

Introduction to Logical Privilege Escalation on Windows

2Hr Workshop

James Forshaw - @tiraniddo

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Agenda of this Workshop

- Windows Internals as relevant to privilege escalation
- Attack surface analysis from sandboxes and normal user
- Bug classes and Vulnerability Exploitation
- Willing to answer questions as I go along, however it might need to be saved till the end depending on the question :-)

Sorry, only so much I can talk about in 2hrs.
Some things are going to be missed :(

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Setup the Tools and Examples

- Download toolset and workbook from link below
- Ideally you want a VM of Windows 10 Anniversary Edition
 - 32 bit preferred, but 64 bit should also work for most things
- Extract contents to *c:\workshop* in the VM
- Read *setup.txt* for instructions on setting up a few things.

<https://goo.gl/HzZ2Gw>

<https://goo.gl/P4Q9GN>

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

What is a Logical Vulnerability?

A security vulnerability which rely on subverting the programmer's original logic rather than abusing unintended behaviour.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Why?

● Why Privilege Escalation?

- Everything is getting sandboxed!
 - Even Firefox (probably).
- Everyone is running as a normal user
 - Or should be, of course there's UAC, but well.

● Why Logical Exploitation?

- Exploiting memory corruption is getting more difficult
 - Stack cookies, hardened heaps
 - Control Flow and Return Flow Guard
 - SMEP preventing trivial kernel code execution
 - ASLR, DEP and all that, including limiting information leakage from kernel.
- Exploiting memory corruptions is boring ;-)

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

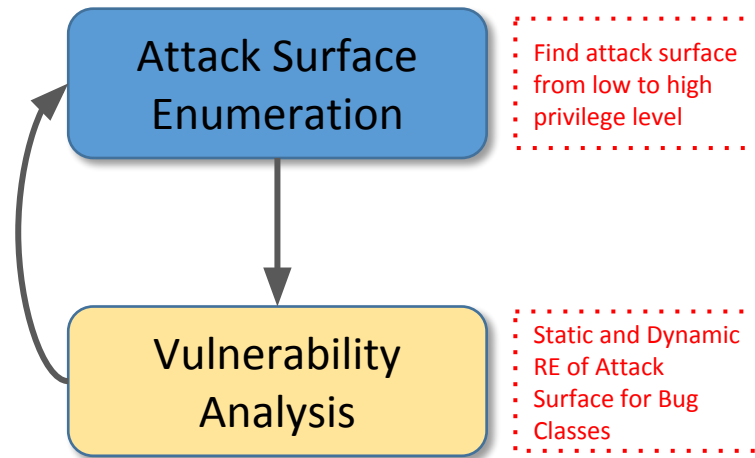
My Approach to Finding Logical Vulnerabilities

Attack Surface
Enumeration

Find attack surface
from low to high
privilege level

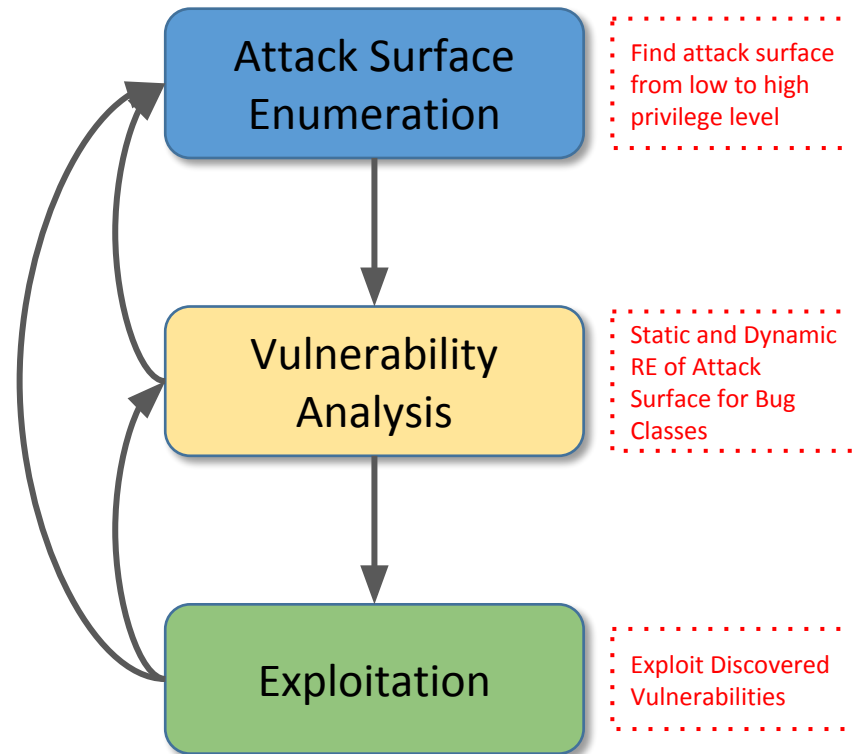
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

My Approach to Finding Logical Vulnerabilities



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

My Approach to Finding Logical Vulnerabilities

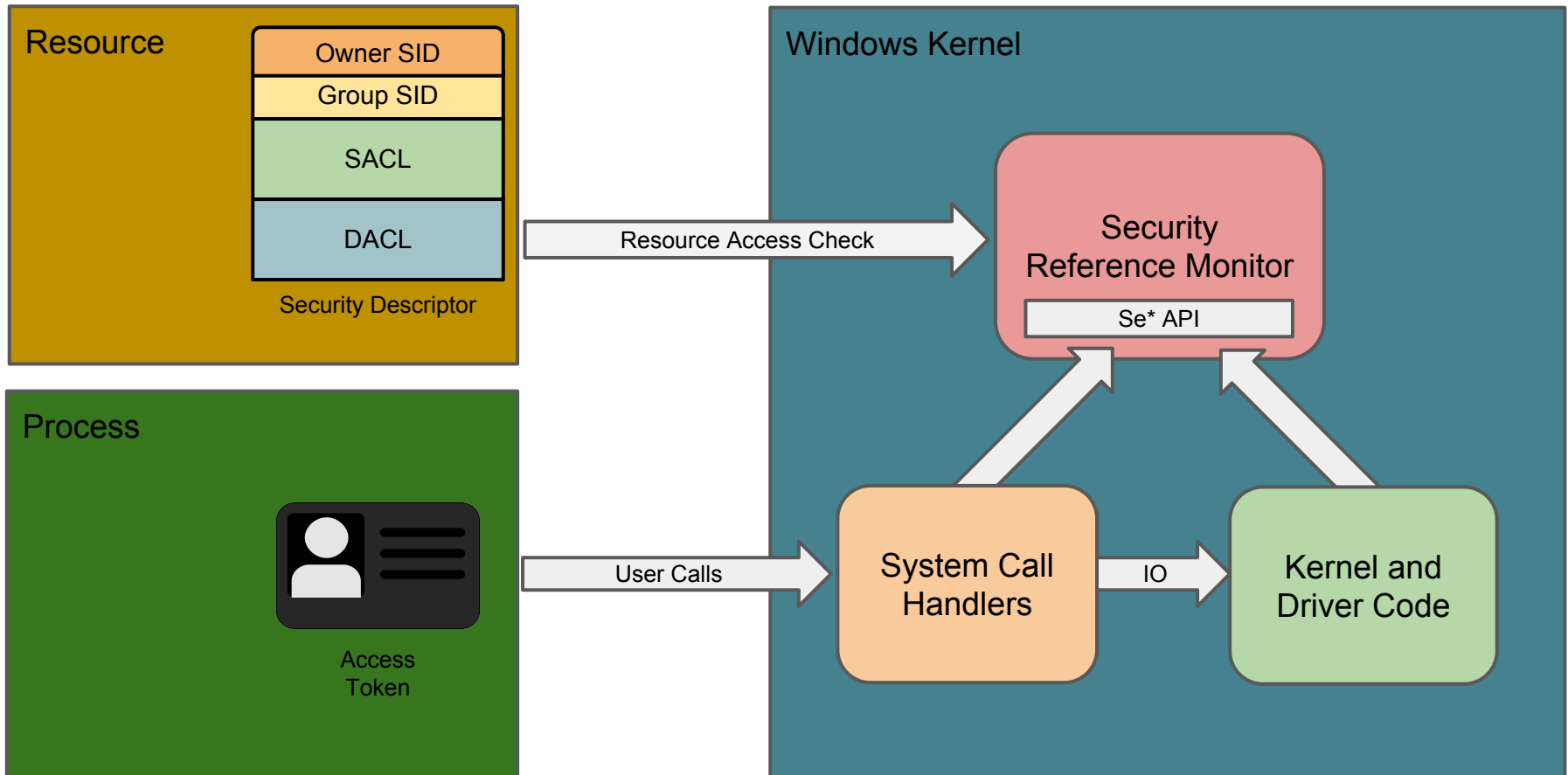


Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows Internals for EoP Hunters

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows Security Components



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Access Token

User Security Identifier

Groups

Mandatory Label

Privileges

The screenshot shows the 'chrome.exe:2716 Properties' dialog box with the 'Security' tab selected. The 'User' field displays 'WIN-32RI1CK49EL\user' and the 'SID' field displays 'S-1-5-21-3711643808-3202222375-1035956708-1001'. Below this, a list of groups is shown, including 'BUILTIN\Administrators', 'BUILTIN\Users', 'CONSOLE LOGON', 'Everyone', 'LOCAL', 'Logon SID (S-1-5-5-0-5071873)', 'Mandatory Label\Medium Mandatory Level', and 'NT AUTHORITY\Authenticated Users'. The 'Mandatory Label\Medium Mandatory Level' entry is highlighted. At the bottom, a list of privileges is shown, including 'SeChangeNotifyPrivilege', 'SeIncreaseWorkingSetPrivilege', 'SeShutdownPrivilege', 'SeTimeZonePrivilege', and 'SeUndockPrivilege'. Red arrows point from labels to these specific fields.

Group	Flags
BUILTIN\Administrators	Deny
BUILTIN\Users	Mandatory
CONSOLE LOGON	Mandatory
Everyone	Mandatory
LOCAL	Mandatory
Logon SID (S-1-5-5-0-5071873)	Mandatory
Mandatory Label\Medium Mandatory Level	Integrity
NT AUTHORITY\Authenticated Users	Mandatory

Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled
SeIncreaseWorkingSetPrivilege	Disabled
SeShutdownPrivilege	Disabled
SeTimeZonePrivilege	Disabled
SeUndockPrivilege	Disabled

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Security Identifiers

- A Security Identifier (SID) is how Windows represents a user or group (think of it like an expanded UID/GID from Unix)



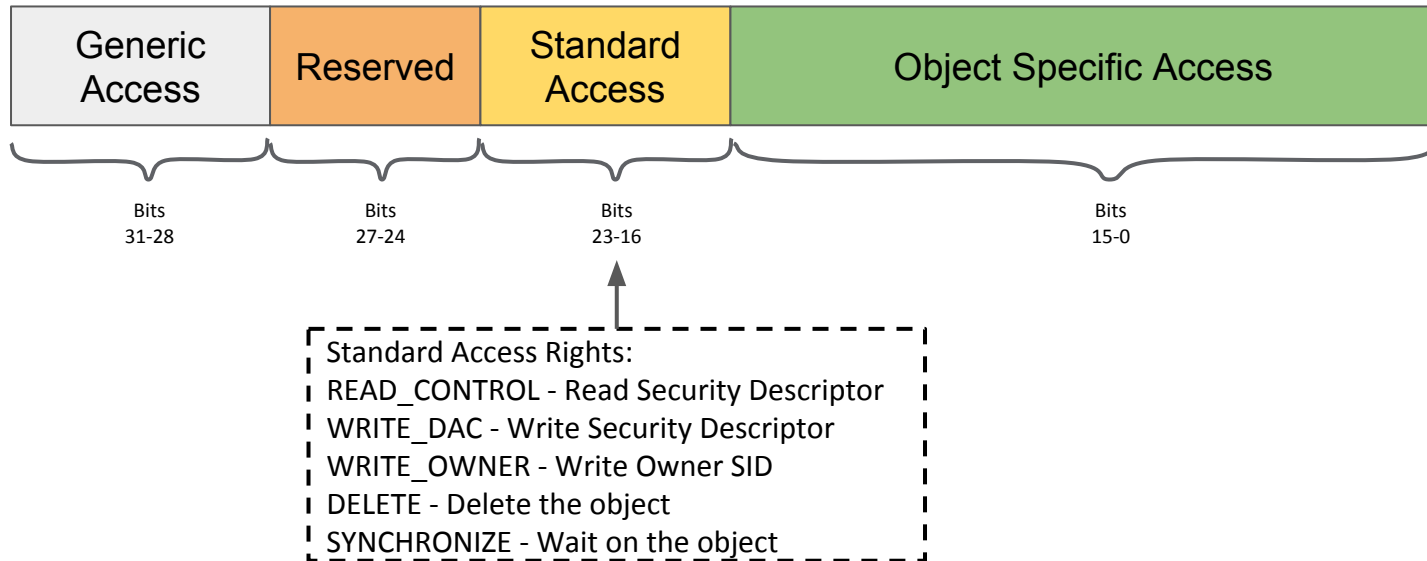
- Some well known SIDs:

World/Everyone	S-1-1-0
Creator Owner	S-1-3-0
Local SYSTEM	S-1-5-18
Authenticated Users	S-1-5-11
Anonymous	S-1-5-7

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

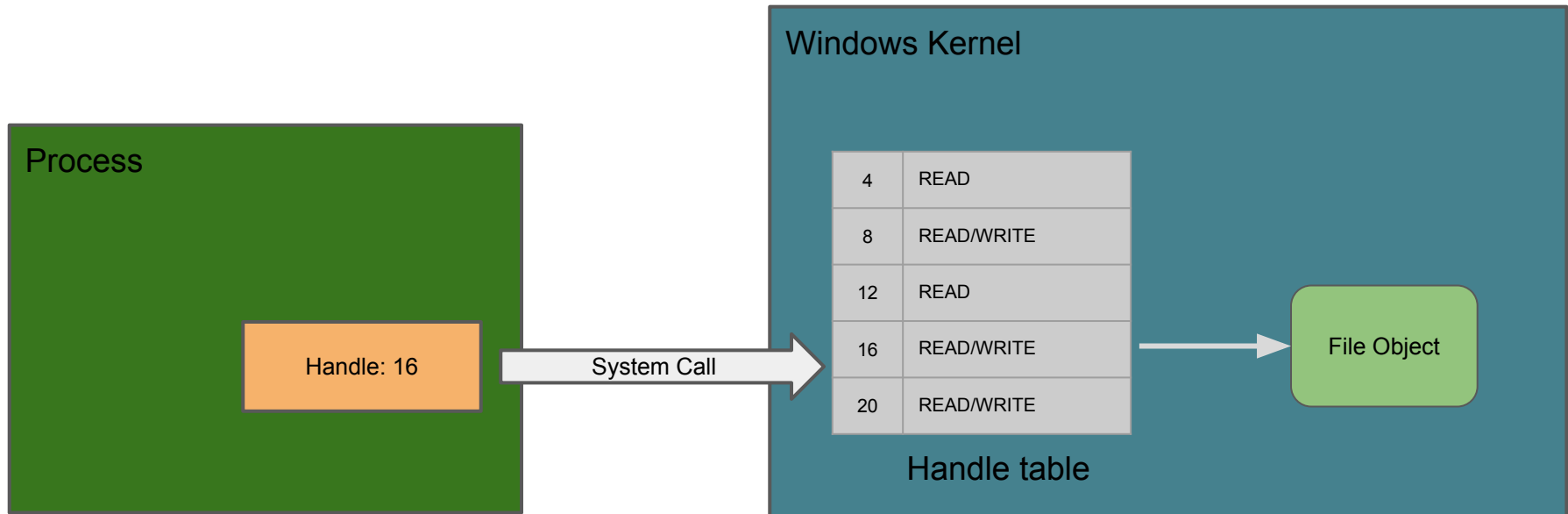
Access Masks

- When opening a handle need specify the access mask.
- Checked against the access mask in the DACL entries



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Handles



```
NTSTATUS ObReferenceObjectByHandle(  
    _In_ HANDLE Handle,  
    _In_ ACCESS_MASK DesiredAccess,  
    _In_opt_ POBJECT_TYPE ObjectType,  
    _In_ KPROCESSOR_MODE AccessMode,  
    _Out_ PVOID *Object,  
    _Out_opt_ POBJECT_HANDLE_INFORMATION HandleInformation  
);
```

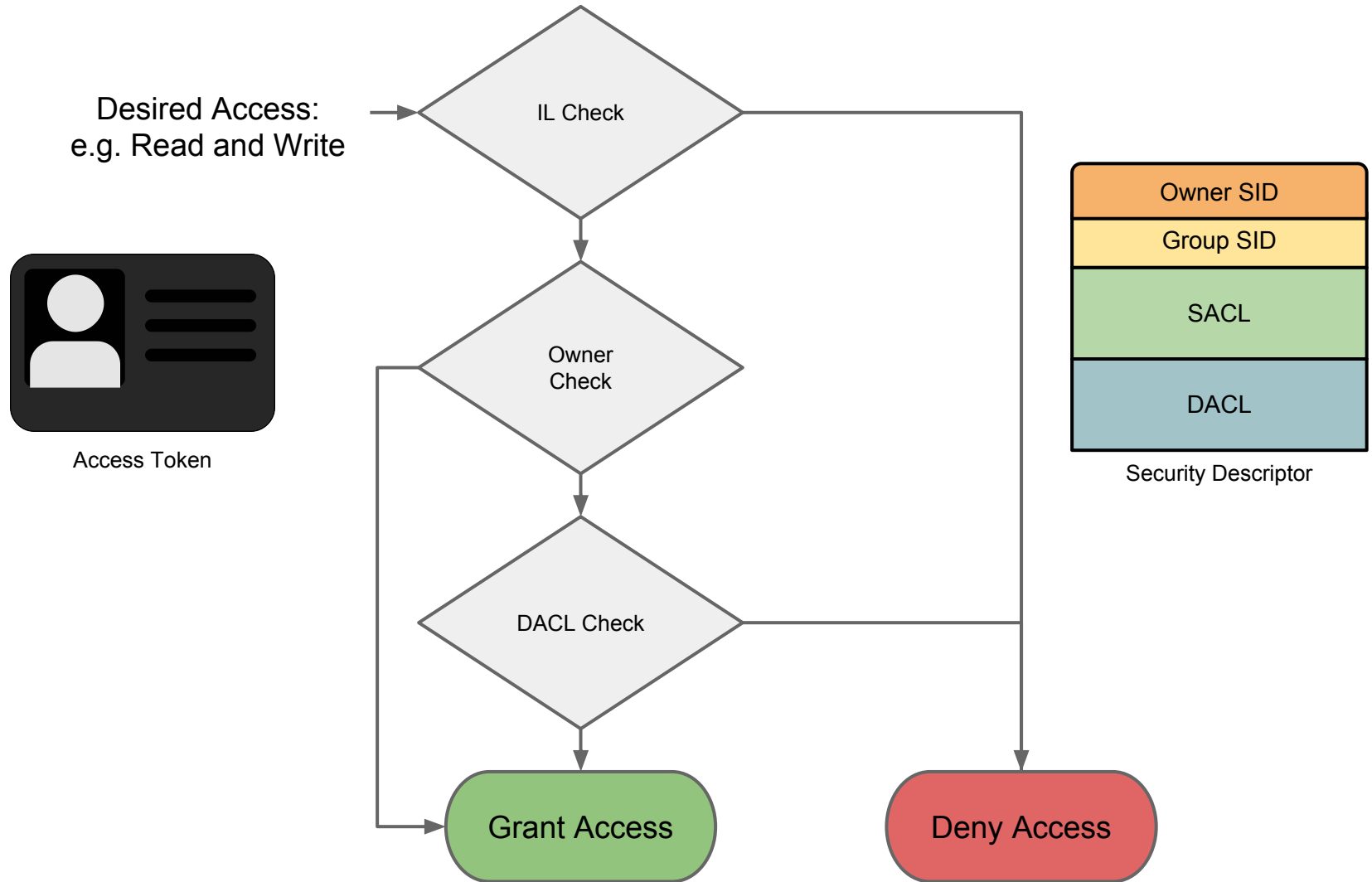
Handle from user mode.

Required Access

Kernel Object Pointer

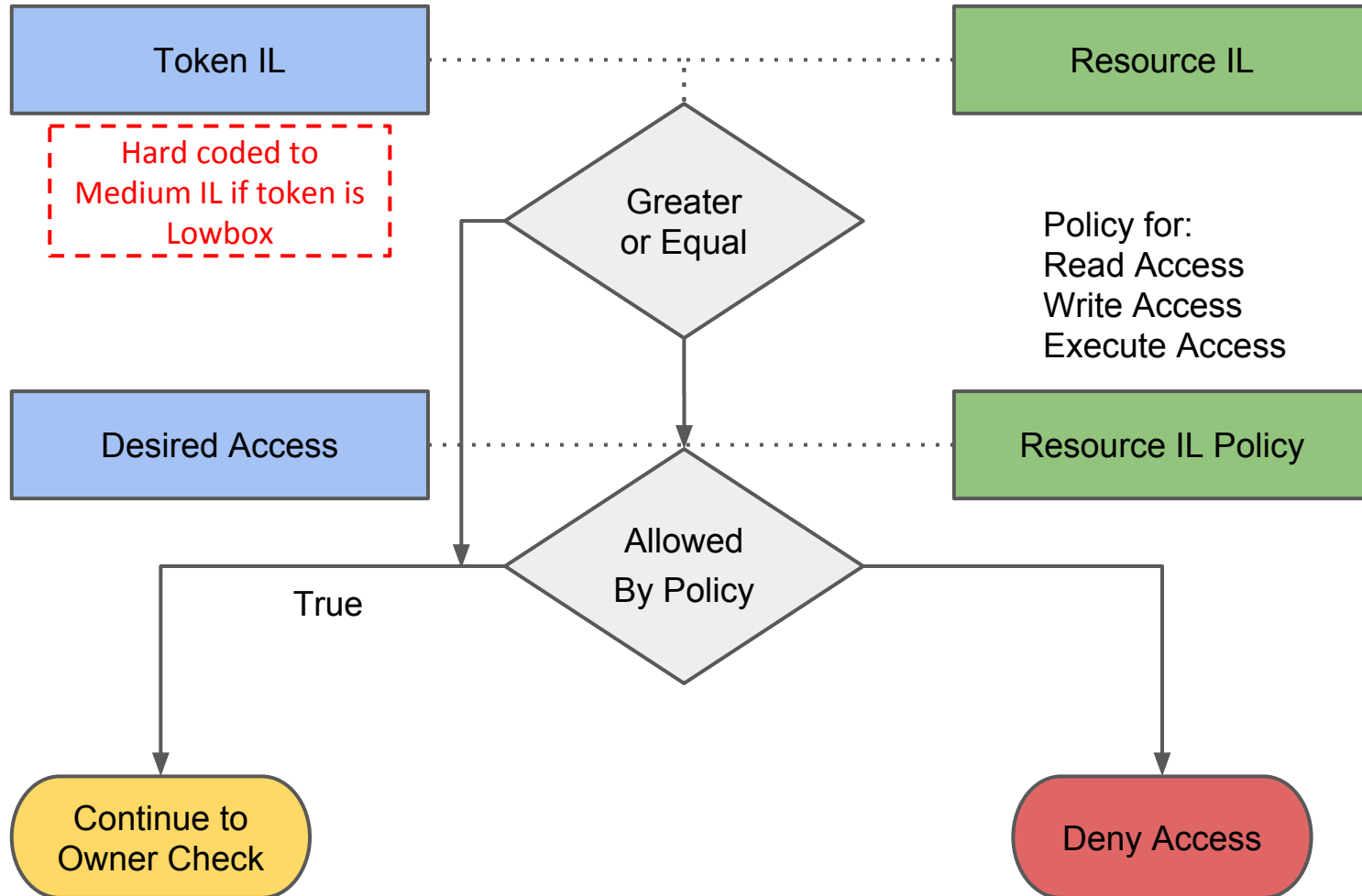
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Security Access Check (SeAccessCheck)



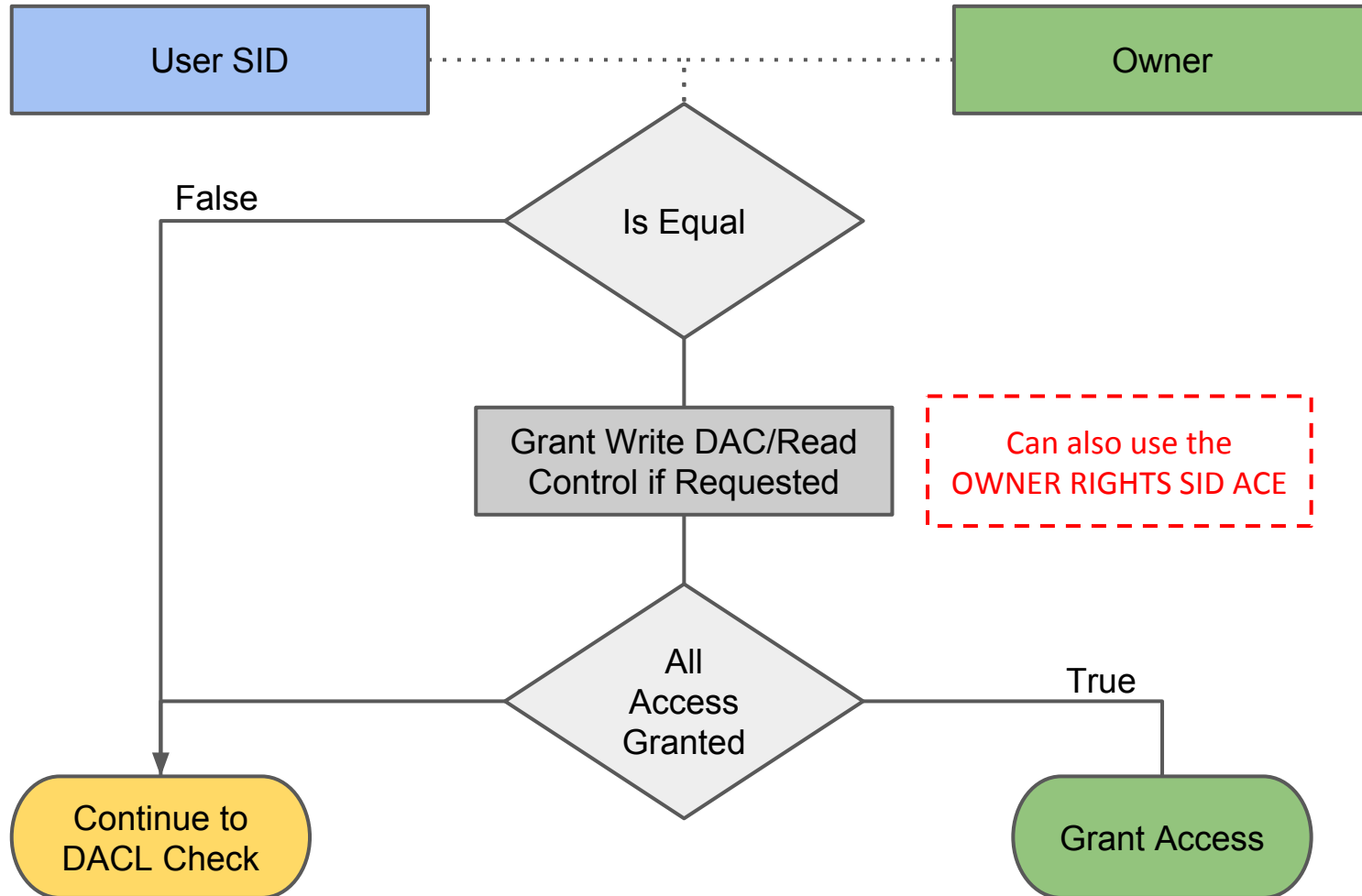
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Mandatory Integrity Level Check



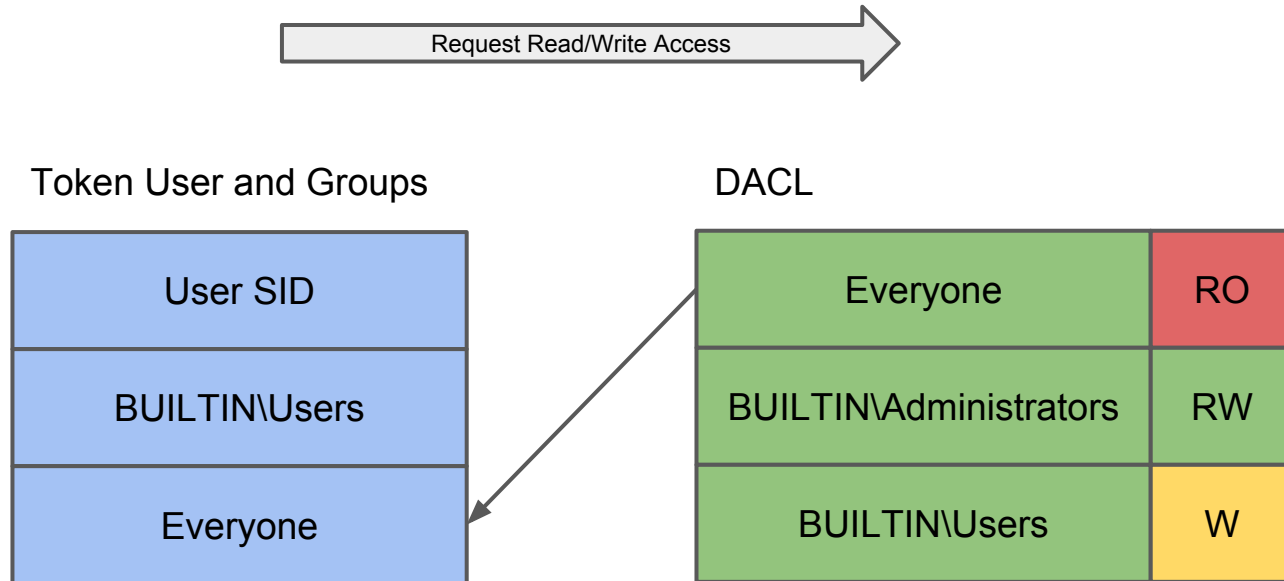
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Owner Check



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

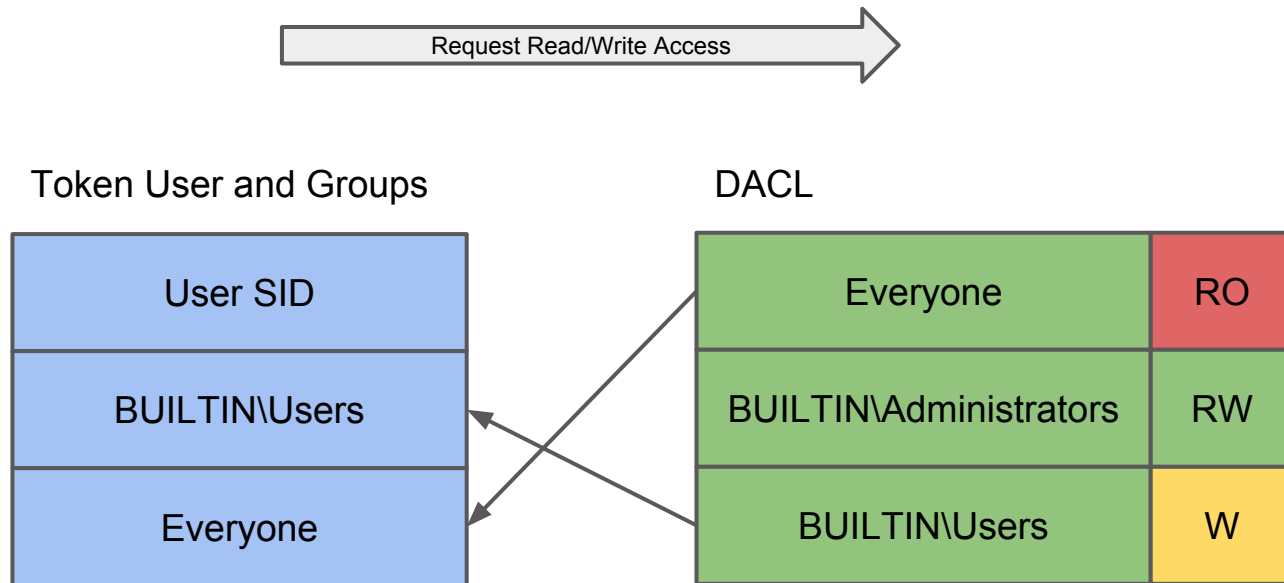
Kernel DACL Check



Current Granted Access: Read Only

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Kernel DACL Check



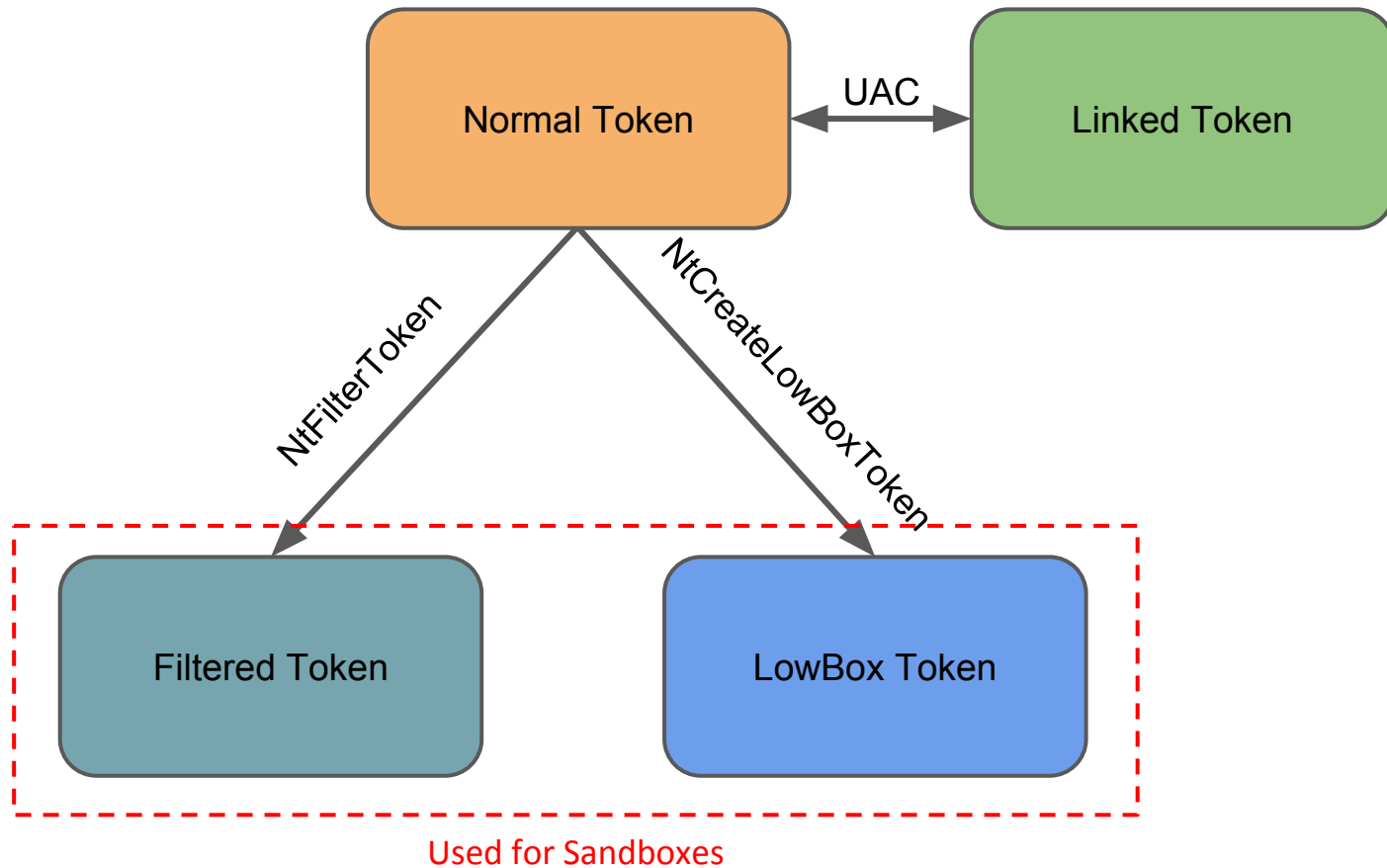
Final Granted Access: Read/Write

Security Descriptors and Inheritance

- New resources by default will inherit Security Descriptor for parent container (be it object directory/file directory/registry key etc.)
- Most resource creation calls can specify explicit SD
- If no inheritable ACEs, uses default DACL.
 - Even for Files, which is an odd behaviour.
- Special ACEs
 - OWNER RIGHTS - Limits/Grants Owner Access
 - CREATOR OWNER - SID replaced during inheritance with current owner SID
 - SELF - Replaced by the SID specified in AccessCheckByType

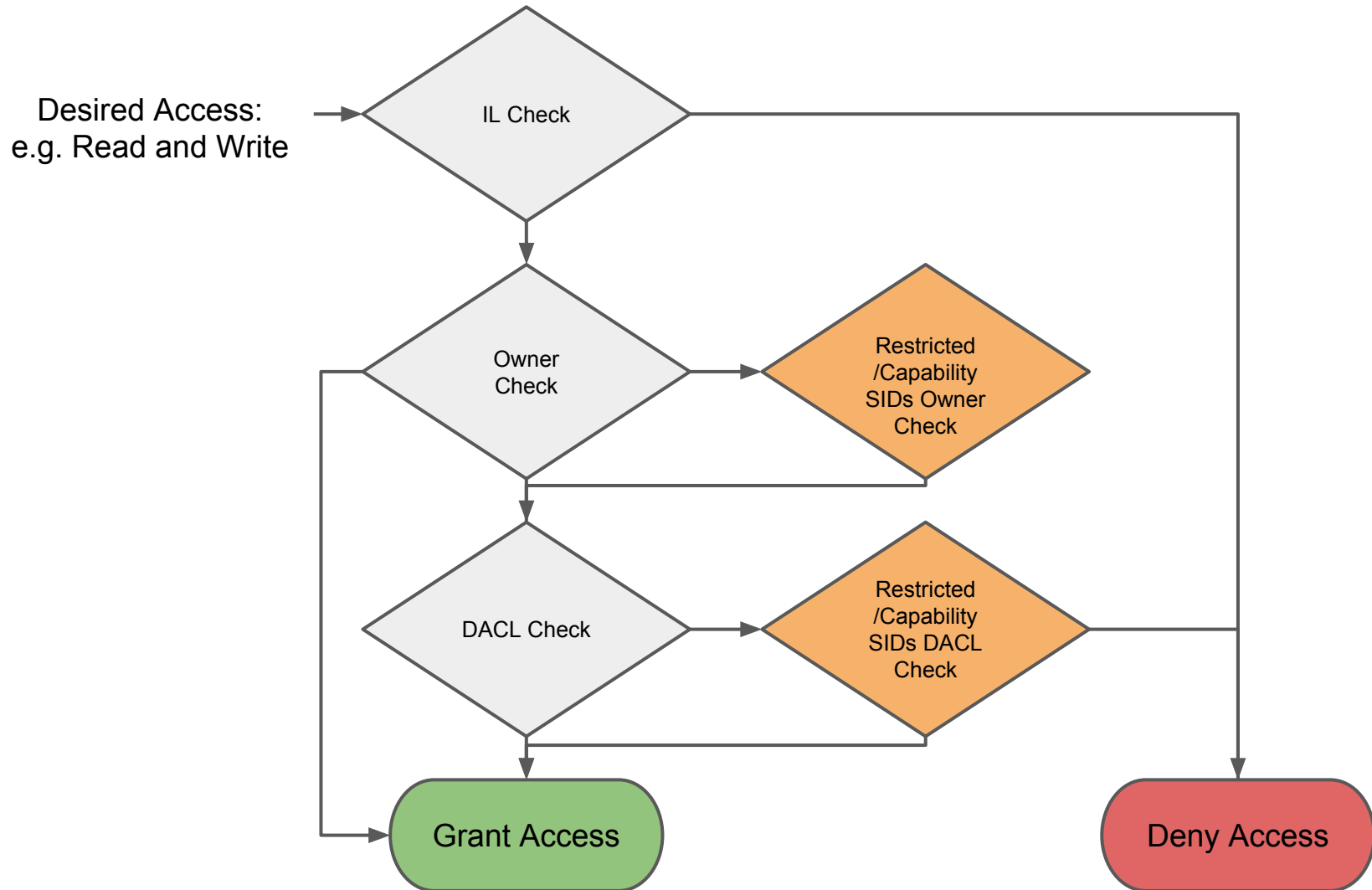
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Token Categories



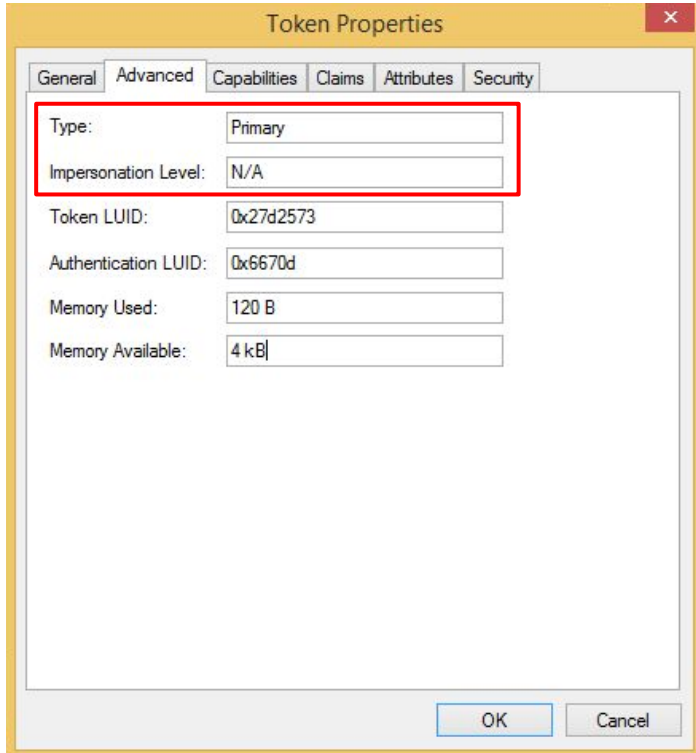
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Restricted/Lowbox Token Access Check



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

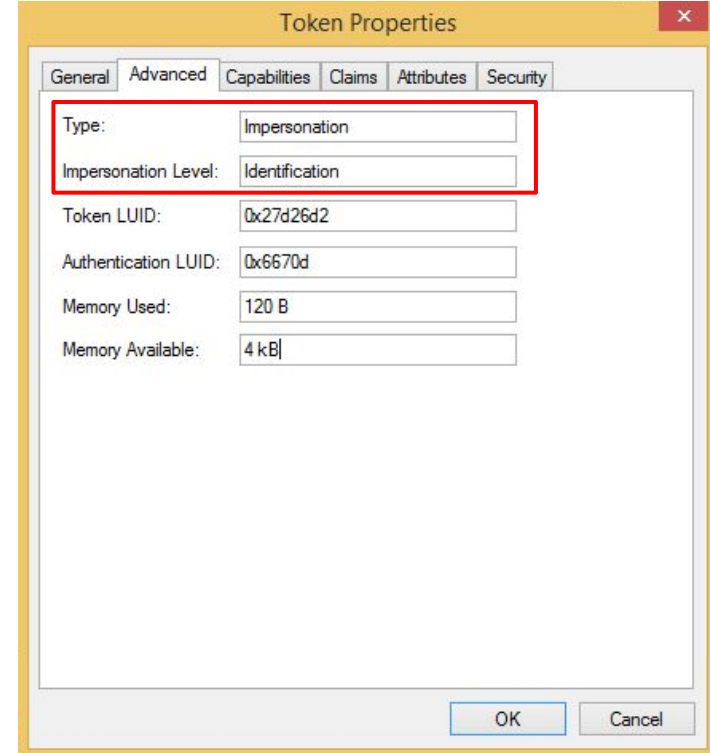
Access Token Types



The image shows a 'Token Properties' dialog box with the 'General' tab selected. A red rectangle highlights the 'Type' and 'Impersonation Level' fields. The 'Type' field is set to 'Primary' and the 'Impersonation Level' field is set to 'N/A'. Other fields include 'Token LUID' (0x27d2573), 'Authentication LUID' (0x6670d), 'Memory Used' (120 B), and 'Memory Available' (4 kB). The 'OK' and 'Cancel' buttons are at the bottom right.

Field	Value
Type	Primary
Impersonation Level	N/A
Token LUID	0x27d2573
Authentication LUID	0x6670d
Memory Used	120 B
Memory Available	4 kB

Primary



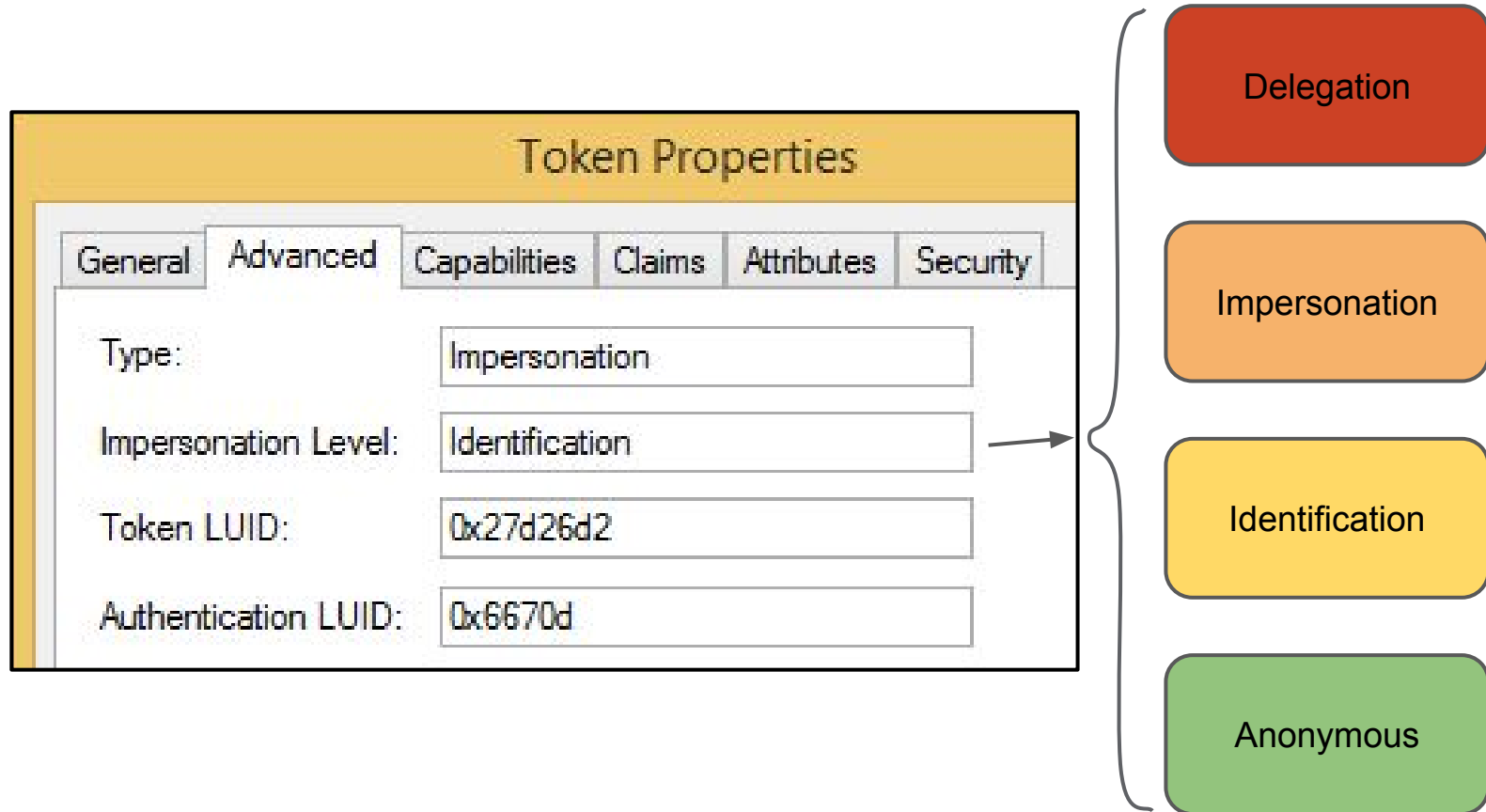
The image shows a 'Token Properties' dialog box with the 'General' tab selected. A red rectangle highlights the 'Type' and 'Impersonation Level' fields. The 'Type' field is set to 'Impersonation' and the 'Impersonation Level' field is set to 'Identification'. Other fields include 'Token LUID' (0x27d26d2), 'Authentication LUID' (0x6670d), 'Memory Used' (120 B), and 'Memory Available' (4 kB). The 'OK' and 'Cancel' buttons are at the bottom right.

Field	Value
Type	Impersonation
Impersonation Level	Identification
Token LUID	0x27d26d2
Authentication LUID	0x6670d
Memory Used	120 B
Memory Available	4 kB

Impersonation

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Impersonation Security Level



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Setting an Impersonation Token

- Impersonation assigns a token to a thread, replaced the token used in access checks for the majority of system calls

Direct Setting

SetThreadToken()
ImpersonateLoggedOnUser()
NtSetInformationThread(...)

Indirect Setting

ImpersonateNamedPipeClient()
RpcImpersonateClient()
CoImpersonateClient()

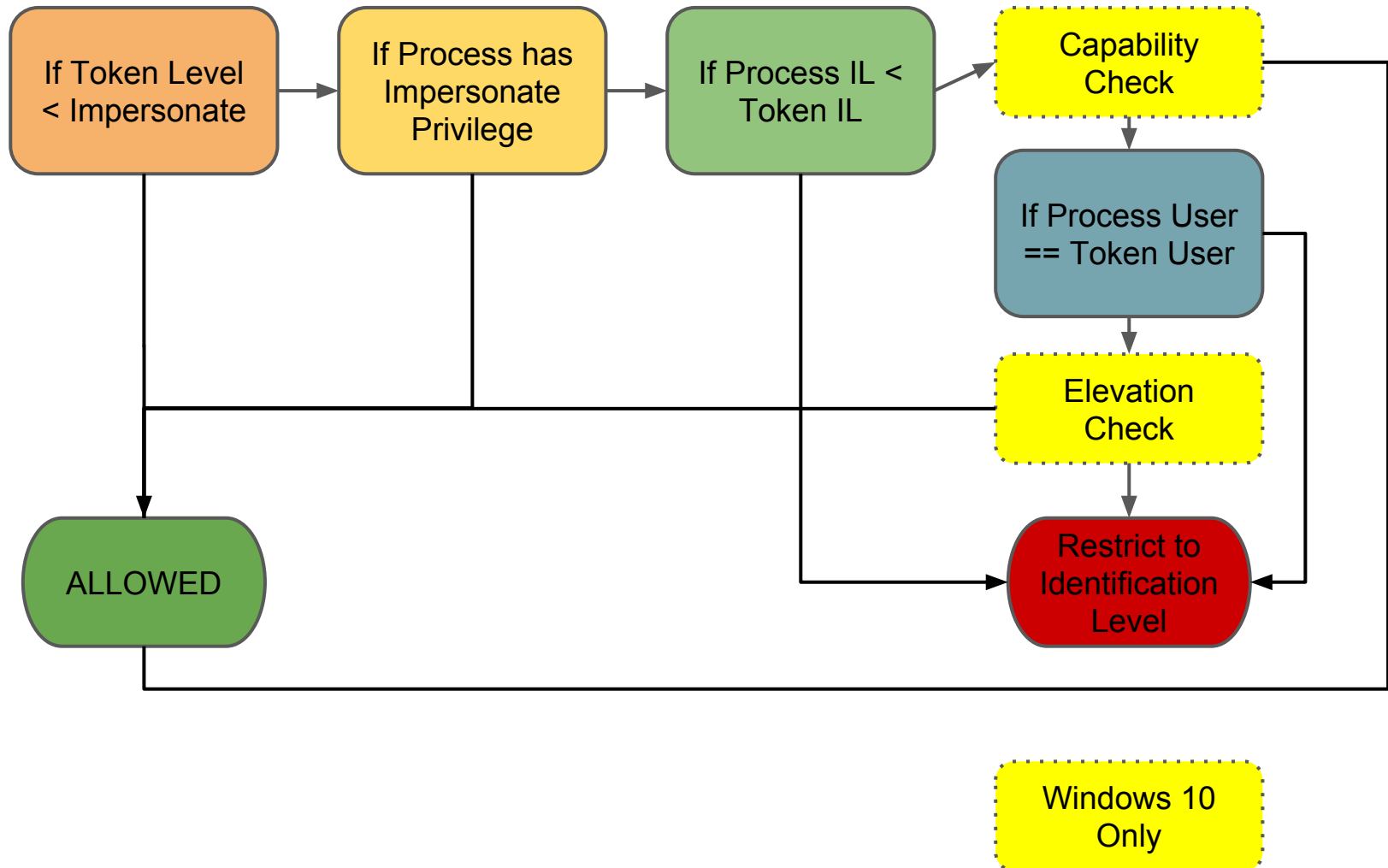
Kernel Setting

PsImpersonateClient()
SeImpersonateClient/Ex()

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

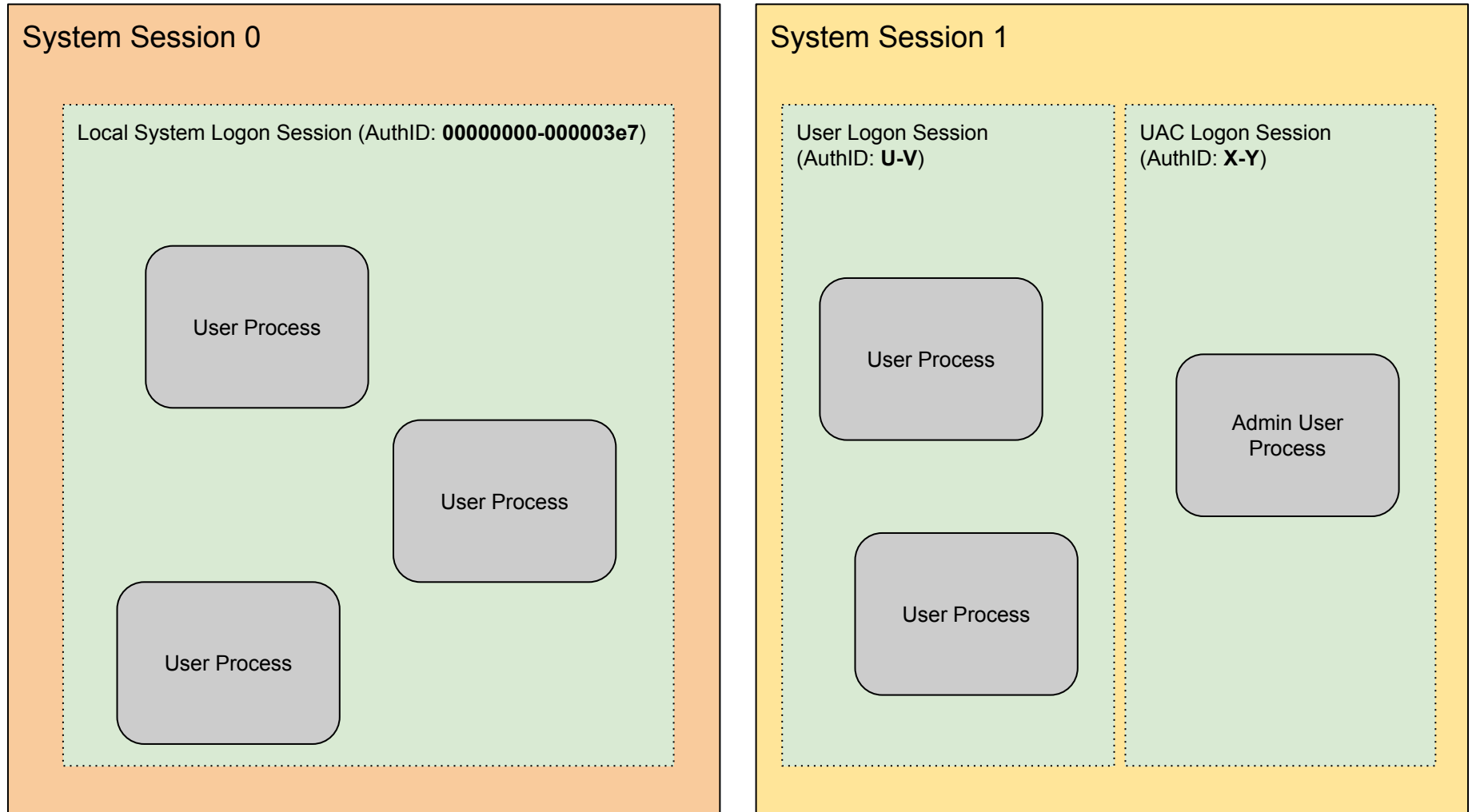
Impersonation Security

PsImpersonateClient(...) ► SeTokenCanImpersonate(...)



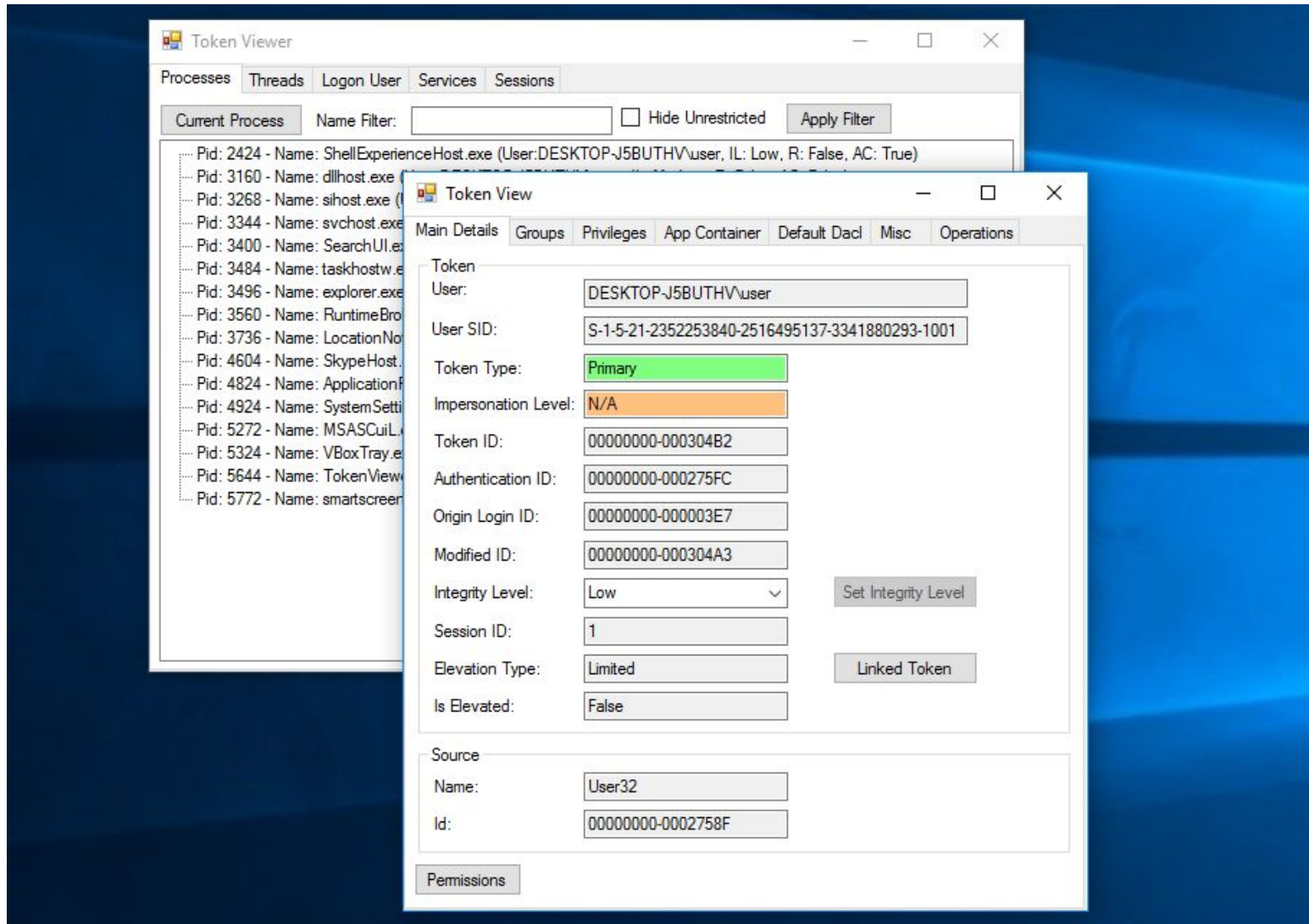
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Login Sessions



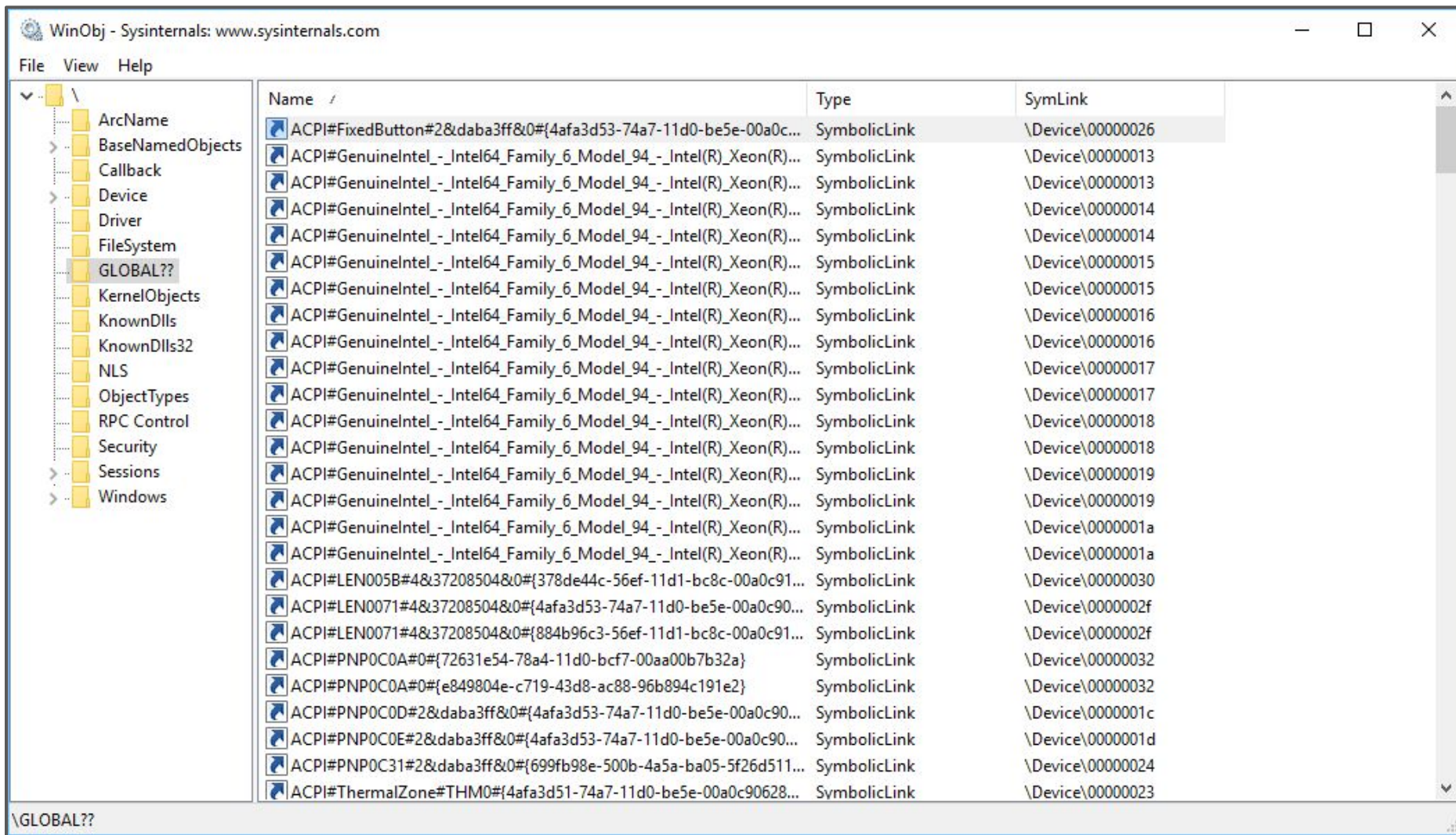
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

DEMO 1: Viewing Token and Security Descriptors



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Object Manager Namespace



WinObj - Sysinternals: www.sysinternals.com

File View Help

GLOBAL??

Name	Type	SymLink
ACPI\FixedButton#2&daba3ff&0#{4afa3d53-74a7-11d0-be5e-00a0c...	SymbolicLink	\Device\00000026
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000013
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000013
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000014
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000014
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000015
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000015
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000016
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000016
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000017
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000017
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000018
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000018
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000019
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\00000019
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\0000001a
ACPI\GenuineIntel_-_Intel64_Family_6_Model_94_-_Intel(R)_Xeon(R)...	SymbolicLink	\Device\0000001a
ACPI\LEN005B#4&37208504&0#{378de44c-56ef-11d1-bc8c-00a0c91...	SymbolicLink	\Device\00000030
ACPI\LEN0071#4&37208504&0#{4afa3d53-74a7-11d0-be5e-00a0c90...	SymbolicLink	\Device\0000002f
ACPI\LEN0071#4&37208504&0#{884b96c3-56ef-11d1-bc8c-00a0c91...	SymbolicLink	\Device\0000002f
ACPI\PNP0C0A#0#{72631e54-78a4-11d0-bcf7-00aa00b7b32a}	SymbolicLink	\Device\00000032
ACPI\PNP0C0A#0#{e849804e-c719-43d8-ac88-96b894c191e2}	SymbolicLink	\Device\00000032
ACPI\PNP0C0D#2&daba3ff&0#{4afa3d53-74a7-11d0-be5e-00a0c90...	SymbolicLink	\Device\0000001c
ACPI\PNP0C0E#2&daba3ff&0#{4afa3d53-74a7-11d0-be5e-00a0c90...	SymbolicLink	\Device\0000001d
ACPI\PNP0C31#2&daba3ff&0#{699fb98e-500b-4a5a-ba05-5f26d511...	SymbolicLink	\Device\00000024
ACPI\ThermalZone#THM0#{4afa3d51-74a7-11d0-be5e-00a0c90628...	SymbolicLink	\Device\00000023

\GLOBAL??

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Important Object Directories

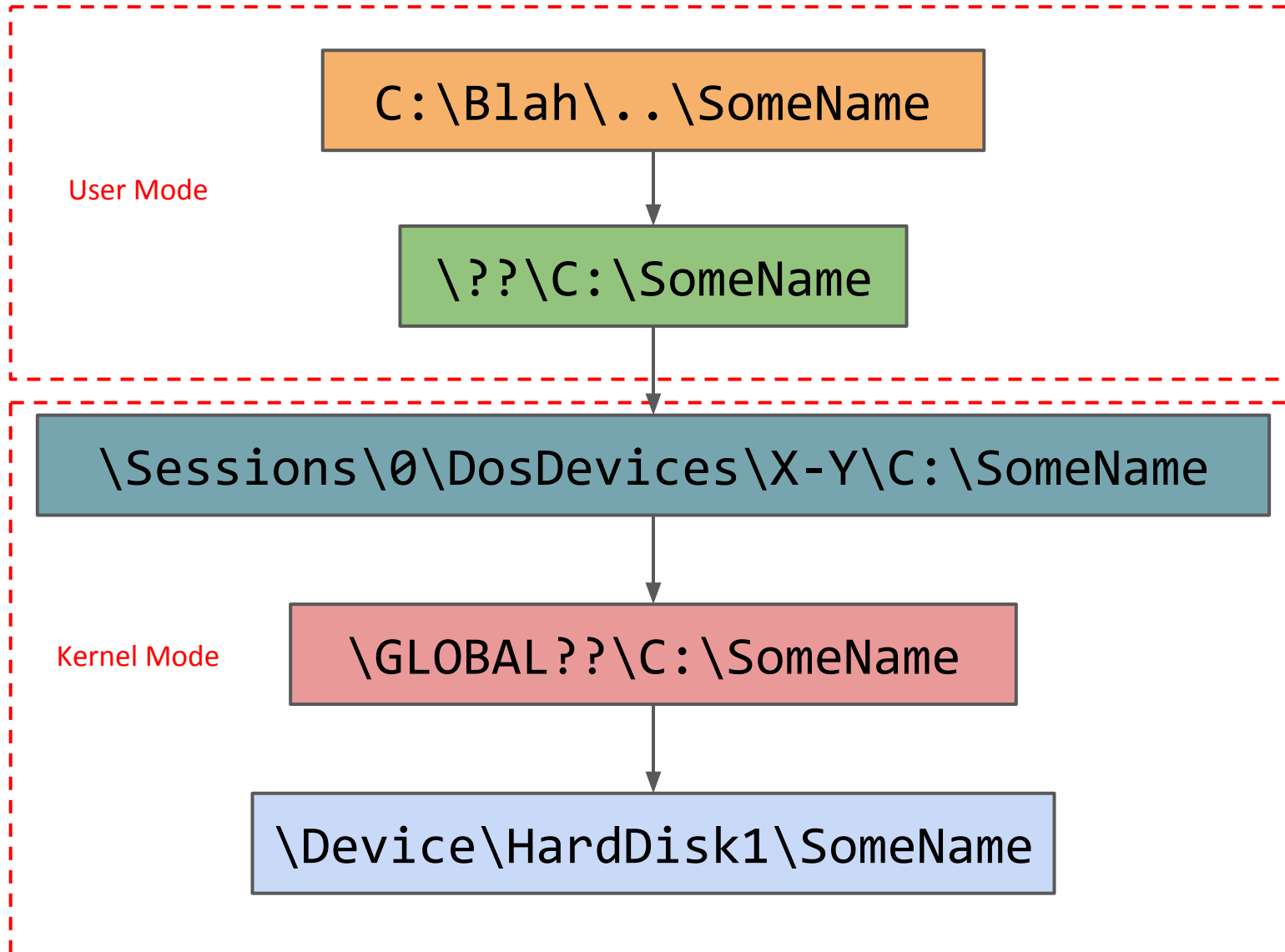
<i>Path</i>	<i>Description</i>
\Device	Default location for kernel driver Device Objects
\GLOBAL??	System location for symbolic links to devices including drive letters
\BaseNamedObjects	System location for named resources
\Sessions\X	Directory for the login session X
\Session\0\DosDevices	Directory for the “Dos Devices” for each logged in user.
\??	“Fake” prefix which refers to per-user Dos Devices.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Win32 Path Support

Path	Description
some\path	Relative path to current directory
c:\some\path	Absolute directory
\\.c:\some\path	Device path, canonicalized
\\?\c:\some\path	Device path, non-canonicalized
\\server\share\path	UNC path to share on server

File Path Handling



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Canonicalization

- Type of Win32 path affects canonicalization behaviour

Path	Result of Canonicalization
c:\path\../badgers	c:\badgers
c:\..\d:/badgers	c:\d:\badgers
\\.\c:\path\../badgers	c:\badgers
\\.\c:\..\d:/badgers	d:\badgers (WTF!)
\\?\c:\path\../badgers	c:\path\../badgers

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

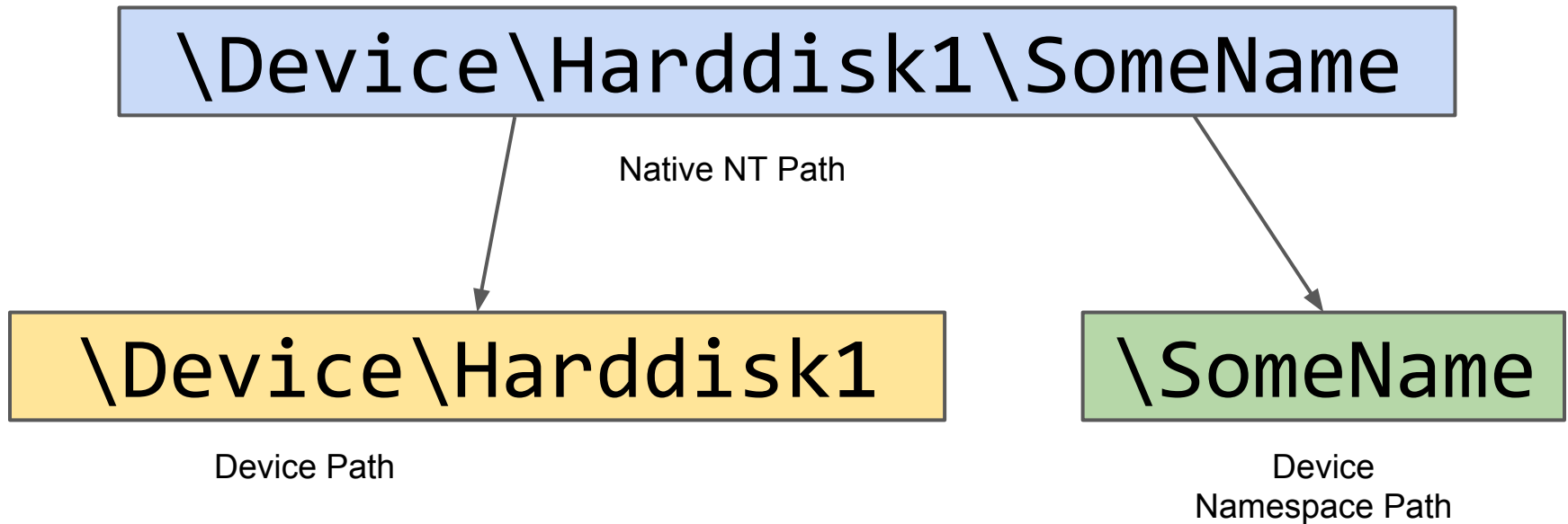
Opening a Device Name

`\Device\Harddisk1\SomeName`

Native NT Path

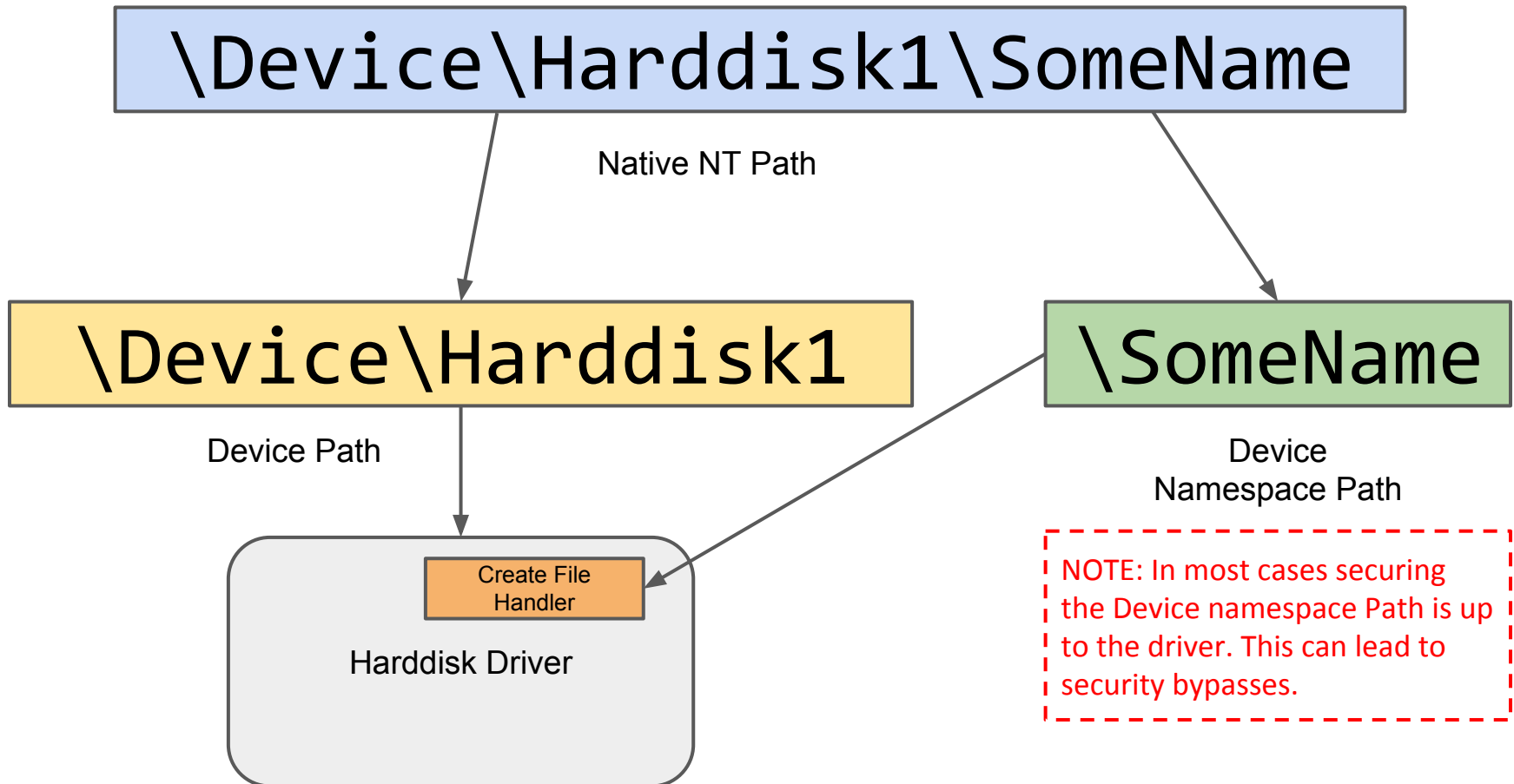
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Opening a Device Name



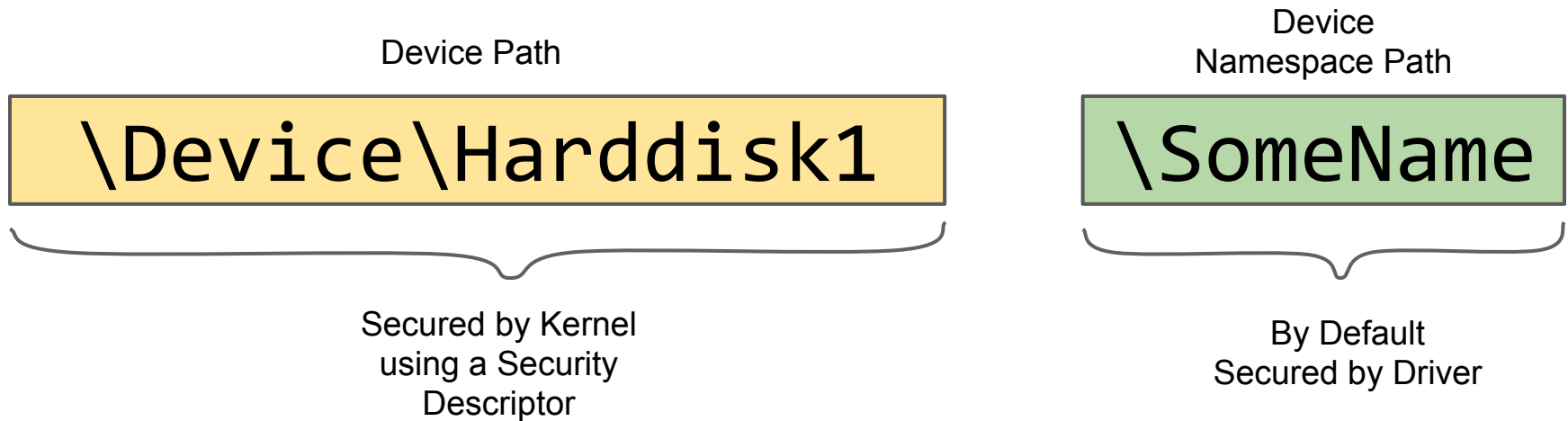
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Opening a Device Name



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Device Namespace Path



Calls to *IoCreateDevice* which specify
FILE_DEVICE_SECURE_OPEN *DeviceCharacteristics* option
secures the namespace using the device security descriptor.

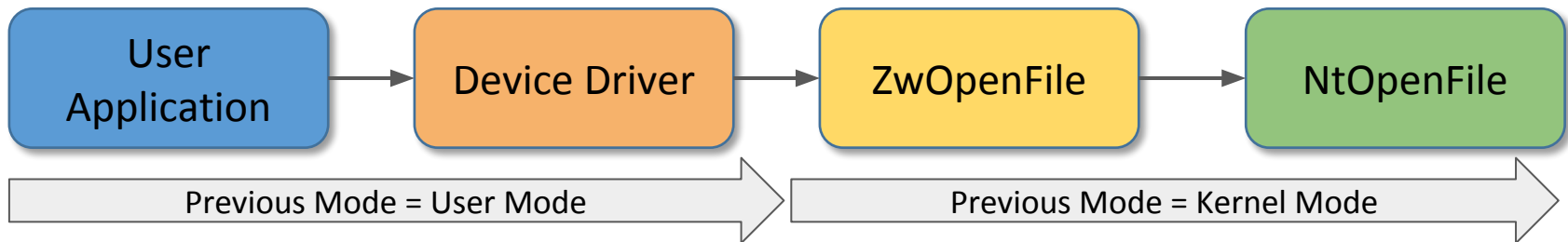
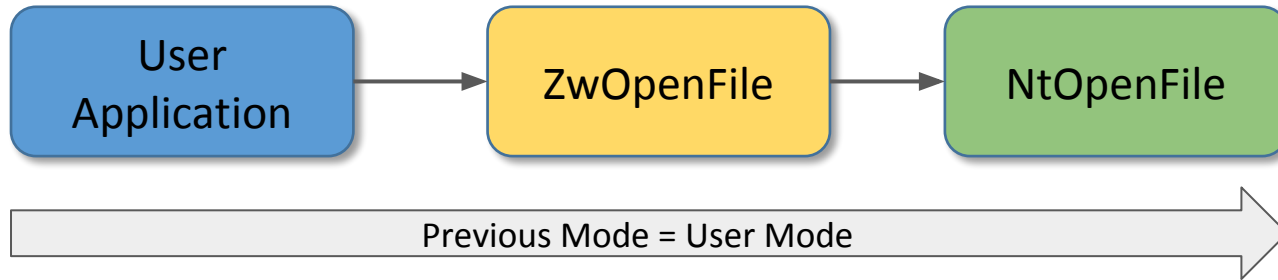
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Kernel Devices and IRPs

- Device objects are exposed in the Object Namespace
- Opened using file APIs (NtCreateFile/NtOpenFile)
- When a file is opened a special packet, an IO Request Packet (IRP) is sent to the driver's handlers
 - IRP_CREATE - Sent when the device object is opened
 - IRP_CLOSE - Sent when all device object handles are closed
 - IRP_DEVICE_CONTROL - DeviceIoControl
 - IRP_READ/WRITE - File Read and Writes


Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Previous Processor Mode



Previous Processor Mode

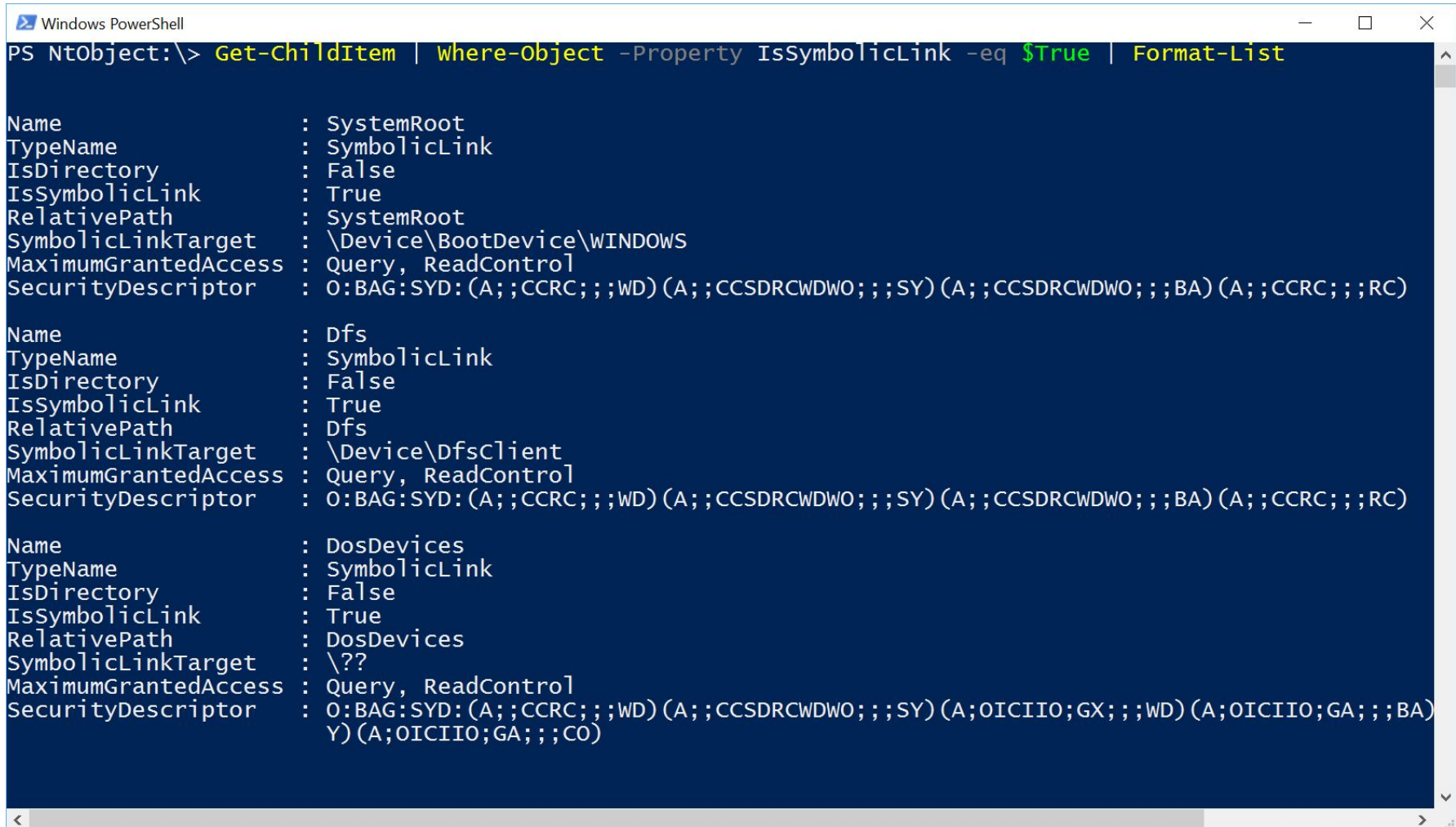
- Previous processor mode used to determine whether to enforce security!

```
BOOLEAN SeAccessCheck(  
    _In_   PSECURITY_DESCRIPTOR    SecurityDescriptor,  
    _In_   PSECURITY_SUBJECT_CONTEXT SubjectSecurityContext,  
    _In_   BOOLEAN                 SubjectContextLocked,  
    _In_   ACCESS_MASK             DesiredAccess,  
    _In_   ACCESS_MASK             PreviouslyGrantedAccess,  
    _Out_  PPRIVILEGE_SET          *Privileges,  
    _In_   PGENERIC_MAPPING        GenericMapping,  
    _In_   KPROCESSOR_MODE         AccessMode,   
    _Out_  PACCESS_MASK            GrantedAccess,  
    _Out_  PNTSTATUS               AccessStatus  
);
```

Explicit processor
mode setting.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 2: Displaying Object Namespace



```
Windows PowerShell
PS NtObject:\> Get-ChildItem | Where-Object -Property IsSymbolicLink -eq $True | Format-List

Name                : SystemRoot
TypeName            : SymbolicLink
IsDirectory          : False
IsSymbolicLink       : True
RelativePath         : SystemRoot
SymbolicLinkTarget   : \Device\BootDevice\WINDOWS
MaximumGrantedAccess : Query, ReadControl
SecurityDescriptor   : O:BAG:SYD:(A;;;CCRC;;;WD)(A;;;CCSDRCWDWO;;;SY)(A;;;CCSDRCWDWO;;;BA)(A;;;CCRC;;;RC)

Name                : Dfs
TypeName            : SymbolicLink
IsDirectory          : False
IsSymbolicLink       : True
RelativePath         : Dfs
SymbolicLinkTarget   : \Device\DfsClient
MaximumGrantedAccess : Query, ReadControl
SecurityDescriptor   : O:BAG:SYD:(A;;;CCRC;;;WD)(A;;;CCSDRCWDWO;;;SY)(A;;;CCSDRCWDWO;;;BA)(A;;;CCRC;;;RC)

Name                : DosDevices
TypeName            : SymbolicLink
IsDirectory          : False
IsSymbolicLink       : True
RelativePath         : DosDevices
SymbolicLinkTarget   : \??
MaximumGrantedAccess : Query, ReadControl
SecurityDescriptor   : O:BAG:SYD:(A;;;CCRC;;;WD)(A;;;CCSDRCWDWO;;;SY)(A;OICIIIO;GX;;;WD)(A;OICIIIO;GA;;;BA)
                    Y(A;OICIIIO;GA;;;CO)
```

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

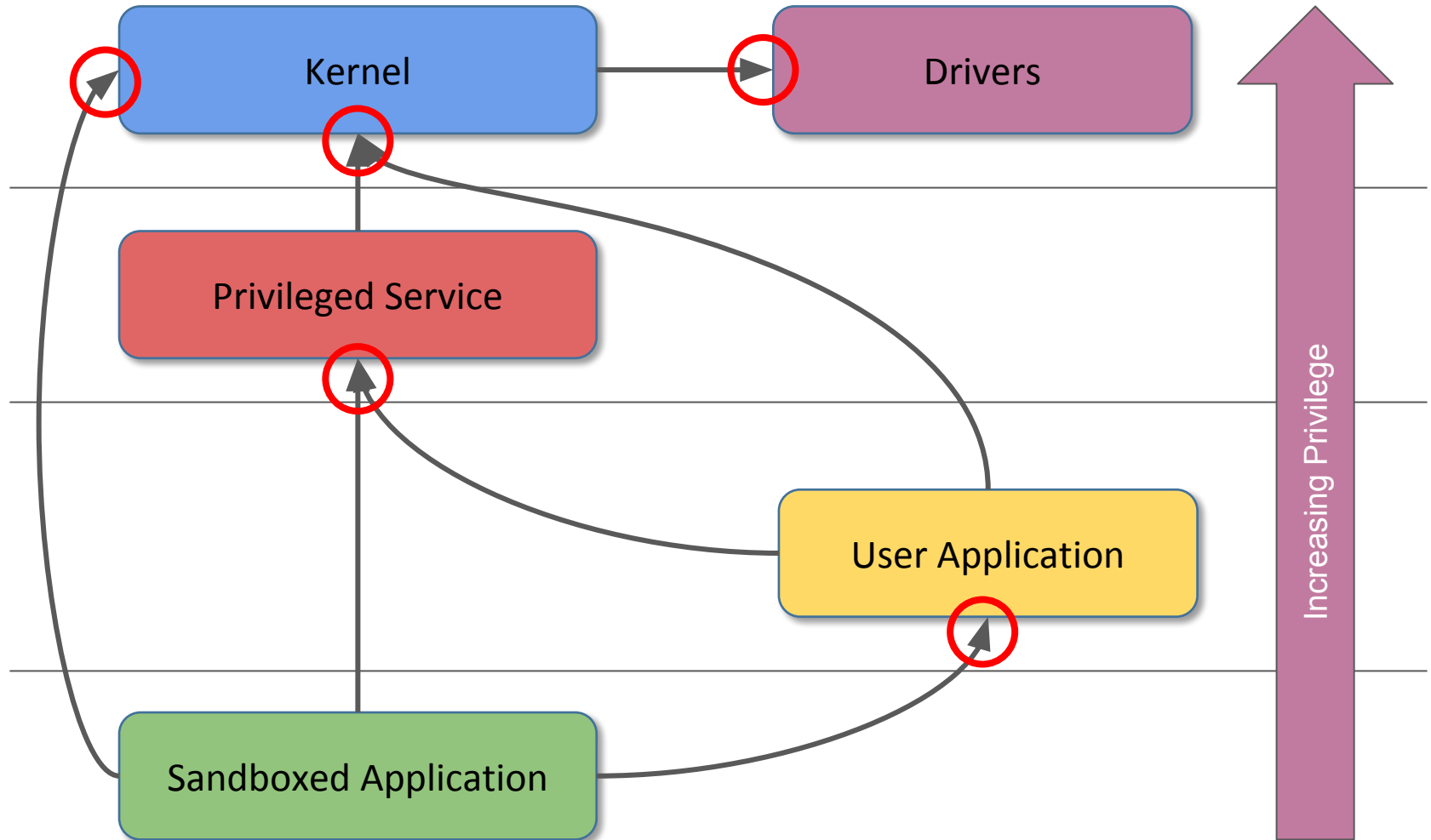
Hunting for Attack Surface

Above all Else, find something to Attack

- Might seem obvious but finding what to attack is the most important part
 - Not that different from hunting for memory corruption, just the types of things you're looking for changes.
 - Need to find interesting functionality accessible from your privilege level
- The attack surface could be Passive or Active
 - Passive means privileged code interacts with a resource the attacker can control
 - Active means the attacker can get privileged code to perform some action for them

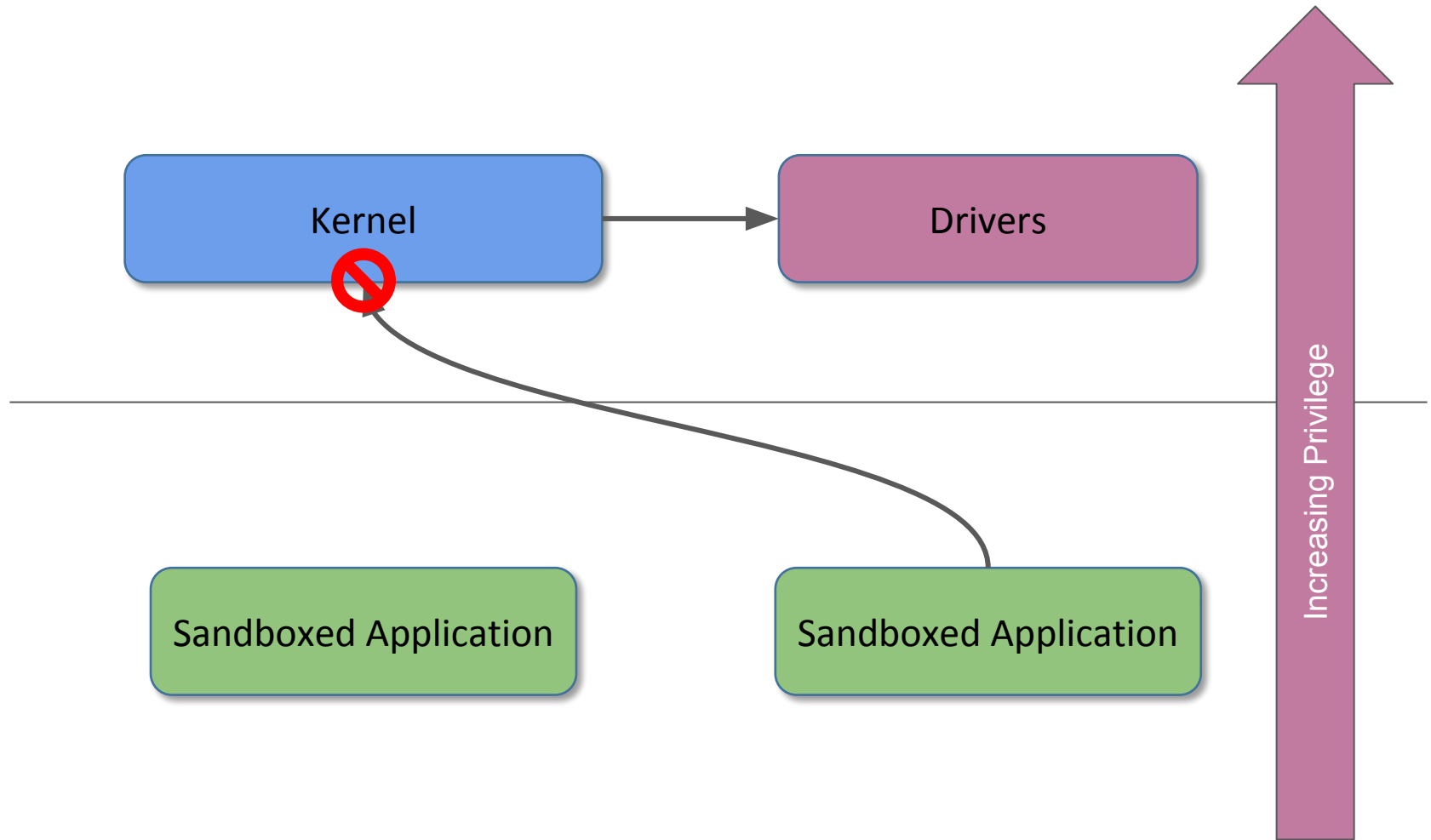
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Privilege Escalation Routes



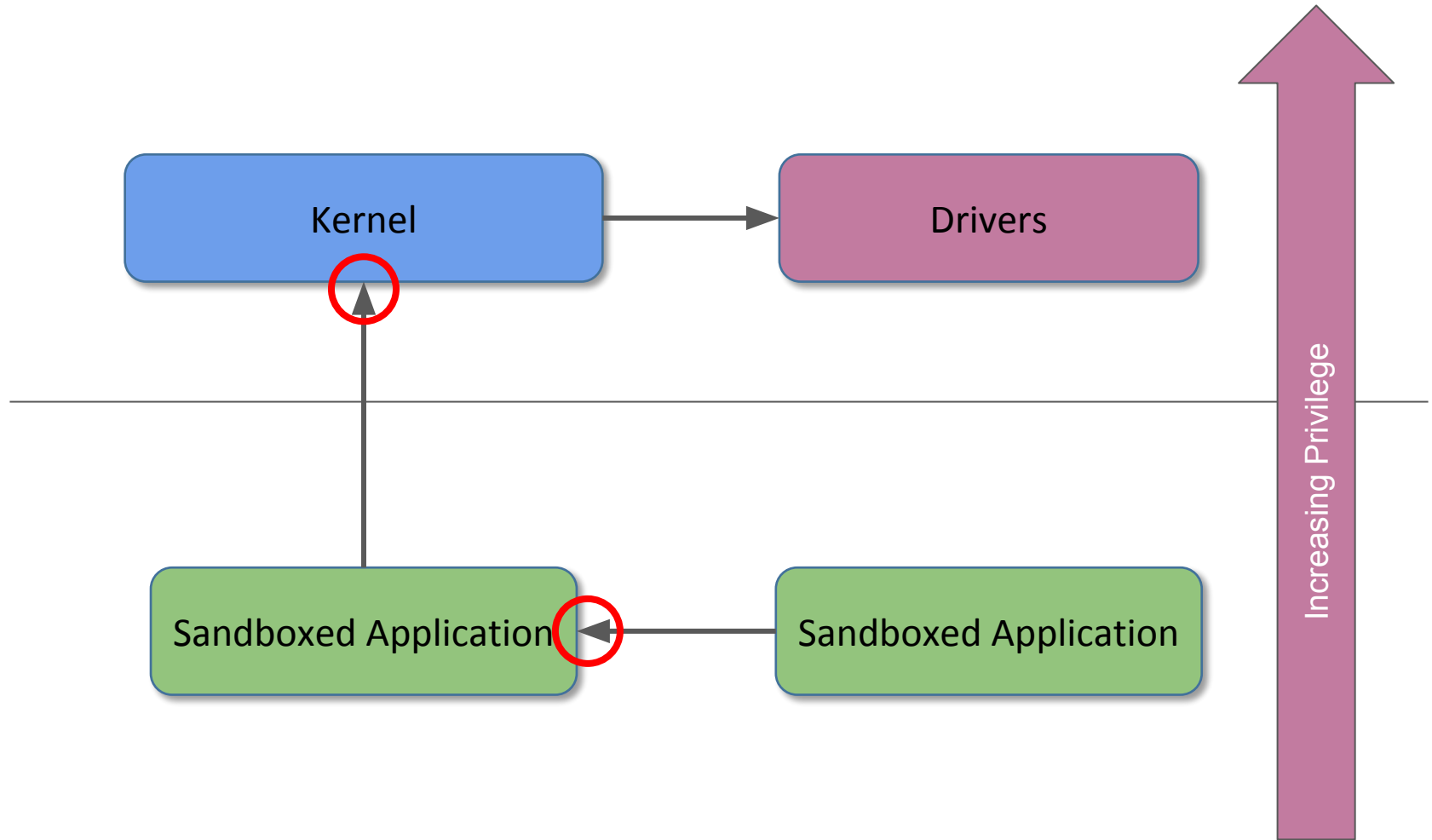
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Don't Always Think of Going Up



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Don't Always Think of Going Up



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Probing Accessible Resources

- Good idea to determine levels of attack surface is to probe what resources you can access from your desired privilege level.
- Primarily interested in WRITE, but in some cases (such as processes) READ is also important.
- This could include:
 - Files
 - Registry Keys
 - Processes and Threads
 - Sections/File Mappings
 - Kernel Driver Device Objects
 - Named Pipes

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Passive Resource Locations

- There's a number of file locations on a default Windows system that a non-administrator can control which could be used by privileged code
 - Subfolders under %SYSTEMDRIVE%\ProgramData
 - %WINDIR%\Temp
- Also some locations inside the Local Machine registry
- General kernel resources could also end up inside the Object Manager namespace
- Find privileged users of this functionality by using something like Process Monitor

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Sandbox Attack Surface Analysis Tools

<i>Tool Name</i>	<i>Description</i>
CheckFileAccess	Enumerate accessible files or named pipes
CheckProcessAccess	Enumerate accessible processes and/or threads
CheckDeviceAccess	Enumerate accessible device objects
CheckRegistryAccess	Enumerate accessible registry keys
CheckObjectManagerAccess	Enumerate accessible names kernel resources (such as Sections/Mutexes/Events etc.)

Tools take a number of common command line arguments:

- p PID : Specify a process to impersonate when doing the access check
- w : Only show writable resources
- r : Recursively enumerate names resources
- k ACCESS: Comma separated list of access rights to check for
- q : Suppress printing errors during enumeration

Services

- Services are also a securable resource
- Typically look for write privileges to change configuration
 - Everyone should already know about this.
- Instead look for start privileges
 - Increase potential attack surface
 - Some services take arguments during start such as the Mozilla Maintenance

Service

```
ServiceController svc = new  
    ServiceController ("blah");  
// Start a service with arbitrary arguments.  
svc.Start(new string[] { "Arg1", "Arg2" });
```

- “CheckServiceAccess -k=Start” is your friend to find what you can start.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Service Triggers

- Some services can be started without an explicit Start privilege
- Windows 7 introduced Service Triggers, starts/stops services on certain events:
 - Access to Named Pipe or RPC Endpoints
 - Creation of Firewall Access Rules
 - Joining of a Domain
 - Custom Event Tracing for Windows event
 - Adding a Hardware Device
- ETW is one of the most common and easiest to execute

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

ETW Event Trigger

- Use the `-t` switch to display service triggers
- For example the WebClient service would print:

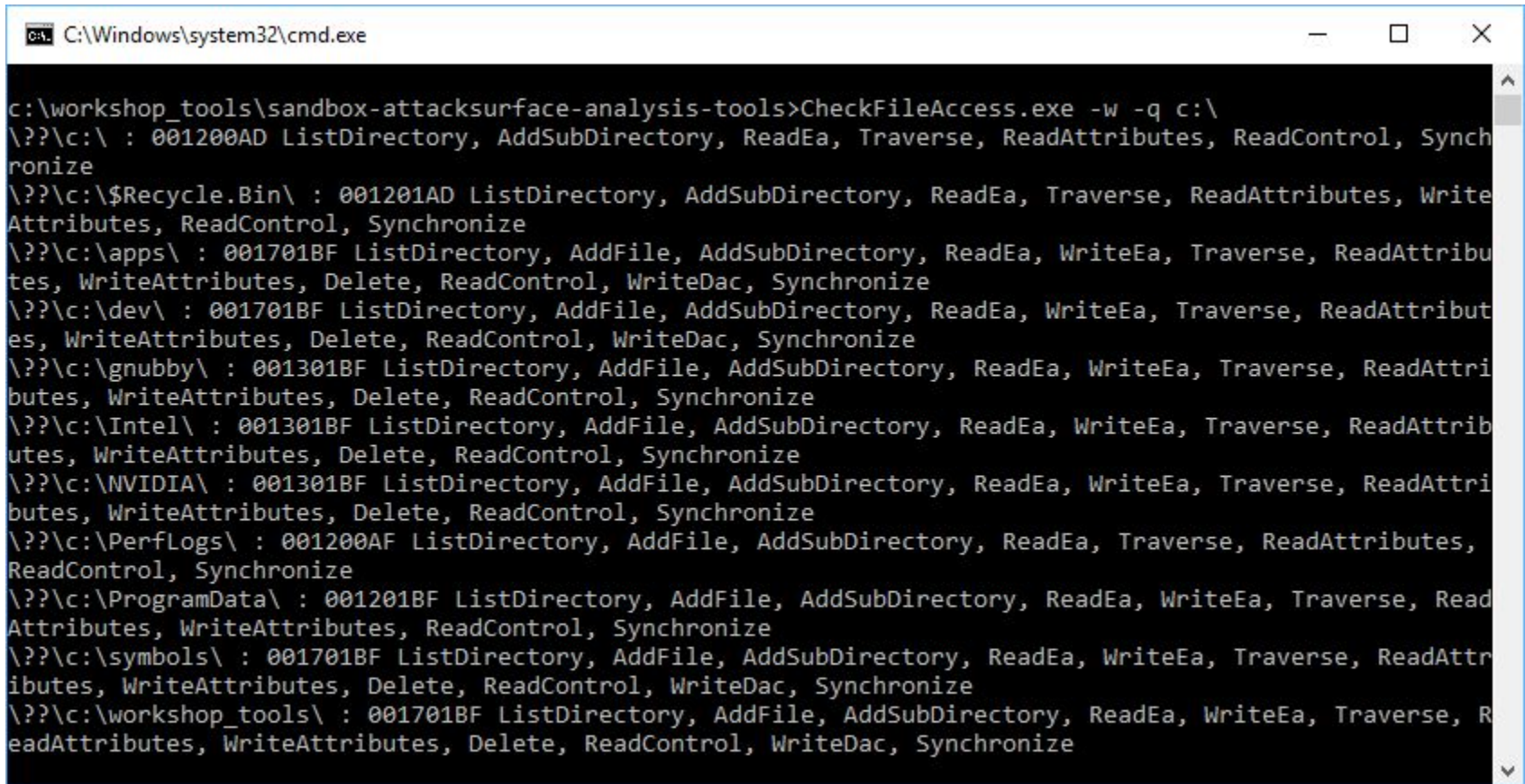
```
WebClient Granted Access: QueryConfig, QueryStatus, EnumerateDependents, Interrogate,  
UserDefinedControl, ReadControl  
Trigger: 0 - Type: Custom - Action: Start  
Subtype: [ETW UUID] {22b6d684-fa63-4578-87c9-effcbe6643c7}
```

- Use the following C++ code to start the WebClient service

```
const GUID _MS_Windows_WebClntLookupServiceTrigger_Provider =  
    { 0x22B6D684, 0xFA63, 0x4578,  
      { 0x87, 0xC9, 0xEF, 0xFC, 0xBE, 0x66, 0x43, 0xC7 } };  
REGHANDLE Handle;  
  
if (EventRegister(  
    &_MS_Windows_WebClntLookupServiceTrigger_Provider,  
    nullptr, nullptr, &Handle) == ERROR_SUCCESS) {  
    EVENT_DESCRIPTOR desc;  
    EventDescCreate(&desc, 1, 0, 0, 4, 0, 0, 0);  
    EventWrite(Handle, &desc, 0, nullptr) == ERROR_SUCCESS;  
    EventUnregister(Handle);  
}
```

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 3 - Enumerating Accessible Resources



```
C:\Windows\system32\cmd.exe

c:\workshop_tools\sandbox-attacksurface-analysis-tools>CheckFileAccess.exe -w -q c:\
\??\c:\ : 001200AD ListDirectory, AddSubDirectory, ReadEa, Traverse, ReadAttributes, ReadControl, Synchroni
ronize
\??\c:\$Recycle.Bin\ : 001201AD ListDirectory, AddSubDirectory, ReadEa, Traverse, ReadAttributes, Write
Attributes, ReadControl, Synchronize
\??\c:\apps\ : 001701BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttribu
tes, WriteAttributes, Delete, ReadControl, WriteDac, Synchronize
\??\c:\dev\ : 001701BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttribu
es, WriteAttributes, Delete, ReadControl, WriteDac, Synchronize
\??\c:\gnubby\ : 001301BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttri
butes, WriteAttributes, Delete, ReadControl, Synchronize
\??\c:\Intel\ : 001301BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttrib
utes, WriteAttributes, Delete, ReadControl, Synchronize
\??\c:\NVIDIA\ : 001301BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttri
butes, WriteAttributes, Delete, ReadControl, Synchronize
\??\c:\PerfLogs\ : 001200AF ListDirectory, AddFile, AddSubDirectory, ReadEa, Traverse, ReadAttributes,
ReadControl, Synchronize
\??\c:\ProgramData\ : 001201BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, Read
Attributes, WriteAttributes, ReadControl, Synchronize
\??\c:\symbols\ : 001701BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, ReadAttr
ibutes, WriteAttributes, Delete, ReadControl, WriteDac, Synchronize
\??\c:\workshop_tools\ : 001701BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse, R
eadAttributes, WriteAttributes, Delete, ReadControl, WriteDac, Synchronize
```

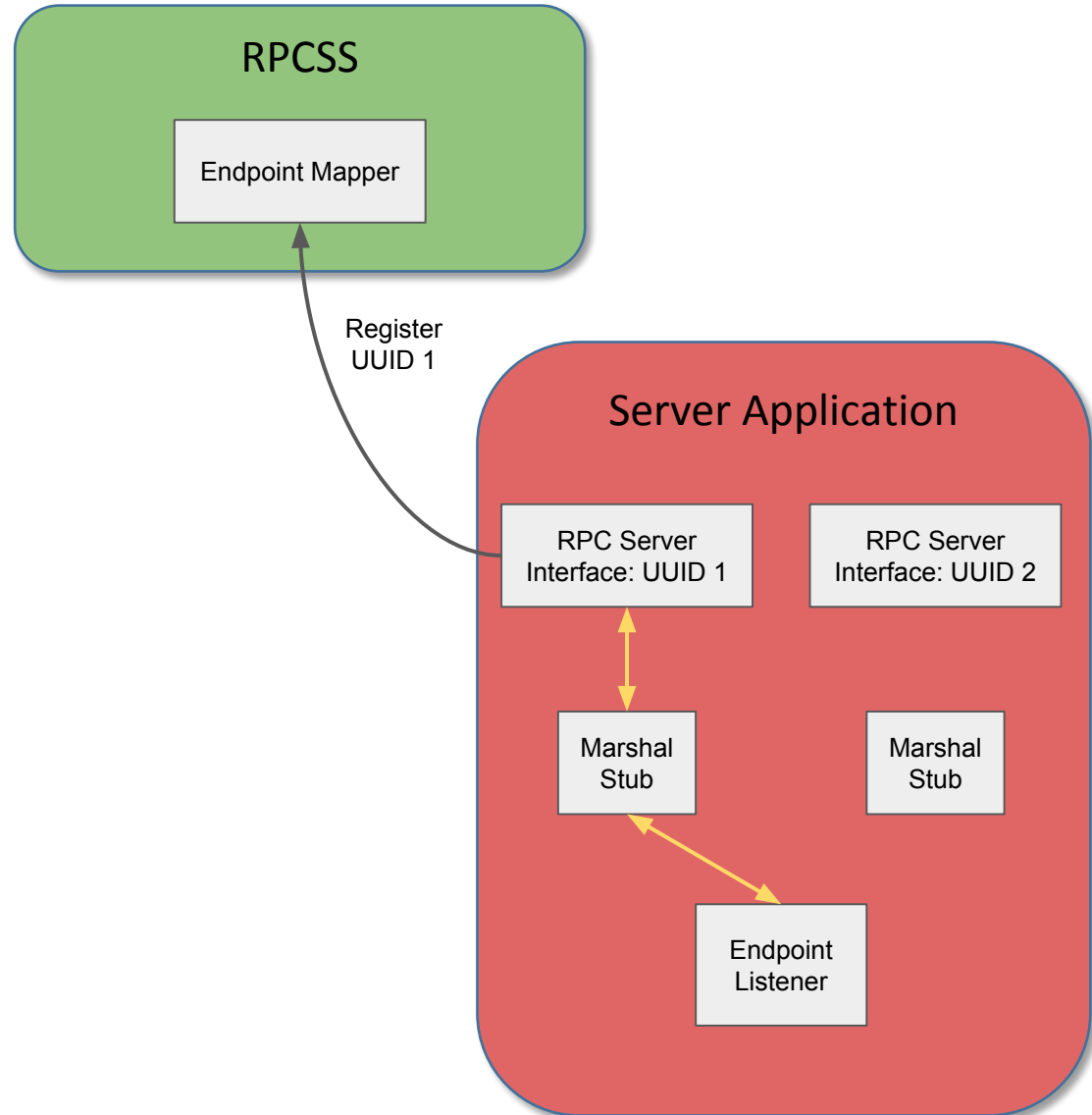
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

RPC Services

- The most common technique on Windows to provide privilege separation between components.
- Used in many common services:
 - Local Security Subsystem (LSASS)
 - AppInfo service (UAC)
 - Secondary Logon service (seclogon)
- Many RPC services are undocumented and contain complex functionality

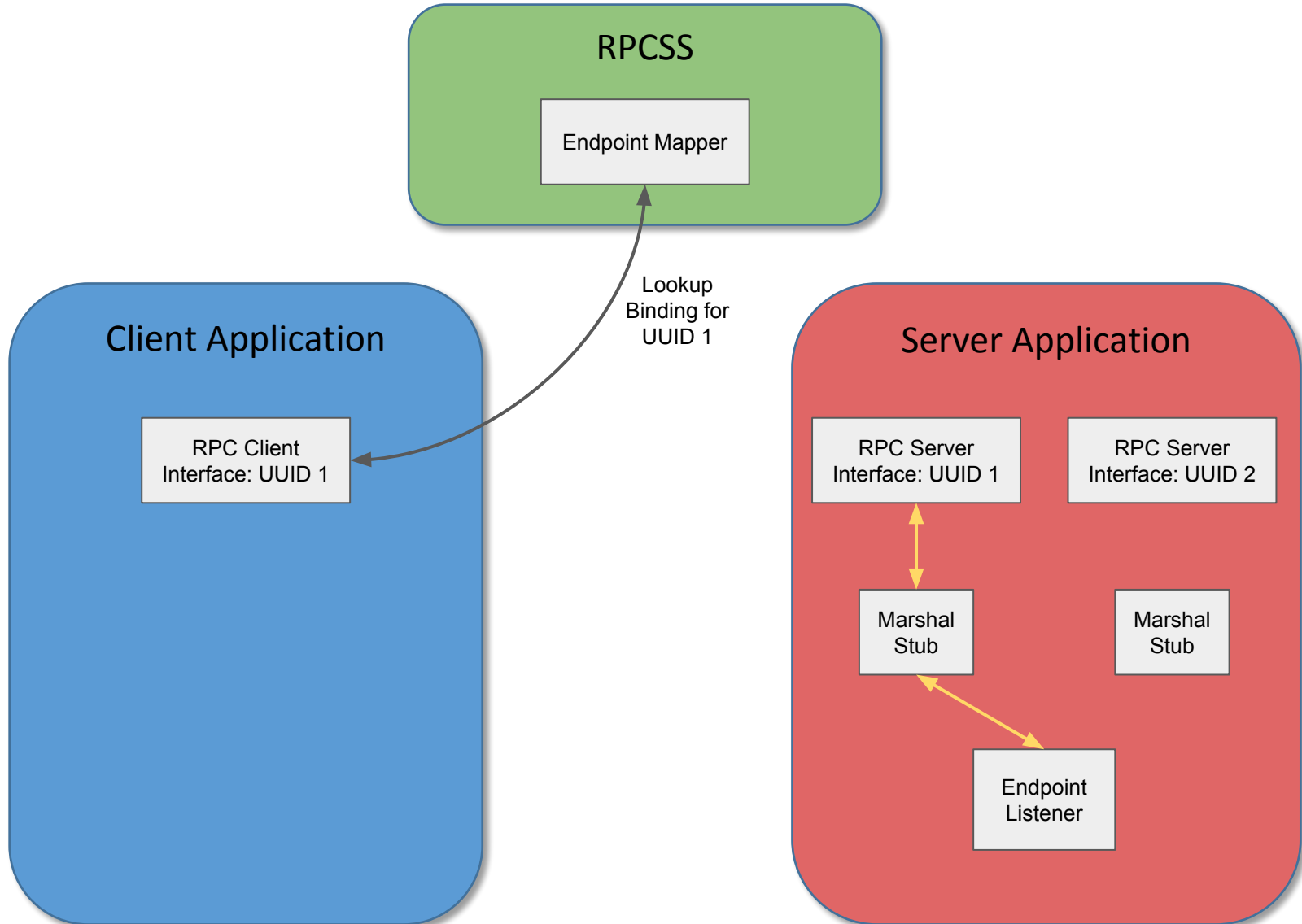
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows RPC Architecture



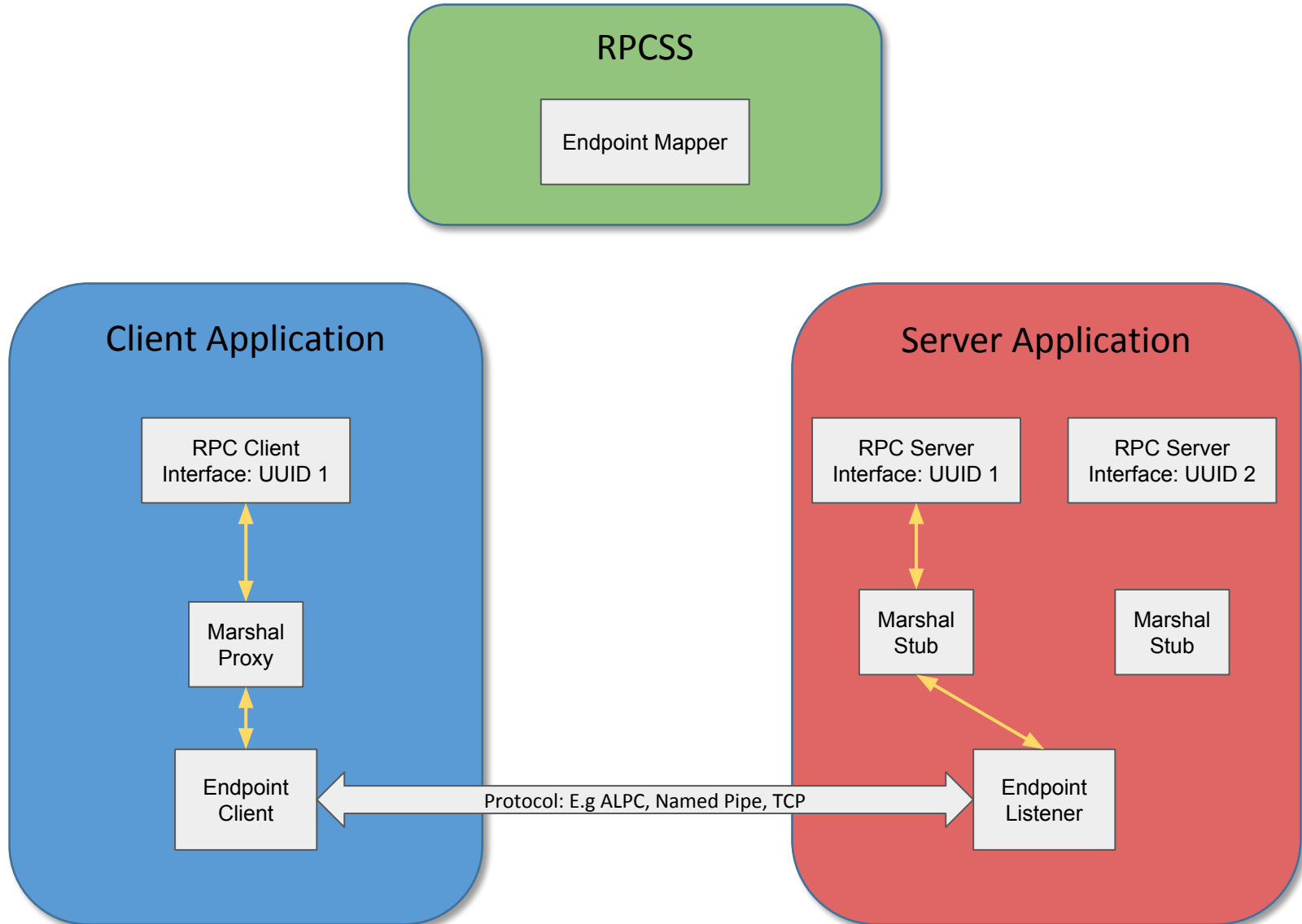
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows RPC Architecture



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows RPC Architecture



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Network Data Representation (NDR)

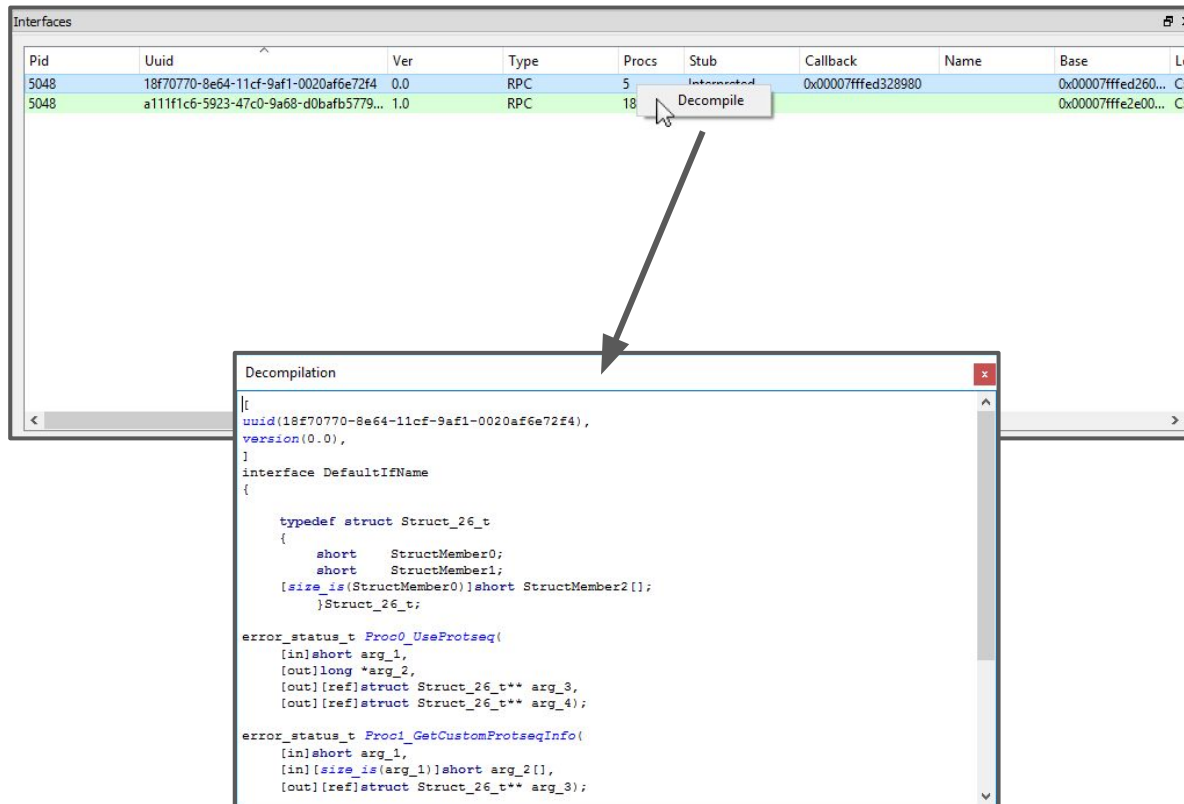
- Server defined interface using an IDL file. Compiler converts to a server Stub build with NDR which handles marshaling of parameters and structures
- Client must have a corresponding Proxy built from the same IDL interface definition otherwise there's likely to be a mismatch.
- Each interface has a defined unique ID (UUID)

```
[  
    uuid (201ef99a-7fa0-444c-9399-19ba84f12a1a),  
    version(1.0),  
]  
interface LaunchAdminProcess  
{  
    long RAiLaunchAdminProcess([in][unique][string] wchar_t* ExecutablePath);  
}
```

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Working with RPC Interfaces

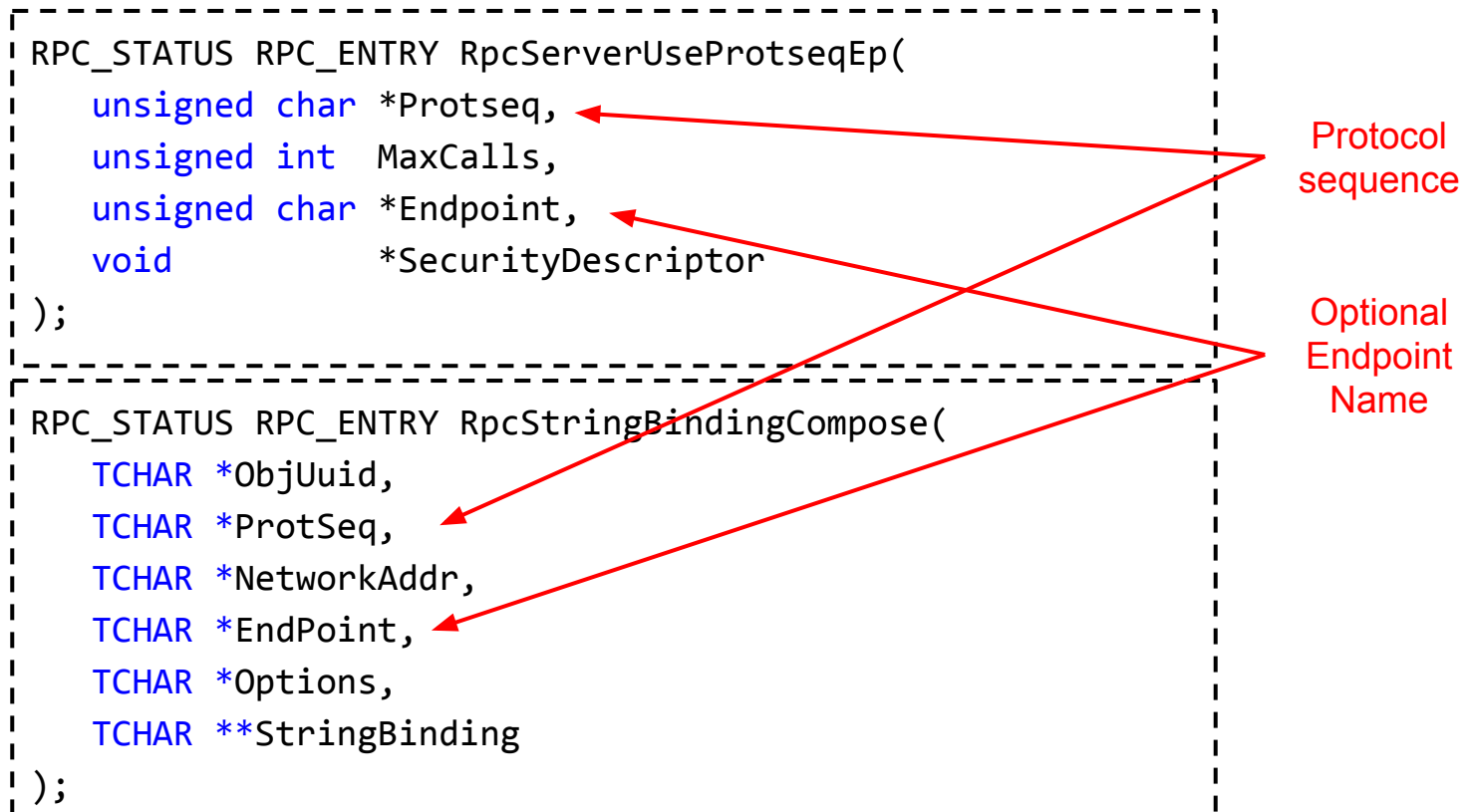
- RPCView Supports basic Decompile of interface definitions.
- Right click interface and choose Decompile



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

RPC Endpoints

- RPC Supports multiple different endpoint protocols.
- Configured on server using `RpcServerUseProtseqEp`
- Configured on client using `RpcStringBindingCompose`



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Protocol Sequences

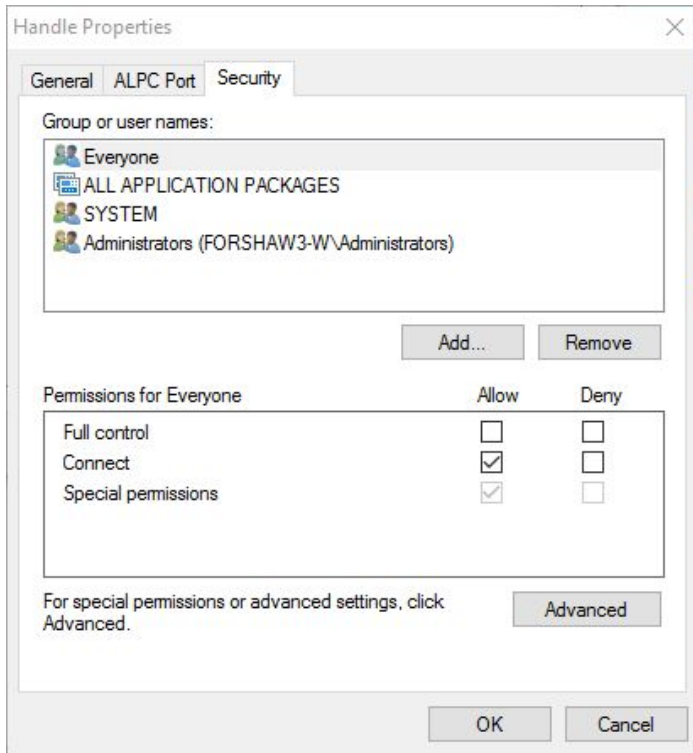
<i>Protocol Sequence</i>	<i>Optional Endpoint Name</i>	<i>Description</i>
ncalrpc	NAME	Local RPC (ALPC)
ncacn_np	\pipe\NAME	Windows Named Pipe
ncacn_ip_tcp	(port number)	TCP/IP
ncacn_ip_udp	(port number)	UDP/IP
ncacn_http	(port number)	HTTP

Note all endpoints and protocol sequences are multiplexed in a single process.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

RPC Security

Connect Time Security (Local)



As all endpoints of multiplexed, you can pick the one with the lowest connect time security

Runtime Security

```
RPC_STATUS RPC_ENTRY RpcServerRegisterIf3(
    _In_      RPC_IF_HANDLE      IfSpec,
    _In_opt_  UUID                *MgrTypeUuid,
    _In_opt_  RPC_MGR_EPV        *MgrEpv,
    _In_      unsigned int       Flags,
    _In_      unsigned int       MaxCalls,
    _In_      unsigned int       MaxRpcSize,
    _In_opt_  RPC_IF_CALLBACK_FN *IfCallbackFn,
    _In_opt_  void               *SecurityDescriptor
);
```


Security callback, run code to verify the client

Static Security Descriptor

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Configuring RPCView Symbols

- Getting Symbol Information gives you a better idea on what functions are exported.
- Configure local path to symbols through Options -> Configure Symbols



Index	Name	Address	Format
0		0x00007ffed3c5fb0	0x00007ffed43e452
1		0x00007ffed3c5cb0	0x00007ffed43e48e
2		0x00007ffed330b60	0x00007ffed43e4c4
3		0x00007ffed3c5f90	0x00007ffed43e500
4		0x00007ffed370eb0	0x00007ffed43e52a

Index	Name	Address	Format
0	UseProtseq	0x00007ffed3c5fb0	0x00007ffed43e452
1	GetCustomProtseqInfo	0x00007ffed3c5cb0	0x00007ffed43e48e
2	UpdateResolverBindings	0x00007ffed330b60	0x00007ffed43e4c4
3	NotifyFDT	0x00007ffed3c5f90	0x00007ffed43e500
4	ControlTracing	0x00007ffed370eb0	0x00007ffed43e52a

RPCView doesn't seem to work with symbol servers. So need to pull symbols manually. Use symchk from Debugging Tools for Windows, cache in a known directory such as c:\symbols.

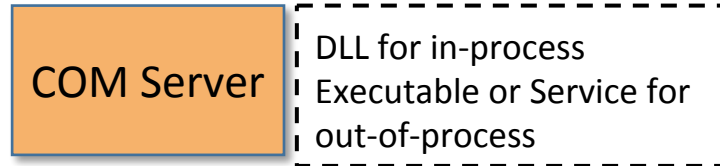
`symchk /s srv*c:\symbols*https://msdl.microsoft.com/download/symbols c:\windows\system32*.dll`

Local COM Services

- COM itself is just an ABI definition to allow multiple programming languages to create and call objects
- Supported out-of-process objects
- Effectively an extension of RPC services.
- Key difference is supports Activation, creation of services from a registry, they don't have to already be running like RPC
- Much bigger attack surface than normal RPC
- Activation brings some fun:
 - RunAs/Different User activation
 - Service starting and object hosting
 - UAC auto-elevation.

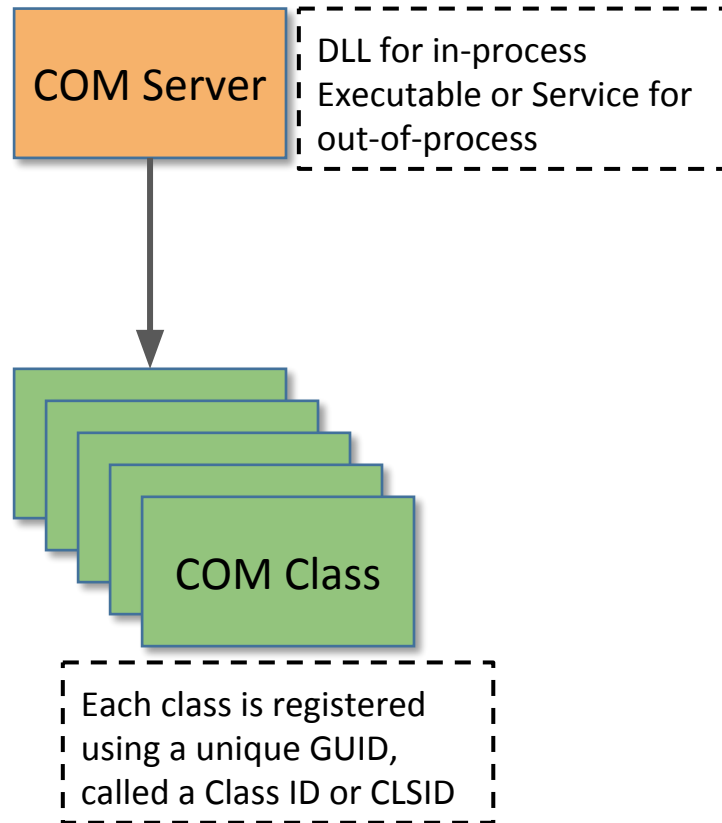
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

COM Technical Architecture



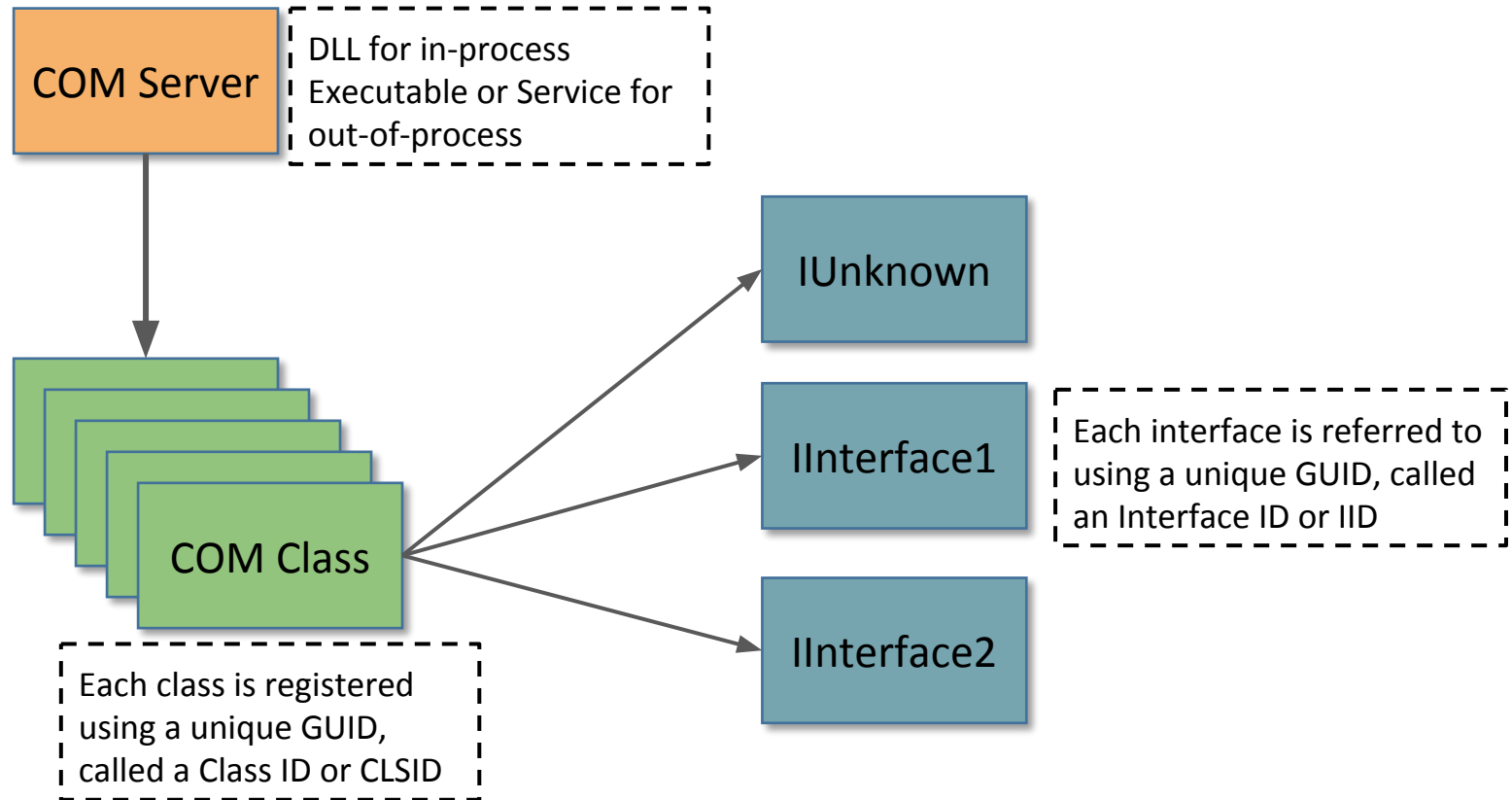
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

COM Technical Architecture



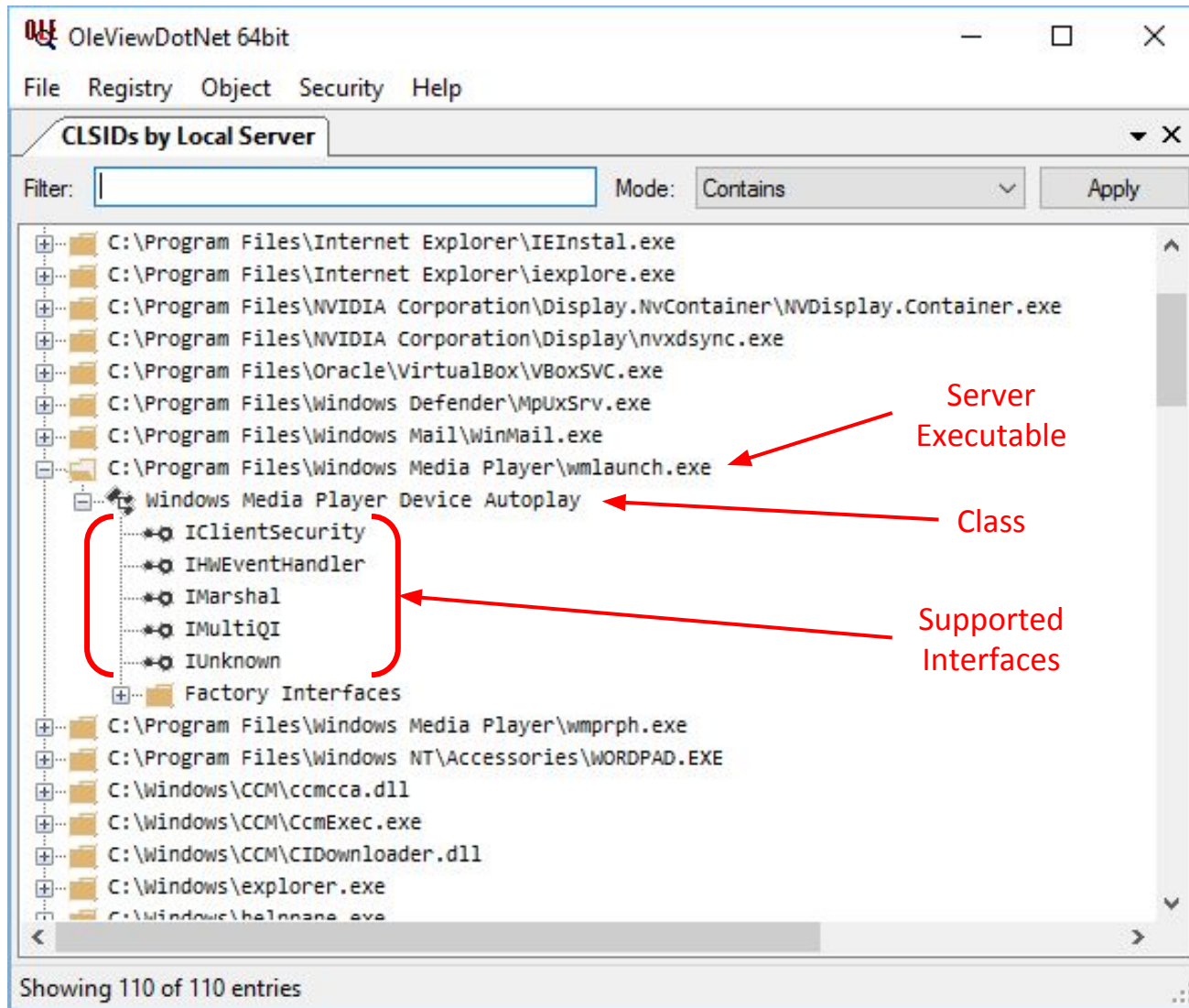
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Technical Architecture



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Technical Architecture

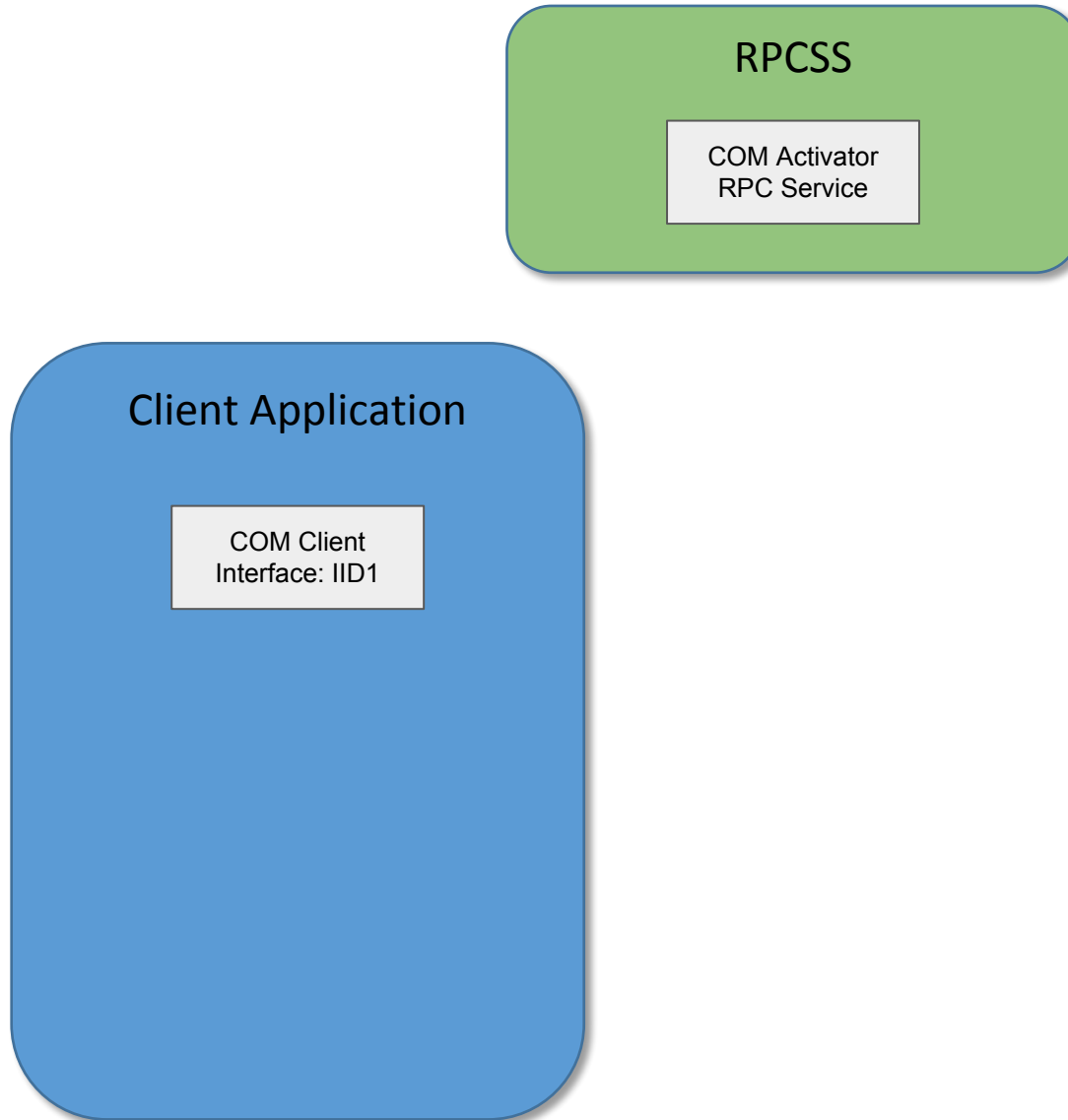


IUnknown, The Root of all COM Evil

- All COM interfaces derive from a special interface, *IUnknown*
- Has the IID `{00000000-0000-0000-C000-000000000046}`
- Supports 3 methods:
 - `QueryInterface` - Used to query for other supported interfaces
 - `AddRef` - Increment object reference count
 - `Release` - Decrement object reference count
- If an object doesn't support an interface *QueryInterface* should return `E_NOINTERFACE (0x80004002)`
- Code should not be casting objects to other interfaces without going through *QueryInterface*. That would be a type confusion bug in waiting :-)

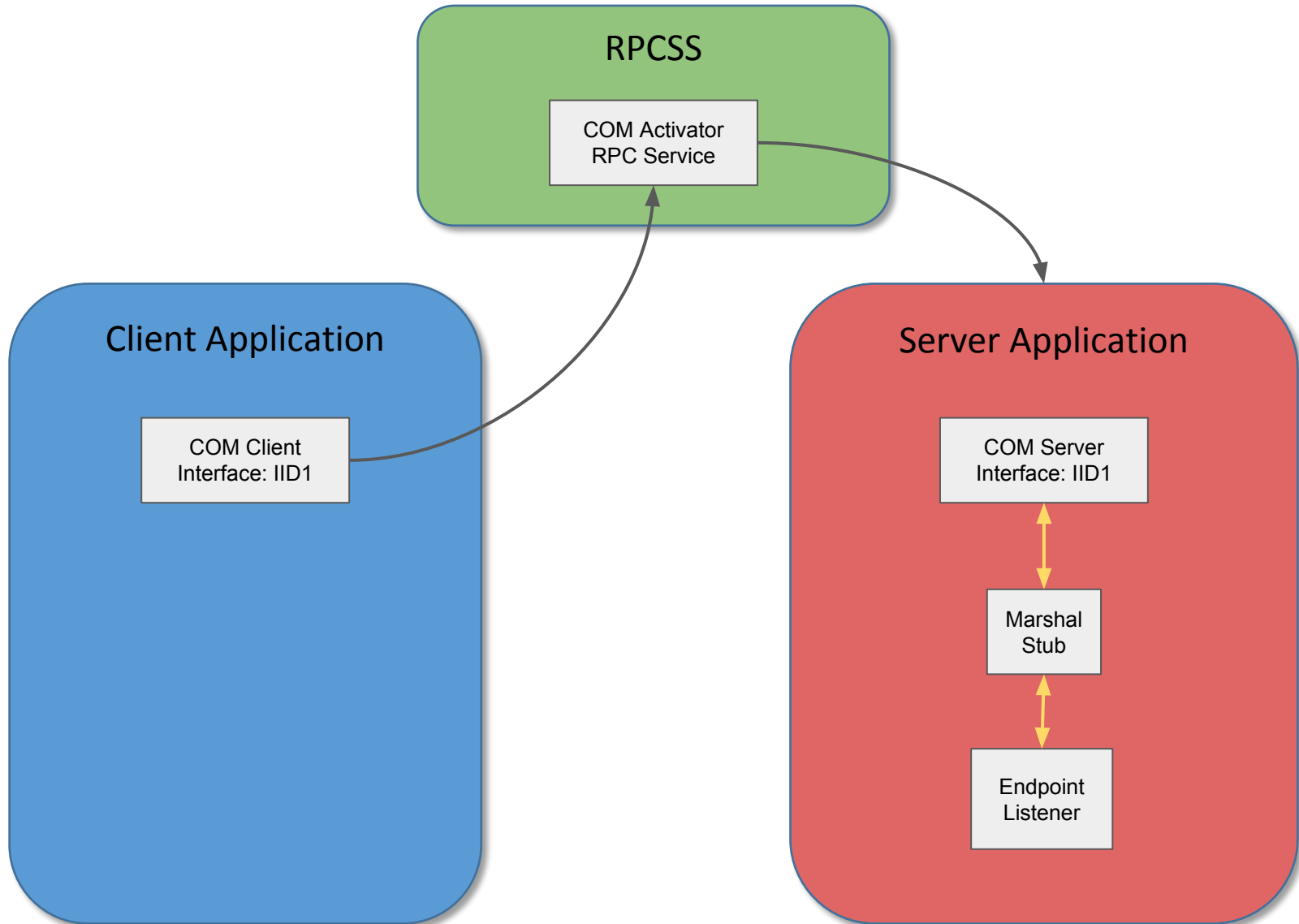
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

COM Activation



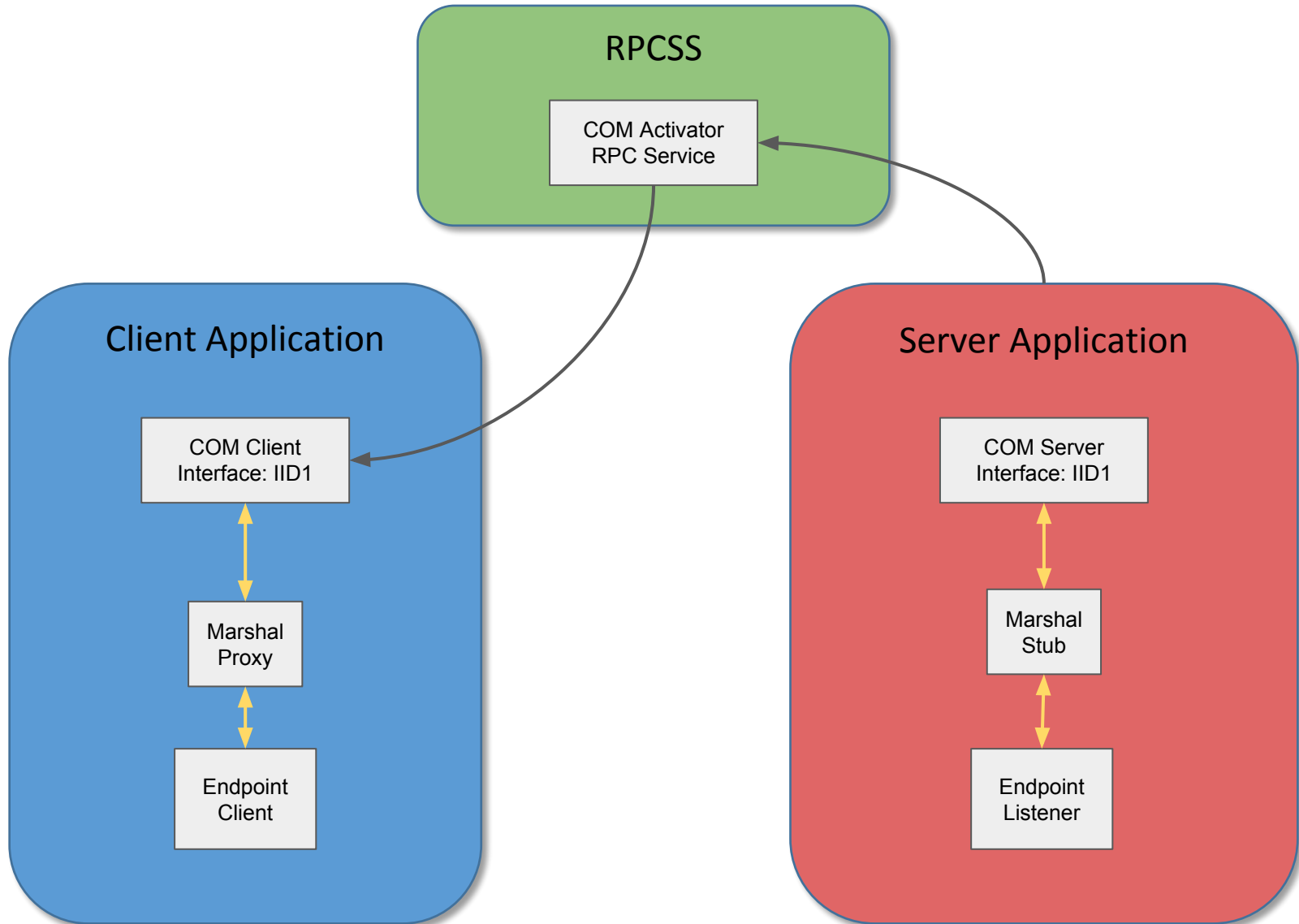
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Activation



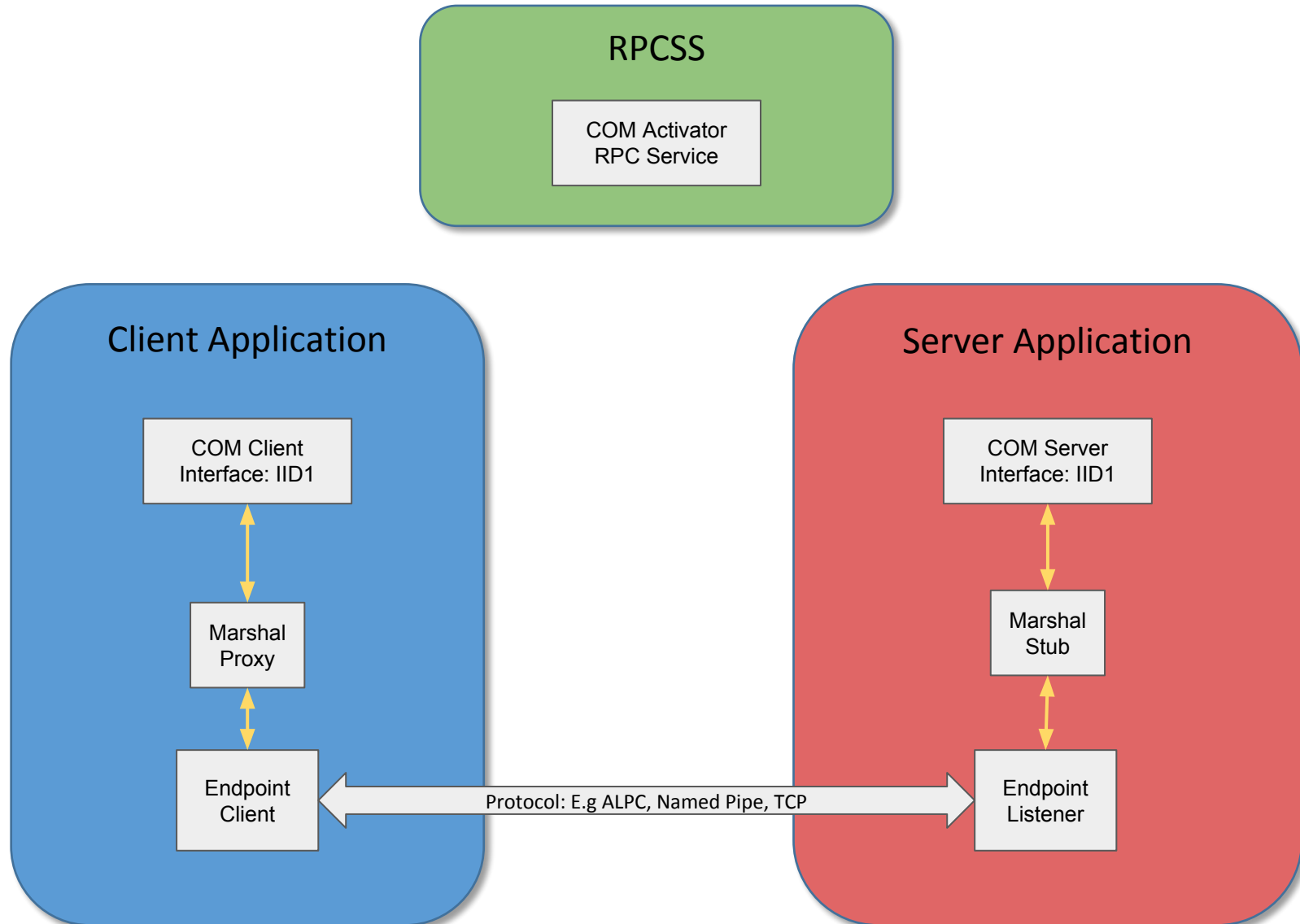
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Activation



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Activation



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Proxies and Stubs

- Like RPC, COM services must define their interface using an IDL file. This must be registered.

OleViewDotNet 64bit

File Registry Object Security Help

PSFactoryBuffer Properties

CLSID Supported Interfaces Proxies

Name: PSFactoryBuffer

CLSID: C90250F3-4D7D-4991-9B69-A5C5BC1C2AE6 Create

Server Type: InProcServer32

Server: C:\Windows\System32\ActXPrxy.dll

CmdLine: N/A

TreatAs: N/A Properties

Threading Model: Both

ProgIDs:

Categories:

PSFactoryBuffer Properties

CLSID Supported Interfaces Proxies

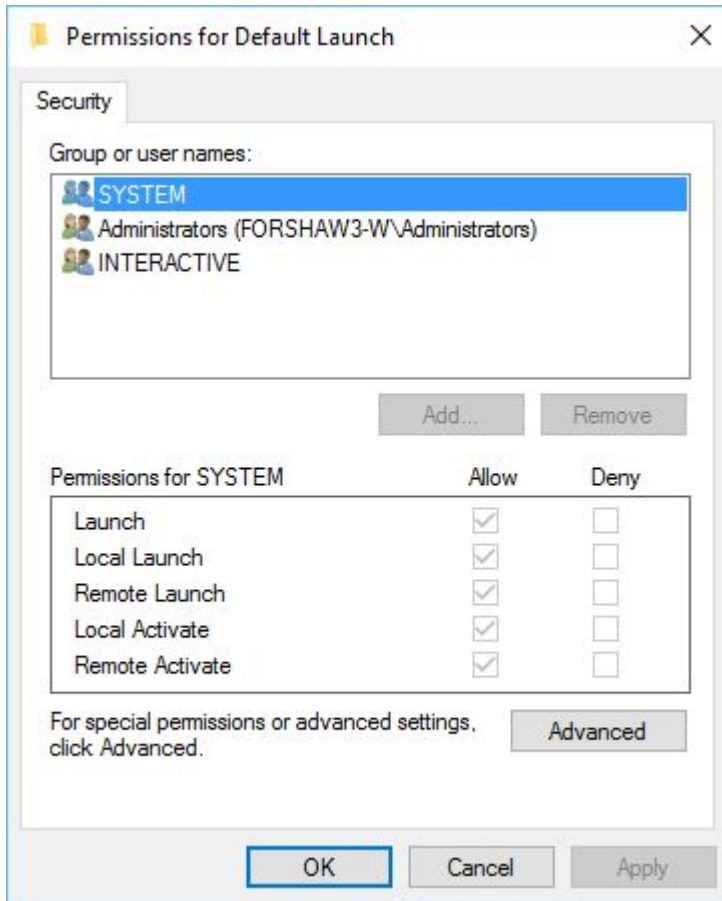
Name	IID
Application Theme::CIAppThemeApi...	C5F80E59-A9FC-439D-9FC4-D290858
IAccessibilityDockingService	8849DC22-CEDF-4C95-998D-051419C
IAccessibilityDockingServiceCallback	157733FD-A592-42E5-B594-248468C!
IAccessibilityDockingServiceInternal	6D2586FD-5559-4747-9A8F-28031D2
IAccessibleObject	95A391C5-9ED4-4C28-8401-AB9E067
IActionProgress	49FF1173-EADC-446D-9285-156453A
IActionProgressDialog	49FF1172-EADC-446D-9285-156453A
IActionSpaceFlow	00D0F427-EBB3-4304-B6F8-AE4FA0F
IActivationErrorPopup	9333BEE8-82CB-475A-8801-E43F128
IActivationErrorPopupFactory	848EAF0A-4435-4D6F-BCDF-8F42FFE
IActivationLayoutPreferences	657A8842-0B5E-40E1-B8CB-9AAFAC
IActivationStoreHelper	A1DFFAF7-0F96-44C0-A97E-5072919
IActivationViewSwitcherFactory	A6B0E1A3-3104-4034-9BAA-52AB1F8
IActiveZBandNotification	0C40987C-7026-499E-ADAD-C164C0F
IActiveZBandNotificationForMonitor	78C8B458-DD5A-4F32-9D8B-D45A19.
IAddressBand	22C440DF-4720-4B3A-A472-0CCB6E6
IAddressBandEx	A695888C-E3FE-47C2-82C0-CCD589C
IAddressEditBox	4ACDA08F-21CF-45AE-A5D5-75CB63
IAdvancedSettingsWriter	5D703587-DA7E-43C0-982C-2BA8E7F
IAdviseOplockCallback	68766F36-3808-42F9-A18F-0ABDE63!
IAggregatedInterruptResponse	21B98AA2-DBD8-4E6B-821C-AC9C01

DLL containing implementation

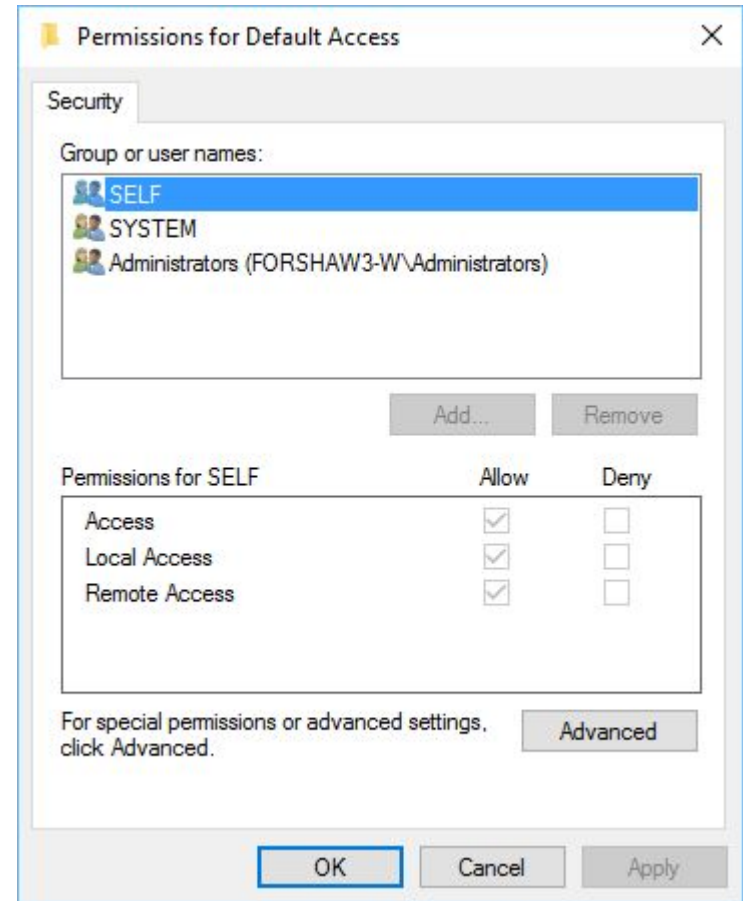
Supported Proxies/Stubs

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

COM Security



Launch = Create a new instance of the server.
Activate = Create new object on existing server.
Enforced in RPCSS



Access = Call methods on existing objects.
Enforced in Server Process

AppIDs and RunAs User

- A COM class be be registered within an AppID (again a GUID)
- The AppID can change the behaviour of the activated COM class
 - Specify non-default Launch or Access permissions
 - Specify a Surrogate Executable to host DLL classes out of the process
 - Specify a Windows Service which will host the class
 - Specify a specific “RunAs” user to run the server under
- RunAs is typically specified as “Interactive User”
 - This means as the current session’s default user, not the caller
 - If a server is registered as “Interactive User” and the caller has permission to Launch/Activate then a sandboxed user can interact with it to try and escape the sandbox

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

User Account Control

- Default Windows user is a “split-token” administrator
- Means that normally user is limited in privileges but can become a full administrator on demand
- Implemented in the AppInfo service
- Plenty of ways of bypassing this, but not normally without prompts
- Also supports special UI Access processes due to block on sending window messages to higher IL procesesses
 - UI Access bypasses the checks

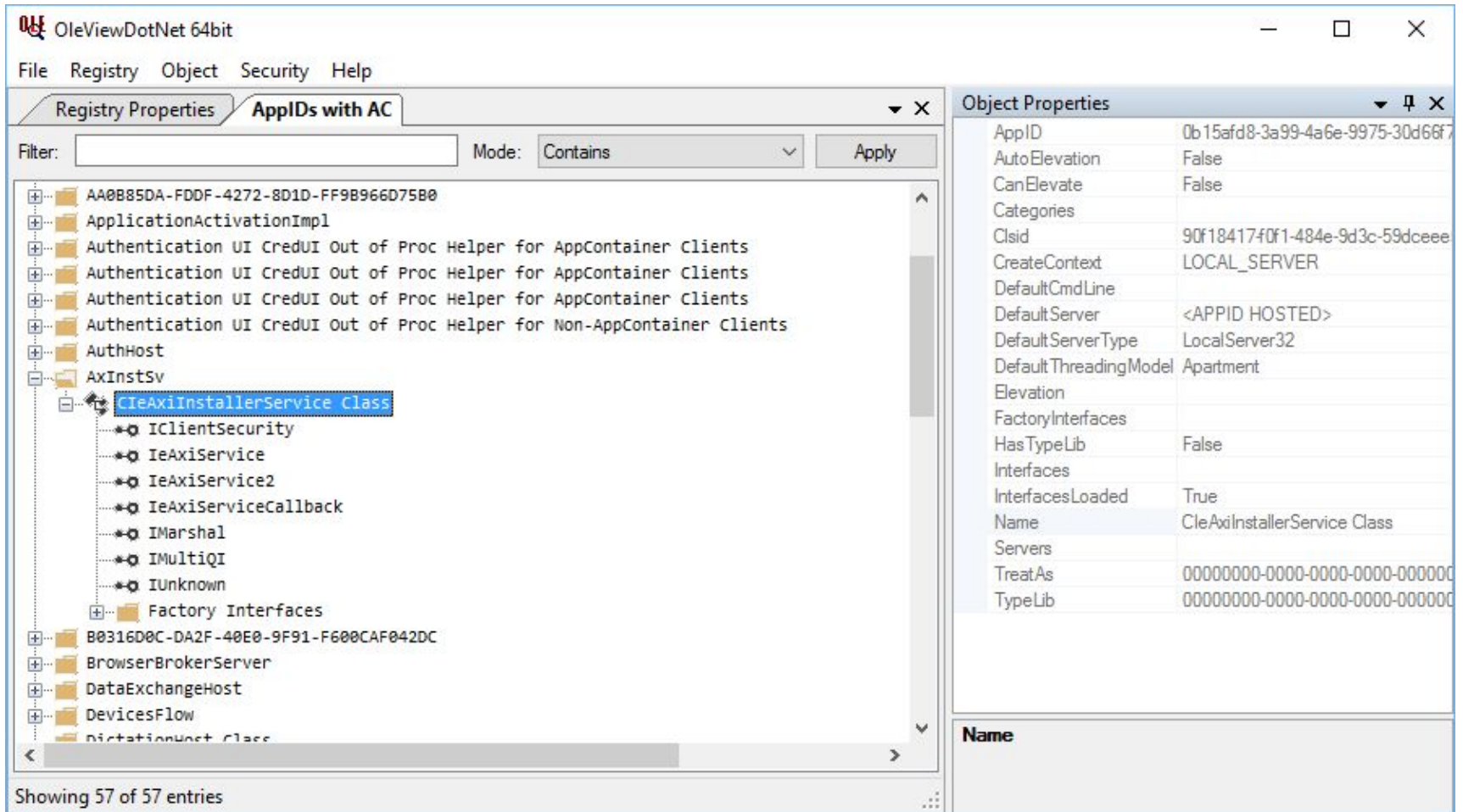
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

UAC Auto-Elevation

- UAC isn't a security boundary, but still most users still run as administrators
- Since UAC was introduced supports auto-elevation for special UI access binaries
 - Signed by a known certificate and in a "secure" location
 - Has the UIAccess element in its manifest
- Since Windows 7, UAC supports auto-elevation for executables if they meet the following criteria
 - Signed by a Microsoft and in a "secure" location
 - Has an autoElevate element in its manifest
- Also supports auto-elevation of out-of-process COM objects

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 4: Inspecting Accessible COM Services



OleViewDotNet 64bit

File Registry Object Security Help

Registry Properties AppIDs with AC

Filter: Mode: Contains Apply

AA0B85DA-FDDF-4272-8D1D-FF9B966D75B0

ApplicationActivationImpl

Authentication UI CredUI Out of Proc Helper for AppContainer Clients

Authentication UI CredUI Out of Proc Helper for AppContainer Clients

Authentication UI CredUI Out of Proc Helper for AppContainer Clients

Authentication UI CredUI Out of Proc Helper for Non-AppContainer Clients

AuthHost

AxInstSv

CieAxInstallerService Class

IClientSecurity

IeAxIService

IeAxIService2

IeAxIServiceCallback

IMarshal

IMultiQI

IUnknown

Factory Interfaces

B0316D0C-DA2F-40E0-9F91-F600CAF042DC

BrowserBrokerServer

DataExchangeHost

DevicesFlow

DictationHost Class

Showing 57 of 57 entries

Object Properties

AppID	0b15afd8-3a99-4a6e-9975-30d66f7
AutoElevation	False
CanElevate	False
Categories	
Clsid	90f18417f0f1-484e-9d3c-59dceee
CreateContext	LOCAL_SERVER
DefaultCmdLine	
DefaultServer	<APPID HOSTED>
DefaultServerType	LocalServer32
DefaultThreadingModel	Apartment
Elevation	
FactoryInterfaces	
HasTypeLib	False
Interfaces	
InterfacesLoaded	True
Name	CieAxInstallerService Class
Servers	
TreatAs	00000000-0000-0000-0000-000000
TypeLib	00000000-0000-0000-0000-000000

Name

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Bug Classes and Exploitation

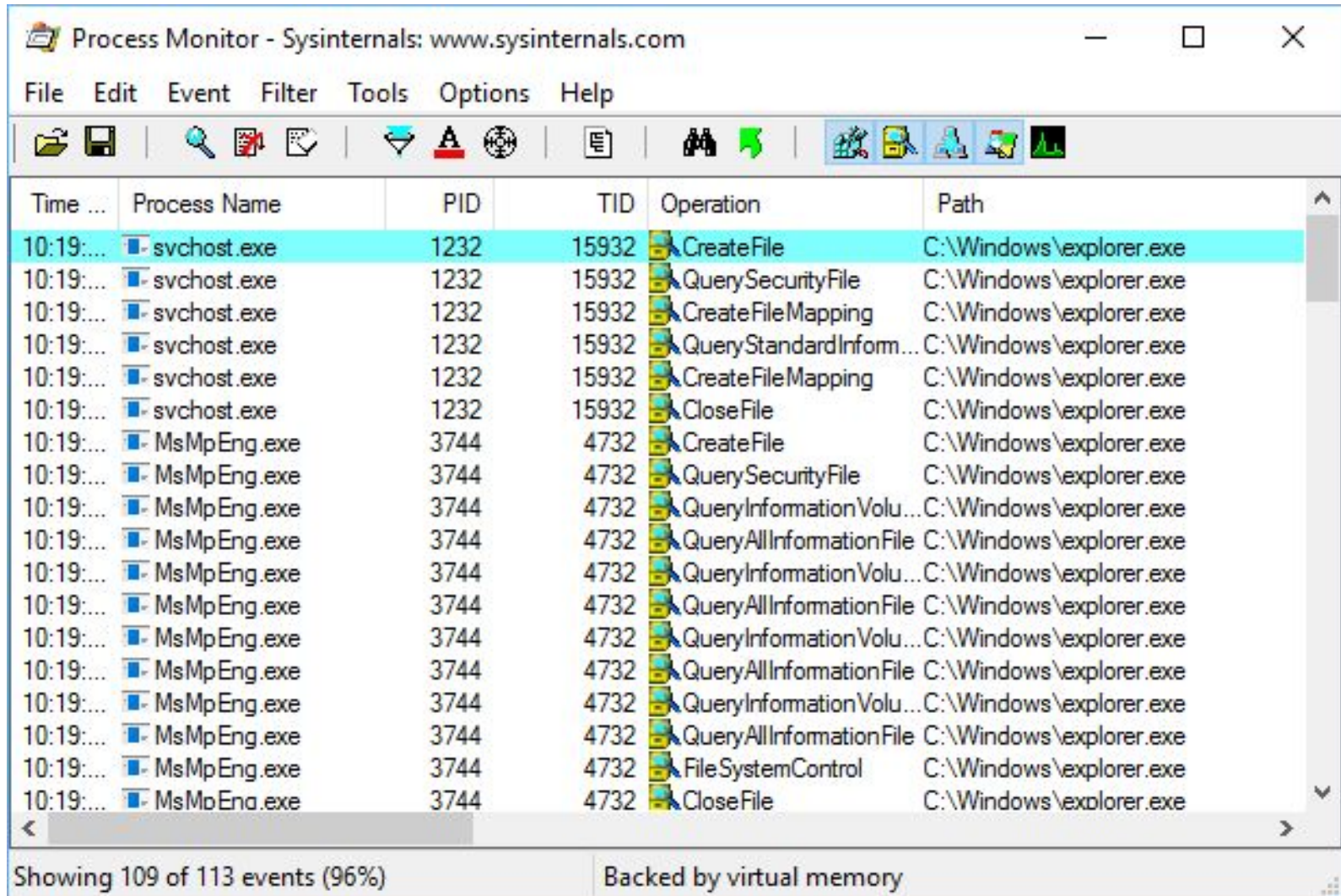
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Example Vulnerabilities

- Toolkit comes with some example vulnerabilities that we can exploit.
 - *LogicalEoPWorkshopDriver* - Simple kernel driver containing a number of logical vulnerabilities
 - *RpcServer* - Simple RPC server containing a number of logical vulnerabilities
 - *COMServer* - Simple COM server in .NET to demonstrate COM based vulnerabilities, is also a client to exploit the an IStorage vulnerability.
- Exploitation Tools
 - *DemoClient* - Simple interface to “exploit” the majority of vulnerabilities
 - *ExploitDotNetDCOMSerialization* - Tool we’ll use to exploit the COM Server via .NET DCOM

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Dynamic Analysis with Process Monitor



Process Monitor - Sysinternals: www.sysinternals.com

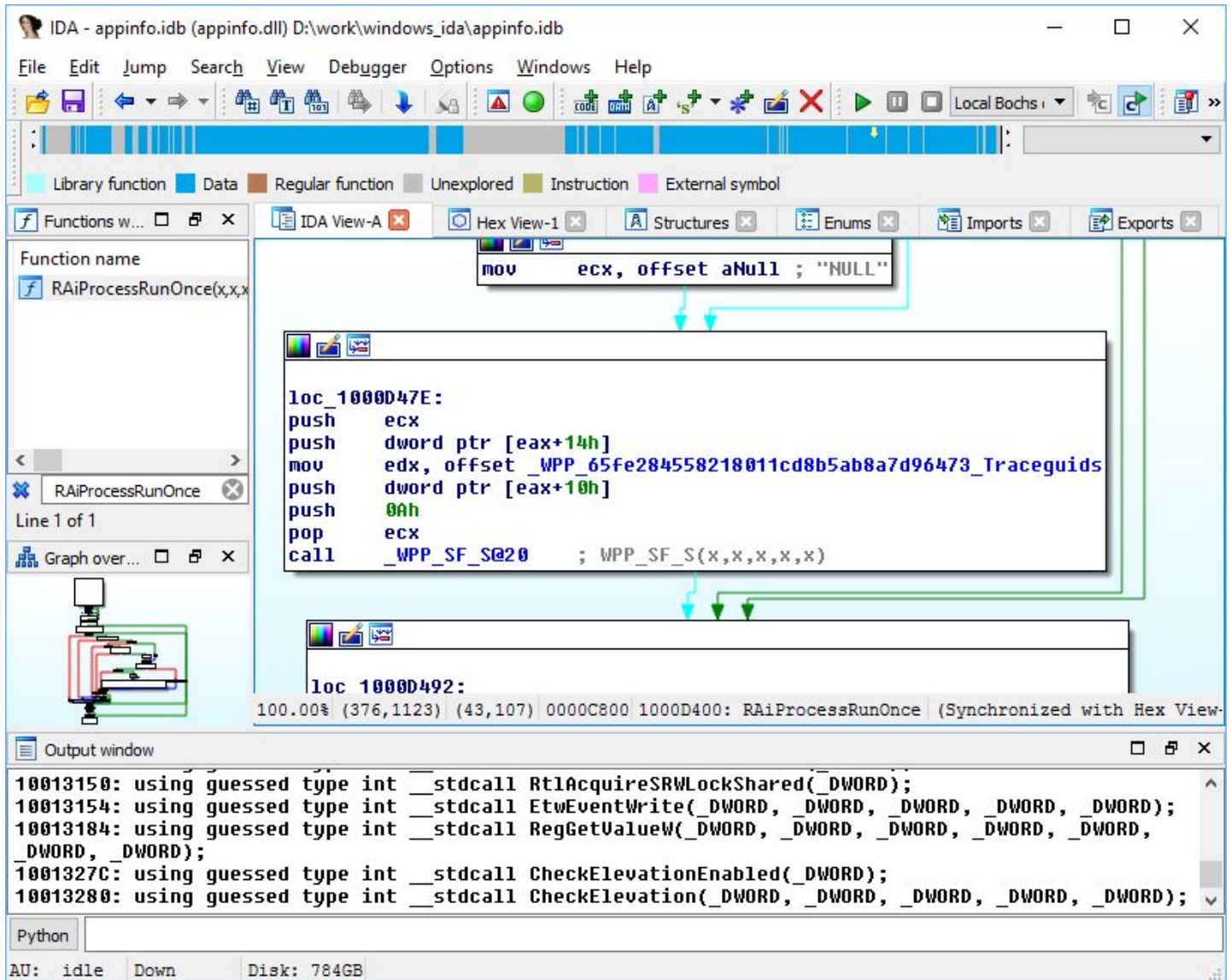
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	TID	Operation	Path
10:19:...	svchost.exe	1232	15932	CreateFile	C:\Windows\explorer.exe
10:19:...	svchost.exe	1232	15932	QuerySecurityFile	C:\Windows\explorer.exe
10:19:...	svchost.exe	1232	15932	CreateFileMapping	C:\Windows\explorer.exe
10:19:...	svchost.exe	1232	15932	QueryStandardInform...	C:\Windows\explorer.exe
10:19:...	svchost.exe	1232	15932	CreateFileMapping	C:\Windows\explorer.exe
10:19:...	svchost.exe	1232	15932	CloseFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	CreateFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QuerySecurityFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryInformationVolu...	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryAllInformationFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryInformationVolu...	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryAllInformationFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryInformationVolu...	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	QueryAllInformationFile	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	File SystemControl	C:\Windows\explorer.exe
10:19:...	MsMpEng.exe	3744	4732	CloseFile	C:\Windows\explorer.exe

Showing 109 of 113 events (96%) Backed by virtual memory

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at: <https://goo.gl/P4Q9GN>

Reverse Engineering with IDA Pro



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Bug Class List

- File path abuse
 - Path Traversal
 - TOCTOU
- Impersonation
 - File/Process Access Under Impersonation
 - Insecure kernel Impersonation and Token Usage
- Insecure Kernel Resource Access
- COM Bugs
 - .NET DCOM Service
 - Bound objects

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Path Canonicalization

```
bool TestLoadLibrary(const wchar_t* name)
{
    wstring full_path = L"C:\\Windows\\" + name;
    HMODULE hModule = LoadLibrary(full_path.c_str());
    if (hModule != nullptr)
    {
        printf("Loaded module: %p\n", hModule);
        FreeLibrary(hModule);
        return true;
    }
    return false;
}
```



No verification
on name


Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 5 : Exploiting Path Canonicalization

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Insecure Path Usage

```
bool TestLoadLibraryCanonical(const wchar_t* name)
{
    if (wcschr(name, '\\') || wcschr(name, '/'))
    {
        printf("Error, name contains path separators\n");
        return false;
    }
    return TestLoadLibrary(name);
}
```



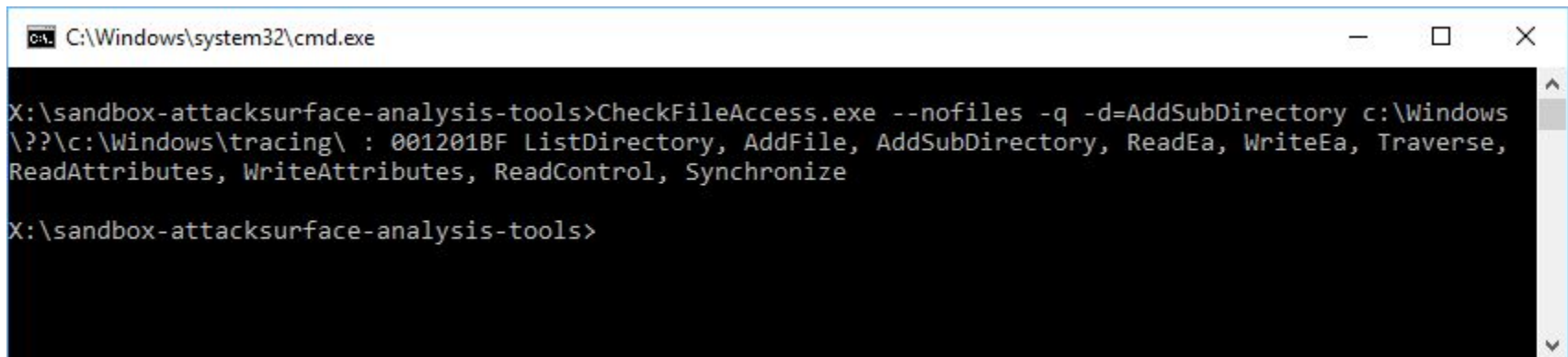
Disallow
canonicalization

- All paths will be c:\windows\name
- We can't write to c:\Windows, or can we?

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

NTFS Alternate Data Streams (ADS)

- ADS allows you to create substreams on NTFS files by separating using the ':' separator
 - E.g. abc:xyz is stream named 'xyz' on the existing file 'abc'
- Also works for directories as if we've got *AddSubDirectory* access



```
C:\Windows\system32\cmd.exe

X:\sandbox-attacksurface-analysis-tools>CheckFileAccess.exe --nofiles -q -d=AddSubDirectory c:\Windows
\??\c:\Windows\tracing\ : 001201BF ListDirectory, AddFile, AddSubDirectory, ReadEa, WriteEa, Traverse,
ReadAttributes, WriteAttributes, ReadControl, Synchronize

X:\sandbox-attacksurface-analysis-tools>
```

C:\Windows\Tracing we can write an ADS to!

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 6: Exploiting Named Streams

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

File Time-of-check Time-of-use

```
bool TestLoadLibraryTocTou(const wchar_t* lib_path) {  
    if (VerifyEmbeddedSignature(lib_path)) {  
        HMODULE hModule = LoadLibrary(lib_path);  
        if (hModule != nullptr) {  
            printf("Loaded module: %p\n", hModule);  
            FreeLibrary(hModule);  
            return true;  
        }  
    }  
    return false;  
}
```

Load the library
path is signed

Verifies that the
DLL is signed

Exploiting TOCTOU

- A few different ways to exploit:
 - Race condition between check time and library loading
 - We could rewrite the file in between the check and the load.
 - Exploit differing path parsing behaviours between functions
 - LoadLibrary will search the PATH for a filename which isn't an absolute path
 - Accessing a file takes path verbatim, LoadLibrary plays some games with extensions

lpFileName [in]

...

If the string specifies a module name without a path and the file name extension is omitted, the function appends the default library extension .dll to the module name.

To prevent the function from appending .dll to the module name, include a trailing point character (.) in the module name string.

c:\abc becomes c:\abc.dll when loaded

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Exploitation

- 1) Copy a signed binary to a known location with the name *abc*.
Doesn't have to be a DLL, can be anything signed.
 - a) Kernel32.dll works on Windows 10 AE.
- 2) Copy the DLL you want to load to *abc.dll* in the same directory.
- 3) Pass path to service specifying *abc* as the filename
- 4) Your desired DLL should be loaded

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

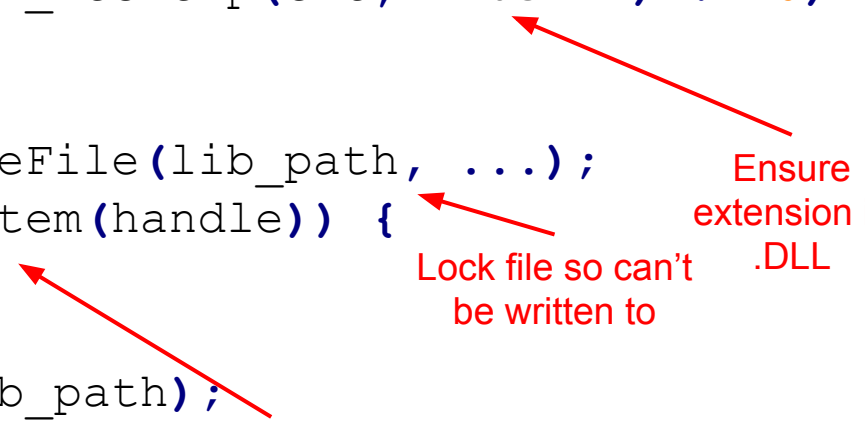
DEMO 7 : TOCTOU on Name

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

File Time-of-check Time-of-use

```
bool TestLoadLibraryTocTouHardened(const wchar_t* lib_path)
{
    LPWSTR ext = PathFindExtensionW(lib_path);
    if (ext == nullptr || _wcsicmp(ext, L".dll") != 0)
        return false;

    HANDLE handle = CreateFile(lib_path, ...);
    if (!CheckFileIsInSystem(handle)) {
        return false;
    }
    return LoadLibrary(lib_path);
}
```



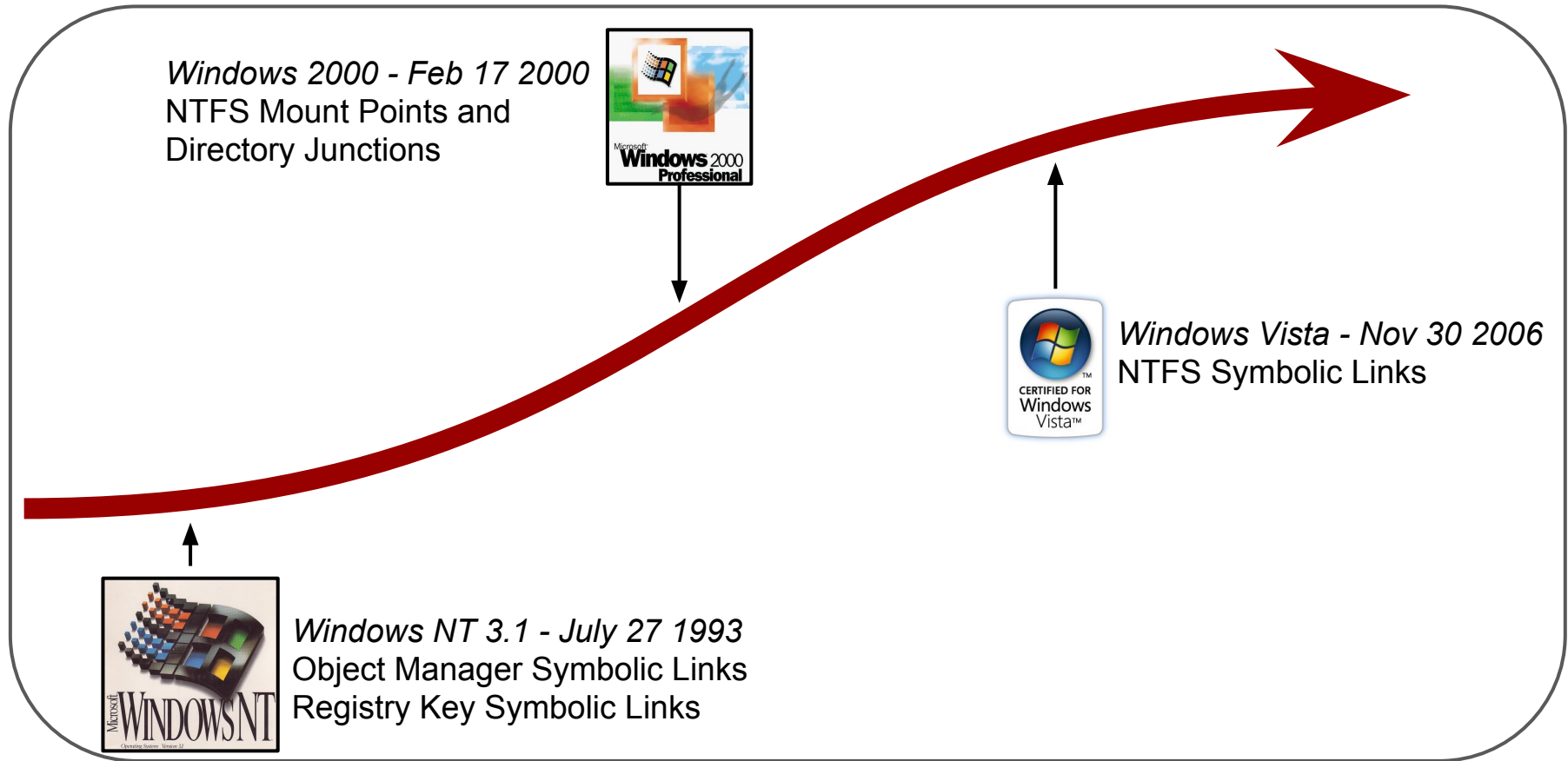
Ensure
extension is
.DLL

Lock file so can't
be written to

Check opened file is in
system directory

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Windows Symbolic Links



Opportunistic Locks (OPLOCK)

- Winning the TOCTOU race means either brute force or finding some what of timing the request.
- We can win the race in many cases using OPLOCKS
 - Locks a file to prevent access, can select Read/Write/Delete or Exclusive
 - Gets a callback when someone else tries to open the file. Closing handle allows that use to continue.

```
DeviceIoControl(g_hFile,  
    FSCTL_REQUEST_OPLOCK_LEVEL_1,  
    NULL, 0,  
    NULL, 0,  
    &bytesReturned,  
    &g_o) ;
```

Note: Must use a Level 1 “Exclusive” lock for system files as normal user always gets Read sharing access.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

DEMO 8 : Symbolic Link TOCTOU

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Reading Files Under Impersonation

```
bool TestCreateProcess() {  
    RpcImpersonateClient();  
    WCHAR cmdline[] = L"c:\\windows\\notepad.exe";  
    if (CreateProcess(cmdline, cmdline, ...)) {  
        return true;  
    }  
    return false;  
}
```

Calling User is Impersonated

Created process uses current process token, not impersonated token

- CreateProcess uses the current process's token by default, not any impersonation token
- However the file is accessed under the identity of the impersonated user
- Can we exploit this?

Current User's DosDevices Directory

- Current user's DosDevices directory is stored in
 \Sessions\0\DosDevices\X-Y
 - X-Y is the current user's login ID
- This is writable by the current user for obvious reasons
- We can re-direct C: to anywhere we like and get arbitrary process running with the identity of the RPC server

NOTE: This will won't work in a sandbox. It also used to work for DLLs but Microsoft fixed that glitch 🙄

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 9 : DosDevices Redirect

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Incorrect Handle Duplication

```
int TestDuplicateHandle(int handle) {  
    unsigned long pid;  
    I_RpcBindingInqLocalClientPID(hBinding, &pid);  
  
    HANDLE process = OpenProcess(PROCESS_DUP_HANDLE, pid);  
    HANDLE ret;  
    DuplicateHandle(process, handle, process,  
        &ret, 0, FALSE, DUPLICATE_SAME_ACCESS))  
  
    return (int)ret;  
}
```

Opening Calling Process

Duplicate handle to and from same process

- Surely this isn't very useful? Can only duplicate a handle we already have back into our own process?

Hard-coded Handle Values

- The Windows kernel supports two pseudo handle values which are used whenever handles are accessed
 - -1 = Handle to the current process
 - -2 = Handle to the current thread
- When duplicating handles -1 refers to source process, which in this case is our own process (useless)
- However -2 refers to the calling thread, which is actually the thread in the RPC server. We can use this to get arbitrary code execution in the server process.

NOTE: If the handle is transported as a DWORD this will fail on 64 bit platforms as the value will zero extended to HANDLE 🤔

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 10 : Handle Duplication

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Insecure Kernel Resource Access

```
NTSTATUS CreateFile(PUNICODE_STRING Path) {  
    OBJECT_ATTRIBUTES obj_attr = { 0 };  
    HANDLE Handle = NULL;  
    ULONG AttributeFlags = OBJ_KERNEL_HANDLE;  
  
    InitializeObjectAttributes(&obj_attr,  
                             Path, AttributeFlags);  
  
    return ZwCreateFile(&Handle,  
                       MAXIMUM_ALLOWED,  
                       &obj_attr ...);  
}
```

Setting attribute
flags

Calling Zw* function,
will transition to Kernel
previous process mode

- Calls to Zw functions transition to kernel mode (when called from kernel code) which disables all security.
- Should be setting OBJ_FORCE_ACCESS_CHECK flag.

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Resource Access

- File/Registry Key/Resource by default created with inherited security descriptor for parent
- However as long as call takes place inside the current process then the OWNER of the file will be the current user
- If inherited descriptor has CREATOR OWNER ACE we get those access rights
- Even if not we're owner so can open for WRITE_DAC access and modify at will


Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 11 : Privileged Resource Creation

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Missing Impersonation Level Checks

```
BOOLEAN GetTokenElevated(PACCESS_TOKEN Token)
{
    PTOKEN_ELEVATION Elevation = NULL;
    BOOLEAN ret = FALSE;
    if (NT_SUCCESS(SeQueryInformationToken(Token,
        TokenElevation, &Elevation)))
    {
        ret = !!Elevation->TokenIsElevated;
    }
    return ret;
}
```



No check that token is
not at impersonation level
< Impersonation

- Kernel might check current caller is an administrator or elevated.
- Must ensure that token is not an identification token

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Exploiting Split-Token Admin

- Numerous ways of getting an Identification level token for a privileged user

Token View window showing token details for User32. The token is Primary, with an Impersonation Level of N/A. The Is Elevated checkbox is unchecked.

Field	Value
User:	DESKTOP-J5BUTHV\user
User SID:	S-1-5-21-2352253840-2516495137-3341880293-1001
Token Type:	Primary
Impersonation Level:	N/A
Token ID:	00000000-000304B2
Authentication ID:	00000000-000275FC
Origin Login ID:	00000000-000003E7
Modified ID:	00000000-000304A3
Integrity Level:	Low
Session ID:	1
Elevation Type:	Limited
Is Elevated:	False
Name:	User32
Id:	00000000-0002758F

Token View window showing token details for User32. The token is Impersonation, with an Impersonation Level of Identification. The Is Elevated checkbox is checked. A red arrow points from the 'Linked Token' button in the left window to the 'Is Elevated' checkbox in this window.

Field	Value
User:	DESKTOP-J5BUTHV\user
User SID:	S-1-5-21-2352253840-2516495137-3341880293-1001
Token Type:	Impersonation
Impersonation Level:	Identification
Token ID:	00000000-003AC83B
Authentication ID:	00000000-000275D3
Origin Login ID:	00000000-000003E7
Modified ID:	00000000-000275FB
Integrity Level:	High
Session ID:	1
Elevation Type:	Full
Is Elevated:	True
Name:	User32
Id:	00000000-0002758F

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Insecure Impersonation in System Thread

```
void SystemThread(PACCESS_TOKEN token) {  
    UNICODE_STRING file;  
    CHECK_STATUS(PsImpersonateClient(KeGetCurrentThread(),  
        token, FALSE, FALSE, SecurityImpersonation));  
    RtlInitUnicodeString(&file, L"\\SystemRoot\\demo.txt");  
    CHECK_STATUS(CreateFileSecure(&file));  
}  
  
void BadImpersonation() {  
    SeCaptureSubjectContext(&subject_context);  
    PsCreateSystemThread(SystemThread,   
        SeQuerySubjectContextToken(&subject_context));  
}
```

Forces Impersonation Level

Runs in a System Thread

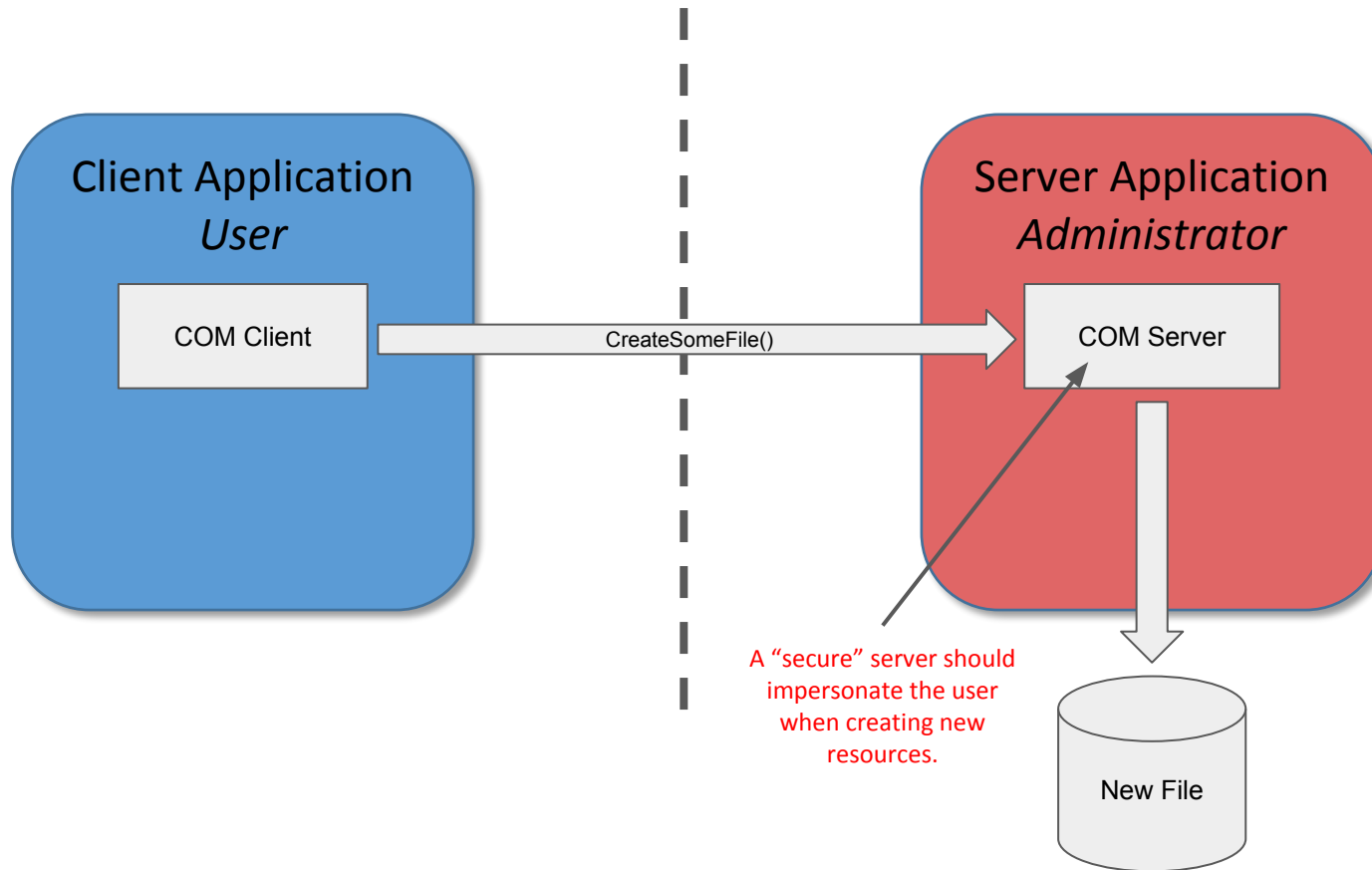
- System Threads run in the System process which has SeImpersonatePrivilege access
- If system thread misuses impersonation can elevate privileges

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

DEMO 12 : Admin Token Check Bypass

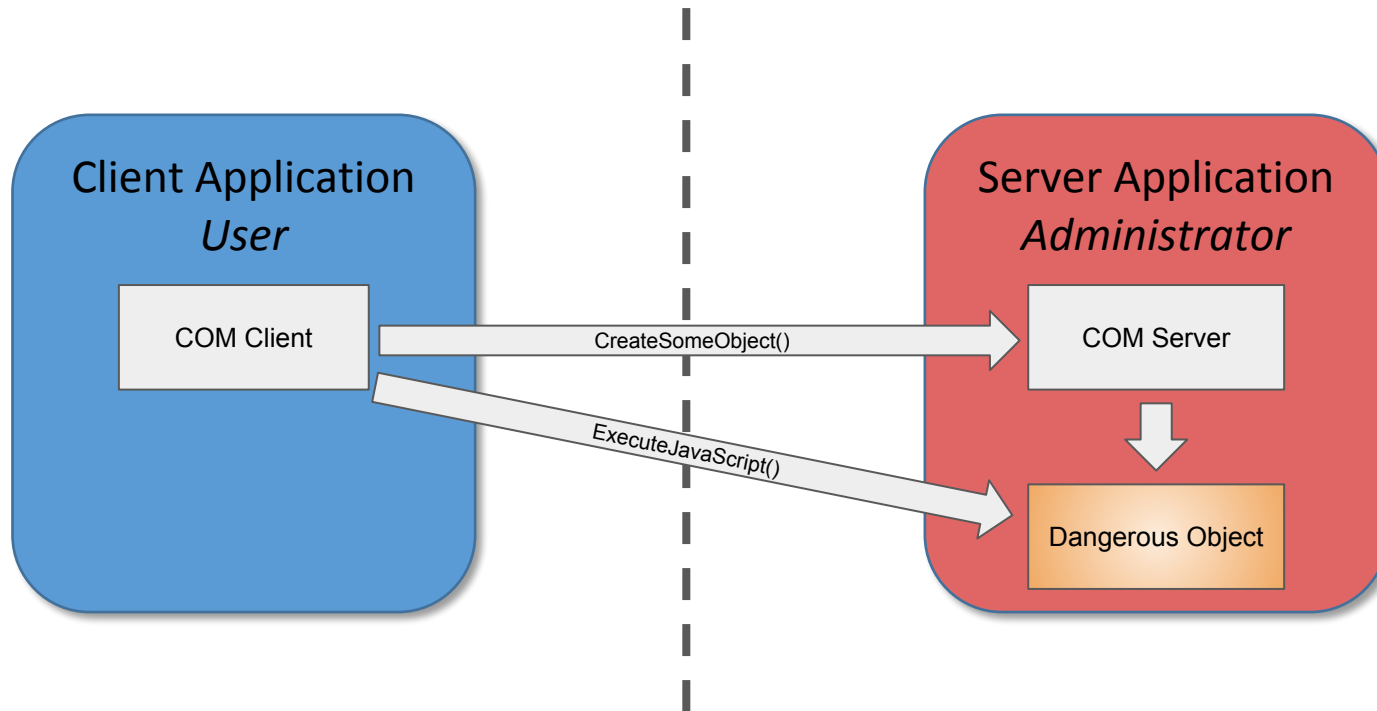
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

Bound COM Objects



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Bound COM Objects



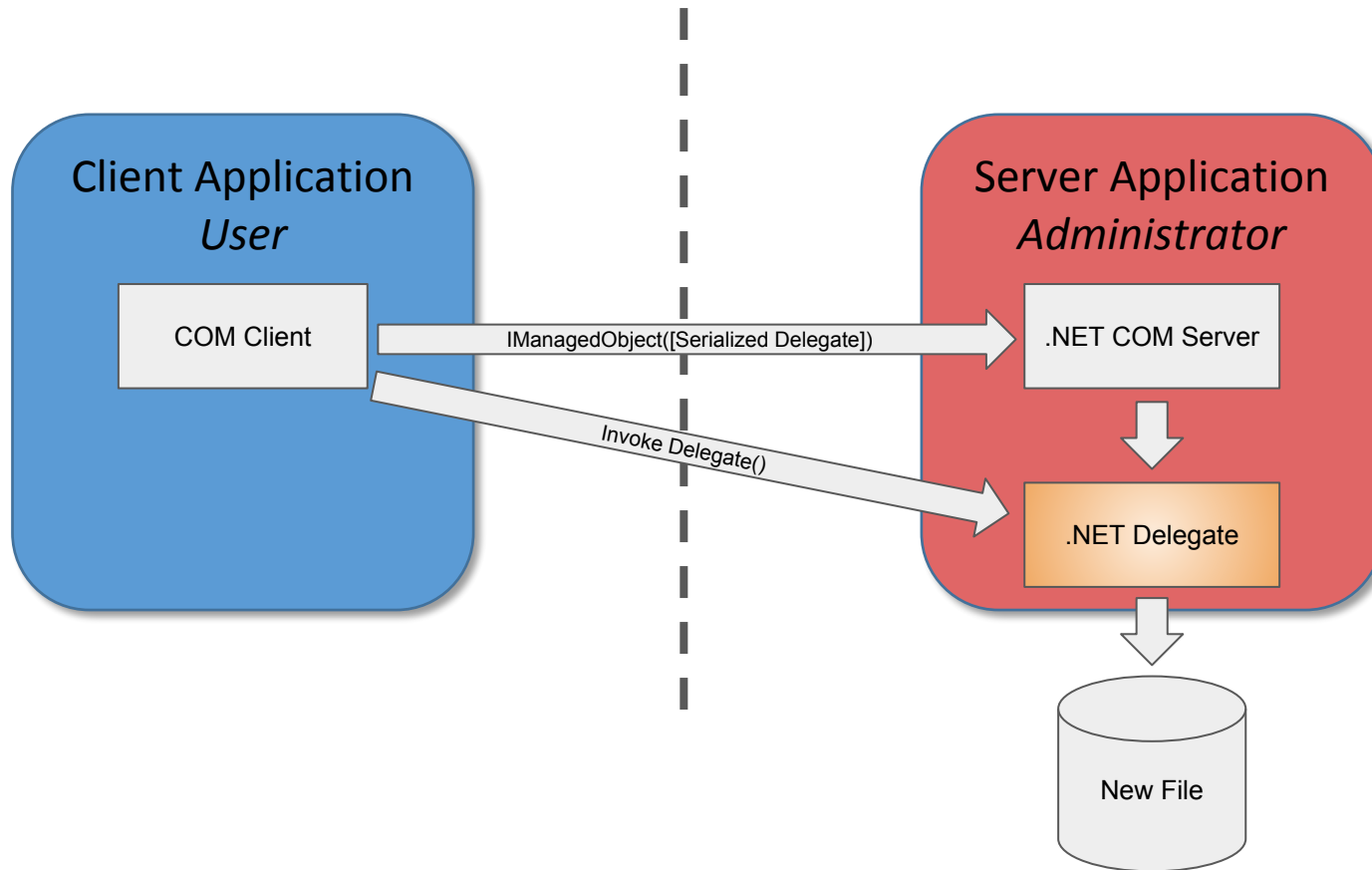
Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

.NET DCOM Services

```
[ComVisible(true),  
Guid("801445A7-C5A9-468D-9423-81F9D13FEE9B")]  
public class COMService : ICOMInterface {  
  
    int cookie = reg_services.RegisterTypeForComClients(  
        typeof(COMService),  
        RegistrationClassContext.LocalServer,  
        RegistrationConnectionType.MultipleUse);  
  
    Console.ReadLine();  
    reg_services.UnregisterTypeForComClients(cookie);  
}
```

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

.NET DCOM Objects (Binary Serialization)



Tools/Examples at: <https://goo.gl/HzZ2Gw> - Workbook at:
<https://goo.gl/P4Q9GN>

DEMO 13 : .NET DCOM Elevation

Tools/Examples at: <https://goo.gl/HzZ2Gw> - Wookbook at:
<https://goo.gl/P4Q9GN>

Resources

<https://github.com/google/sandbox-attacksurface-analysis-tools>

<https://github.com/google/symboliclink-testing-tools>

<https://github.com/tyranid/oleviewdotnet>

<https://github.com/tyranid/ExploitDotNetDCOM>