How to Implement COM Monitor

- Advanced COM worm -

Speaker: AmesianX(Park Young Ho, amesianx@nate.com)

What Am I Going to Present?

- Introduction to COM Monitoring
- Kinds of COM Monitoring
- Information We Can Get Through COM Monitoring
- Why Did I Make COM Monitor?
- New Vulnerabilities
- Changes of Various Attack

What is COM Monitoring?

➔ COM Monitoring is watching components' communication.

Advantage of COM Monitoring

➔ COM Monitoring can monitor 'Method Invoke' between client and component and control the action.

What is COM?

- Component Object Model (COM) is a platform for software componentry introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology. The term COM is often used in the software development world as an umbrella term that encompasses the OLE, OLE Automation, ActiveX, COM+ and DCOM technologies. Although COM was introduced in 1993, Microsoft did not begin emphasizing the name COM until 1997.

  for more information...
  http://en.wikipedia.org/wiki/Component_Object_Model

❑ Binary Hooking

- ➔ Hooking COM by Code Injection/Interception
- ➔ It is impossible to every COM(Customized)

❑ Wrapped Control

- ➔ WrapControl uses exiting programming methods because of COM's structure
- ➔ Making exiting COM called 'A' into a new COM called 'B' and then aggregate
- ➔ It is possible to change a COM into a new COM, but impossible to hook every COM(Customized)

❑ COM Interface Hooking(InterfaceHooking)

- ➔ COM Interface Hooking is well-known hooking method
- ➔ When making COM through hooking the API CoCreateInstance, hijacking Interface-ID and change it, so this leads to hijacking COM creation(Global)
- ➔ This is mentioned in MS's Detours Hooking Library.

Universal Wrapper Control

- wrapping every COM by making universal COM control
- Every COM has virtual function table, so it is a design structure of the intermediate,delegate of methods among virtual function table
- Delegate(the intermediate)maps the virtual function table, and then hijacking method call general-purposely.
- Actually, when we make a wrapper control, it is possible to make a wrapper control if a COM module to be wrapped is made as a 'aggregatable' development model. But if a COM module is not made as a aggregatable model, the wrapper development technique to aggregate the COM module is needed. We call this technique "Universal Delegator".

❑ Probability of information disclosure

- ➔ We can almost get nothing through COM hooking in common programs
- ➔ It is very clear that information disclosure is possible in the component-based host(client) applications like Internet Explorer.
- ➔ Example of getting information or controlling through COM Monitoring

  Ex) “Memory Hacking” in ‘KBS1 TV’
  Ex2) “Demonstration Movie” in ‘Real Internet Banking Web Site’

❑ Technical Contents That Couldn't Be Opened in News

- ➔ Users' PC is infected by virus, so universal wrapper control is installed.
- ➔ The ActiveX(COM) of PKI solution is already hijacked by universal wrapper control.
- ➔ When a user accesses finance web pages by using Internet Explorer, universal wrapper control(fake control) will operate instead of a real PKI module(ActiveX).
- ➔ The method of PKI ActiveX(COM) module is controlled by the universal wrapper control to a cracker's programming.
- ➔ PKI module is an encryption module, so it uses the two methods, Encrypt and Decrypt through the whole service.
- ➔ Cracker fabricates the normal transfer data as an argument of Encrypt function as his own data(transfer the data to cracker)
- ➔ All data of finance web page go through the encryption of PKI module, so Decrypt method will take the encrypted data and decrypt it, and then it shows the decrypted values on the web.
- ➔ Cracker shows "Transferred successfully."(auto – previous information collection)

❑ Common Important Information Disclosure

(What is important information?)

- ➔ Making Secure Communication Module by COM
  Even if SecureModule is made by COM, there will be plaintext section.
  (ex: HTTPS communication implementation module)
- ➔ Making Payment Module by COM
  Payment module can be forged by hacker when using payment data
  (I demonstrate by using common pc instead of real finance web sites owing to the request of finance.)
- ➔ Believing XML(being charge of asynchronous communication in Web 2.0 era) blindly
  (MSXML.DLL is an extension module of COM. It is easy to hijack the communication of XML.)
- ➔ Using COM as a core module in HTS
  HTS is programmed by various languages. To meet the standard of communication, COM is used. One company

❑ Common Important Information Disclosure

(What is important information?)

➔ Common Application(COM Host Functioning Application)

* Application that can include COM is called 'host' or 'client'.
(ex: Internet Explorer)

* The reason of COM use is to call method(function) in the language such as Visual Basic which can't use pointer or script language.

* When making one application by using different programming languages, you make the core module by COM and the communication information is monitored.

* I will demonstrate the hijacking of the calls in common application and in script.

❑ Applied Research(Development of ActiveCheck)

Research to apply the features of Universal Delegator

- ➔ The motive of this research began from very funny idea.
- ➔ The purpose of this research was to make a fuzzer which can find vulnerabilities of ActiveX.
- ➔ Three men(I – make engine, main programmer, general producer) began to make a fuzzer. This project began from one colleague's suggestion that I needed time to be accustomed to my new company.
- ➔ To make a better ActiveX fuzzer than COMRaider of iDefense, many ideas came out.
- ➔ In the beginning of project, I considered the relation of methods. If we have to call methods like "install() -> init() -> start() -> vulnmethod()", and then overflow can be happened, we need to know the order of calling method.
- ➔ If initialization begins when arguments like init("ILovePH")passes, how can a fuzzer identify this?
- ➔ These kinds of things brought about the need of monitoring.

LINK – http://www.develop.com/technology/resourcedetail.aspx?id=76121ebc-beab-42a9-a891-869c4be57771

❑ Most Serious Problem

On the progress of development, I heard about Jikto.

➔ Fuzzer can detect buffer overflow, but can't find vulnerabilities of normal functions. To complement this defect, monitoring should be possible.

➔ Jikto is a java script scanner. If a hacker performs malicious a XSS attack and injects a script into a normal web site, a PC of a user becomes a tool of remote vulnerability. In this case, a user may become a victim of attack because Jikto call a method through the use of normal ActiveX XML. Even though the attack of Jikto is going on, if we can monitor, we can detect the running of Jikto. Not only Jikto but many ActiveXes used in Korea can detect whether they are attacked by normal method or not.

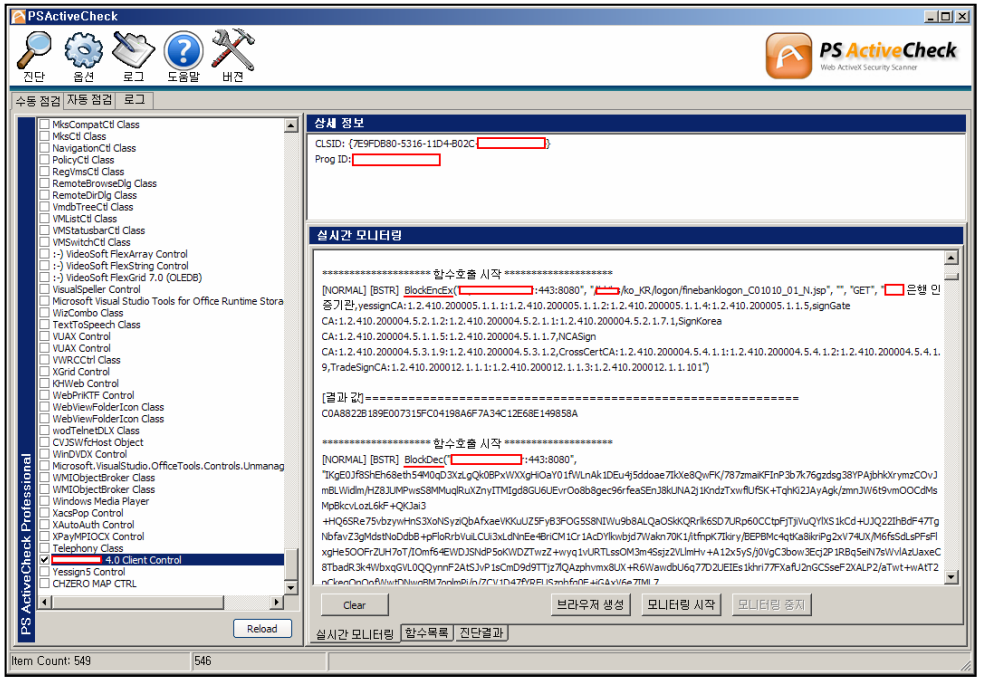
LINK - http://www.develop.com/technology/resourcedetail.aspx?id=76121ebc-beab-42a9-a891-869c4be57771

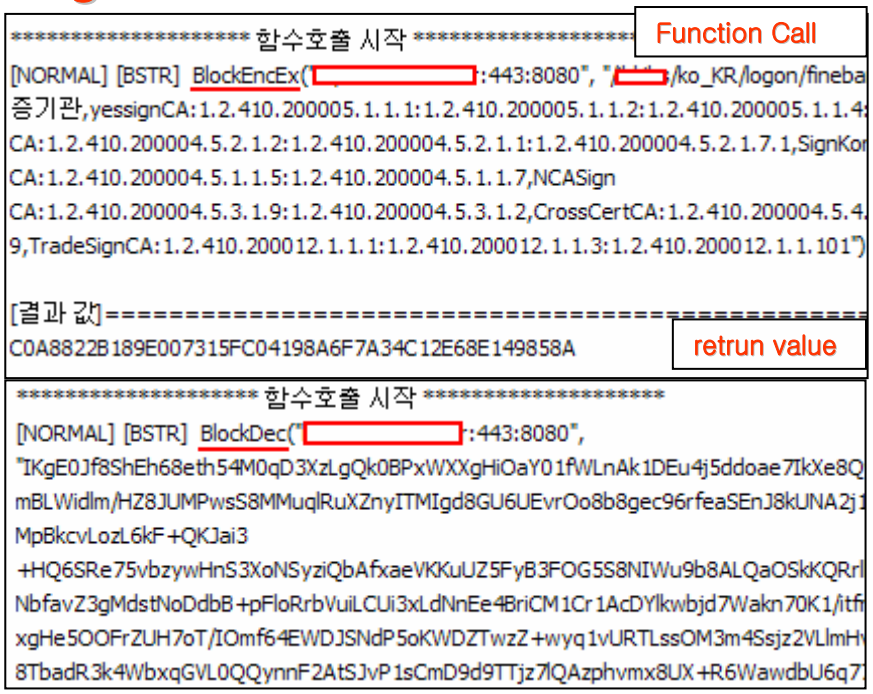
❑ Rising of New Vulnerability

Right after I developed monitoring and fuzzing features, I found out a serious problem.

[Internet Explorer – Running..]

➔ Internet Banking System RealTime Tracing

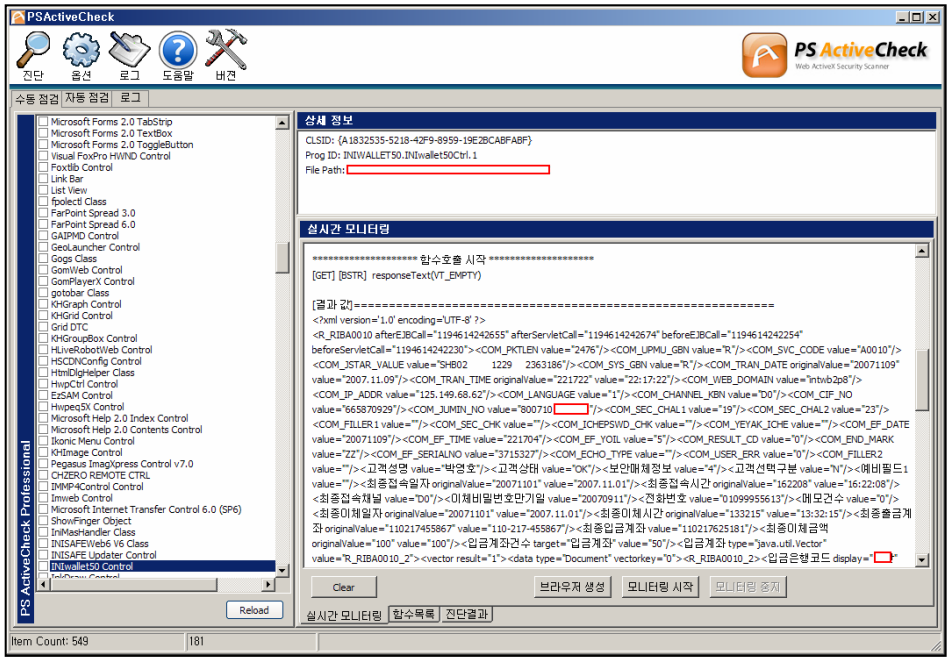




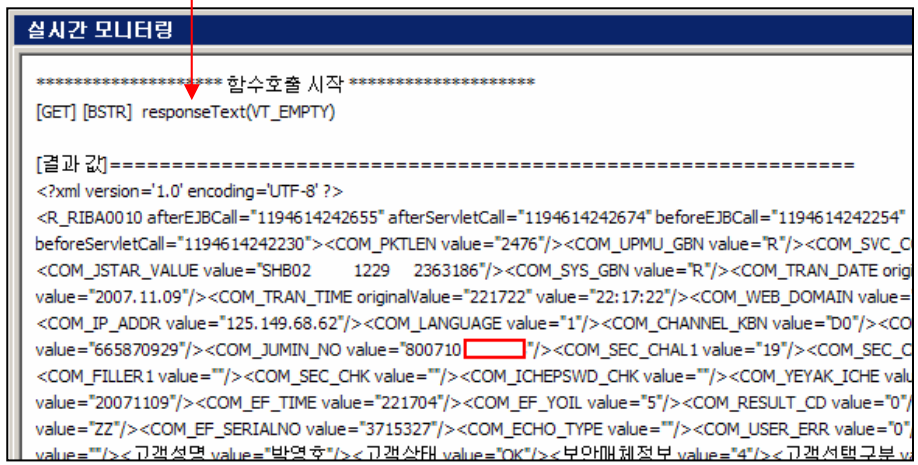
➔ XML Communication RealTime Tracing

Class	ProgId	CLSID
XML HTTP	Msxml2.XMLHTTP	{F6D90F16-9C73-11D3-B32E-00C04F990BB4}
XML HTTP 2.6	Msxml2.XMLHTTP.2.6	{f5078f1e-c551-11d3-89b9-0000f81fe221}
XML HTTP 3.0	Msxml2.XMLHTTP.3.0	{F5078F35-C551-11D3-89B9-0000F81FE221}
XML HTTP 4.0	Msxml2.XMLHTTP.4.0	{88D969C5-F192-11D4-A65F-0040963251E5}
XML HTTP 5.0	Msxml2.XMLHTTP.5.0	{88D969EA-F192-11D4-A65F-0040963251E5}
XML HTTP 6.0	Msxml2.XMLHTTP.6.0	{88d96a0a-f192-11d4-a65f-0040963251e5}
XML HTTP Request	Microsoft.XMLHTTP.1.0	{ED8C108E-4349-11D2-91A4-00C04F7969E8}

[OneClick]
XML Modules

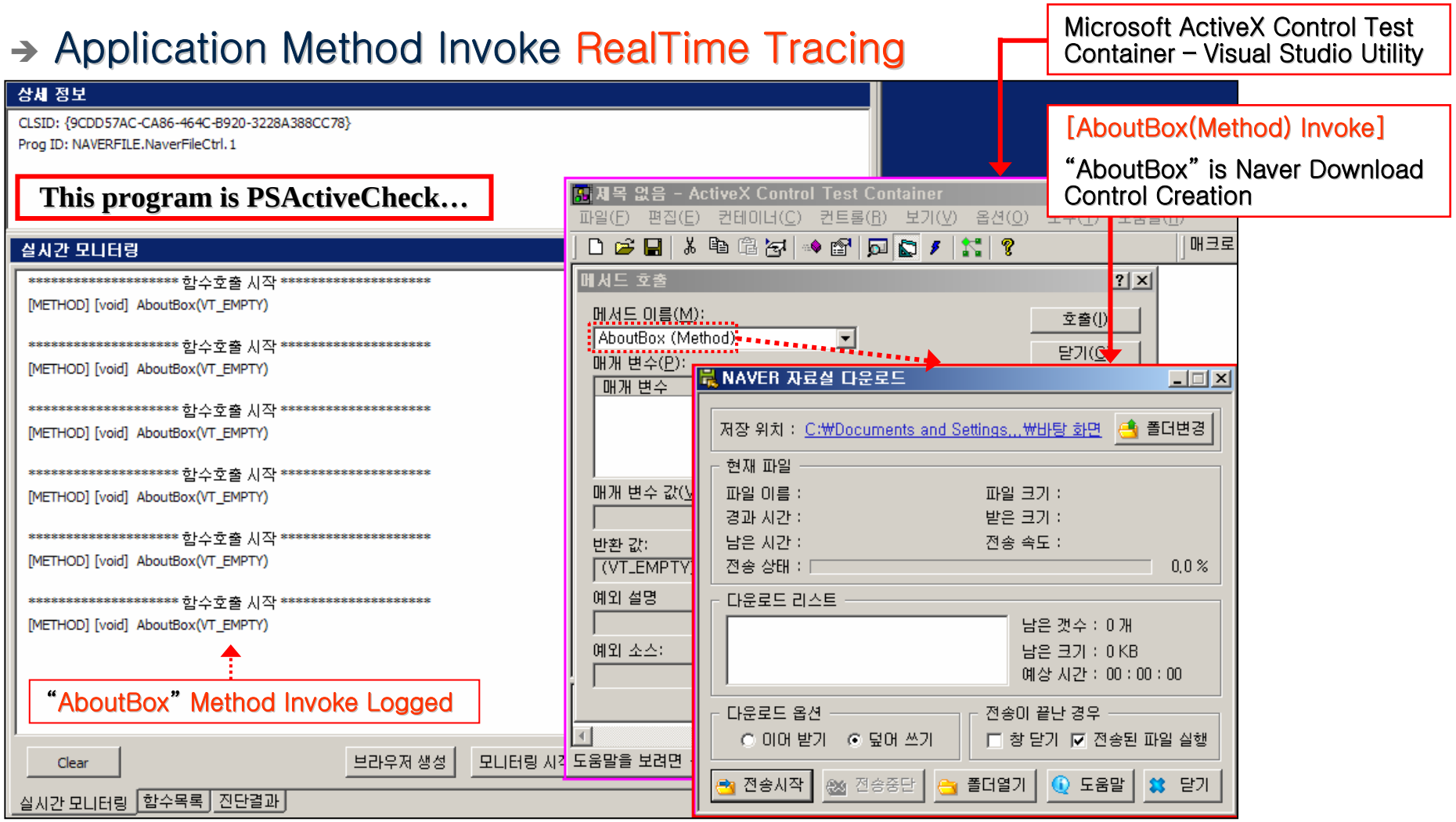


XML "responseText" Function Call



return value

➔ Application Method Invoke RealTime Tracing



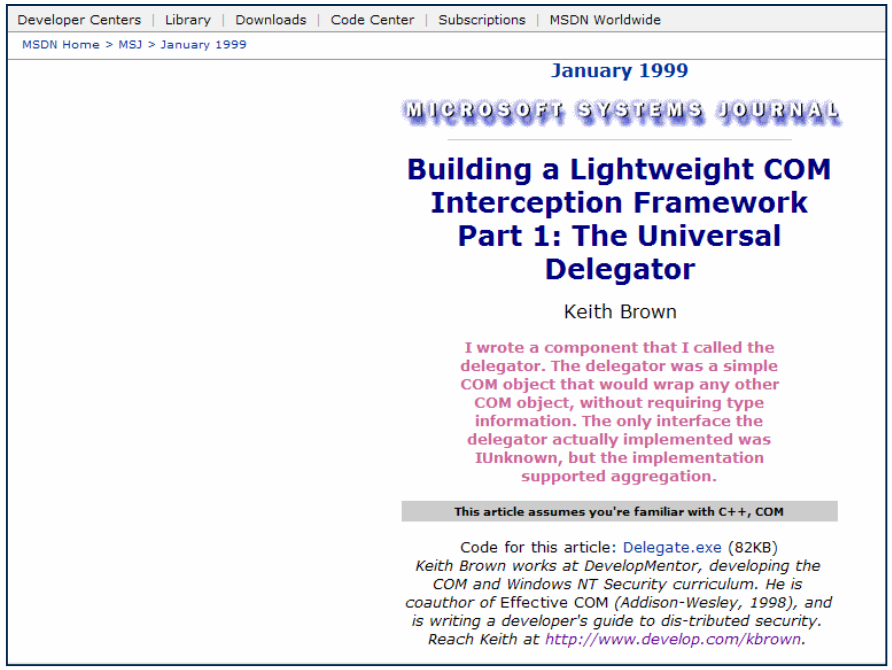
❑ Two theories of Attack

I will demonstrate COM monitoring, and then explain the technique.
And I will mention the exiting hooking techniques, too.

➔ Now I will demonstrate the operation of PSActiveCheck.
And I will also give a technical explanation about the operation.
Thanks for the permission of my company.

❑ Appearance of Universal Delegator

- ➔ One article of "Microsoft System Journal" in 1999, 1
- ➔ Title – "Building a Lightweight COM Interception Framework"
- ➔ Keith Brown opened his research result
- ➔ The applied research of "Universal Delegator" by Keith was derived.
- ➔ Progress rate of derived research is very big, but the scale isn't big



Developer Centers | Library | Downloads | Code Center | Subscriptions | MSDN Worldwide

MSDN Home > MSJ > January 1999

January 1999

MICROSOFT SYSTEMS JOURNAL

Building a Lightweight COM Interception Framework Part 1: The Universal Delegator

Keith Brown

I wrote a component that I called the delegator. The delegator was a simple COM object that would wrap any other COM object, without requiring type information. The only interface the delegator actually implemented was IUnknown, but the implementation supported aggregation.

This article assumes you're familiar with C++, COM

Code for this article: Delegate.exe (82KB)

Keith Brown works at DevelopMentor, developing the COM and Windows NT Security curriculum. He is coauthor of Effective COM *(Addison-Wesley, 1998), and is writing a developer's guide to dis-tributed security. Reach Keith at http://www.develop.com/kbrown.*

LINK – http://www.develop.com/technology/resourcedetail.aspx?id=76121ebc-beab-42a9-a891-

❑ Keith Brown was said that Universal Delegator for Description.

Create custom COM interception plumbing

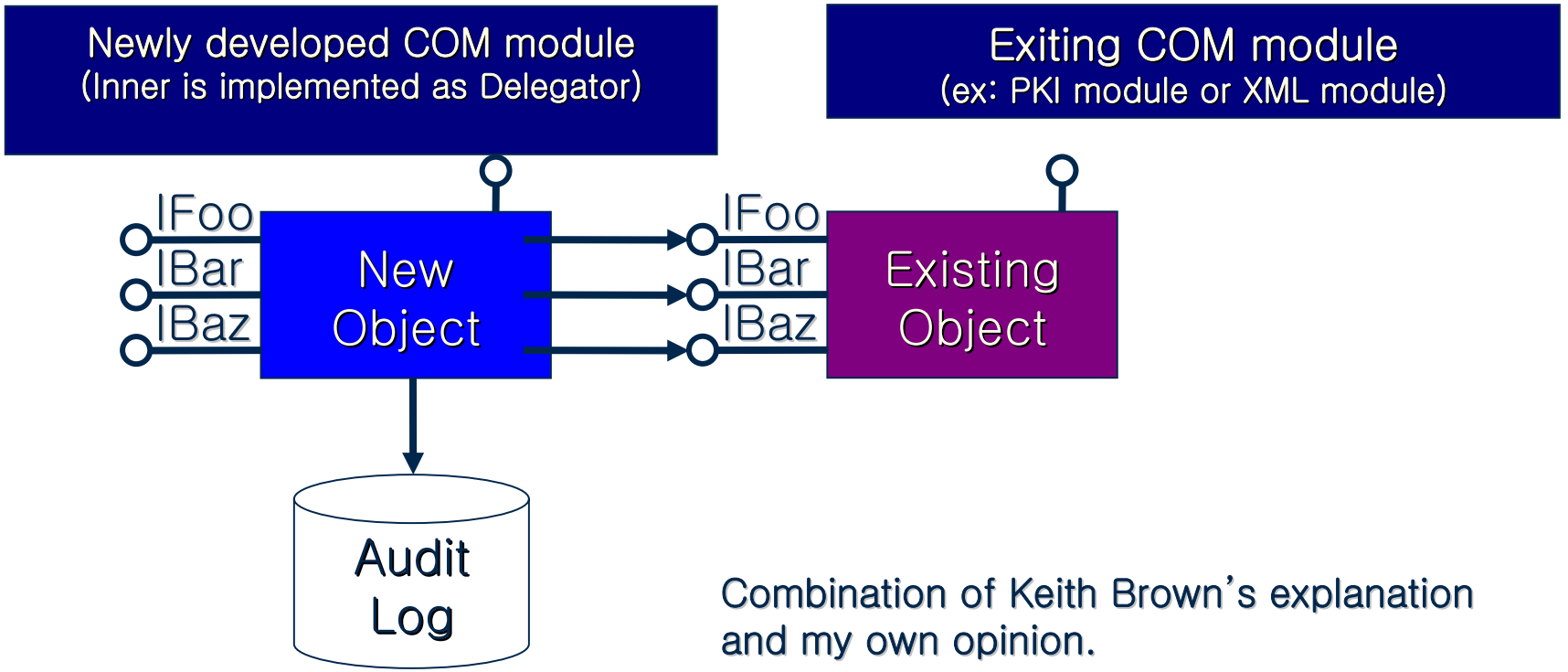
➔ This hookable component wraps another object and transparently exposes all of its interfaces.

➔ You can combine this with a hook that performs arbitrary preprocessing and/or postprocessing of every method call, or a simpler hook (like the Anonymous Delegator Hook) which gets a peek at each interface pointer on the wrapped object (and drops the authn level, etc.).

LINK - http://www.develop.com/technology/resourcedetail.aspx?id=76121ebc-beab-42a9-a891-

❑ Structure of Universal Delegator

➔ Newly developed COM module can choose one (can be loaded first) of many methods such as interface hooking, system registry forgery, API function forgery, or kernel hooking, etc.



Combination of Keith Brown's explanation and my own opinion.

```
// included by delegate.cpp
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
DELEGATOR_ENTRY_POINTS(9)
… 생략 …
DELEGATOR_ENTRY_POINTS(1022)
DELEGATOR_ENTRY_POINTS(1023)
```

```
Virtual method table of 1024
method table. The purpose is
1:1 mapping(Delegate) all the
existing COM methods.
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
```

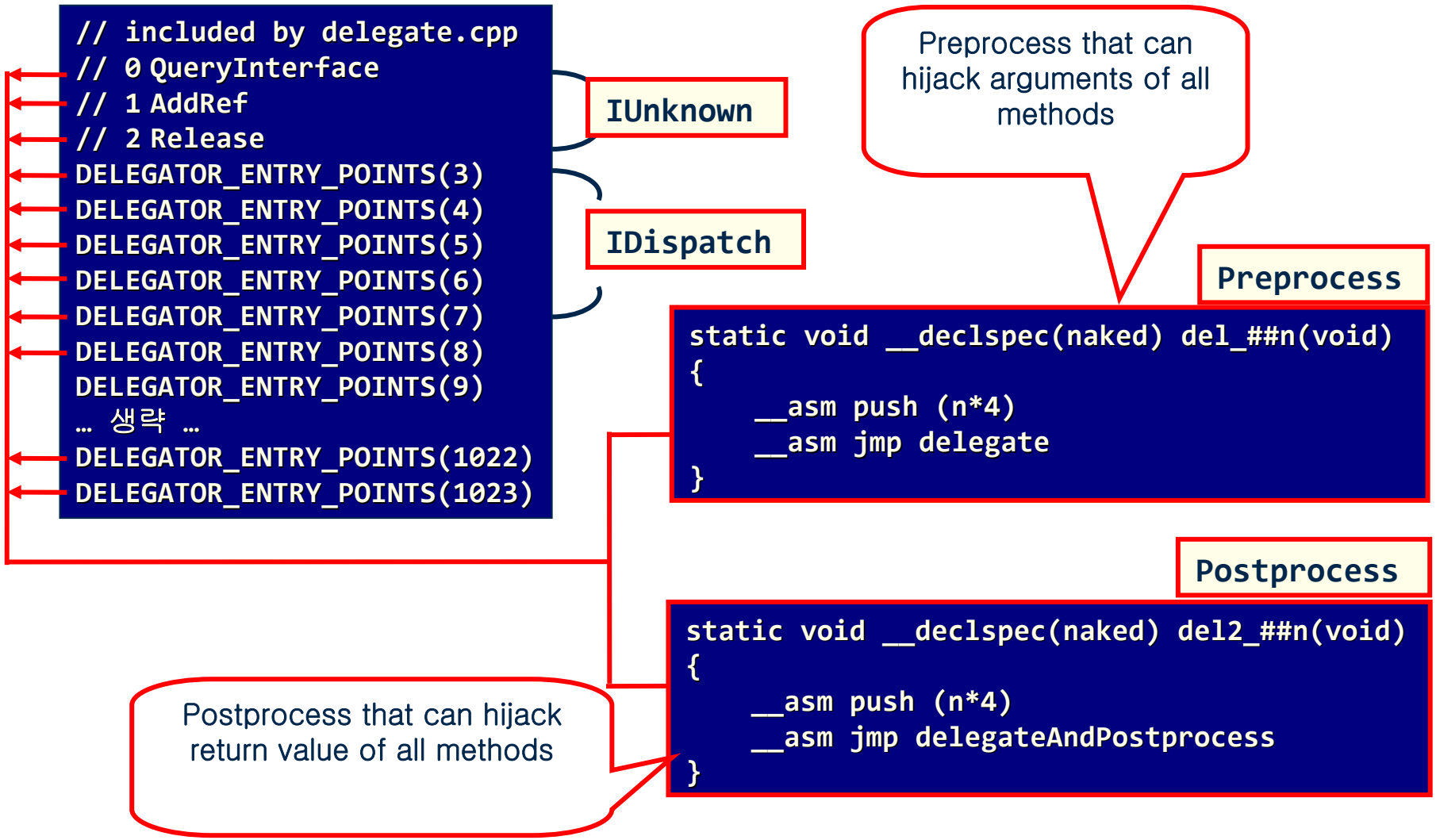
IUnknown

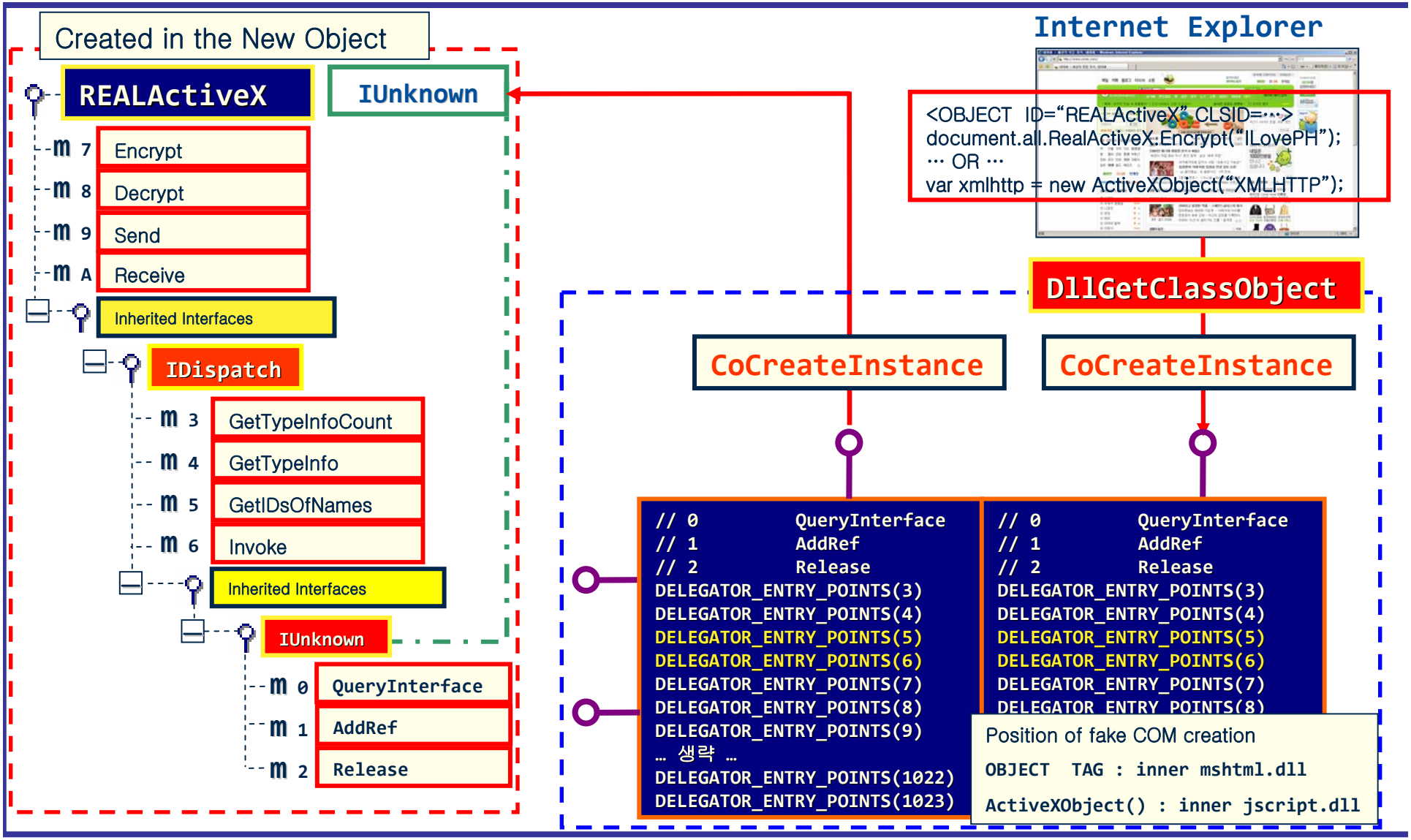
IDispatch

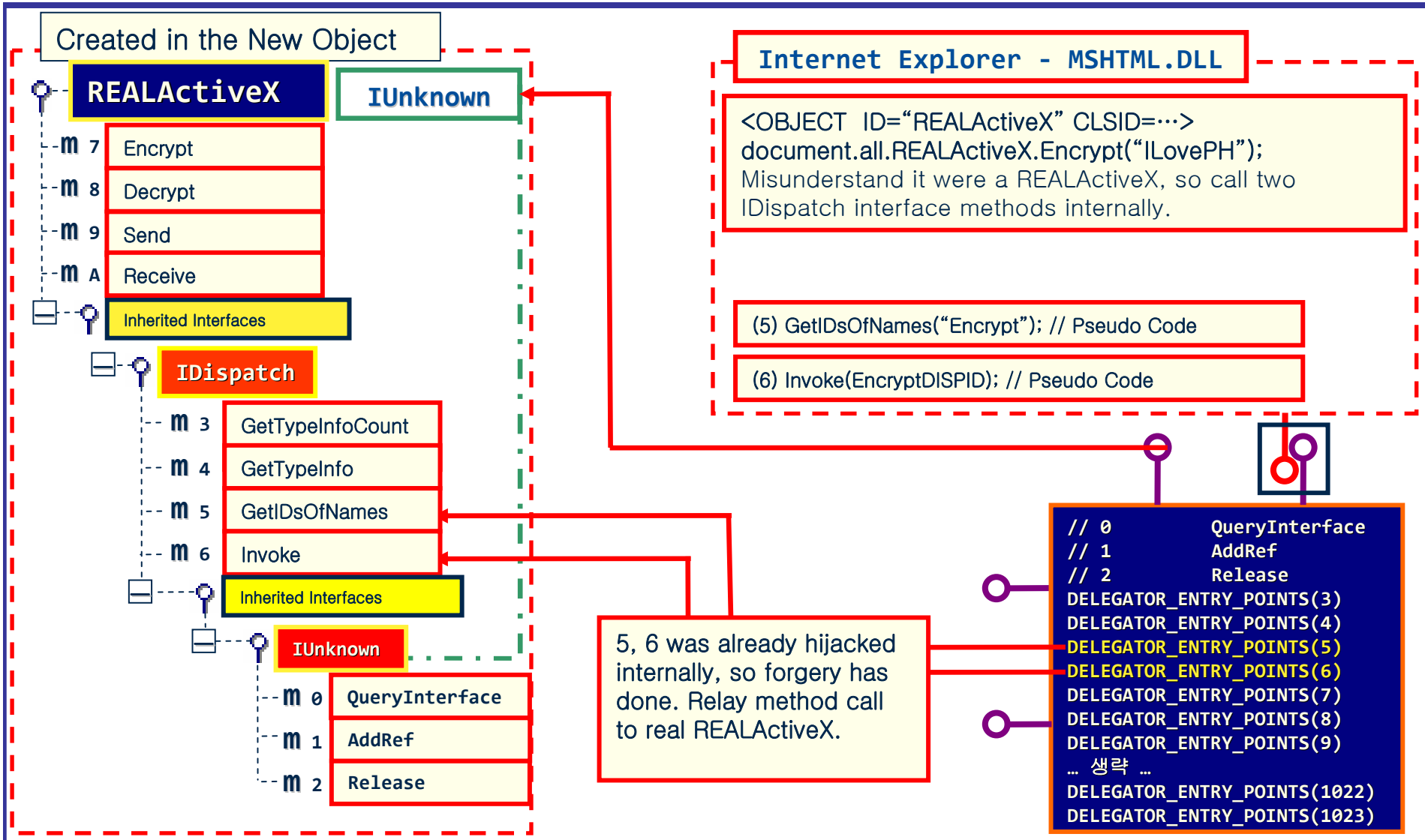
DELEGATOR_ENTRY_POINTS

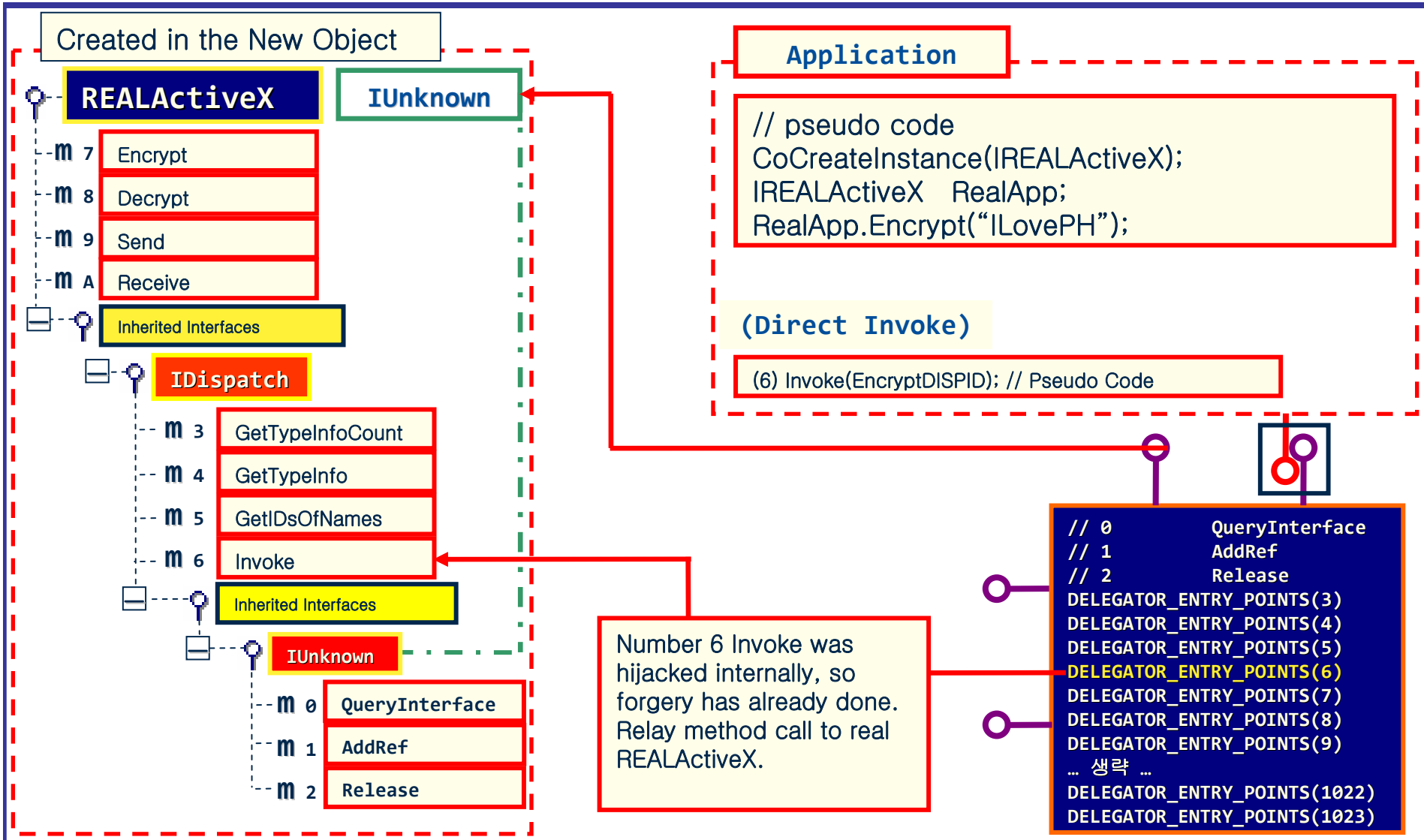
```
#define DELEGATOR_ENTRY_POINTS(n)
static void __declspec(naked) del_##n(void)
{ __asm push (n*4) __asm jmp delegate }
static void __declspec(naked) del2_##n(void)
{ __asm push (n*4) __asm jmp delegateAndPostprocess }

#include "entrypoints.inc"
```


// included by delegate.cpp
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
DELEGATOR_ENTRY_POINTS(9)
… 생략 …
DELEGATOR_ENTRY_POINTS(1022)
DELEGATOR_ENTRY_POINTS(1023)
IUnknown
IDispatch
Preprocess that can hijack arguments of all methods
Preprocess
static void __declspec(naked) del_##n(void)
{
__asm push (n*4)
__asm jmp delegate
}
Postprocess
static void __declspec(naked) del2_##n(void)
{
__asm push (n*4)
__asm jmp delegateAndPostprocess
}
Postprocess that can hijack return value of all methods


Created in the New Object
REALActiveX
IUnknown
m 7 Encrypt
m 8 Decrypt
m 9 Send
m A Receive
Inherited Interfaces
IDispatch
m 3 GetTypeInfoCount
m 4 GetTypeInfo
m 5 GetIDsOfNames
m 6 Invoke
Inherited Interfaces
IUnknown
m 0 QueryInterface
m 1 AddRef
m 2 Release
Internet Explorer
<OBJECT ID="REALActiveX" CLSID=…>
document.all.RealActiveX.Encrypt("ILovePH");
… OR …
var xmlhttp = new ActiveXObject("XMLHTTP");
DllGetClassObject
CoCreateInstance
CoCreateInstance
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
DELEGATOR_ENTRY_POINTS(9)
… 생략 …
DELEGATOR_ENTRY_POINTS(1022)
DELEGATOR_ENTRY_POINTS(1023)
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
Position of fake COM creation
OBJECT TAG : inner mshtml.dll
ActiveXObject() : inner jscript.dll

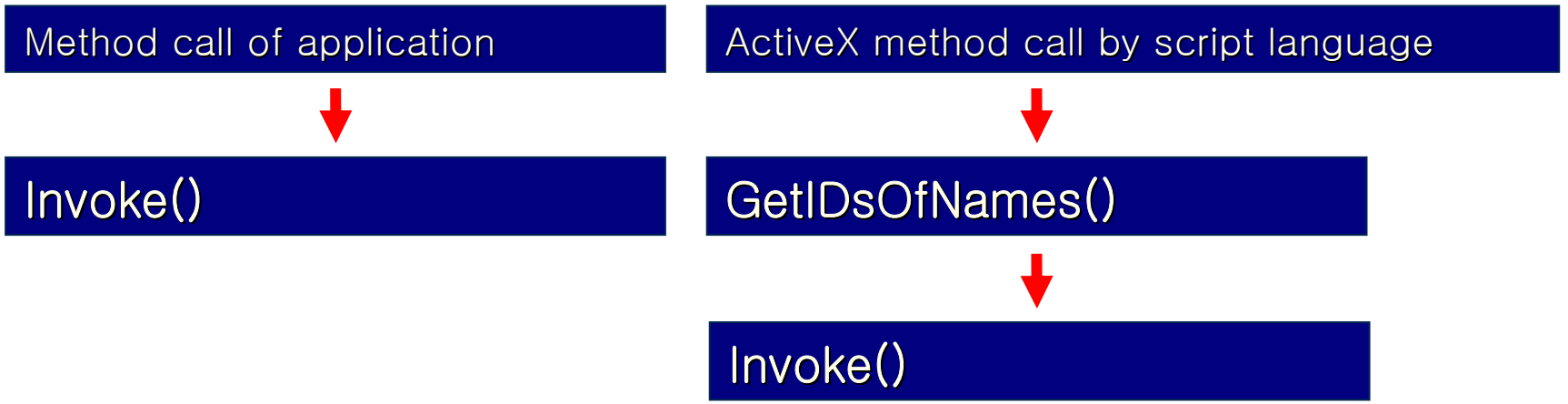

Created in the New Object
REALActiveX
IUnknown
m 7 Encrypt
m 8 Decrypt
m 9 Send
m A Receive
Inherited Interfaces
IDispatch
m 3 GetTypeInfoCount
m 4 GetTypeInfo
m 5 GetIDsOfNames
m 6 Invoke
Inherited Interfaces
IUnknown
m 0 QueryInterface
m 1 AddRef
m 2 Release
Internet Explorer - MSHTML.DLL
<OBJECT ID="REALActiveX" CLSID=…>
document.all.REALActiveX.Encrypt("ILovePH");
Misunderstand it were a REALActiveX, so call two IDispatch interface methods internally.
(5) GetIDsOfNames("Encrypt"); // Pseudo Code
(6) Invoke(EncryptDISPID); // Pseudo Code
5, 6 was already hijacked internally, so forgery has done. Relay method call to real REALActiveX.
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
DELEGATOR_ENTRY_POINTS(9)
… 생략 …
DELEGATOR_ENTRY_POINTS(1022)
DELEGATOR_ENTRY_POINTS(1023)


Created in the New Object
REALActiveX
IUnknown
m 7 Encrypt
m 8 Decrypt
m 9 Send
m A Receive
Inherited Interfaces
IDispatch
m 3 GetTypeInfoCount
m 4 GetTypeInfo
m 5 GetIDsOfNames
m 6 Invoke
Inherited Interfaces
IUnknown
m 0 QueryInterface
m 1 AddRef
m 2 Release
Application
// pseudo code
CoCreateInstance(IREALActiveX);
IREALActiveX RealApp;
RealApp.Encrypt("ILovePH");
(Direct Invoke)
(6) Invoke(EncryptDISPID); // Pseudo Code
Number 6 Invoke was hijacked internally, so forgery has already done. Relay method call to real REALActiveX.
// 0 QueryInterface
// 1 AddRef
// 2 Release
DELEGATOR_ENTRY_POINTS(3)
DELEGATOR_ENTRY_POINTS(4)
DELEGATOR_ENTRY_POINTS(5)
DELEGATOR_ENTRY_POINTS(6)
DELEGATOR_ENTRY_POINTS(7)
DELEGATOR_ENTRY_POINTS(8)
DELEGATOR_ENTRY_POINTS(9)
… 생략 …
DELEGATOR_ENTRY_POINTS(1022)
DELEGATOR_ENTRY_POINTS(1023)

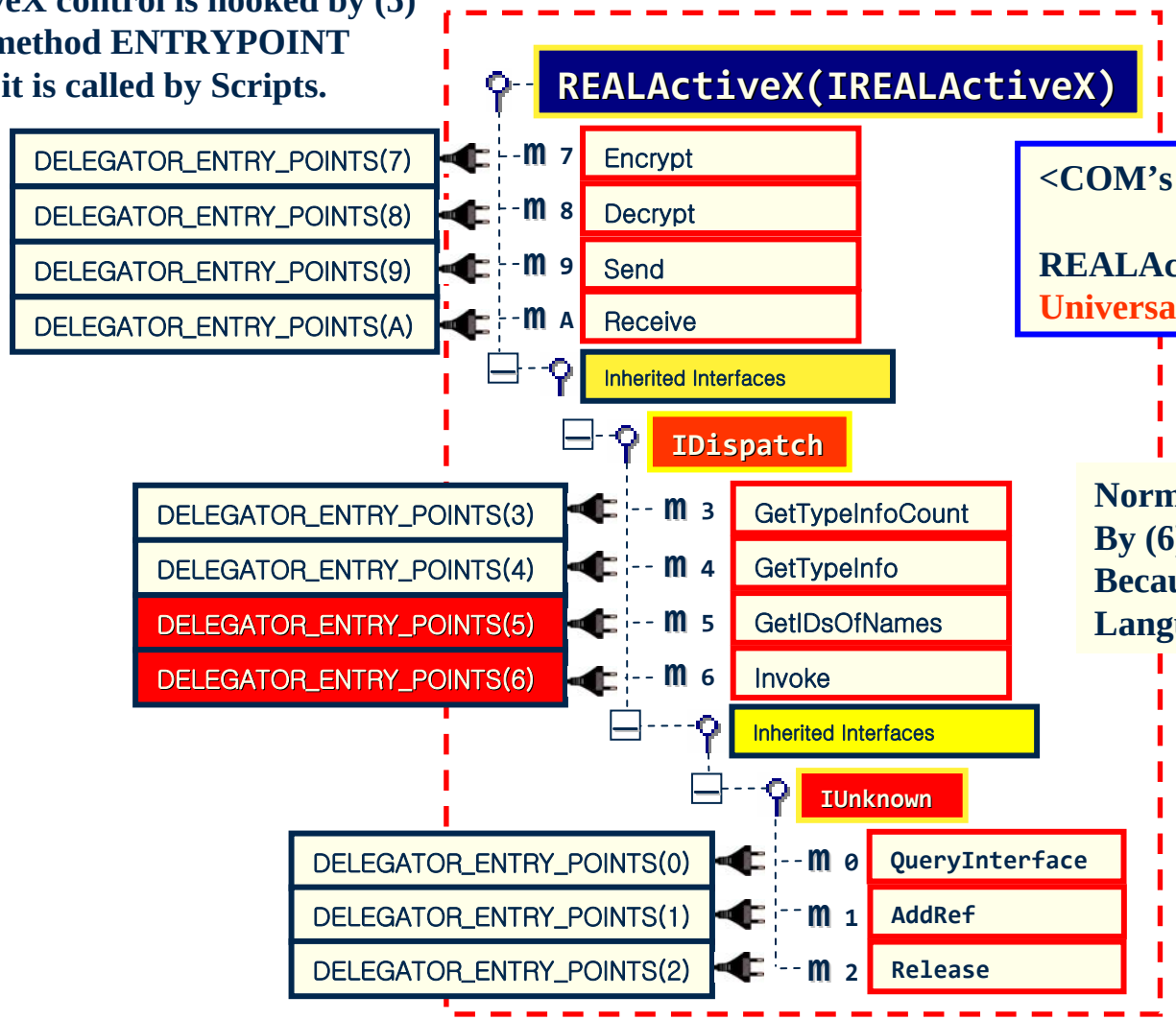
❑ Invisible Inner Operation

➔ Method Call in common application (ex: MfcApp.exe)
When you call method in common application, you use ‘pointer’.

➔ Method Call in script language (inside of IE: Jscript.dll)
You can’t use pointer in script language, so you call method through IDispatch.(This is the reason of slow Javascript.)



All ActiveX control is hooked by (5) and (6) method ENTRYPOINT Because it is called by Scripts.



<COM's Virtual Function Table>

REALActiveX's lpVtbl is mapped by Universal Delegator's Method Tables.

Normal COM module is hooked By (6) method ENTRYPOINT Because it is called by C++. (Pointer Language)

How to Implement COM Monitor(Universal Version)

- This structure is a basis of Universal Wrapper hooking mechanism.
- In order to trace COM modules in real-time, it is needed more implementation.

Preprocess

```
static void __declspec(naked) del_##n(void)
{
    __asm push (n*4)
    __asm jmp delegate
}
```

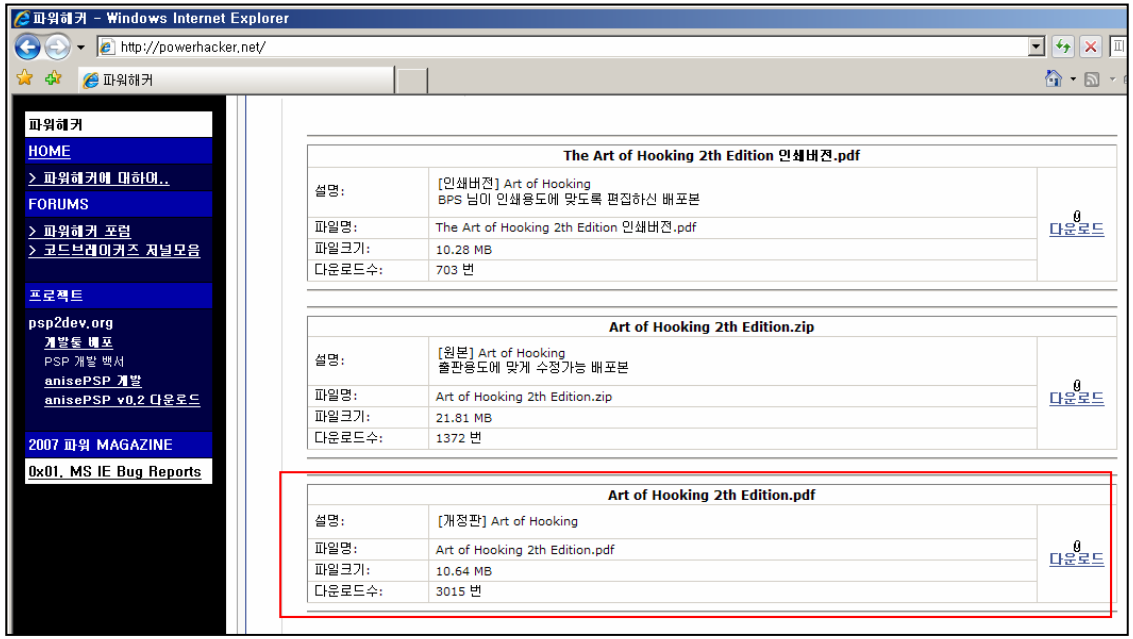
Postprocess

```
static void __declspec(naked) del2_##n(void)
{
    __asm push (n*4)
    __asm jmp delegateAndPostprocess
}
```

All of functionality is place of two delegators.

How to Implement COM Monitor(Custom Version)

- this is the old-age inline patching technique
- See the articles (in powerhacker.net)
- Title - "Art of Hooking" by AmesianX (COM Hooking Part is Lesson-2 in this articles.)



Art of Hooking - http://powerhacker.net/forums/download.php?id=5 - Internet Articles
Art of Hooking - http://powerhacker.net/forums/download.php?id=113 - Printable Articles

nothing is impossible..