Detecting Union Type Confusion in Component Object Model

Yuxing Zhang¹, Xiaogang Zhu², Daojing He⁺¹,³, Minhui Xue⁴, Shouling Ji⁵, Mohammad Sayad Haghighi², Sheng Wen², and Zhiniang Peng⁶

¹East China Normal University, ²Swinburne University of Technology, ³Harbin Institute of Technology, Shenzhen, ⁴CSIRO’s Data61, ⁵Zhejiang University, ⁶Sangfor Technologies Inc.
The Microsoft Component Object Model (COM) defines a binary interoperability standard for creating reusable software libraries that interact at run time.

The Microsoft Component Object Model (COM) defines a binary interoperability standard for creating reusable software libraries that interact at run time.

COM is the foundation for many other technologies, such as Microsoft's OLE and Active, Windows Runtime.

The Microsoft Component Object Model (COM) defines a binary interoperability standard for creating reusable software libraries that interact at run time.

COM is the foundation for many other technologies, such as Microsoft's OLE and Active, Windows Runtime.

Many COM server run in high privilege, union type confusion in COM has the potential to be used in the development of 100%-reliable exploits.

Union In COM

```c
// Non-Encapsulated Union
union Union_C {
    struct Struct_0 Arm_0;
    struct Struct_1 Arm_1;
};

// Encapsulated Union
struct Union_B {
    uint Selector;
    union Union_C member8;
    /* ... */
};

// Re-encapsulated Union
struct Union_A {
    /* ... */
    struct Union_B member4;
};
```

Union in COM

- Customized Union
- Built-in Union
- Non-encapsulated Union
- Encapsulated Union
- Re-encapsulated Union
- VARIANT
- PROPVARIANT
Each analyzed Windows operating system has over 10,000 unions in its COM.

50% of all customized unions in COM contain both pointer and non-pointer members.
Union Type Confusion Example

```c
#define NAME 1
#define ID 2

typedef union MetaInfo {
    char* name;
    int id;
} MetaInfo;

typedef struct UserInfo {
    int type;
    MetaInfo info;
} UserInfo;

void print_user_name(UserInfo user){
    printf("User Name: %s\n", user.info.name);
}

int main() {
                         {NAME,\"Louis\"}, {ID,1002}};
    int s = sizeof(users) / sizeof(UserInfo);
    for (int i = 0; i < s; i++) {
        print_user_name(users[i]);
    }
    return 0;
}
```

Type Confusion occurs when print users[1] and users[3] Integer is interpreted as a string pointer
Introduction to Union Type Confusion

Type confusion occurs: an integer is interpreted as a pointer.

Root cause: The type of union member is not properly checked before being used.
Union Type Confusion in COM (Attack Scenario)

Client-server model.
⇒ Client can pass a union to the server directly when the interface accepts a union parameter.
⇒ Regular clients can trigger server-side high-privilege bugs.
Union Type Confusion in COM (Attack Scenario)

Marshalling process

⇒ Client can not pass arbitrary union descriptor.
⇒ No union type confusion occurs if server use the union member before checking the descriptor.
⇒ Without descriptor verification, servers may face union type confusion.
⇒ We can detect union type confusion by verifying the process's descriptor check accuracy.

Client-server model.

⇒ Client can pass a union to the server directly when the interface accepts a union parameter.
⇒ Regular clients can trigger server-side high-privilege bugs.
Our Contribution

- We, in a study that was the first of its kind, analyzed different forms of unions in Windows COM, and discovered that the extensive use of unions has resulted in the creation of union type confusions. We further showed how such type confusions can be used in the development of exploits.

- We created COMFUSION, a novel framework that systematically breaks down the complex problem of identifying union type confusions in COM binaries into smaller, more manageable sub-problems. Each of these sub-problems can be solved using available techniques, but we have adapted and combined them specifically for COM analysis.

- We analyzed 79,195 COM objects in three popular releases of Windows, i.e., Windows 10 version 1809, Windows 10 version 21H2, and Windows 11 version 21H2 with COMFUSION and successfully found 36 union type confusions. 19 of these type confusions have been confirmed to possess the ability to corrupt memory, exposing 4 confirmed CVEs.
Overview Of COMFusion

1. Extracting COM objects
   - Registry
   - COM Objects
   - Database
   - Extractor
   - MIDL file
   - Binary file
   - Interface Function Table

2. Explore Unions Declaration
   - DAG of customized structures
   - Unions Declaration
   - Discriminants and Legitimate values
   - Selector: \{(0,1,2,3,4}\}
   - Member0->Selector: \{(0,1,2,3,4}\}

3. Locate Union Variables
   - Interface_0
   - Interface_1
   - Taint Propagation
   - COM Server Binaries

4. Identify Union Type Confusion
   - Disassembled Code
   - Symbolic Execution
   - Bugs

0x180099EB3 push r14
0x1802408B5 sub rsp, 20h
//... ...*/

Windows Registry

Registry
...

Interface
Function Table

Binary file
MIDL file

Extracting COM objects

COM Server Binaries

Taint Propagation

Interface_1

Proc6

Proc5

Proc4(a,b)
The locations of the binary files that implement the interface functions are registered in

HKEY_CLASSES_ROOT/CLSID/$CLSID/InprocServer32 OR
HKEY_CLASSES_ROOT/CLSID/$CLSID/LocalServer32.

Exported objects includes:
• MIDL files for each COM interface.
• The COM server binary.
• Interface functions table.
Explore Unions Declarations

1. Extracting COM objects
   - Registry
   - Database
   - Extractor
   - COM Objects
     - MIDL file
     - Binary file
     - Interface and Function Table

2. Explore Unions Declaration
   - DAG of customized structures
   - Unions Declaration
     - Selector: {{0,1,2,3,4}}
     - MemberSelector: {{0,1,2,3,4}}
   - Discriminants and Legitimate values

3. Locate Union Variables
   - Interface_0
   - Proc4(a,b)
   - Proc5
   - Proc6
   - Interface_1
   - Taint Propagation
   - COM Server Binaries

4. Identify Union Type Confusion
   - An exported DAG of customized structures.
     (CLSID:0b2c9183-c9fa-4c53-ae21-c900b0c39965
     IID:0c738a7a-2a1c-11ce-ade5-00aa0044773d)

   - The recovered MIDL file includes all customized structure declaration.
     - struct Struct_20 {
       BSTR Member0;
       int Member8;
       VARIANT Member10;
     }
   - We use DAG(Directed Acyclic Graph) to explore the relationships of all union.
     - Each node represents a structure
     - Edge(u->v) means u includes v.
Locate Union Variables in Binaries

1. Extracting COM objects
2. Explore Unions Declaration
3. Locate Union Variables
4. Identify Union Type Confusion

Taint Propagation
Locate Union Variables in Binaries

1. Extracting COM objects
   - Registry
   - COM Objects
   - MIDL file
   - Binary file
   - Interface Function Table

2. Explore Unions Declaration
   - DAG of customized structures
   - Unions Declaration
   - Discriminants and Legitimate values

3. Locate Union Variables
   - Interface_0
   - Interface_1
   - Taint Propagation

4. Identify Union Type Confusion
   - Disassembled Code
   - Symbolic Execution
   - Bugs

Taint Propagation

Taint Source:

Sensitive Interface Function

Taint Specification:

- Two kinds of taints:
  - discriminant
  - union member
- Function call:
  - summarized function
  - internal function
  - external function

Avoid Path explosion:

- LOOP_THOLD
- CALL_THOLD
- TOTAL_THOLD
Checking strategies:

Calculate **possible values** of union descriptor when union member is used

Symbolic Execution
Identify Union Type Confusion

Checking strategies:

Calculate possible values of union descriptor when union member is used

Initialization

• Interface Function Arguments
• The this Object

Execution Strategies

• Along with taint propagation
• Prune safe branch

How to define “use”

• Parameters of not analyzed functions.
• Target of memory access
• Operand of comparison and arithmetic instruction.
RQ1: How effective is COMFUSION in analyzing off the shelf COM binaries for sensitive interface functions?

Among 79,195 COM objects in 3 popular releases of Windows, we totally find 3940 sensitive interface functions.

<table>
<thead>
<tr>
<th>Platform</th>
<th>#COMs</th>
<th>#Bins</th>
<th>#Funcs</th>
<th>#Intfs Funcs</th>
<th>#Sens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win10 1809</td>
<td>26929</td>
<td>1316</td>
<td>1945915</td>
<td>62555</td>
<td>1801</td>
</tr>
<tr>
<td>Win10 21H2</td>
<td>26124</td>
<td>1241</td>
<td>2028735</td>
<td>60326</td>
<td>1728</td>
</tr>
<tr>
<td>Win11 21H2</td>
<td>26142</td>
<td>1305</td>
<td>2461951</td>
<td>60849</td>
<td>411</td>
</tr>
</tbody>
</table>

Statistics of COM objects and sensitive functions exported by COMFusion
• RQ2: How precisely can COMFUSION identify union type confusions?

We have totally found 78 Union Type Confusions with 42 false positives. The 36 true positives include 19 Confusion of Pointers (CoP) and 17 Confusion of Non-Pointers (CoNP).

<table>
<thead>
<tr>
<th>Platform</th>
<th>#UC</th>
<th>#FP</th>
<th>#FDR</th>
<th>#FPS</th>
<th>#TP</th>
<th>#TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win10 1809</td>
<td>38</td>
<td>18</td>
<td>47.4%</td>
<td>1(FP_I)</td>
<td>10(FP_II)</td>
<td>11(CoP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7(FP_III)</td>
<td></td>
</tr>
<tr>
<td>Win10 21H2</td>
<td>31</td>
<td>17</td>
<td>54.9%</td>
<td>0(FP_I)</td>
<td>12(FP_II)</td>
<td>6(CoP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5(FP_III)</td>
<td></td>
</tr>
<tr>
<td>Win11 21H2</td>
<td>9</td>
<td>7</td>
<td>77.8%</td>
<td>0(FP_I)</td>
<td>7(FP_II)</td>
<td>2(CoP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0(FP_III)</td>
<td></td>
</tr>
</tbody>
</table>

Statistics of Union Type Confusion discovered by COMFusion
Evaluation (3/4)

- RQ3: If there are false positives, how are they generated?

- Type I: [In, Out] Only ‘Write’ but No ‘Read’.

- Type II: Mismatch in the Number of Function Arguments.

- Type III: Discriminant Checking Affected by Wrongly-assigned Symbolic Variable

Example of Type III false positive.
- RQ4: How dangerous are those union type confusion bugs? Can they cause severe damages?

<table>
<thead>
<tr>
<th>Affected Applications or Binaries</th>
<th>Windows Version</th>
<th>Function Name</th>
<th>Impact</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 UPnPHost service</td>
<td>Windows 10 1809</td>
<td>OnXXXEdSafe</td>
<td>Elevation of Privilege</td>
<td>CVE-2020-1519</td>
</tr>
<tr>
<td>2 WalletService</td>
<td>Windows 10 1809</td>
<td>WaAXXXXPropertyValue</td>
<td>Elevation of Privilege</td>
<td>CVE-2021-26871</td>
</tr>
<tr>
<td>3 Diagnostic Execution Service</td>
<td>Windows 10 1809</td>
<td>ComXXXents</td>
<td>Elevation of Privilege</td>
<td>CVE-2020-1393</td>
</tr>
<tr>
<td>4 Diagnostic Execution Service</td>
<td>Windows 10 1809</td>
<td>GetXXXdates</td>
<td>Elevation of Privilege</td>
<td>CVE-2020-1130</td>
</tr>
<tr>
<td>5 UPnPHost service</td>
<td>Windows 10 1809</td>
<td>HrQ XXXXible</td>
<td>Elevation of Privilege</td>
<td>Confirmed</td>
</tr>
<tr>
<td>6-7 ieframe.dll (two CLSIDs)</td>
<td>Windows 10 1809</td>
<td>NaAXXXXBindCtx</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>8 ieframe.dll</td>
<td>Windows 10 1809</td>
<td>CDXXXXXexc(Line 74)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>9 ieframe.dll</td>
<td>Windows 10 1809</td>
<td>CDXXXXXexc(Line 75)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>10 ieframe.dll</td>
<td>Windows 10 1809</td>
<td>_CXXXXXDialog</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>11 exploreframe.dll</td>
<td>Windows 10 1809</td>
<td>SHXXXXbject</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>12-13 ieframe.dll (two CLSIDs)</td>
<td>Windows 10 21H2</td>
<td>NaAXXXXBindCtx</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>14 ieframe.dll</td>
<td>Windows 10 21H2</td>
<td>CDXXXXXexc(Line 74)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>15 ieframe.dll</td>
<td>Windows 10 21H2</td>
<td>CDXXXXXexc(Line 75)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>16 ieframe.dll</td>
<td>Windows 10 21H2</td>
<td>_CXXXXXDialog</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>17 ieframe.dll</td>
<td>Windows 10 21H2</td>
<td>CDoXXXxView</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>18 WMSPDMO.DLL</td>
<td>Windows 11 21H2</td>
<td>CWXXXXXrite(Line 104)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
<tr>
<td>19 WMSPDMO.DLL</td>
<td>Windows 11 21H2</td>
<td>CWXXXXXrite(Line 139)</td>
<td>Denial of Service</td>
<td>Confirmed</td>
</tr>
</tbody>
</table>

Confusion of Pointers discovered by COMFusion
We proposed COMFUSION, the first tool for discovering union type confusion vulnerabilities in Windows COM.

COMFUSION applied taint analysis and symbolic execution based on MIDL files to identify union type confusions in COM objects.

COMFUSION analyzed 79,195 COM objects and discovered 36 union type confusions, of which four that run in high privilege services are now given four CVE identifiers.
Thanks for listening!

Q&A

Contact: Yuxing Zhang, 52194501006@stu.ecnu.edu.cn