Windows Exploit Techniques

Atum

About me

- Atum
  - @blue-lotus
  - @Tea Deliverers
  - @Peking University
- Keywords
  - Software Security, System Security
  - CTF PWNer, Weak Chicken
- lgcpku@gmail.com

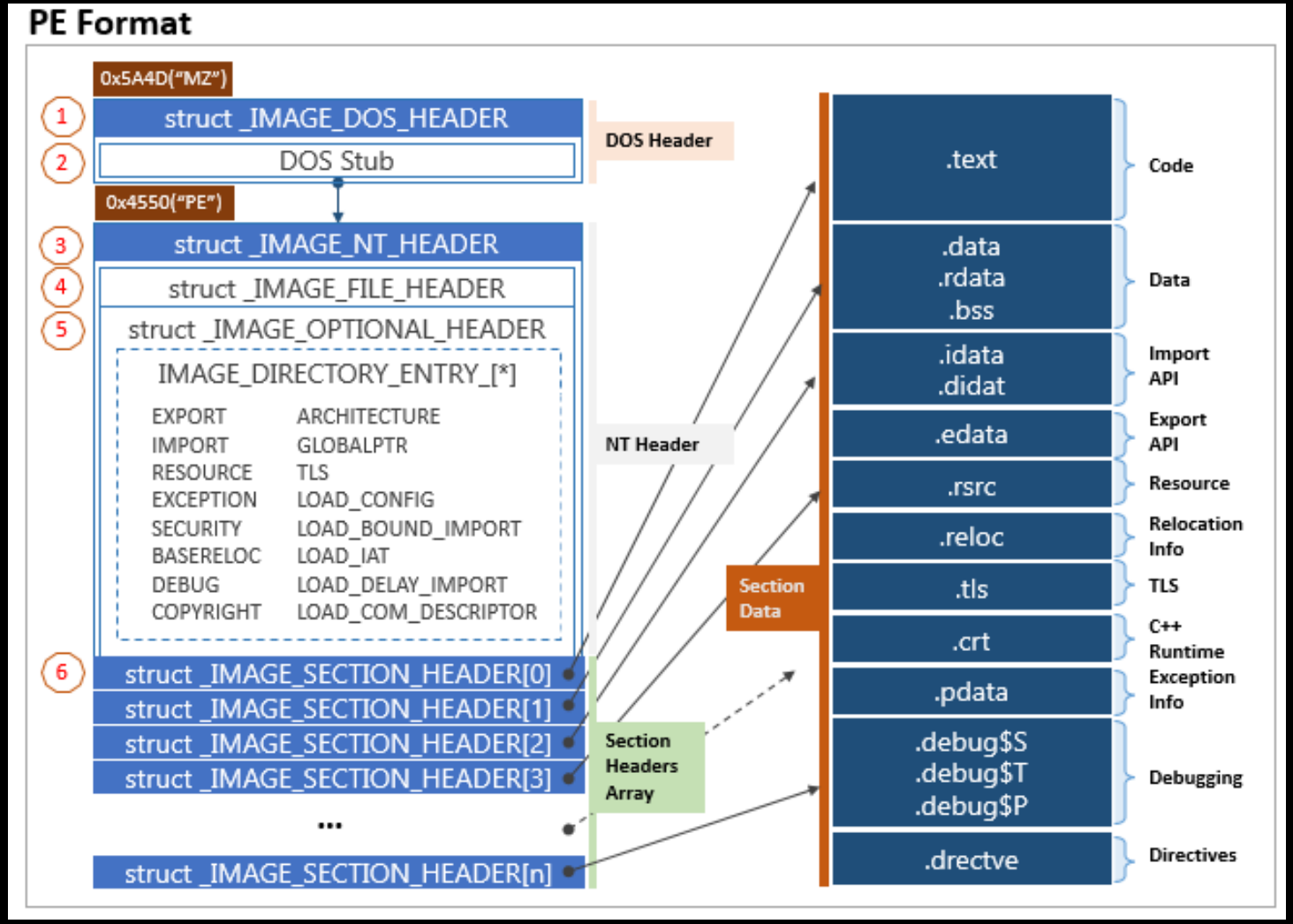
Outline

- Basics
- Windows Exploit Toolchains
- Basics Exploit Technique
- Stack Based Exploit Technique
- Heap Based Exploit Technique
- Windows Exploit Summary

Basics

PE/COFF FILE Format

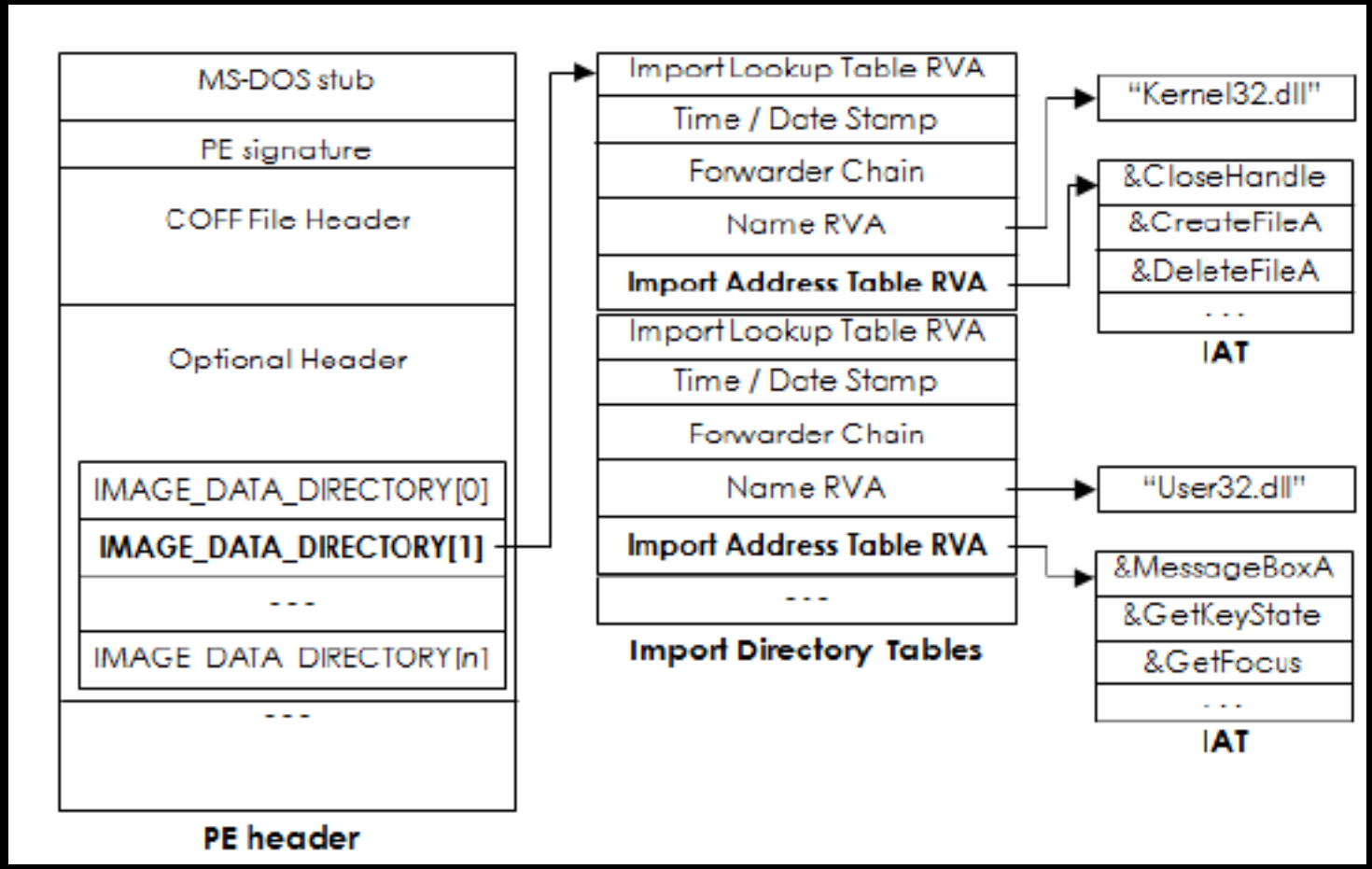
- DOS Header
  - MZ signature
- PE FILE HEADER
  - EntryPoint
  - DataDirectory
- Section Table
  - Table of Section Headers



Basics

PE/COFF FILE Format

- Import Address Table
  - Similar as ELF GOT
  - Read Only
- Export Address Table
  - Exported functions of a Module
  - Read Only



Basics

Important DLLs

- ntdll.dll
  - Interface of userspace and kernel
  - exports the Windows Native API
  - Reside in write-protected page; shared base among processes
- kernel32.dll
  - Imports ntdll.dll
  - exports the Windows API
  - Reside in write-protected page; shared base among processes
- ucrtbase.dll
  - C runtime library(similar to glibc)

Windows Exploit Toolchains

General Tools

- Cygwin
  - A bash environment on Windows
- socket lib
  - Used to interact with executables like pwntools
- Process Hacker
  - An enhanced version of tasklist
- Visual Studio
  - Developer Command Prompt

Windows Exploit Toolchains

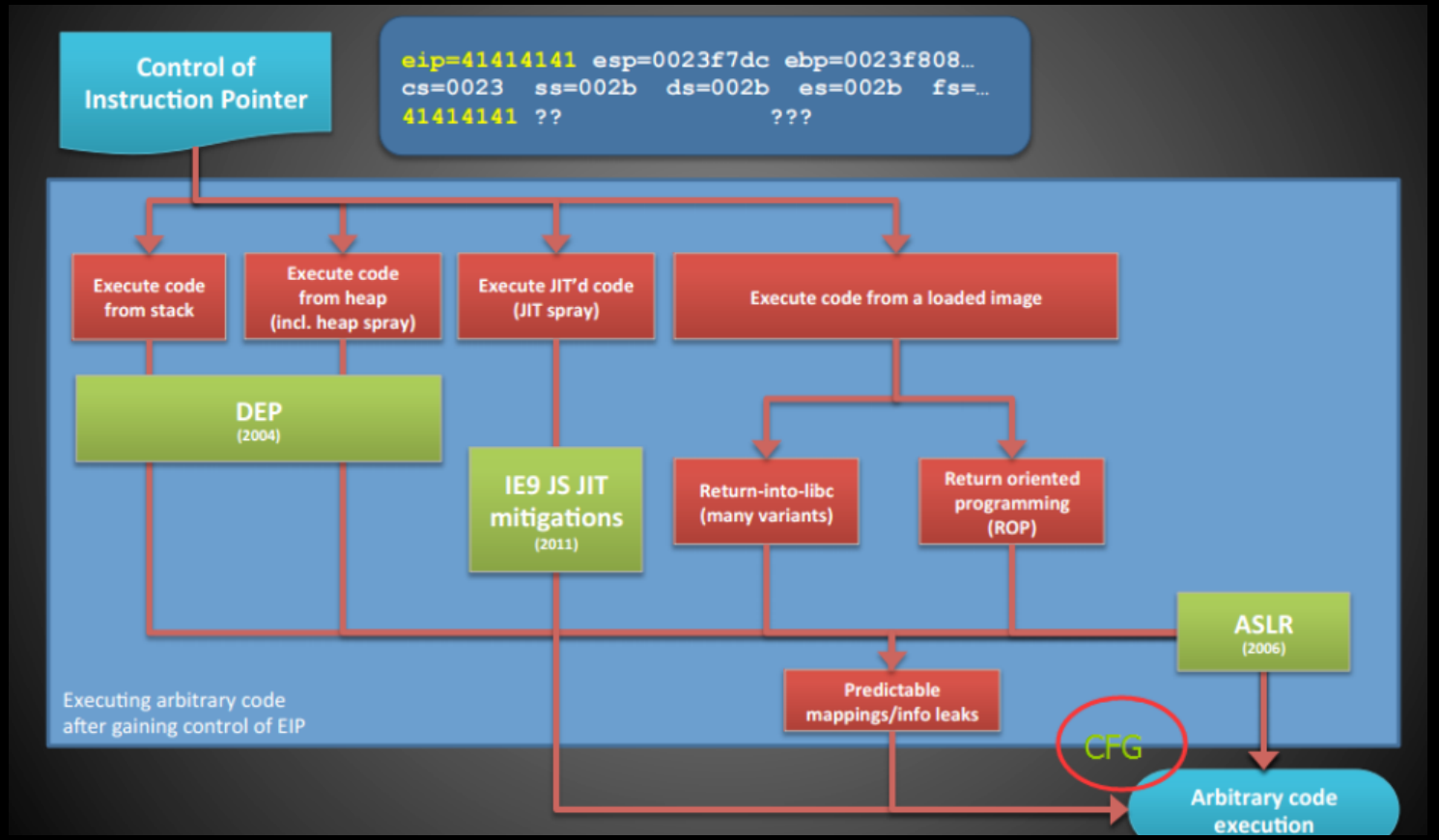
Debuggers

- Windbg
  - Recommend, very powerful
- IDA Pro Debugger
  - A debugger front-end, support multi-backend debugger such as gdb, windbg.
- Ollydbg
  - Easy & powerful, but cannot debug x64 program
- X64dbg
  - Similar to ollydbg, can debug x64 program

Basics Exploit Techniques

General Exploit Mitigations

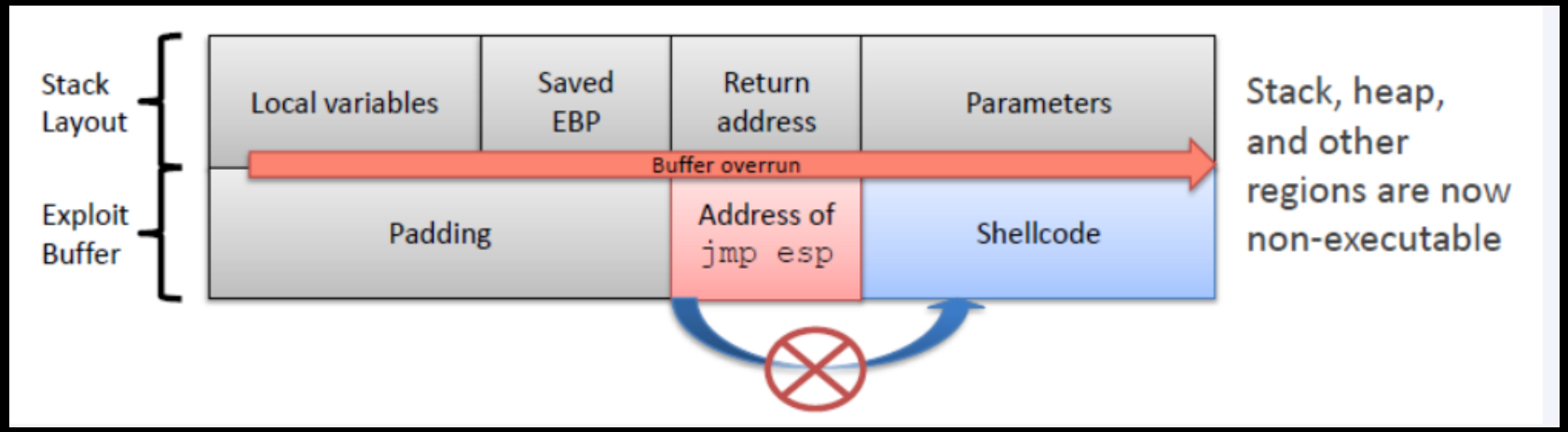
- DEP
- ASLR
- CFG



Basics Exploit Techniques

DEP

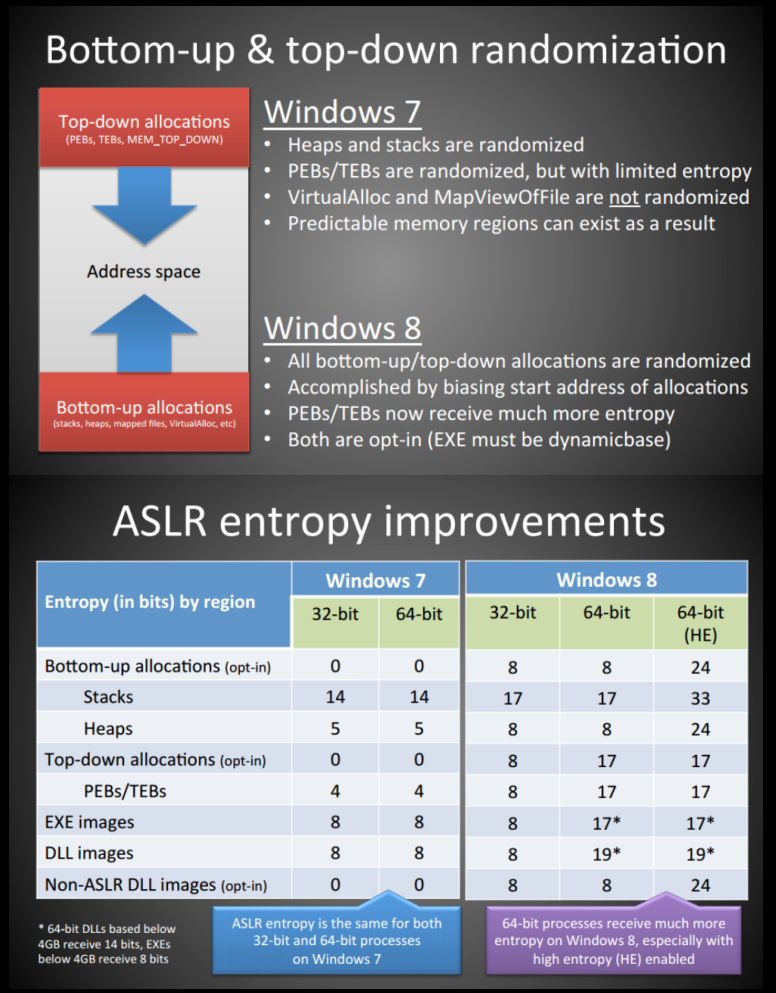
- NX on Linux
- Bypassed by
  - ROP
  - JIT page, VirualProtect etc.



Basics Exploit Techniques

ASLR

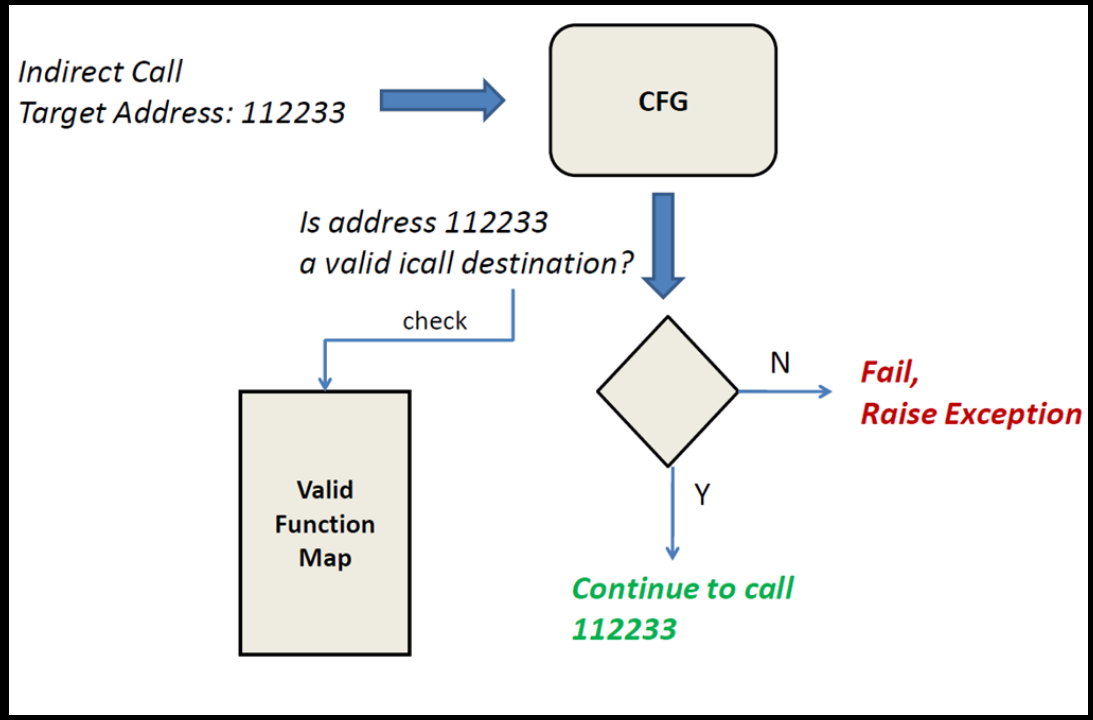
- Slightly different from PIE&ASLR on Linux
  - Image randomization base changed every time system booted
  - TEB/PEB/heap/stack randomization base changed every time process start
  - Some kernel related dlls (such as ntdll.dll kernel32.dll) share base among all processes
- Bypassed by
  - Info leak(cross process is OK)
  - brute-force (win7 x64, win10 x86)
  - Attack Non-ASLR images or top down alloc(win7)



Windows Security Mitigations

Control Flow Guard

- All indirect call are checked by predefined read-only bitmap
- Attack Vtable is history now.
- Bypassed by
  - Overwrite CFG unprotected value (return address, SEH handler, etc.).
  - Overwrite CFG disabled module
  - COOP++



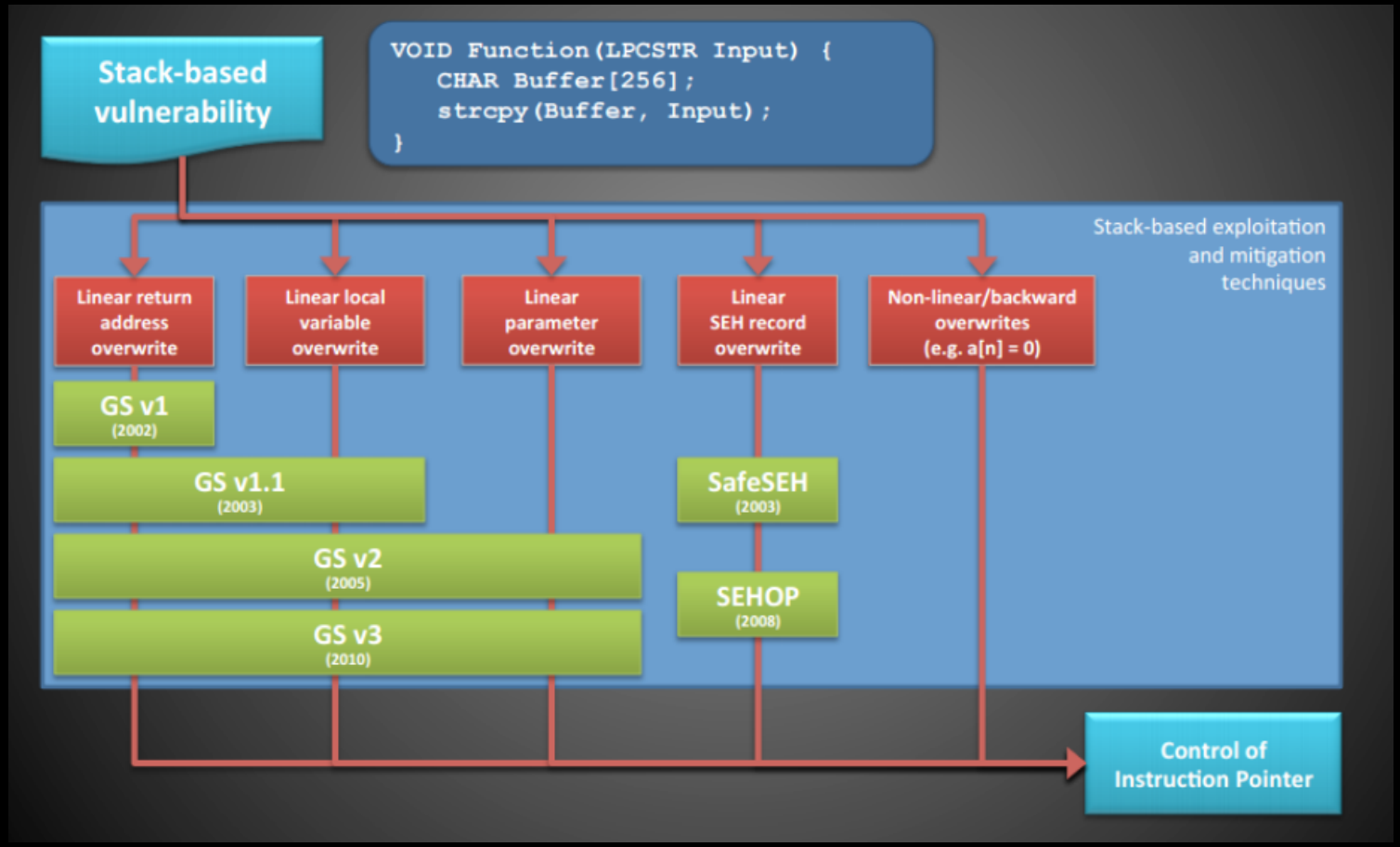
Example

- babyrop
- Help you guys familiar with Windows Exploit toolchains && Windows PWN skills
- Leak ucrtbase
- Stack Overflow
- Return to system('cmd')

Stack Based Exploit Techniques

Stack Based Vulnerability Mitigations

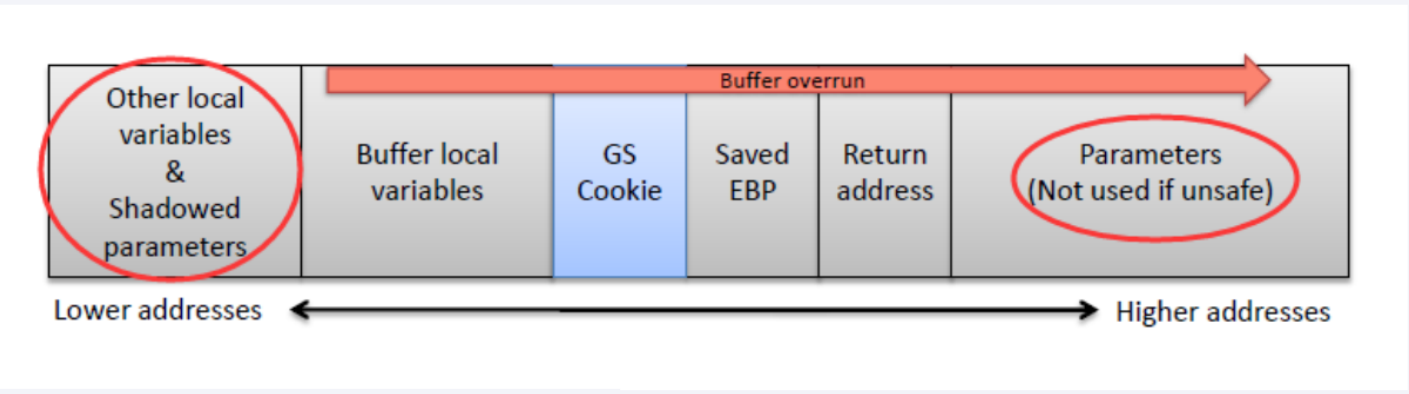
- GS
- SafeSEH
- SEHOP

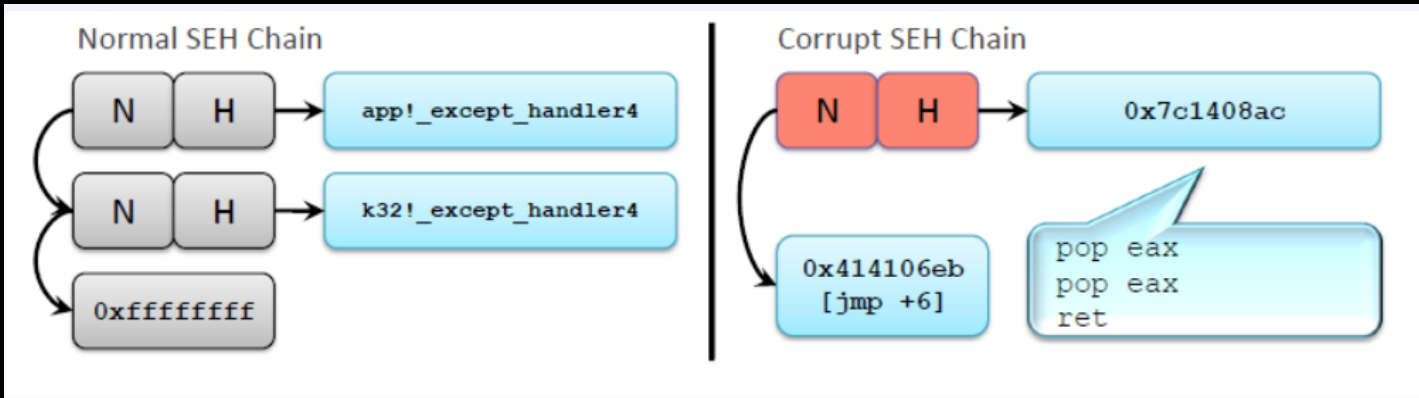


Stack Based Exploit Techniques

GS

- Similar to stack canary
- Bypassed by
  - corrupt SEH(x86)
  - Stack underflow
  - nonlinear write

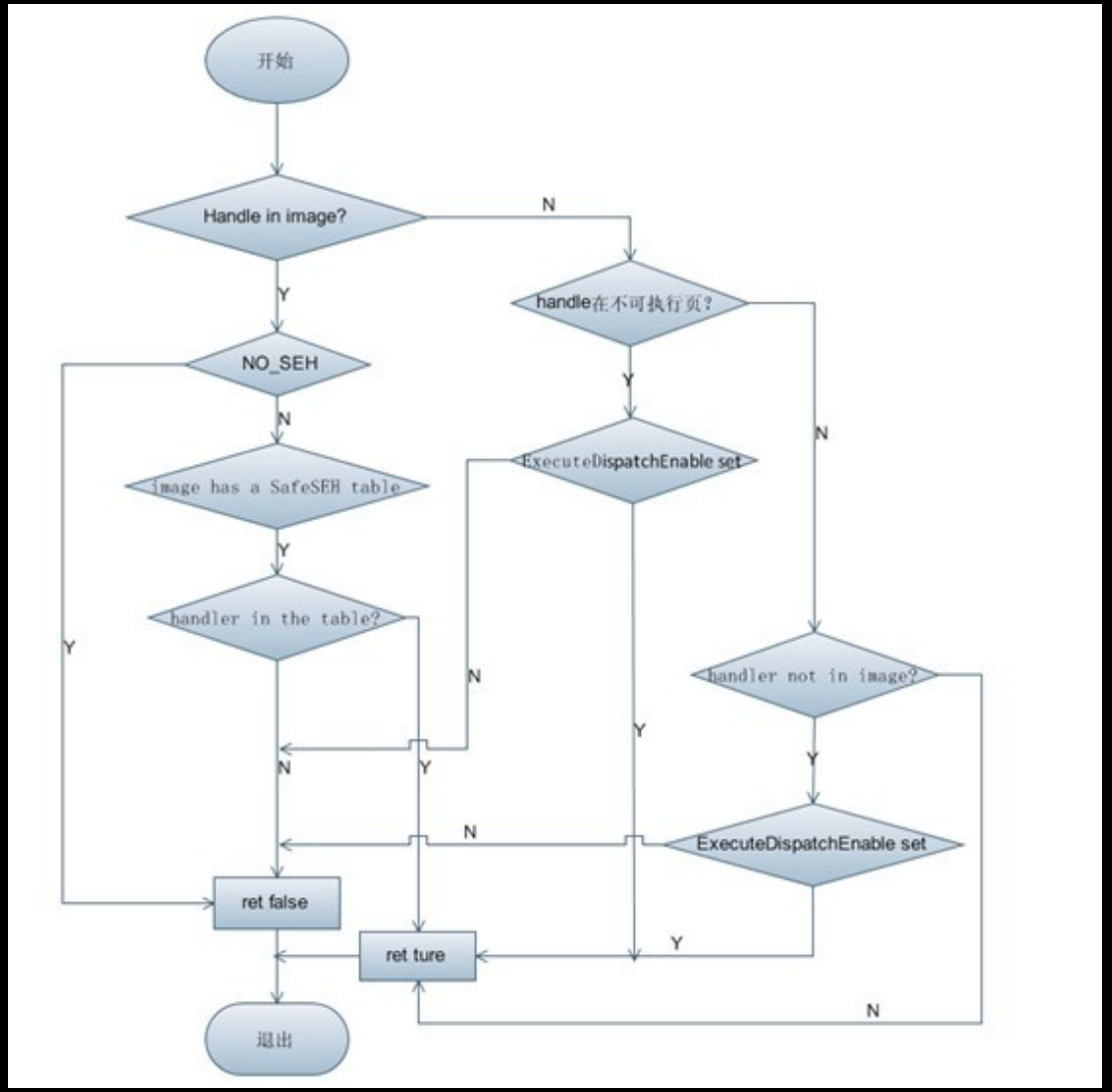




Stack Based Exploit Techniques

SafeSEH(x86)

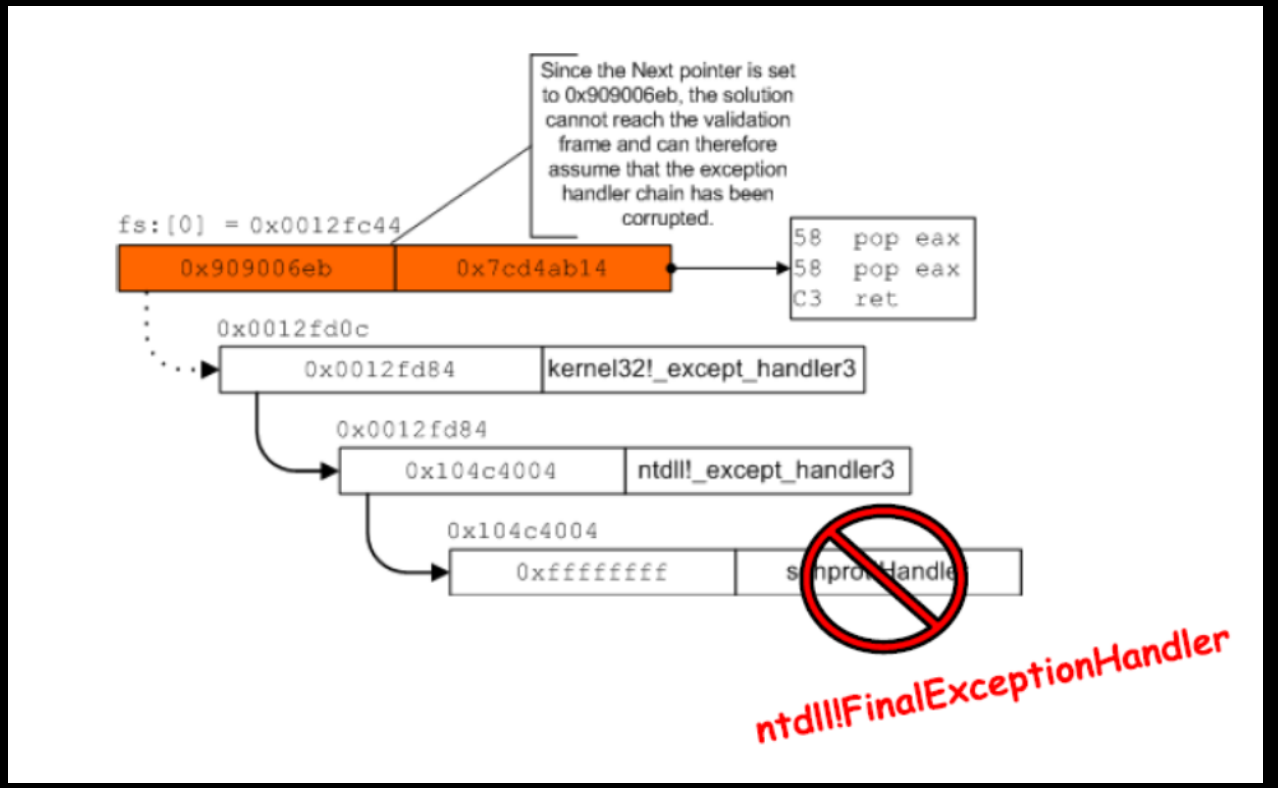
- Check whether handler is valid before calling the exception handler
- Bypassed by:
  - corrupt handler to an image with seh but without safeseh



Stack Based Exploit Techniques

SEHOP(x86)

- Check whether SEH chain ends with ntdll!FinalExceptionHandler
- Bypassed by
  - Leak stack address and recover the SEH chain



Stack Based Exploit Techniques

- What is SEH
  - For function contains try..except block, a VC_EXCEPTION_REGISTRATION struct will be pushed into stack
  - Overwrite handler and trigger a exception to hijack control flow

```
struct VC_EXCEPTION_REGISTRATION
{
    VC_EXCEPTION_REGISTRATION* prev;
    FARPROC handler;
    scopetable_entry* scopetable; //指向scopetable 数组指针
    int _index; //在scopetable_entry 中索引
    DWORD _ebp; //当前EBP 值
}
```

寄存器和局部变量	
ebp ^ cookie	-1c
esp	-18
xxxx	-14
fs:[0]	-10
handler	-c
scopetable^cookie	-8
trylevel	-4
original_ebp	ebp
Ret_addr	+4

Stack Based Exploit Techniques

Bypass GS by overwriting SEH

- Bypass SafeSEH
  - Corrupt handler to an image with SHE but without safeSEH. (only way, see ntdll.dll!RtlIsValidHandler)
- Bypass SEHOP
  - Leak stack address, recover SEH chains
- A little hard

```
bool RtlIsValidHandler(handler)
{
    if (handler image has a SafeSEH table) {
        if (handler found in the table)
            return TRUE;
        else
            return FALSE;
    }
    if (ExecuteDispatchEnable|ImageDispatchEnable bits set in the process flags)
        return TRUE
    if (handler is on a executable page){
        if (handler is in an image) {
            if (image has the IMAGE_DLLCHARACTERISTICS_NO_SEH flag set)
                return FALSE;
            if (image is a .NET assembly with the ILonly flag set)
                return FALSE;
            return TRUE
        }
        if (handler is not in an image) {
            if (ImageDispatchEnable bit set in the process flags)
                return TRUE;
            else
                return FALSE;
        }
    }
    if (handler is on a non-executable page) {
        if (ExecuteDispatchEnable bit set in the process flags)
            return TRUE;
        else
            raise ACCESS_VIOLATION;
    }

}
```

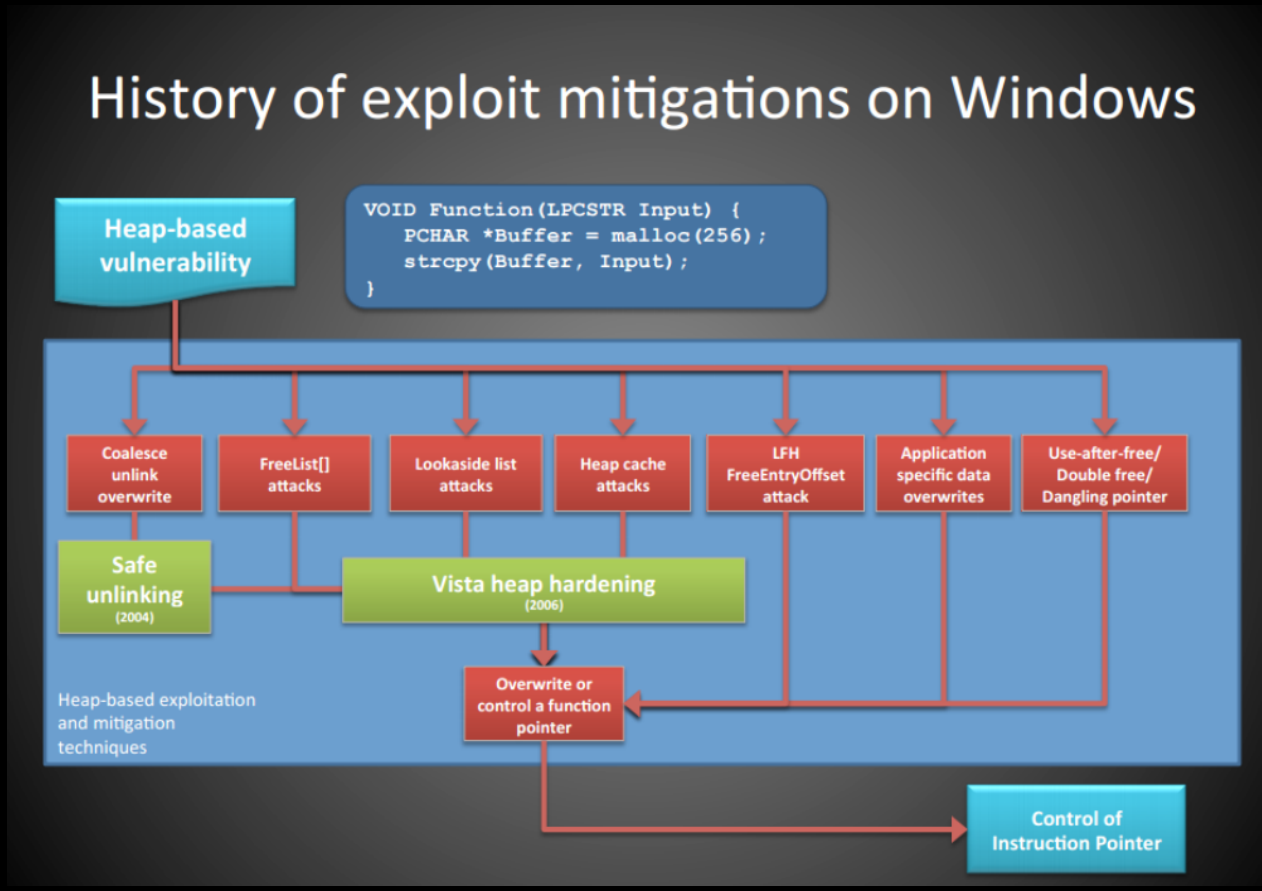
Example

- Babyseh
- Play with SEH
- Stack overflow with gs enable
- Overwrite handler

Heap-Based Exploit Techniques

Heap-based vulnerability mitigations

- Metadata check & hardening
- LFH allocation randomization
- VirtualAlloc randomization



Heap-Based Exploit Techniques

Metadata check & hardening

- Almost impossible to attack heap meta-data
  - Safe unlink
  - Replace lookaside lists with LFH
  - Heap cookies & Guard pages
    - Heap cookies are checked in some places such as entry free
    - Zero Permission Guard pages after VirtualAlloc memory
  - Metadata encoding
  - Pointer encoding
    - Almost all function pointer are encoded such as VEH, UEF, CommitRoutine, etc.
- Bypassed by
  - Overflow User data

Heap-Based Exploit Techniques

Metadata check & hardening

Change in Windows 8	Impact
LFH is now a bitmap-based allocator	LinkOffset corruption no longer possible [8]
Multiple catch-all EH blocks removed	Exceptions are no longer swallowed
HEAP handle can no longer be freed	Prevents attacks that try to corrupt HEAP handle state [7]
HEAP CommitRoutine encoded with global key	Prevents attacks that enable reliable control of the CommitRoutine pointer [7]
Validation of extended block header	Prevents unintended free of in-use heap blocks [7]
Busy blocks cannot be allocated	Prevents various attacks that reallocate an in-use block [8,11]
Heap encoding is now enabled in kernel mode	Better protection of heap entry headers [19]

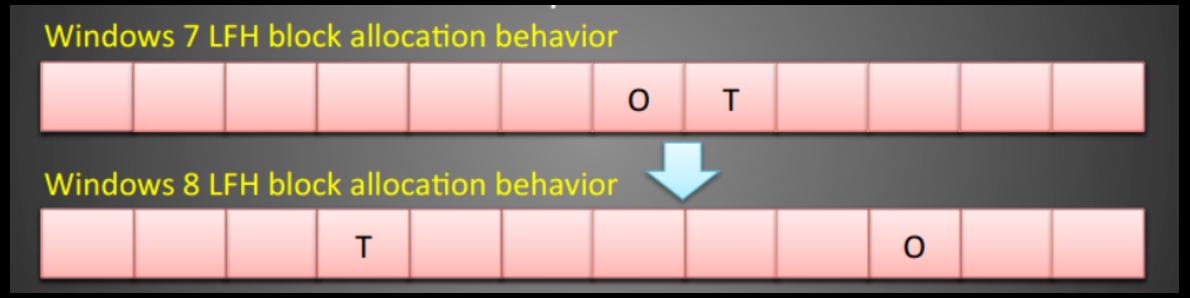
Heap-Based Exploit Techniques

VirtualAlloc randomization

- Ptr=VirtualAlloc(size+random), return ptr+random

LFH allocation randomization

- GetNextFreedLFHblock(random_start_index)
- Bypassed by
  - allocate LFH unhandled size(larger than 0x4000)
  - allocate LFH disabled size(specific-sized LFH will enable only if allocation times exceeded some threshold)
  - heap spray
  - brute-force



Example

- babyvtable
- Simple heap overflow, Play with heap && LFH
- Defeat LFH randomization
- Overflow vtable to hijack control flow
- Overflow data pointer to AAR&&AAW
- ROP to prompt a shell

Windows Exploit Summary

General Exploit Techniques

- Return oriented programming
- Get RWX pages via VirtualProtect like function
- Address space brute-force
- Heap manipulation
- Stack canary leak && overwrite
- Shellcode
  - Syscall style shellcodes are hard to use

Windows Exploit Summary

Information Leak Techniques

- Cross Binary/Process Shared address
- Leak share object base via GOT/GOT_PLT
- Dynamic search function address via AAR
- Leak stack address via non-stack address(such as libc environ)
- Leak Address via Format String Bug
- Leak Stack/SO/Binary Base Address via uninitialized stack buffer (OK)

Windows Exploit Summary

Control Flow hijack Techniques

- Return address overwrite (bypass CFG)
- SEH handler overwrite(bypass CFG)
- User function pointer overwrite
- vtable overwrite
  - If CFG enable, the targets are limited the overwrite value to function start
- Internal function pointers overwrite
  - Some function pointers are encoded or removed
    - UEF VEH encoded, PEB RtlEnterCriticalSection, RtlLeaveCriticalSection Removed.
  - Some function pointer such as SEH handler are still available to write

Homework

- Play with debugger
- Play with these challenges & pwn it
  - babyrop
  - babyrop2
  - babyvtable
  - drevil

Thanks

Atum