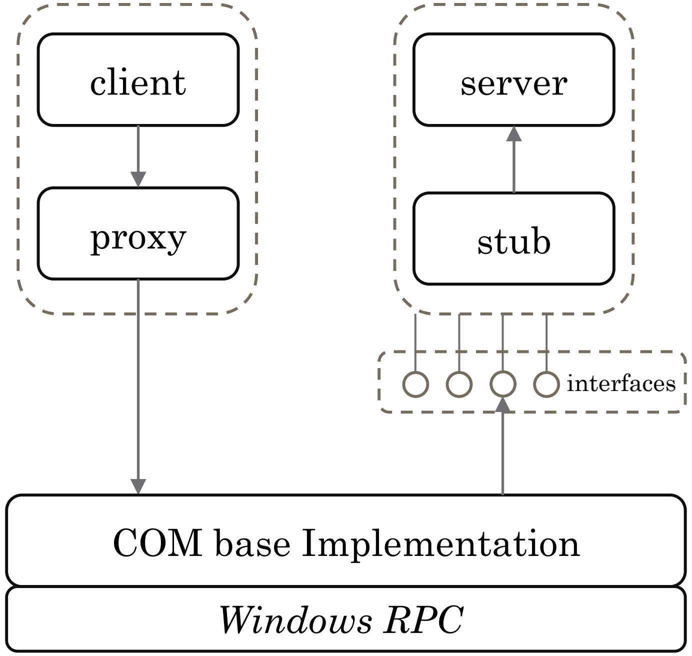
Detecting Union Type Confusion in Component Object Model

Yuxing Zhang[1], Xiaogang Zhu[2], Daojing He†[1,3], Minhui Xue[4], Shouling Ji[5], Mohammad Sayad Haghighi[2], Sheng Wen[2], and Zhiniang Peng[6]

[1]*East China Normal University,* [2]*Swinburne University of Technology,* [3]*Harbin Institute of Technology, Shenzhen,* [4]*CSIRO's Data61,* [5]*Zhejiang University,* [6]*Sangfor Technologies Inc.*

Introduction To COM(Component Object Model)

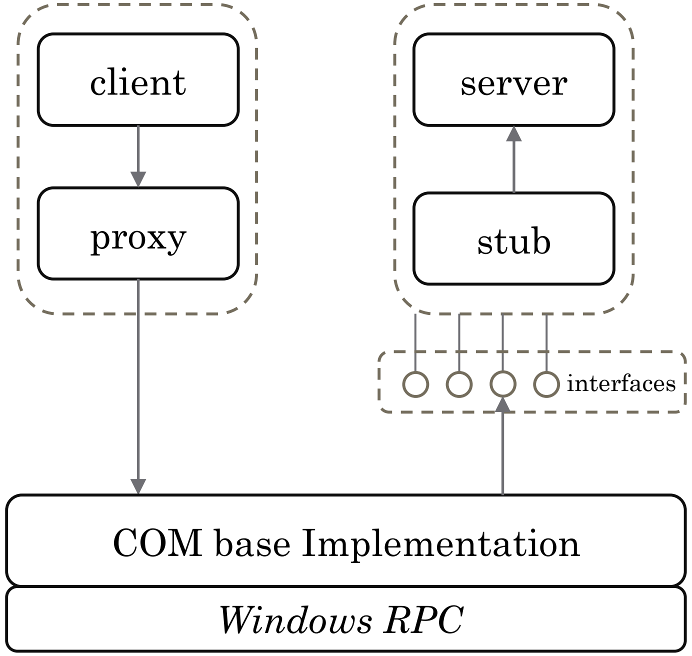
- The Microsoft Component Object Model (COM) defines a binary interoperability standard for creating reusable software libraries that interact at run time.



Ref：https://learn.microsoft.com/en-us/windows/win32/com/com-technical-overview

Introduction To COM(Component Object Model)

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

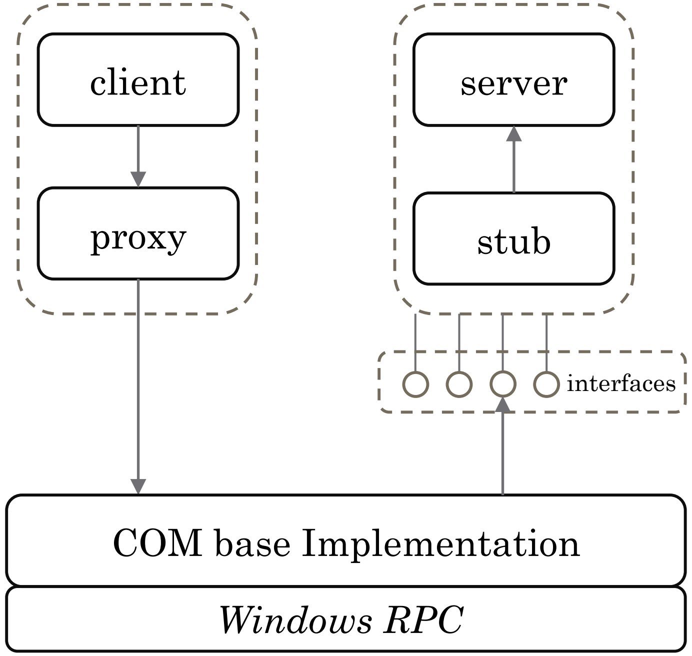


COM is the foundation for many other technologies , such as Microsoft's OLE and Active, Windows Runtime.

Ref：https://learn.microsoft.com/en-us/windows/win32/com/com-technical-overview

Introduction To COM(Component Object Model)

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

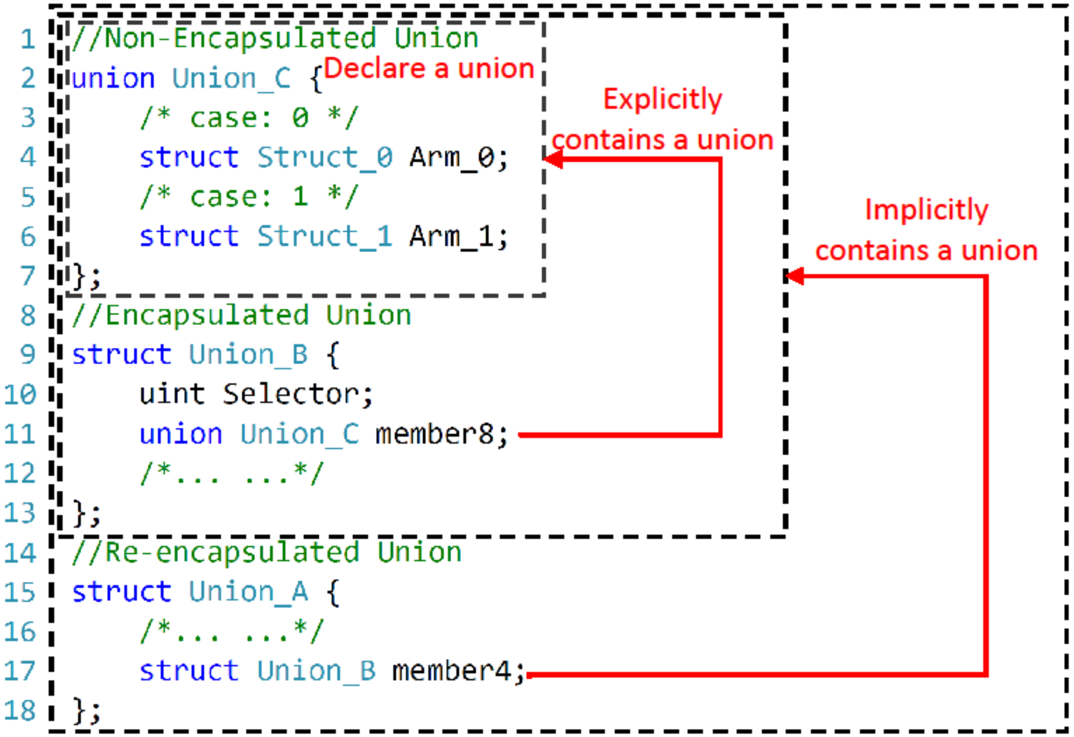


COM is the foundation for many other technologies , such as Microsoft's OLE and Active, Windows Runtime.

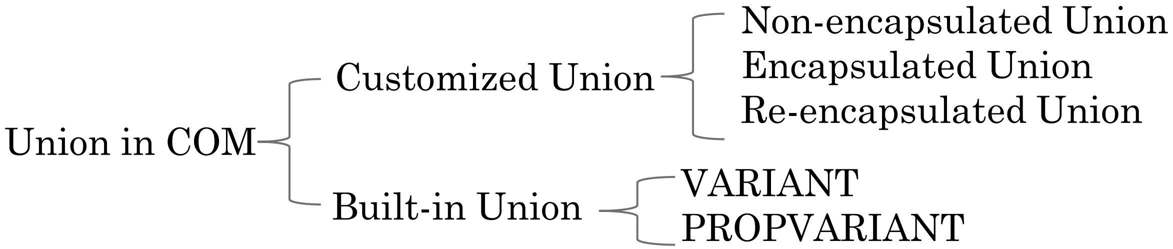
Many COM server run in high privilege, union type confusion in COM has the potential to be used in the development of 100%-reliable exploits

Ref： https://learn.microsoft.com/en-us/windows/win32/com/com-technical-overview

Union In COM

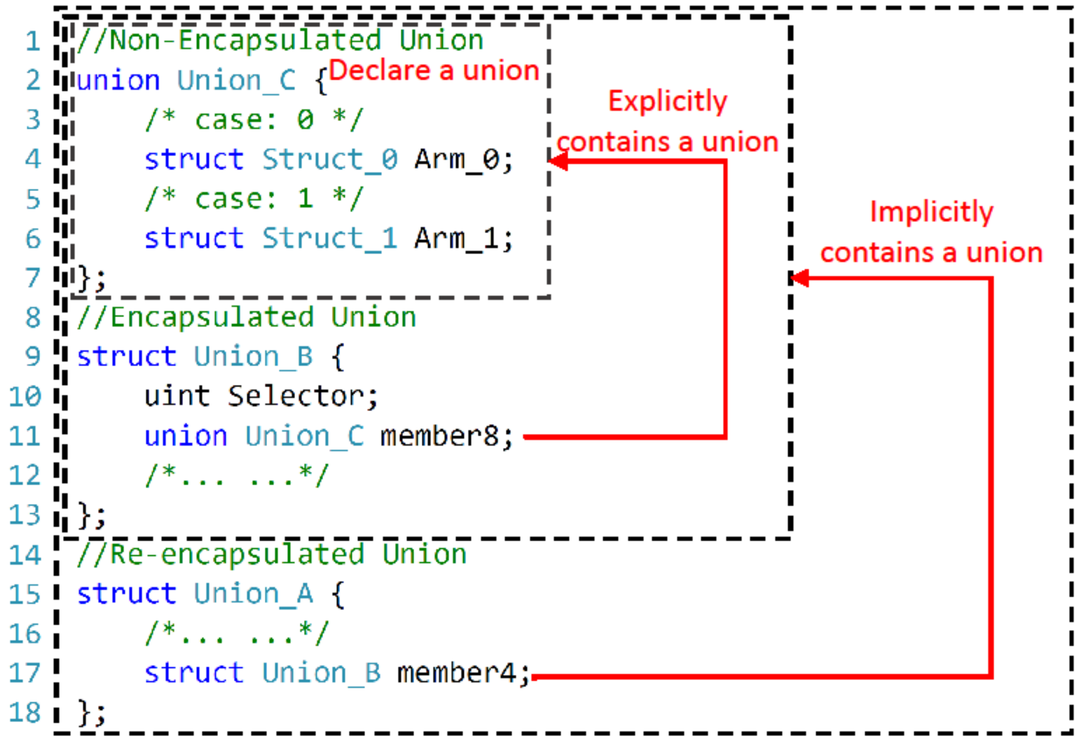


```
//Non-Encapsulated Union
union Union_C {
    /* case: 0 */
    struct Struct_0 Arm_0;
    /* case: 1 */
    struct Struct_1 Arm_1;
};
//Encapsulated Union
struct Union_B {
    uint Selector;
    union Union_C member8;
    /*... ...*/
};
//Re-encapsulated Union
struct Union_A {
    /*... ...*/
    struct Union_B member4;
};
```

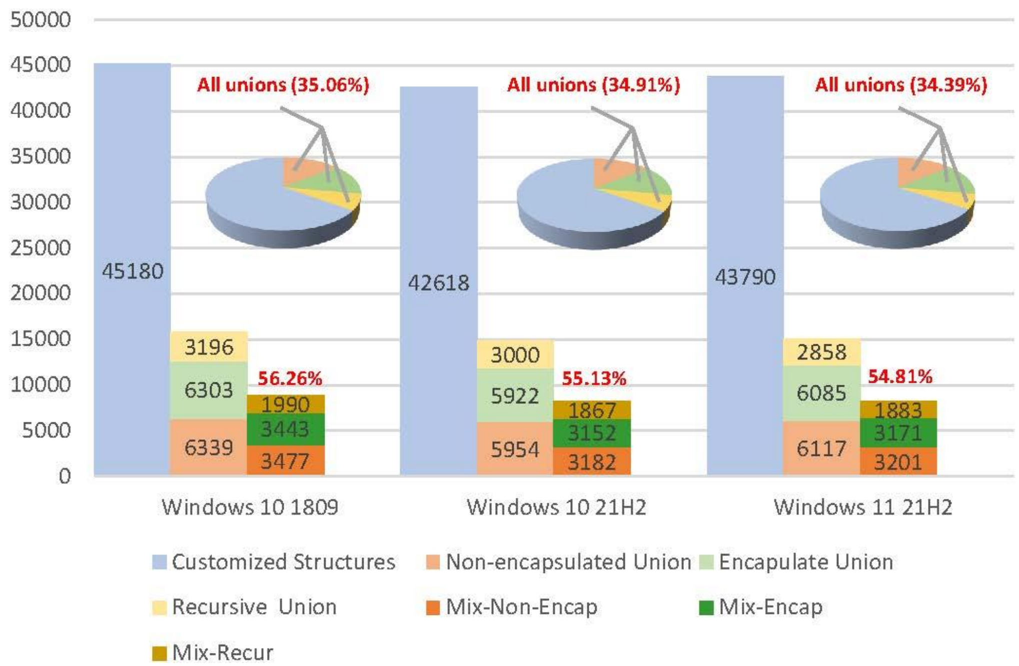


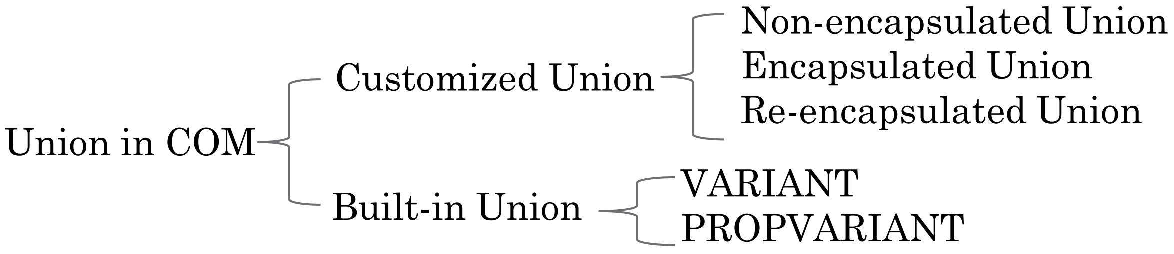
- Union in COM
  - Customized Union
    - Non-encapsulated Union
    - Encapsulated Union
    - Re-encapsulated Union
  - Built-in Union
    - VARIANT
    - PROPVARIANT

Union In COM



```
//Non-Encapsulated Union
union Union_C {
    /* case: 0 */
    struct Struct_0 Arm_0;
    /* case: 1 */
    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;
};
```



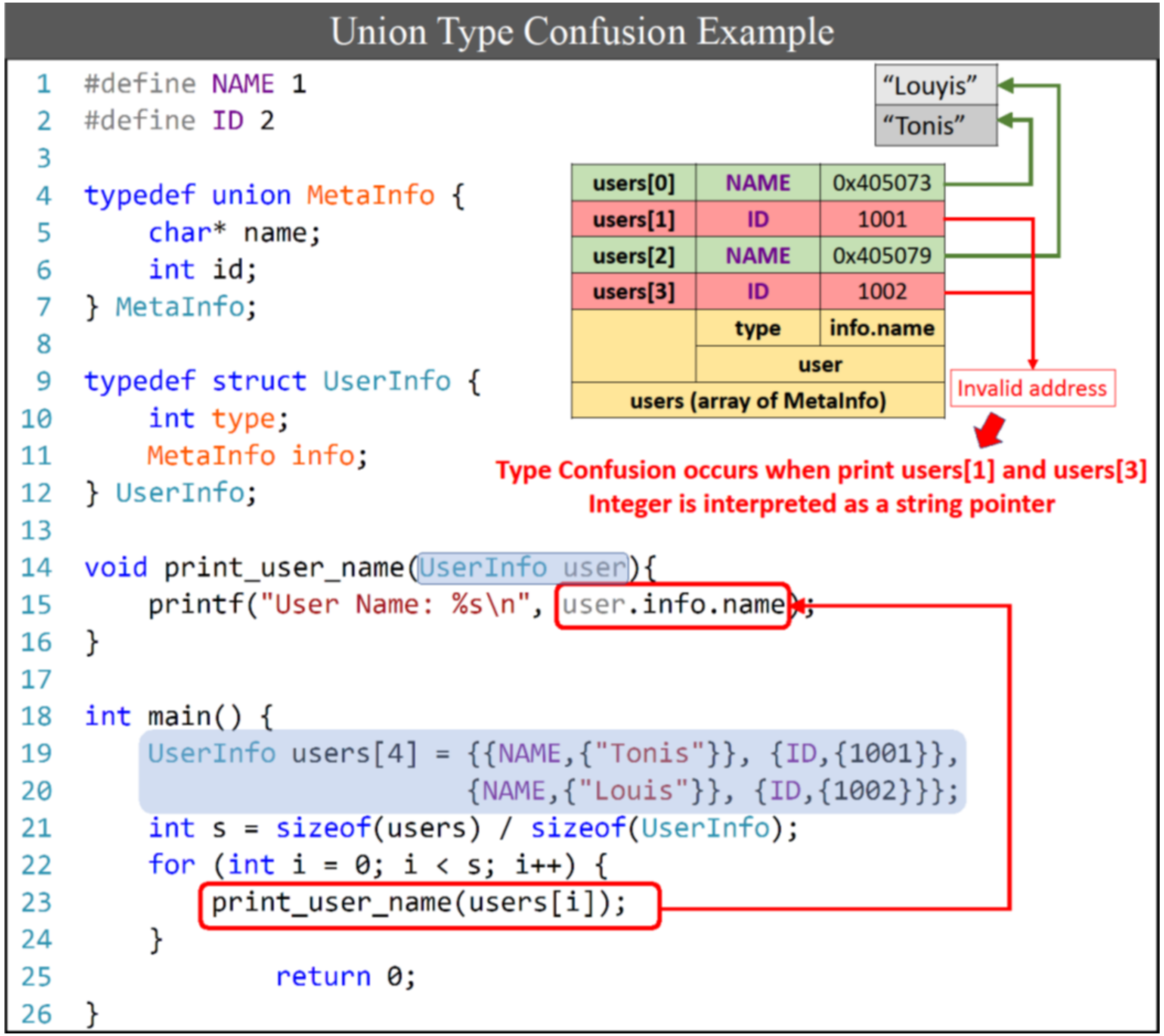


Union in COM
- Customized Union
  - Non-encapsulated Union
  - Encapsulated Union
  - Re-encapsulated Union
- Built-in Union
  - VARIANT
  - PROPVARIANT

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



Union Type Confusion Example

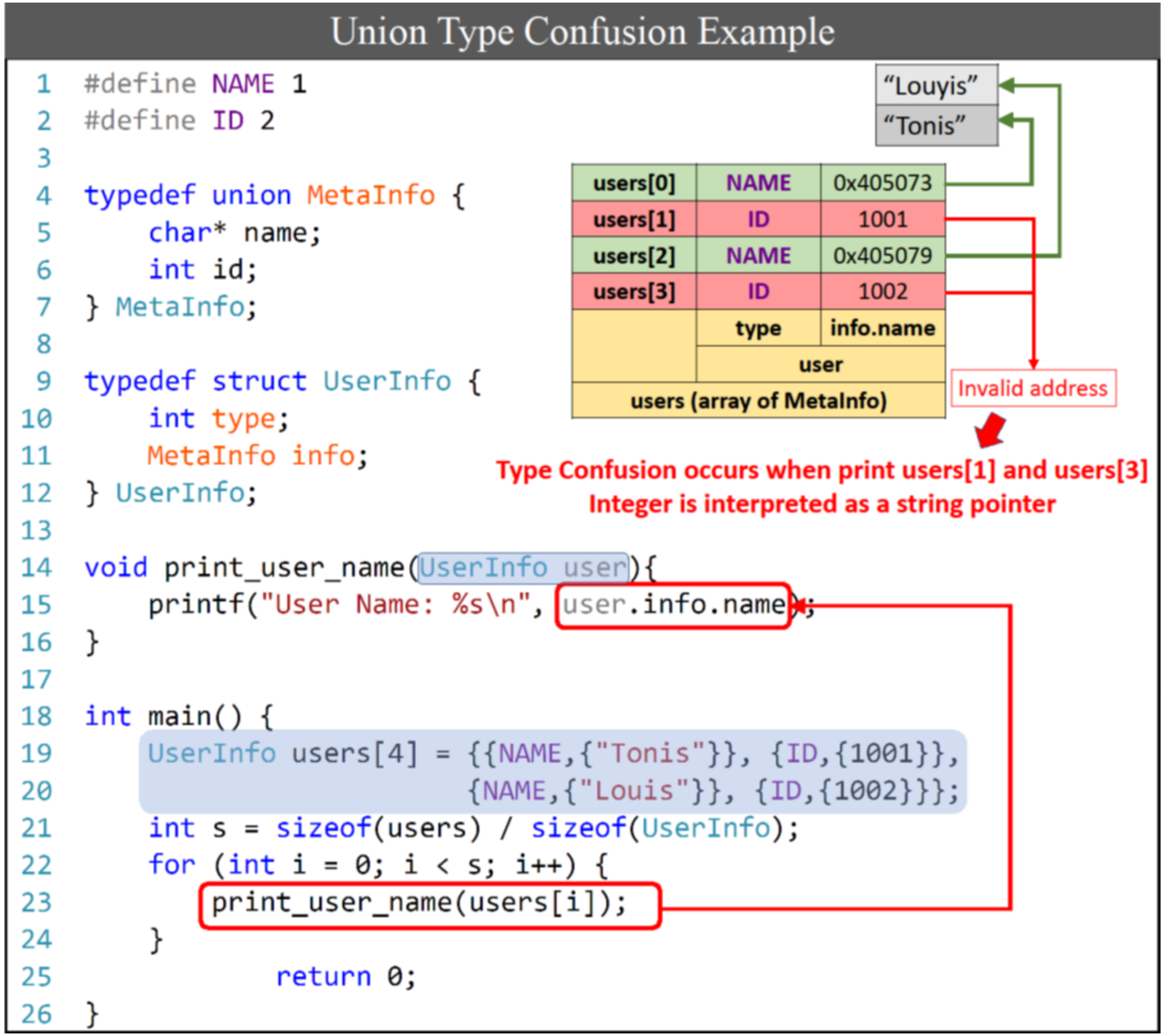
```c
#define NAME 1
#define ID 2

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() {
    UserInfo users[4] = {{NAME,{"Tonis"}}, {ID,{1001}},
                         {NAME,{"Louis"}}, {ID,{1002}}};
    int s = sizeof(users) / sizeof(UserInfo);
    for (int i = 0; i < s; i++) {
        print_user_name(users[i]);
    }
            return 0;
}
```

users[0]	NAME	0x405073
users[1]	ID	1001
users[2]	NAME	0x405079
users[3]	ID	1002
	type	info.name
	user	
users (array of MetaInfo)		

Type Confusion occurs when print users[1] and users[3]

Integer is interpreted as a string pointer

Introduction to Union Type Confusion



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() {
    UserInfo users[4] = {{NAME,{"Tonis"}}, {ID,{1001}},
                        {NAME,{"Louis"}}, {ID,{1002}}};
    int s = sizeof(users) / sizeof(UserInfo);
    for (int i = 0; i < s; i++) {
        print_user_name(users[i]);
    }
            return 0;
}
```

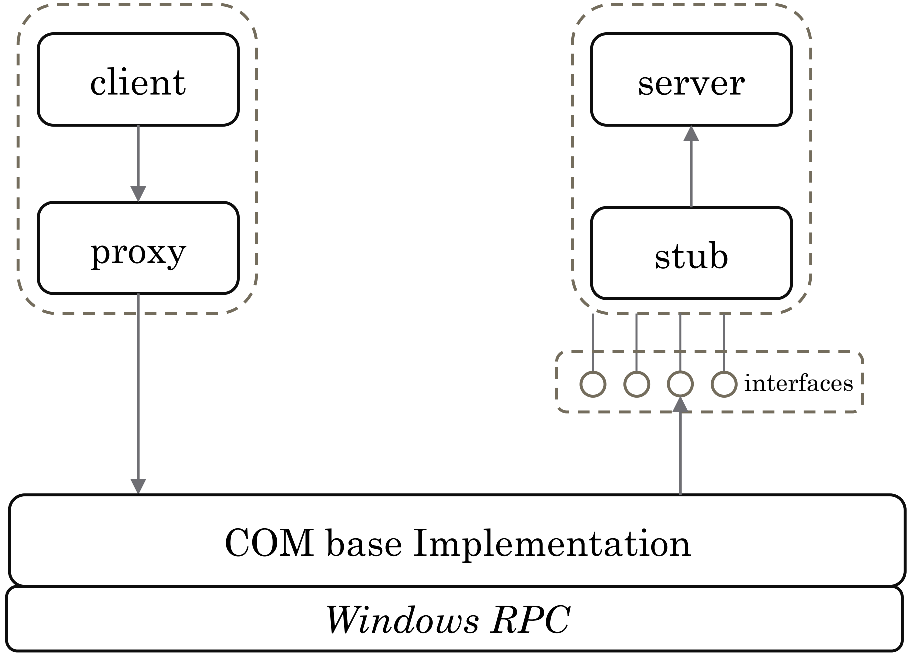
users[0]	NAME	0x405073
users[1]	ID	1001
users[2]	NAME	0x405079
users[3]	ID	1002
	type	info.name
	user	
users (array of MetaInfo)		

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

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)

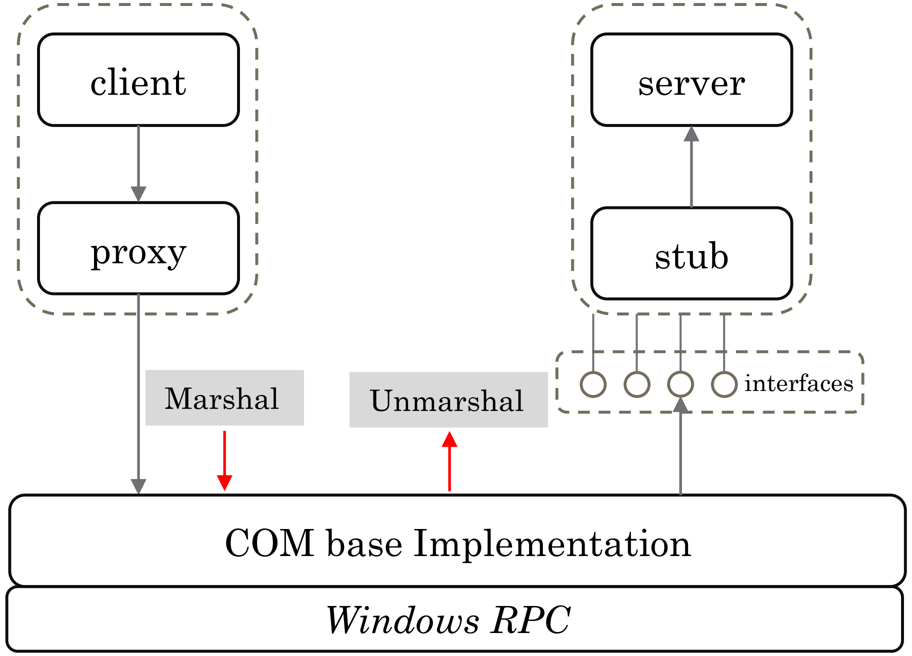


Client-server model.

⇒ 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)



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.

Marshalling process

⇒ Client can not pass arbitrary union descriptor.

⇒ No union type confusion occurs if server use the union member before checking the descriptor.

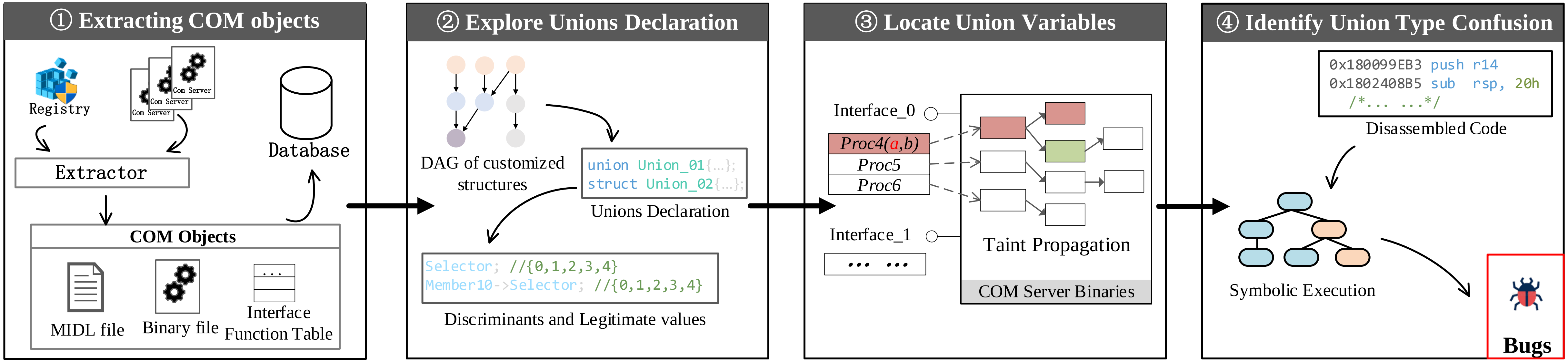
⇒ Without descriptor verification, servers may face union type confusion.

⇒ We can detect union type confusion by verifying the process's descriptor check accuracy.

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

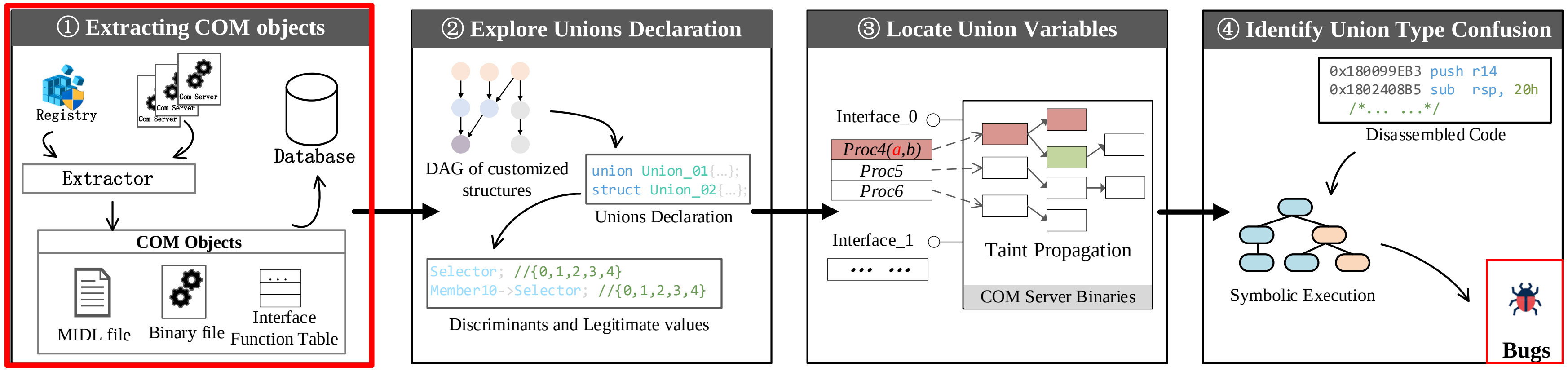


```
union Union_01{...};
struct Union_02{...};
```

```
Selector; //{0,1,2,3,4}
Member10->Selector; //{0,1,2,3,4}
```

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

Extract COM Objects



```
union Union_01{...};
struct Union_02{...};
```

```
Selector; //{0,1,2,3,4}
Member10->Selector; //{0,1,2,3,4}
```

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

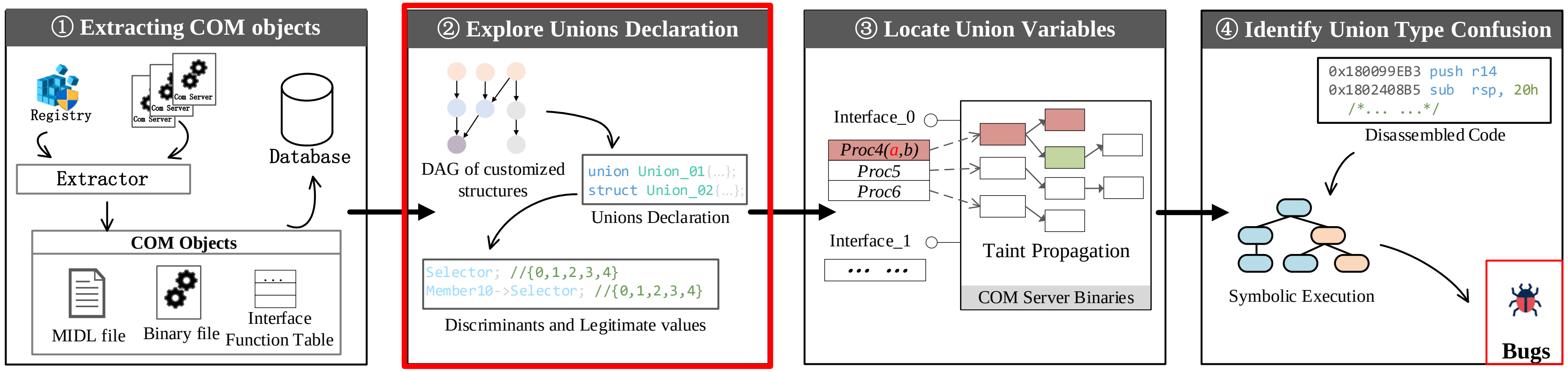
- The locations of the binary files that implement the interface functions are registered in *HKEY_CLASSES_ROOT/CLSID/$CLSID/InprocServer32* or *HKEY_CLASSES_ROOT/CLSID/$CLSID/LocalServer32*.

- Exported objects includes:
  - MIDL files for each COM interface.
  - The COM server binary.
  - Interface functions table.

```
struct Struct_20 {
    BSTR Member0;
    int Member8;
   VARIANT Member10;
};
[Guid("204810b4-73b2-11d4-bf42-00b0d0118b56")]
interface IUPnPEventSink : IUnknown {
   HRESULT Proc3([In] int p0, [In] int[] p1);
   HRESULT Proc4([In] VARIANT* p0);
   HRESULT Proc5([In] Struct_20* p0);
}
```

An example of exported MIDL file.

Explore Unions Declarations



① Extracting COM objects

Registry

Com Server

Extractor

Database

COM Objects

MIDL file

Binary file

Interface Function Table

② Explore Unions Declaration

DAG of customized structures

```
union Union_01{...};
struct Union_02{...};
```

Unions Declaration

```
Selector; //{0,1,2,3,4}
Member10->Selector; //{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

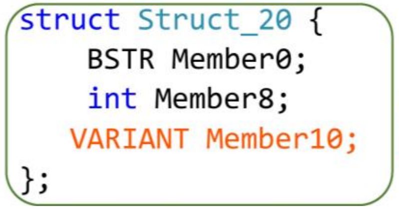
```
0x180099EB3 push r14
0x1802408B5 sub  rsp, 20h
  /*... ...*/
```

Disassembled Code

Symbolic Execution

Bugs

- The recovered MIDL file includes all customized structure declaration.

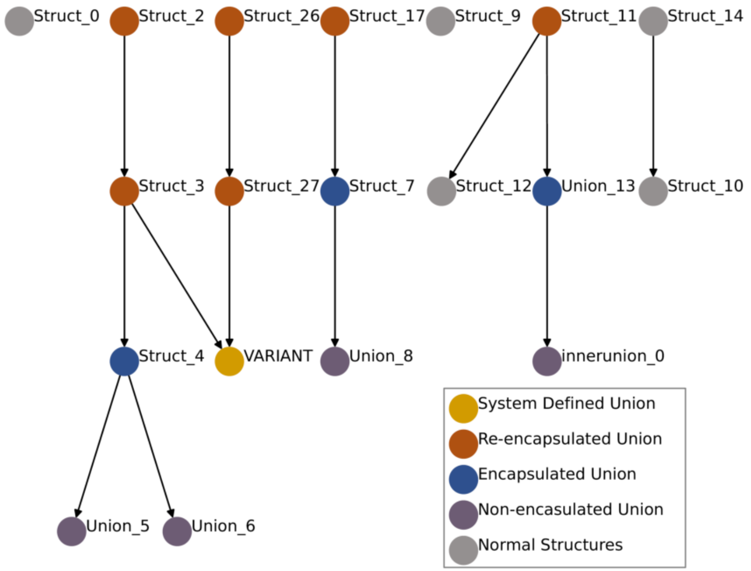


```
struct Struct_20 {
    BSTR Member0;
    int Member8;
   VARIANT Member10;
};
```

Customized Structure

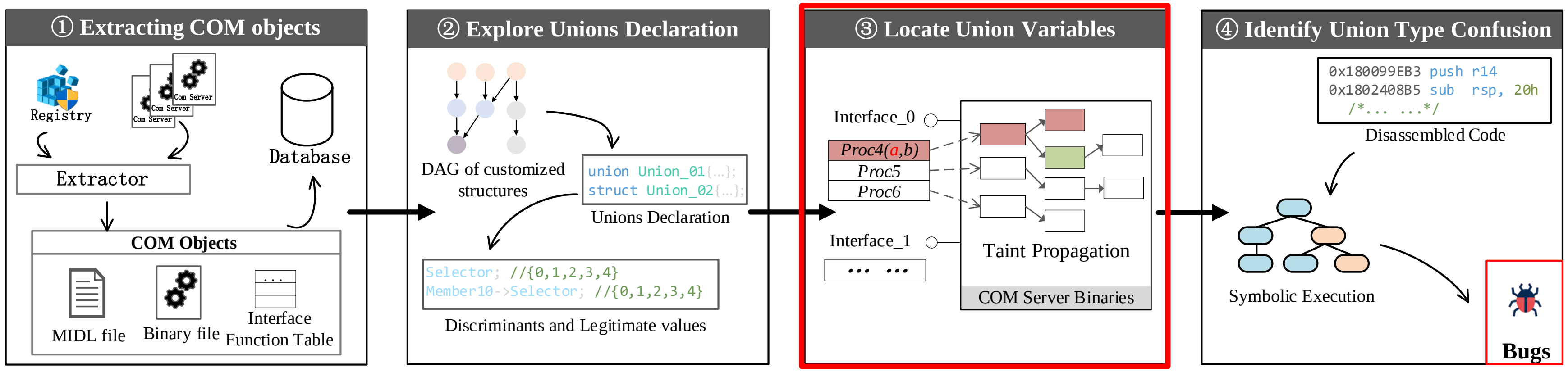
Re-encapsulated union

- 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.



An exported DAG of customized structures.
(CLSID:0b2c9183-c9fa-4c53-ae21-c900b0c39965 IID:0c733a8a-2a1c-11ce-ade5-00aa0044773d)

Locate Union Variables in Binaries



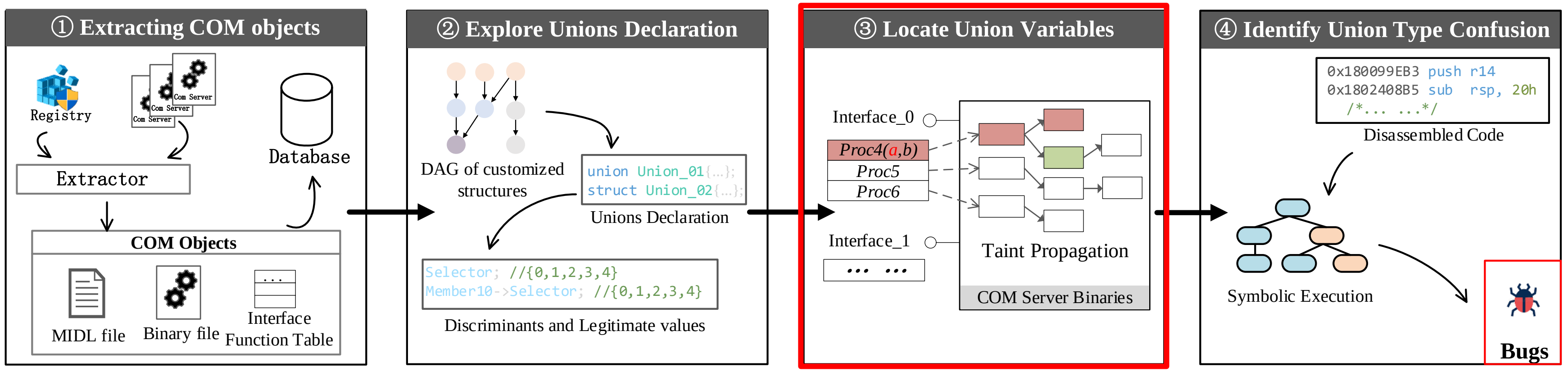
```
union Union_01{...};
struct Union_02{...};
```

```
Selector; //{0,1,2,3,4}
Member10->Selector; //{0,1,2,3,4}
```

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

Taint Propagation

Locate Union Variables in Binaries



Taint Propagation

Taint Source:

Sensitive Interface Function

```
[Guid("204810b4-73b2-11d4-bf42-00b0d0118b56")]
interface IUPnPEventSink : IUnknown {
   HRESULT Proc3([In] int p0, [In] int[] p1);
   HRESULT Proc4([In] VARIANT* p0);
   HRESULT Proc5([In] Struct_20* p0);
}
```

Potentially vulnerable

Supported Interface Function

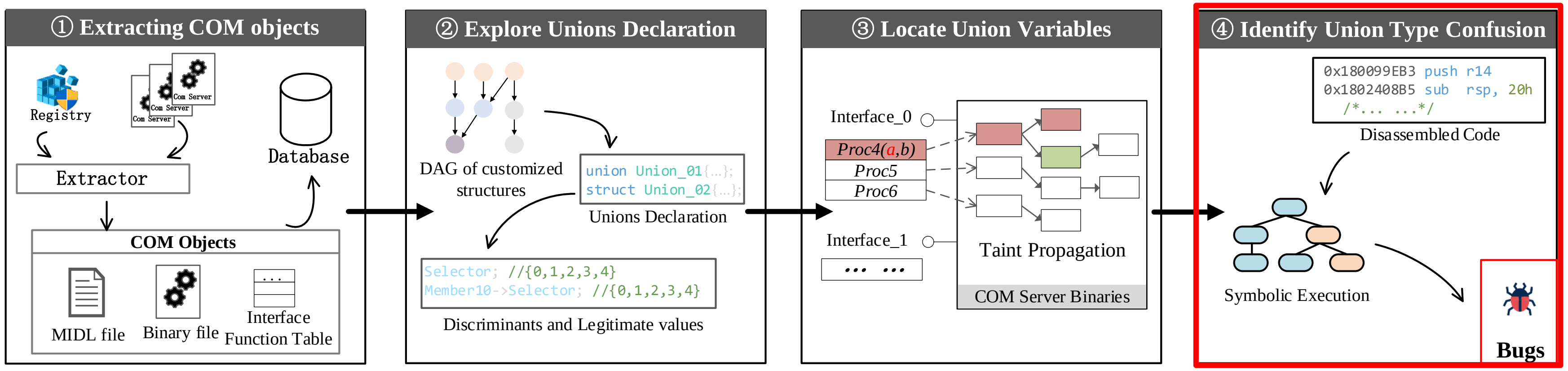
Taint Specification:

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

Avoid Path explosion:

- LOOP_THOLD
- CALL_THOLD
- TOTAL_THOLD

Identify Union Type Confusion

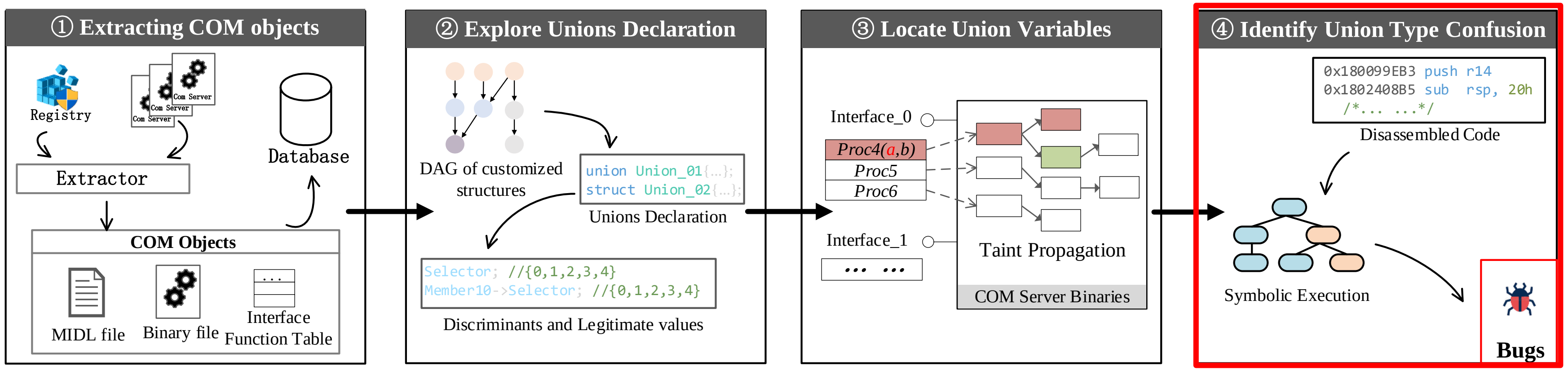


Checking strategies:

Calculate possible values of union descriptor when union member is used

→ Symbolic Execution

Identify Union Type Confusion



```
union Union_01{...};
struct Union_02{...};
```

```
Selector; //{0,1,2,3,4}
Member10->Selector; //{0,1,2,3,4}
```

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

Checking strategies:

Calculate possible values of union descriptor when union member is used

Symbolic Execution

Initialization

- Interface Function Arguments
- The **this** Object

Execution Strategies

- Along with taint propagation
- Prune safe branch

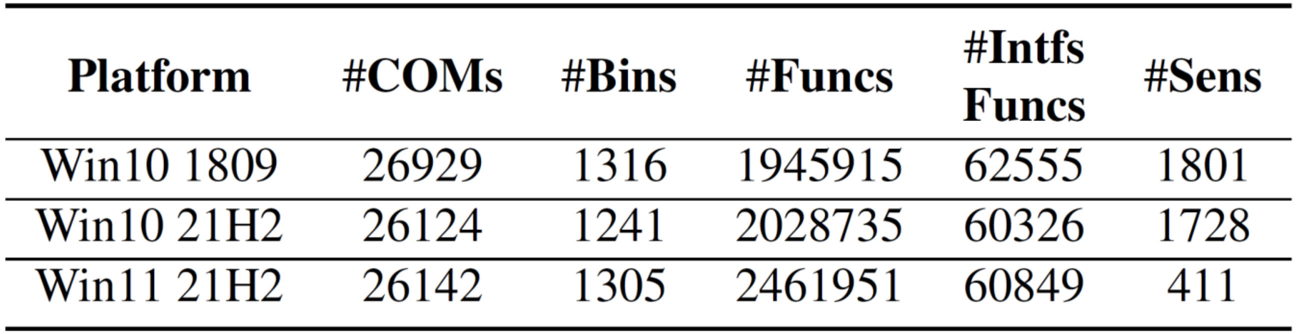
How to define "use"

- Parameters of not analyzed functions.
- Target of memory access
- Operand of comparison and arithmetic instruction.

Evaluation (1/4)

- 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.



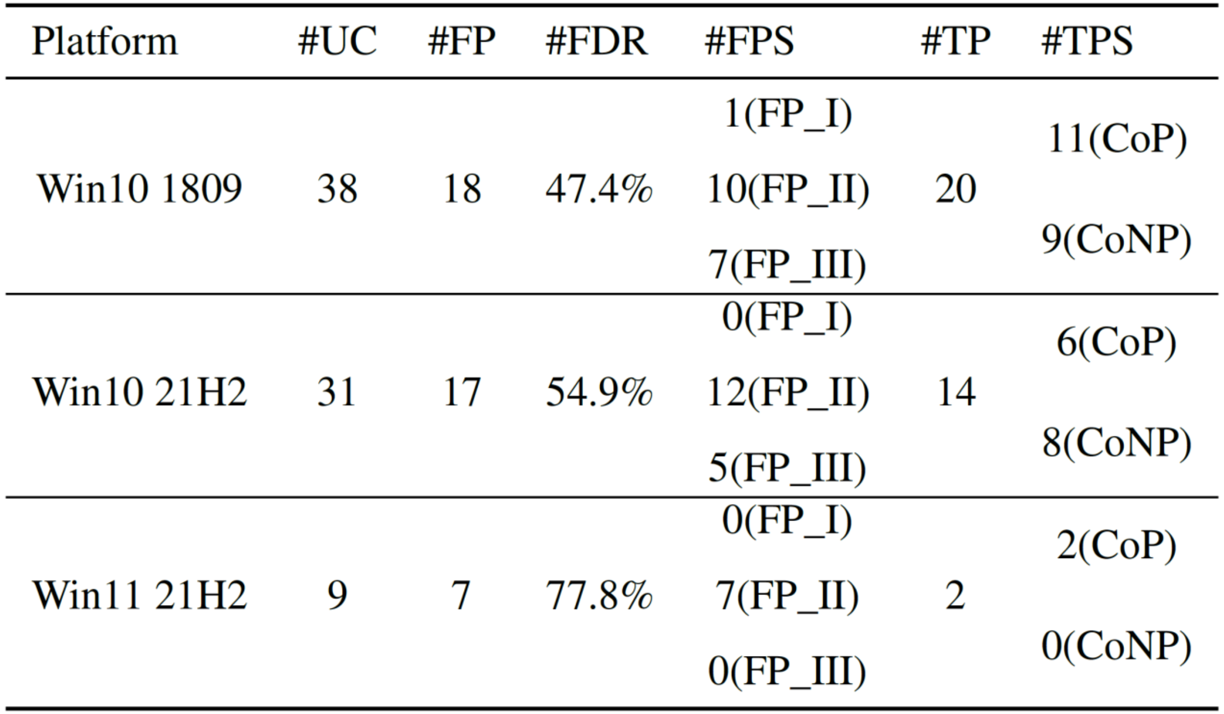
Platform	#COMs	#Bins	#Funcs	#Intfs Funcs	#Sens
Win10 1809	26929	1316	1945915	62555	1801
Win10 21H2	26124	1241	2028735	60326	1728
Win11 21H2	26142	1305	2461951	60849	411

Statistics of COM objects and sensitive functions exported by COMFusion

Evaluation (2/4)

- 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).



Platform	#UC	#FP	#FDR	#FPS	#TP	#TPS

Win10 1809	38	18	47.4%	1(FP_I)
				10(FP_II)
				7(FP_III)
					20	11(CoP)
						9(CoNP)

Win10 21H2	31	17	54.9%	0(FP_I)
				12(FP_II)
				5(FP_III)
					14	6(CoP)
						8(CoNP)

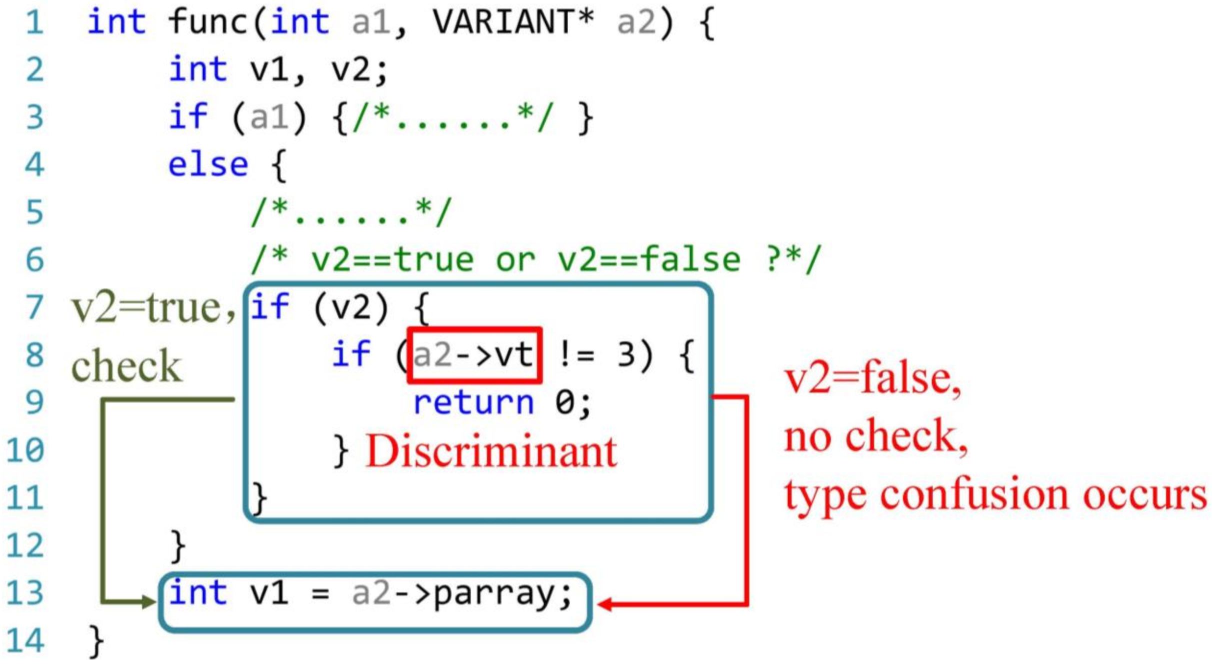
Win11 21H2	9	7	77.8%	0(FP_I)
				7(FP_II)
				0(FP_III)
					2	2(CoP)
						0(CoNP)

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

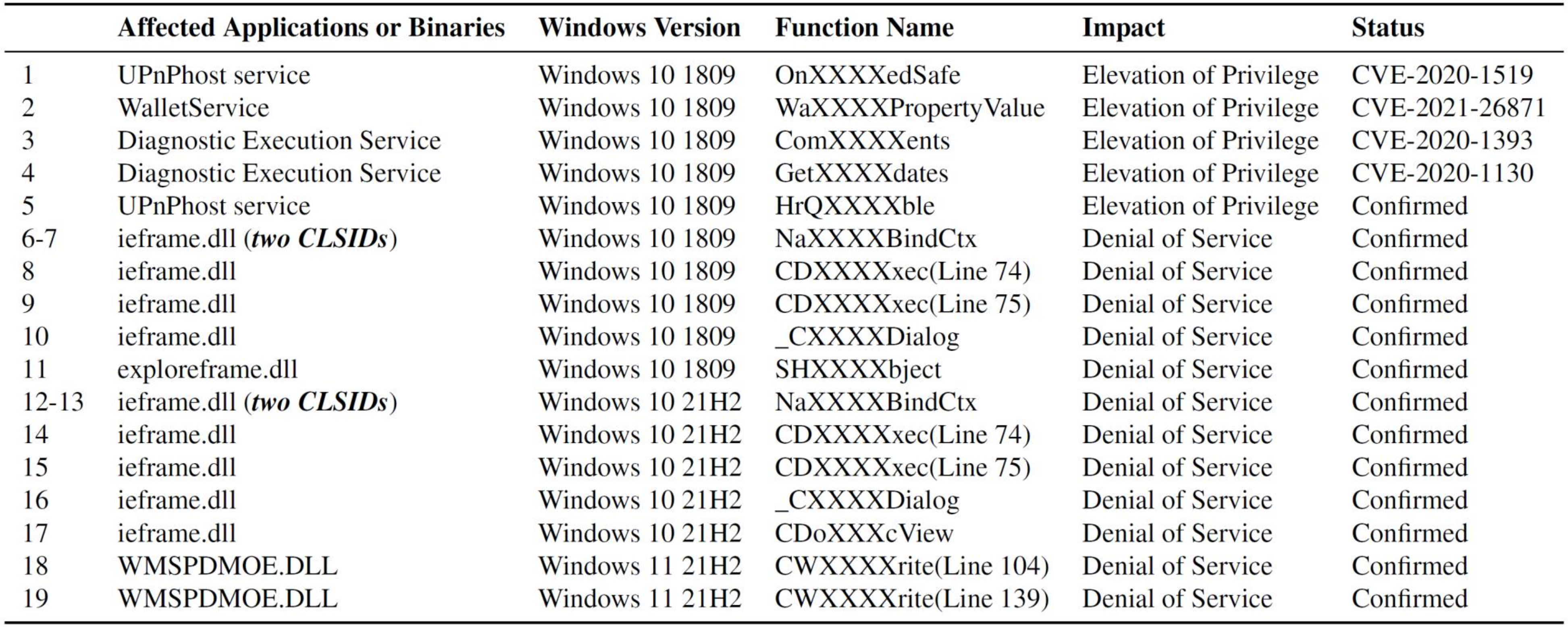


```
int func(int a1, VARIANT* a2) {
    int v1, v2;
    if (a1) {/*......*/ }
    else {
        /*......*/
        /* v2==true or v2==false ?*/
        if (v2) {
            if (a2->vt != 3) {
                return 0;
            }
        }
    }
    int v1 = a2->parray;
}
```

Example of Type III false positive.

Evaluation (4/4)

- RQ4: How dangerous are those union type confusion bugs? Can they cause severe damages?



	Affected Applications or Binaries	Windows Version	Function Name	Impact	Status
1	UPnPhost service	Windows 10 1809	OnXXXXedSafe	Elevation of Privilege	CVE-2020-1519
2	WalletService	Windows 10 1809	WaXXXXPropertyValue	Elevation of Privilege	CVE-2021-26871
3	Diagnostic Execution Service	Windows 10 1809	ComXXXXents	Elevation of Privilege	CVE-2020-1393
4	Diagnostic Execution Service	Windows 10 1809	GetXXXXdates	Elevation of Privilege	CVE-2020-1130
5	UPnPhost service	Windows 10 1809	HrQXXXXble	Elevation of Privilege	Confirmed
6-7	ieframe.dll (***two CLSIDs***)	Windows 10 1809	NaXXXXBindCtx	Denial of Service	Confirmed
8	ieframe.dll	Windows 10 1809	CDXXXXxec(Line 74)	Denial of Service	Confirmed
9	ieframe.dll	Windows 10 1809	CDXXXXxec(Line 75)	Denial of Service	Confirmed
10	ieframe.dll	Windows 10 1809	_CXXXXDialog	Denial of Service	Confirmed
11	exploreframe.dll	Windows 10 1809	SHXXXXbject	Denial of Service	Confirmed
12-13	ieframe.dll (***two CLSIDs***)	Windows 10 21H2	NaXXXXBindCtx	Denial of Service	Confirmed
14	ieframe.dll	Windows 10 21H2	CDXXXXxec(Line 74)	Denial of Service	Confirmed
15	ieframe.dll	Windows 10 21H2	CDXXXXxec(Line 75)	Denial of Service	Confirmed
16	ieframe.dll	Windows 10 21H2	_CXXXXDialog	Denial of Service	Confirmed
17	ieframe.dll	Windows 10 21H2	CDoXXXcView	Denial of Service	Confirmed
18	WMSPDMOE.DLL	Windows 11 21H2	CWXXXXrite(Line 104)	Denial of Service	Confirmed
19	WMSPDMOE.DLL	Windows 11 21H2	CWXXXXrite(Line 139)	Denial of Service	Confirmed

Confusion of Pointers discovered by COMFusion

Conclusion

- 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