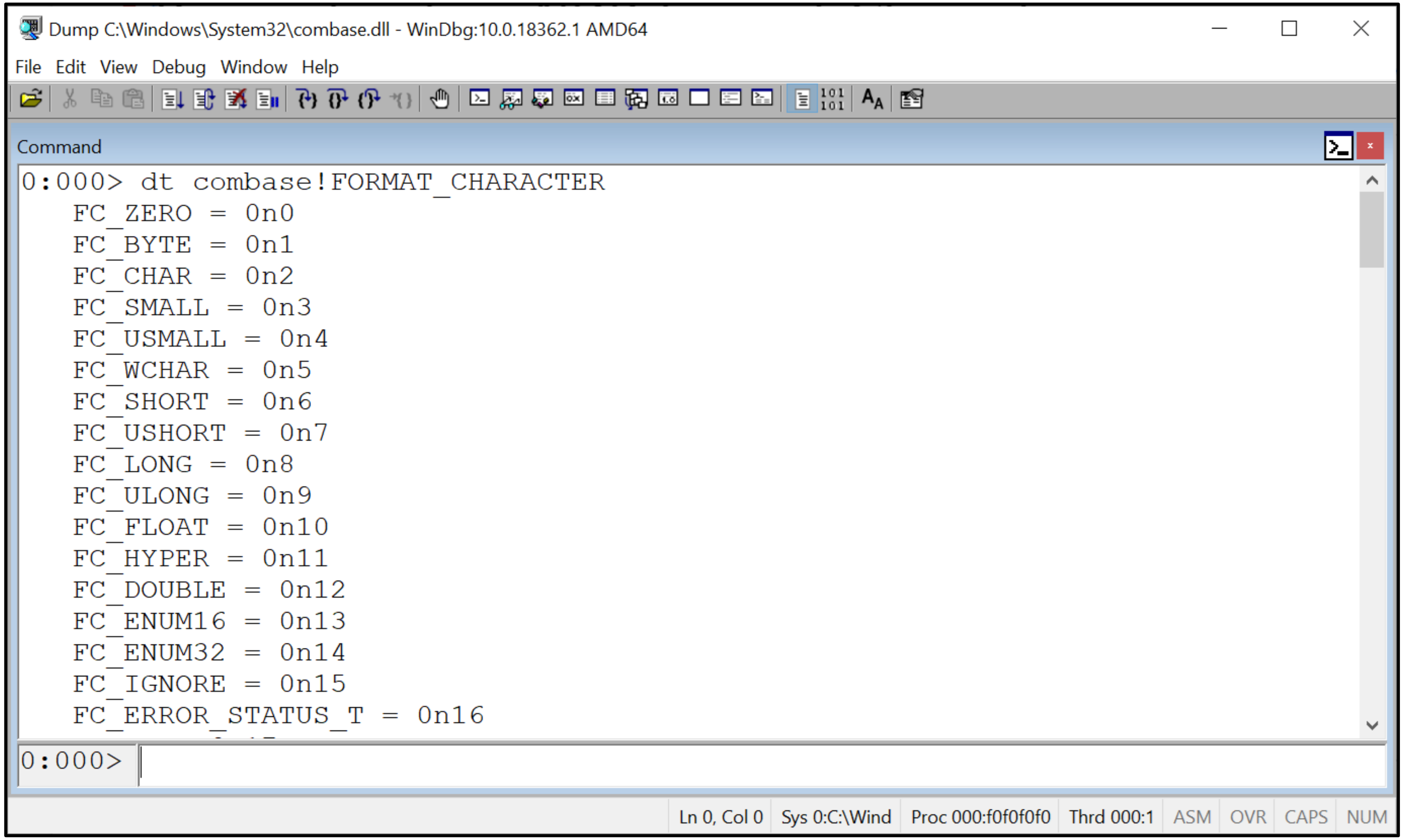
About 2,730 results (0.52 seconds)

Did you mean: ***inttypes***.h

wine/ndrtypes.h at master · alexhenrie/wine · GitHub
https://github.com › alexhenrie › wine › blob › master › include › ndrtypes
Contribute to alexhenrie/wine development by creating an account on GitHub.

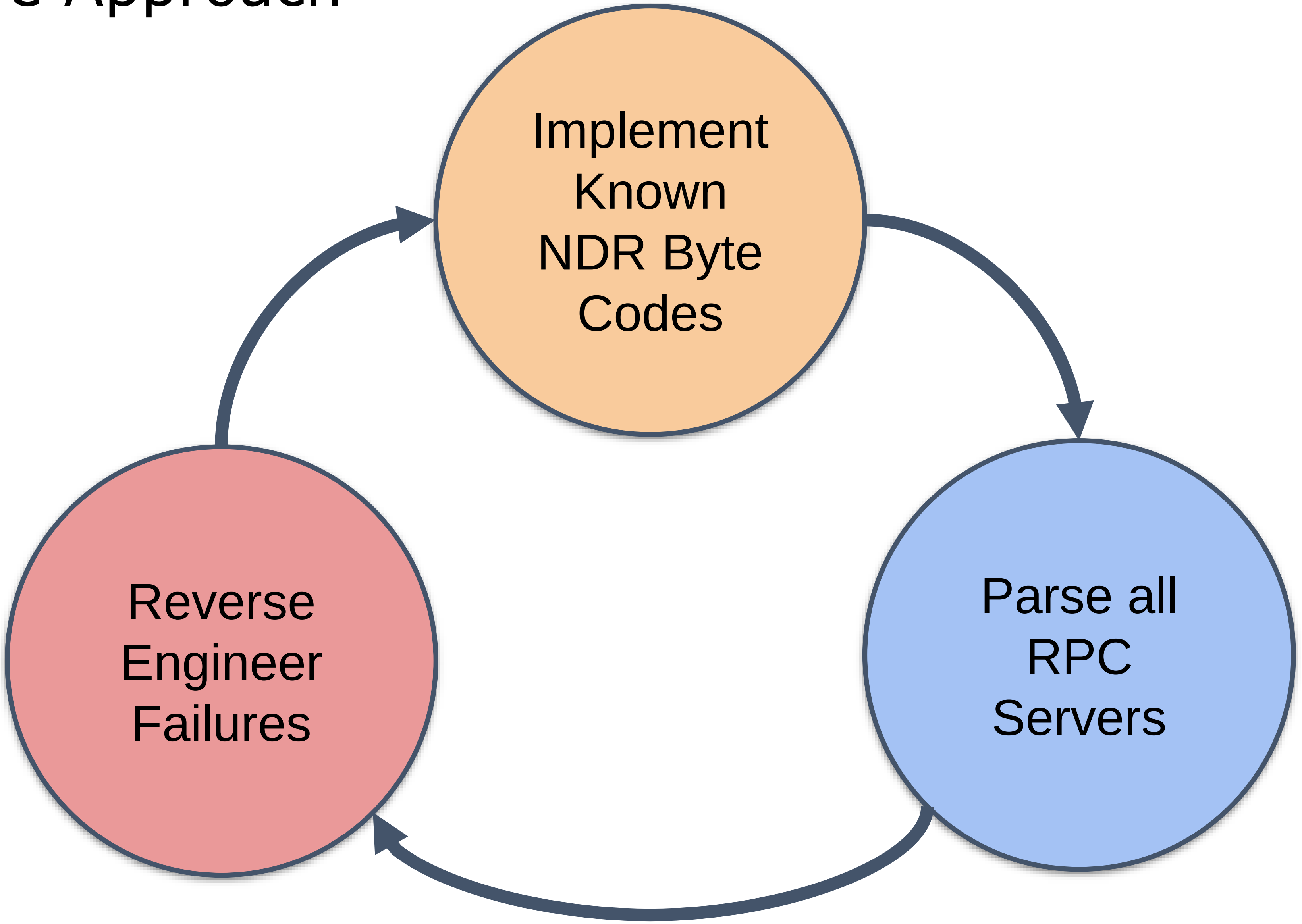
mingw-w64/ndrtypes.h at master · Alexpux/mingw-w64 · GitHub
https://github.com › blob › master › mingw-w64-headers › include › ndrty...
#include <limits.**h**>. #ifdef __cplusplus. extern "C" {. #endif. #define UNION_OFFSET16_MIN -32512. #define PTR_WIRE_SIZE 4. #define NT64_PTR_SIZE 8.

Reconstructing NDR Format Characters

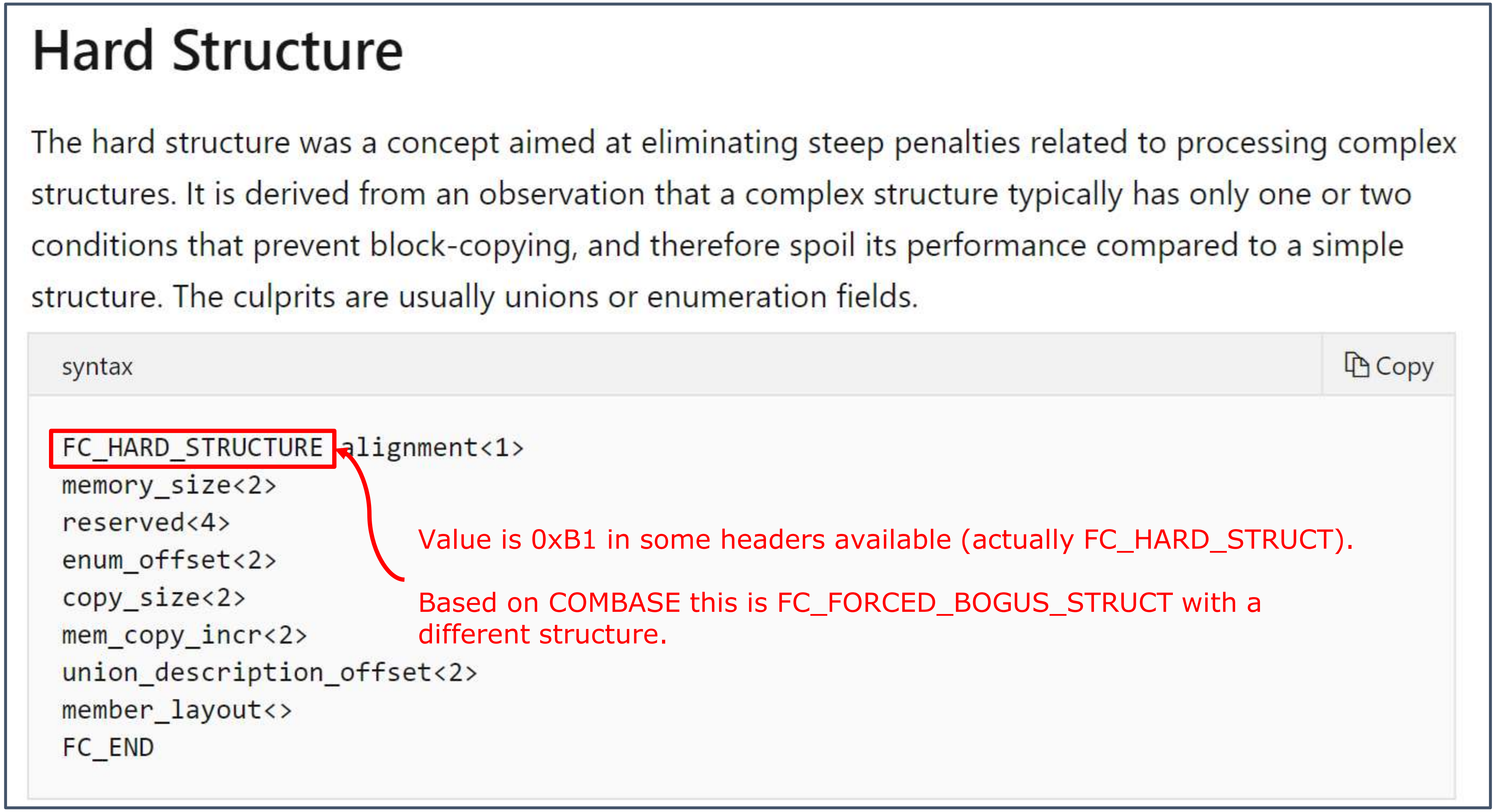

Dump C:\Windows\System32\combase.dll - WinDbg:10.0.18362.1 AMD64
File Edit View Debug Window Help
Command
Ln 0, Col 0
Sys 0:C:\Wind
Proc 000:f0f0f0f0
Thrd 000:1
ASM
OVR
CAPS
NUM

```
0:000> dt combase!FORMAT_CHARACTER
   FC_ZERO = 0n0
   FC_BYTE = 0n1
   FC_CHAR = 0n2
   FC_SMALL = 0n3
   FC_USMALL = 0n4
   FC_WCHAR = 0n5
   FC_SHORT = 0n6
   FC_USHORT = 0n7
   FC_LONG = 0n8
   FC_ULONG = 0n9
   FC_FLOAT = 0n10
   FC_HYPER = 0n11
   FC_DOUBLE = 0n12
   FC_ENUM16 = 0n13
   FC_ENUM32 = 0n14
   FC_IGNORE = 0n15
   FC_ERROR_STATUS_T = 0n16
0:000>
```

Iterative Approach



Incorrectly Documented Byte Codes



Hard Structure

The hard structure was a concept aimed at eliminating steep penalties related to processing complex structures. It is derived from an observation that a complex structure typically has only one or two conditions that prevent block-copying, and therefore spoil its performance compared to a simple structure. The culprits are usually unions or enumeration fields.

syntax

Copy

```
FC_HARD_STRUCTURE alignment<1>
memory_size<2>
reserved<4>
enum_offset<2>
copy_size<2>
mem_copy_incr<2>
union_description_offset<2>
member_layout<>
FC_END
```

Value is 0xB1 in some headers available (actually FC_HARD_STRUCT).

Based on COMBASE this is FC_FORCED_BOGUS_STRUCT with a different structure.

Undocumented Byte Codes

FC_UNUSED4 (0x3C) became
FC_SYSTEM_HANDLE in Windows 8.

```
interface IChakraJIT {
    HRESULT ConnectProcessWithProcessHandle(
        [in] handle_t binding,
        [in, system_handle(sh_process, 1)] HANDLE processHandle
    );
}
```

Type of handle

Optional Desired Access

```
0x3c,                   /* FC_SYSTEM_HANDLE */
0x4,                    /* 4 (Type) */
NdrFcLong( 0x1 ),      /* 1 (Access Mask) */
```

Sets the global symbol resolver to use WinDBG's DBGHELP

```
PS> Set-GlobalSymbolResolver "c:\dbg\dbghelp.dll"
```

Carve out the RPC services in RPCSS.DLL

```
PS> Get-RpcServer "c:\windows\system32\rpcss.dll"
```

Enumerate all DLLs in system32 and extract RPC servers.

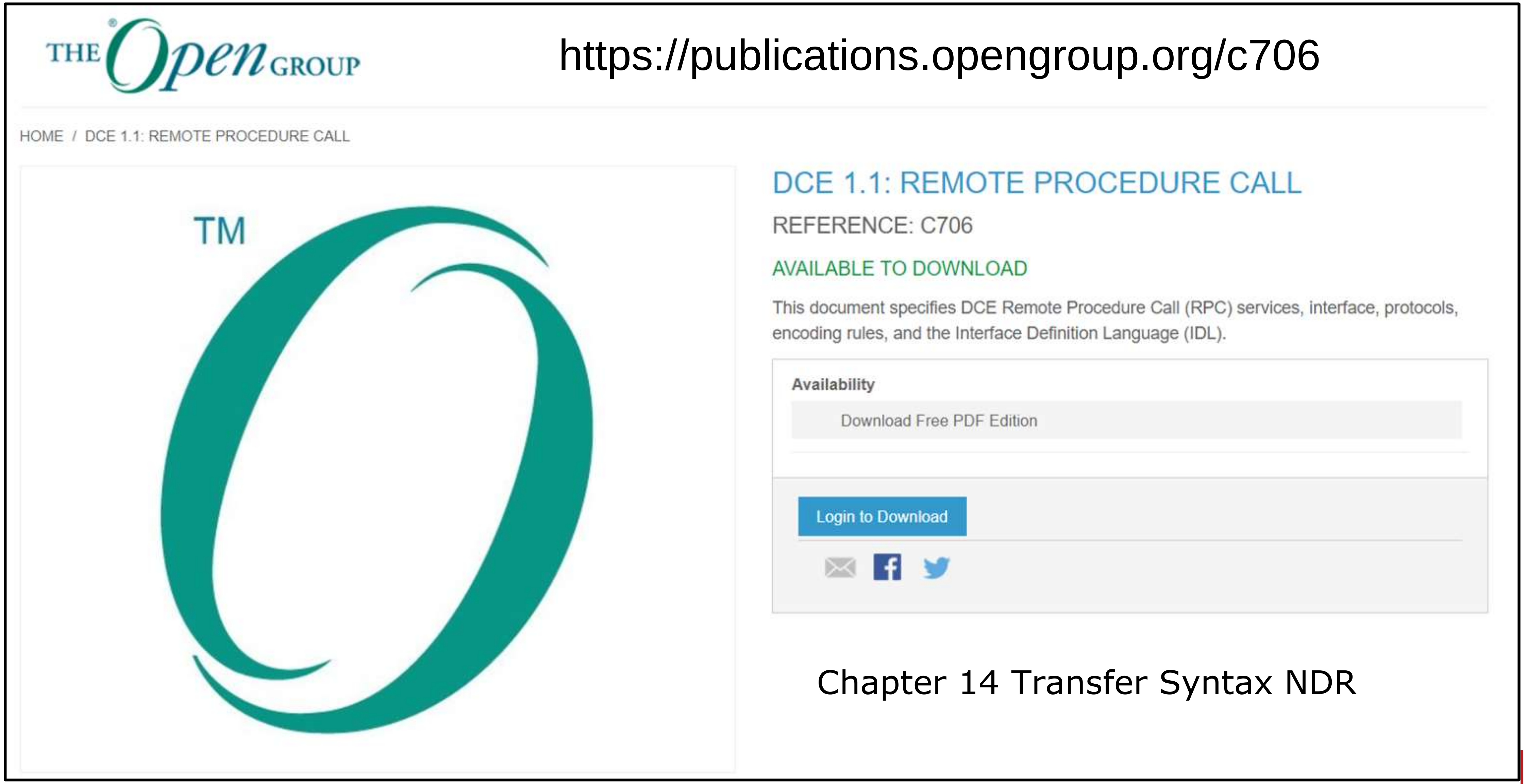
```
PS> ls "c:\windows\system32\*.dll" | Get-RpcServer
```

Format an RPC endpoint as text.

```
PS> $rpc_server | Format-RpcServer
```

Implementing NDR Marshaller

Going to the Standards



https://publications.opengroup.org/c706

HOME / DCE 1.1: REMOTE PROCEDURE CALL

DCE 1.1: REMOTE PROCEDURE CALL

REFERENCE: C706

AVAILABLE TO DOWNLOAD

This document specifies DCE Remote Procedure Call (RPC) services, interface, protocols, encoding rules, and the Interface Definition Language (IDL).

Availability

Download Free PDF Edition

Chapter 14 Transfer Syntax NDR

Lots of Types to Implement

Primitive Types
Signed Integers
Unsigned Integers
Booleans
ENUM16 + ENUM32

Strings
Fixed ANSI
Fixed Unicode
Varying
Conformant

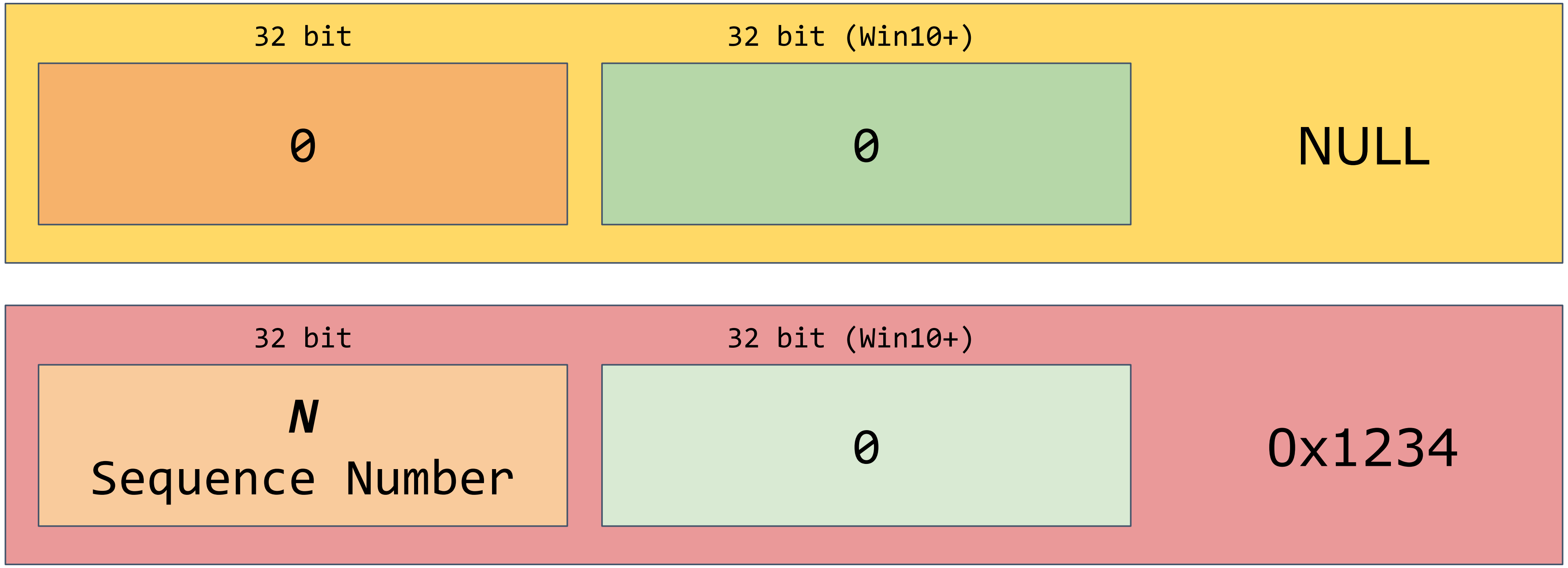
Pointers
Reference
Unique
Full
Embedded

Arrays
Fixed
Varying
Conformant
Conformant Varying

Miscellaneous
Context Handles
System Handles
Interface Pointers
Pipes

Undocumented NDR Types - System Handle

```
0x3c,                  /* FC_SYSTEM_HANDLE */
0x4,                   /* 4 (Type) */
NdrFcLong( 0x1 ),     /* 1 (Access Mask) */
```

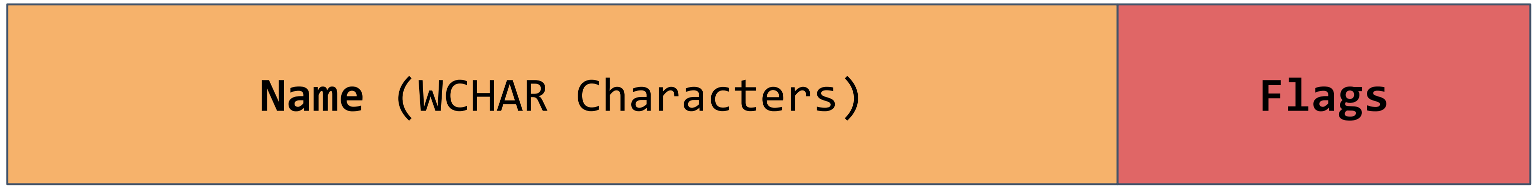


So Many Structures

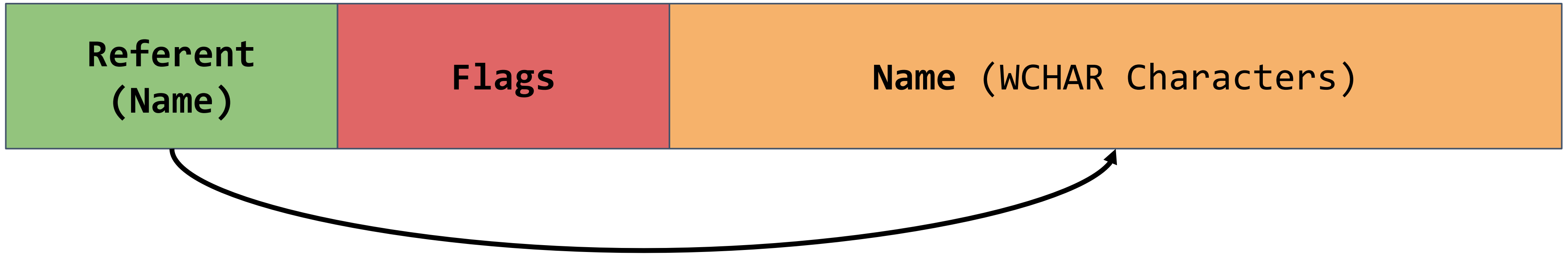
Simple Structure	Simple	Simple with Pointers	
Conformant Structure	Conformant	Conformant with Pointers	Conformant Varying
Complex Structure	“Bogus”	“Forced Bogus”	
Unions	Non-Encapsulated	Non-Encapsulated	

Structures with Pointers

```
struct NAME_ENTRY {
    [string] LPWSTR Name;
    DWORD Flags;
};
```



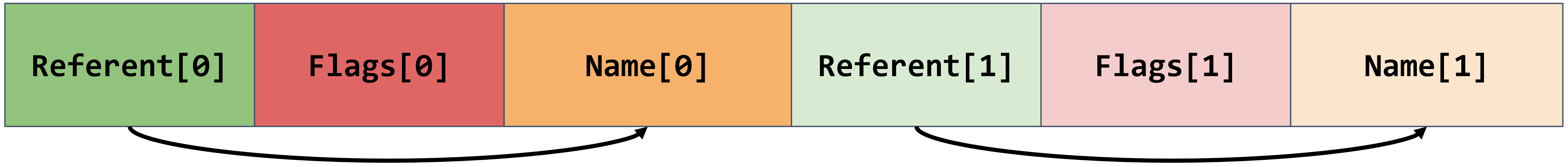
Marshalled like this?

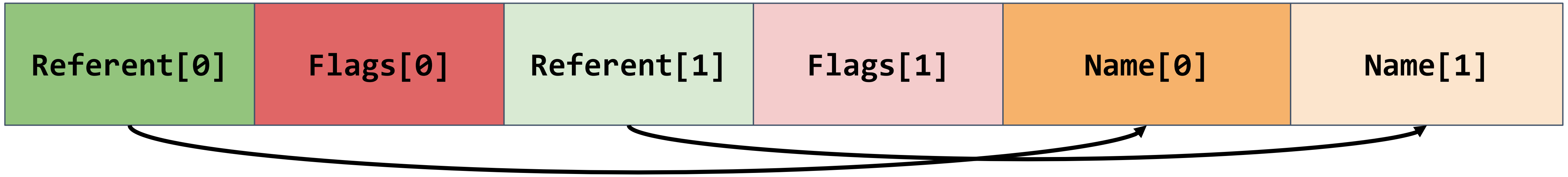


Arrays of Structures with Pointers

```
struct NAME_ENTRY {
    [string] LPWSTR Name;
    DWORD Flags;
} ARR[2];
```

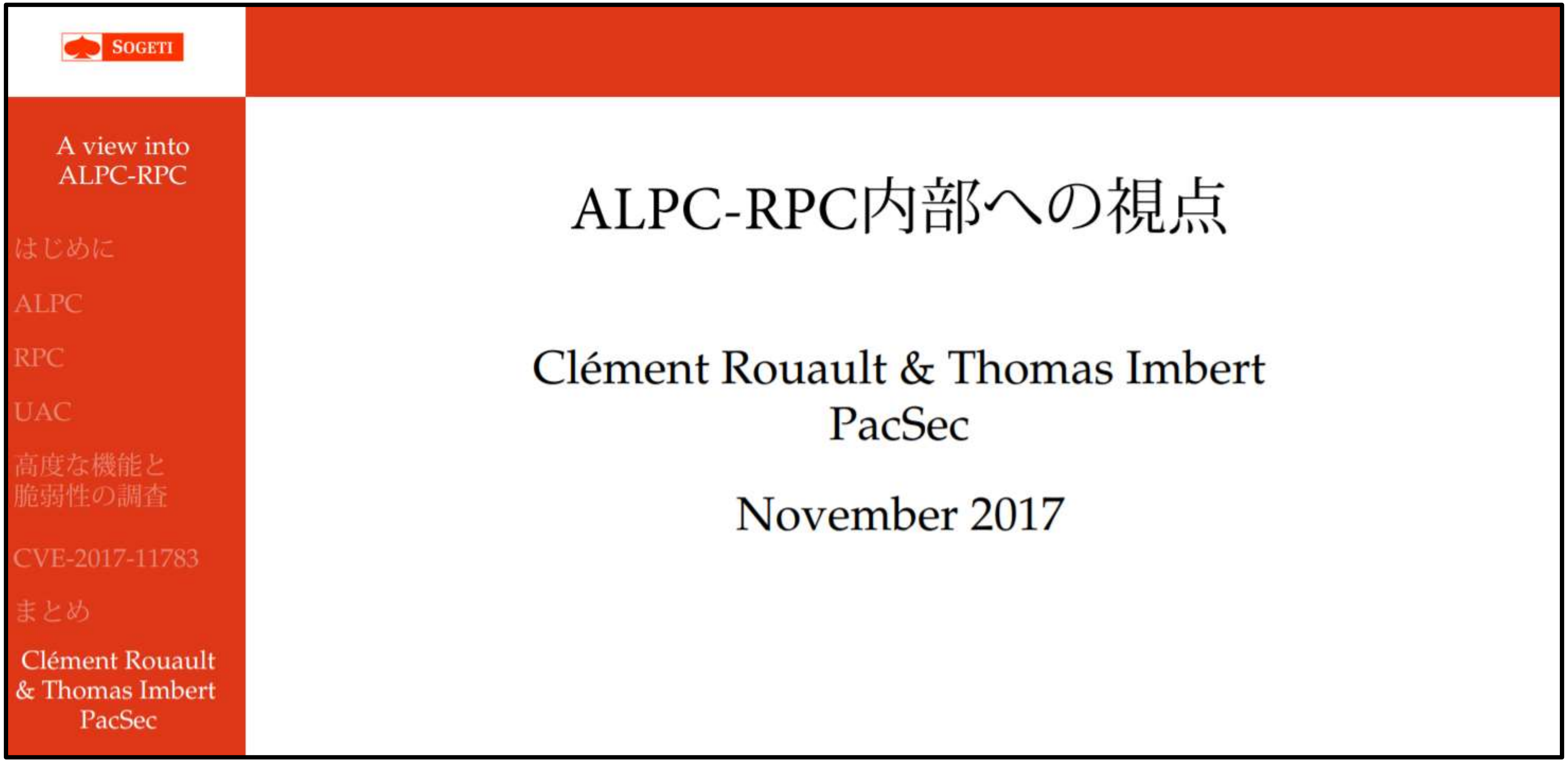
Marshalled like this?





Implementing the ALPC Client

Implementing ALPC/RPC Transport



SOGETI

A view into ALPC-RPC

はじめに

ALPC

RPC

UAC

高度な機能と脆弱性の調査

CVE-2017-11783

まとめ

Clément Rouault & Thomas Imbert PacSec

ALPC-RPC内部への視点

Clément Rouault & Thomas Imbert
PacSec

November 2017

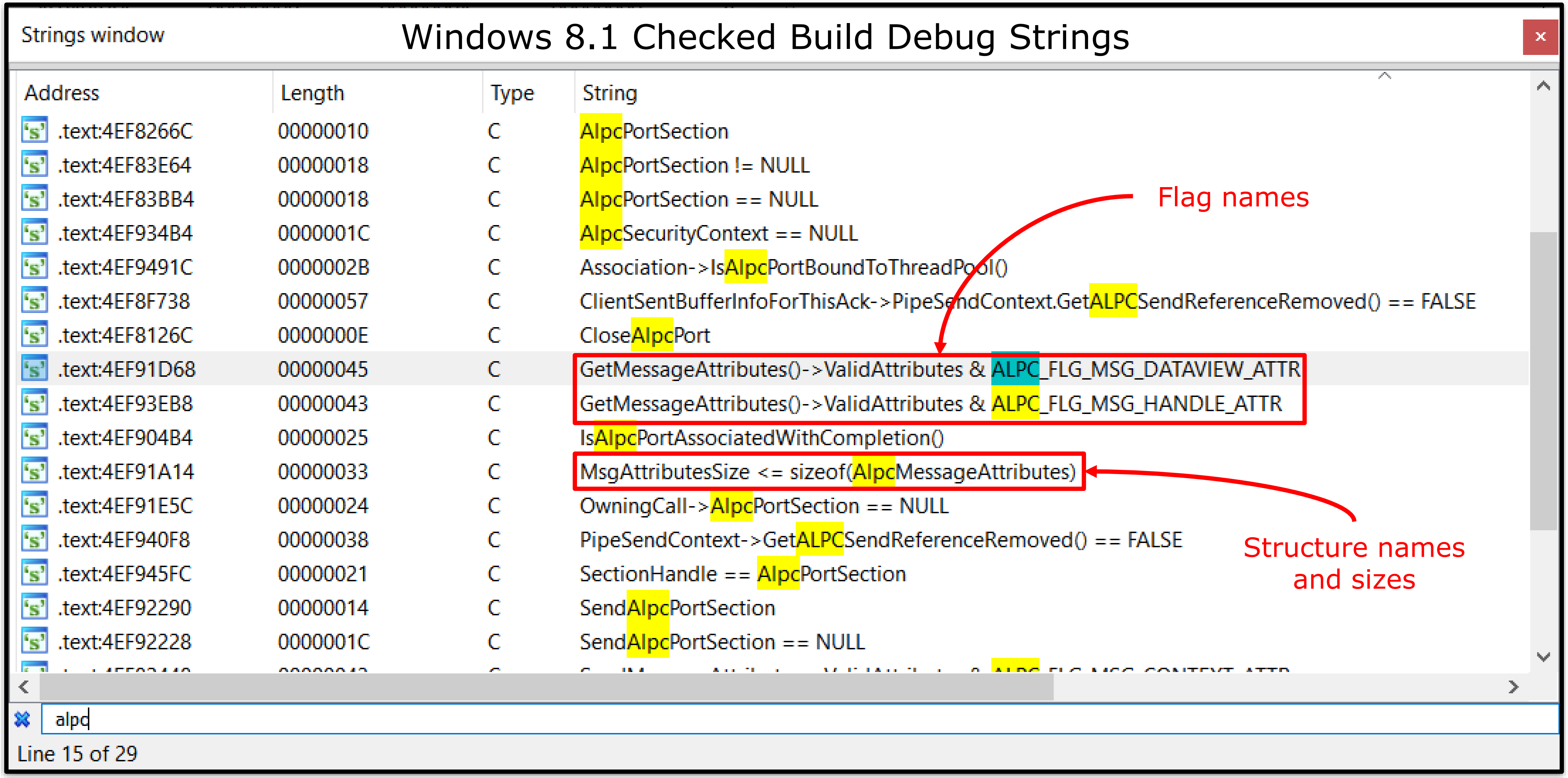
https://pacsec.jp/psj17/PSJ2017_Rouault_Imbert_alpc_rpc_pacsec.pdf

Fair Amount Unknown

```
class ALPC_RPC_BIND(ctypes.Structure):
  _fields_ = [
    ("request_type", gdef.DWORD),
    ("UNK1", gdef.DWORD),
    ("UNK2", gdef.DWORD),
    ("target", gdef.RPC_IF_ID),
    ("flags", gdef.DWORD),
    ("if_nb_ndr32", gdef.USHORT),
    ("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),
]
```

```
KNOW_REQUEST_TYPE =
gdef.FlagMapper(
    gdef.RPC_REQUEST_TYPE_CALL,
    gdef.RPC_REQUEST_TYPE_BIND)

KNOW_RESPONSE_TYPE =
  gdef.FlagMapper(
    gdef.RPC_RESPONSE_TYPE_FAIL,
    gdef.RPC_RESPONSE_TYPE_SUCCESS,
    gdef.RPC_RESPONSE_TYPE_BIND_OK)
```

A Simple Alex Ionescu Trick



Strings window

Windows 8.1 Checked Build Debug Strings

Address	Length	Type	String
.text:4EF8266C	00000010	C	AlpcPortSection
.text:4EF83E64	00000018	C	AlpcPortSection != NULL
.text:4EF83BB4	00000018	C	AlpcPortSection == NULL
.text:4EF934B4	0000001C	C	AlpcSecurityContext == NULL
.text:4EF9491C	0000002B	C	Association->IsAlpcPortBoundToThreadPool()
.text:4EF8F738	00000057	C	ClientSentBufferInfoForThisAck->PipeSendContext.GetALPCSendReferenceRemoved() == FALSE
.text:4EF8126C	0000000E	C	CloseAlpcPort
.text:4EF91D68	00000045	C	GetMessageAttributes()->ValidAttributes & ALPC_FLG_MSG_DATAVIEW_ATTR
.text:4EF93EB8	00000043	C	GetMessageAttributes()->ValidAttributes & ALPC_FLG_MSG_HANDLE_ATTR
.text:4EF904B4	00000025	C	IsAlpcPortAssociatedWithCompletion()
.text:4EF91A14	00000033	C	MsgAttributesSize <= sizeof(AlpcMessageAttributes)
.text:4EF91E5C	00000024	C	OwningCall->AlpcPortSection == NULL
.text:4EF940F8	00000038	C	PipeSendContext->GetALPCSendReferenceRemoved() == FALSE
.text:4EF945FC	00000021	C	SectionHandle == AlpcPortSection
.text:4EF92290	00000014	C	SendAlpcPortSection
.text:4EF92228	0000001C	C	SendAlpcPortSection == NULL

alpc

Line 15 of 29

Cleaned up Bind

```
struct LRPC_BIND_MESSAGE {
  public LRPC_HEADER Header;
  public int RpcStatus;
  public RPC_SYNTAX_IDENTIFIER Interface;
  public TransferSyntaxSetFlags TransferSyntaxSet;
  public short DceNdrSyntaxIdentifier;
  public short Ndr64SyntaxIdentifier;
  public short FakeNdr64SyntaxIdentifier;
  public bool RegisterMultipleSyntax;
  public bool UseFlowId;
  public long FlowId;
  public int ContextId;
}
```

```
struct LRPC_HEADER {
  public LRPC_MESSAGE_TYPE MessageType;
  public int Padding;
}
```

```
enum LRPC_MESSAGE_TYPE {
  lmtRequest = 0,
  lmtBind = 1,
  lmtFault = 2,
  lmtResponse = 3,
  lmtCancel = 4,
  lmtReservedMessage = 5
  lmtCallbackAck = 7,
  lmtCallbackNack = 8,
  lmtCallbackRequest = 9,
  lmtCallbackReply = 10,
  lmtCallbackFault = 11,
  lmtPipePull = 12,
  lmtPipeAck = 13
}
```

Finding the ALPC Port

```
RPC_STATUS RpcEpResolveBinding(
  RPC_BINDING_HANDLE Binding,
  RPC_IF_HANDLE      IfSpec
);
```

Query the Endpoint Mapper

```
using (var dir = NtDirectory.Open(@"\RPC Control")) {
    foreach (var port in dir.Query()) {
        try {
            using (var server = new RpcClient(
                                    interface_id, interface_version)) {
                server.Connect(port.Name);
                    return port.Name;
            }
        }  catch {}
    }
}
```

Brute Force

Generating C# Code

Mapping Types

Type	*IDL*	*C# (In)*	*C# (Out)*	*C# (In/Out)*
Primitive	int	int	out int	ref int
Structure	GUID*	Guid	out Guid	ref Guid
String ANSI	[string] char*	string	out string	ref string
String Unicode	[string] wchar_t*	string	out string	ref string
Unique Pointer	[unique] GUID*	Nullable<Guid>	out Nullable<Guid>	ref Nullable<Guid>
Unique Pointer	[unique] char*	string	out string	ref string
Array	BYTE[]	byte[]	out byte[]	ref byte[]
Binding Handle	handle_t	REMOVED	N/A	N/A
Context Handle	CUSTOM*	NdrContextHandle	out NdrContextHandle	ref NdrContextHandle
System Handle	HANDLE	NtFile	out NtFile	ref NtFile

Simple Example

```
int Func([in] handle_t h, [in] int i, [out] int* o);
```

```
int Func(int p0, out int p1) {
  NdrMarshalBuffer m = new NdrMarshalBuffer();
  m.WriteInt32(p0);
  NdrUnmarshalBuffer u = SendReceive(1, m);
  p1 = u.ReadInt32();
  return u.ReadInt32();
}
```

Procedure Number

Dealing with Out Parameters in PowerShell

```
PS> $p1 = 0
PS> $client.Func(1234, [ref]$p1)
```

```
struct Func_Ret {
  public int p1;
  public int retval;
}
```

```
Func_Ret Func(int p0) {
  var m = new NdrMarshalBuffer();
  m.WriteInt32(p0);
  var u = SendReceive(1, m);
  Func_Ret r = new Func_Ret();
  r.p1 = u.ReadInt32();
  r.retval = u.ReadInt32();
  return r;
}
```

General and compile a client for a parsed RPC server.

```
PS> $client = Get-RpcClient $rpc_server
```

Connect an RPC client, try and lookup ALPC port from Endpoint Mapper

```
PS> Connect-RpcClient $client
```

Connect an RPC client, try and lookup ALPC port by brute force.

```
PS> Connect-RpcClient $client -FindAlpcPort
```

Format one or more RPC servers as a C# a write to c:\out.

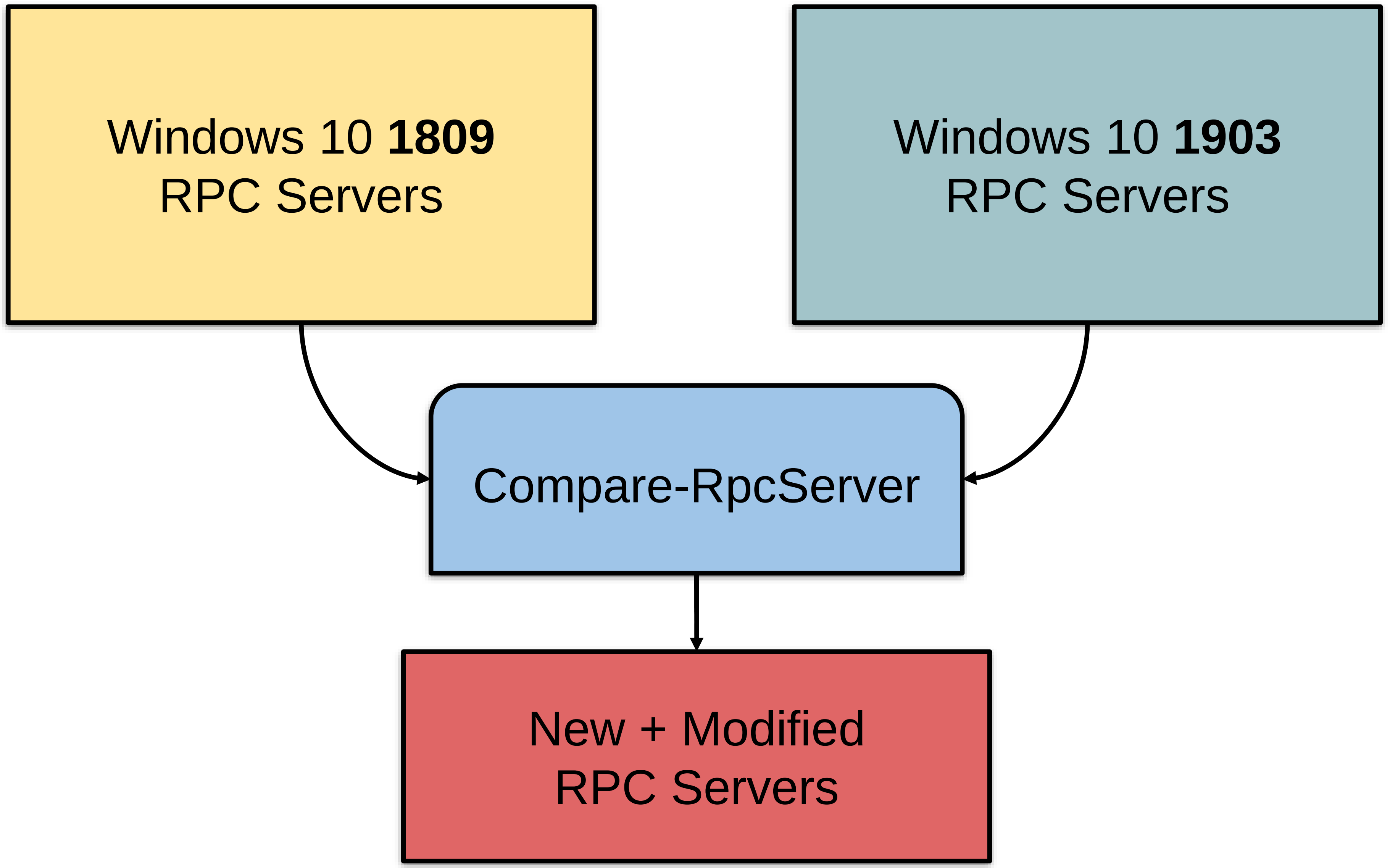
```
PS> $rpc | Format-RpcClient -OutputPath "c:\out"
```

Constructors for “Struct_1” exposed off client’s “New” Property

```
PS> $client.New.Struct_1()
```

Use Cases

Find New Windows RPC Apis

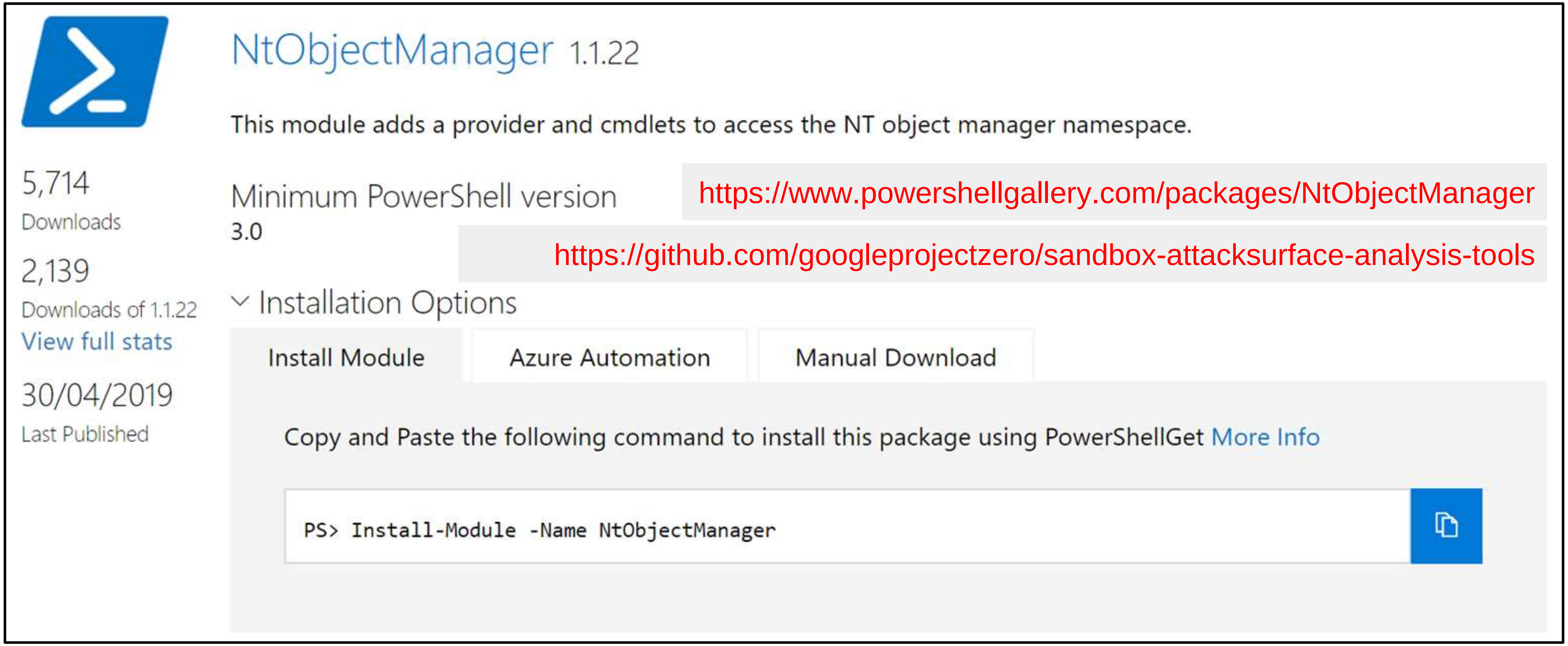


Fuzzing

```
static void FuzzCall(RpcClientBase client) {
    MethodInfo[] mis = client.GetType()
             .GetMethods(BindingFlags.DeclaredOnly);

    foreach (var mi in mis) {
      List<object> ps = new List<object>();
      foreach (var pi in mi.GetParameters()) {
        ps.Add(FuzzParam(pi));
      }
      mi.Invoke(client, ps.ToArray());
    }
}
```

Wrapping Up

Managed RPC Client All Available Today



NtObjectManager 1.1.22

This module adds a provider and cmdlets to access the NT object manager namespace.

5,714
Downloads

2,139
Downloads of 1.1.22

View full stats

30/04/2019
Last Published

Minimum PowerShell version
3.0

https://www.powershellgallery.com/packages/NtObjectManager

https://github.com/googleprojectzero/sandbox-attacksurface-analysis-tools

Copy and Paste the following command to install this package using PowerShellGet More Info

```
PS> Install-Module -Name NtObjectManager
```

DEMO

Possible Future Work

- Implement parsing NDR64 byte code and NDR64 wire format.
- Support Pipes and some misc other types.
- Implement asynchronous support.
- Implement transports for Named Pipes and TCP.
- Add server support.