The Memory With Python

파이썬으로 치트엔진 만들기 #본격 사서 고생 프로젝트

CodeEngn

Outline

1. 시작하며

사원 A씨의 게임 해킹툴 분석

알고싶다 치트엔진

The Memory with Python : PyCheat

2. 개발기

관전 포인트 살펴보기

모방은 창조의 어머니 #1 Memory Search

모방은 창조의 어머니 #2 Find Accessing Address

목마른 사람이 우물을 파는 법 #3 Auto DLL Injection

목마른 사람이 우물을 파는 법 #4 Inline Hooking

삽질 어워드 # Enum Modules

3. 사용기

만든 것은 써먹어야 제맛 #1 취약 지점 찾기

만든 것은 써먹어야 제맛 #2. 해킹툴 제작하기

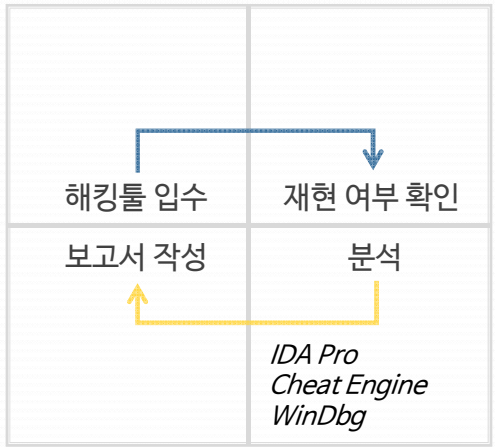
4. 후기

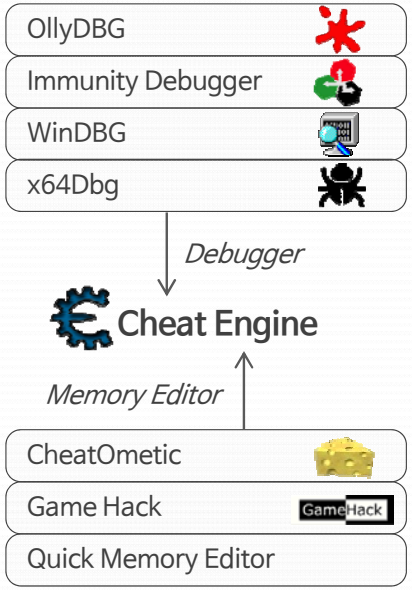
Reference #그래서, 좀 더 나아졌는가?

결론 # CheatEngine 과 비교를 한다면? #자문자답의 시간

Part 1. 시작하며

사원 A씨의 게임 해킹툴 분석





분석 도구는 분석가의 취향대로 사원 A씨의 취향 저격 Cheat Engine

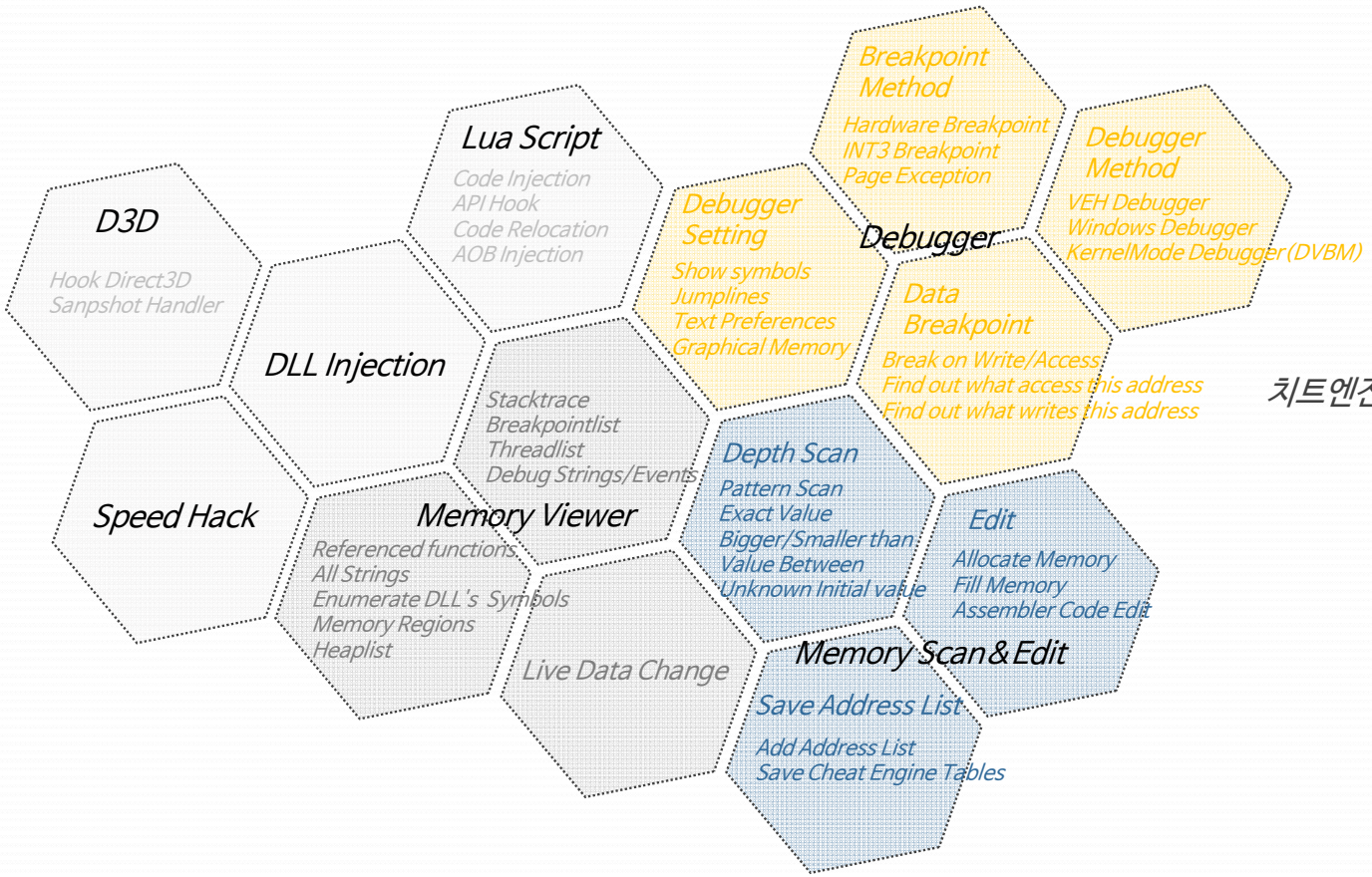
Debugger 기능과 Memory Edit 기능을 하나에!!

Part 1. 알고 싶다 치트엔진

#1. 실전에서 살펴보는 치트엔진 기능

Part 1. The Memory with Python : PyCheat

원대한 포부 # 엄청난 것을 만들어보겠다는 마음



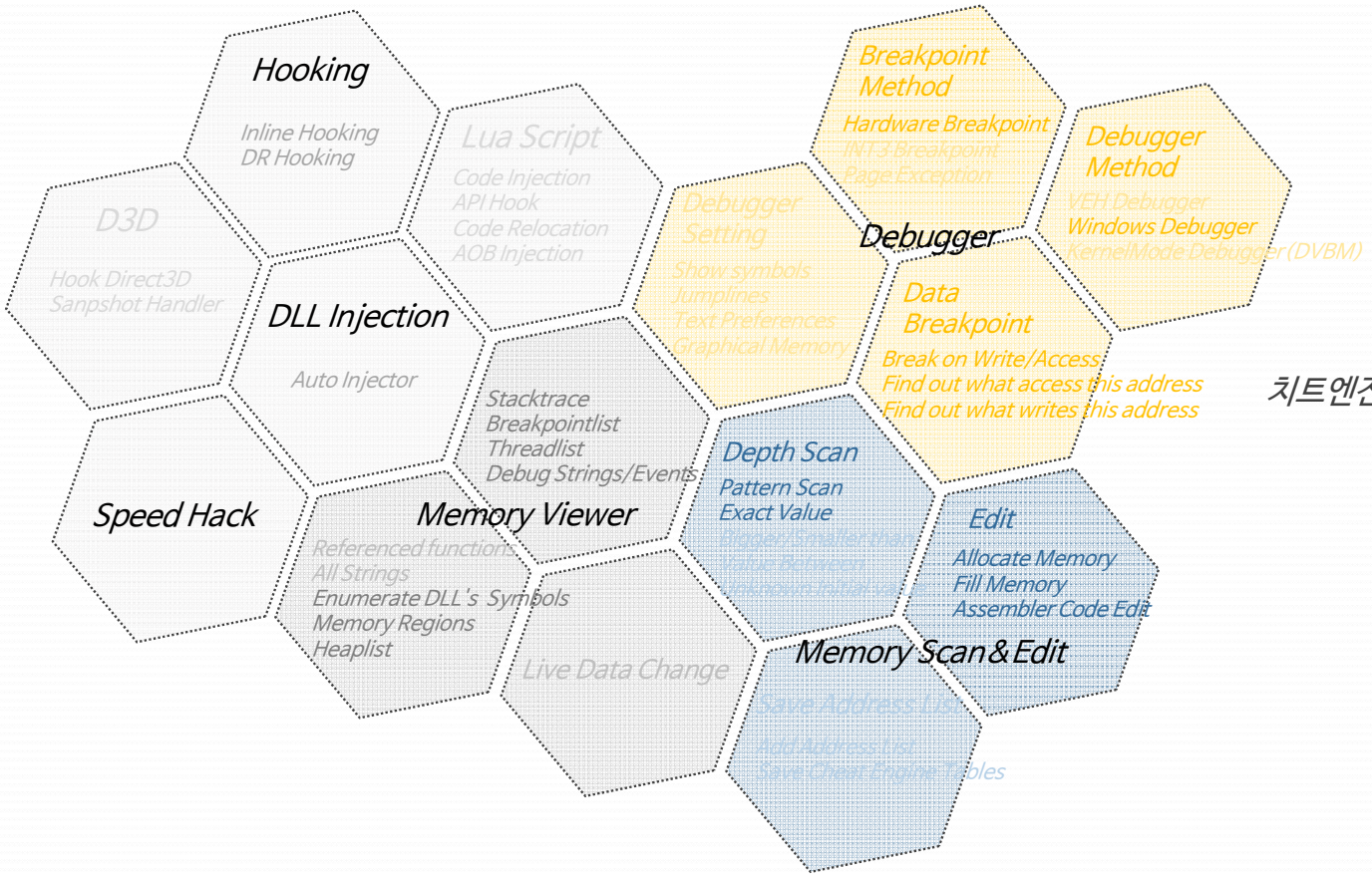
#커스터마이징 #DIY

치트엔진에서 자주 사용하는 기능은 가져오고

필요한 기능은 추가한다

Part 1. The Memory with Python : PyCheat

그러나 #현실과의 타협



#커스터마이징 #DIY

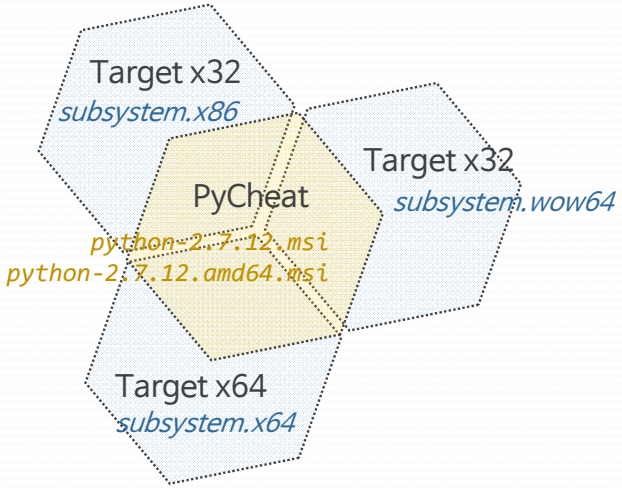
치트엔진에서 자주 사용하는 기능은 가져오고

필요한 기능은 추가한다

Part 2. 관전 포인트 살펴보기

PyCheat #관전 포인트를 알고 보면 꿀 재미가 가득

1. 아키텍처 호환, 그 전쟁의 서막



subsystem	common	engine
define.py	winapi.py	debugger.py
x86.py	process.py	hooking.py
wow64.py	address.py	injection.py
x64.py	util.py	pattern.py

2. 요리사가 쓰면 도구, 범죄자가 쓰면 흉기

Part 2. 모방은 창조의 어머니

Memory Search

MEMORY_BASIC_INFORMATION

```
class MEMORY_BASIC_INFORMATION32(Structure):
    _fields_ = [
        ("BaseAddress",         c_ulong),
        ("AllocationBase",      c_ulong),
        ("AllocationProtect",   c_ulong),
        ("RegionSize",          c_ulong),
        ("State",               c_ulong),
        ("Protect",             c_ulong),
        ("Type",                c_ulong)
    ]
```

```
class MEMORY_BASIC_INFORMATION64(Structure):
    _fields_ = [
        ("BaseAddress",         c_uint64),
        ("AllocationBase",      c_uint64),
        ("AllocationProtect",   c_uint32),
        ("__alignment1",        c_uint32),
        ("RegionSize",          c_uint64),
        ("State",               c_uint32),
        ("Protect",             c_uint32),
        ("Type",                c_uint32),
        ("__alignment2",        c_uint32)
]
```

Pattern Search

kernel32.GetSystemInfo

kernel32.VirtualQueryEx

kernel32.ReadProcessMemory

Pattern Scan

lpMinumumApplicationAddress 0x00001000

```
00010000   Commit    Read+Write    Mapped    10000
00020000   Commit    Read+Write    Private   2000
00022000   Reserve                 Private   C000
                        ...
00400000   Commit    Read          Image     1000
00401000   Commit    Execute+Read  Image     E000
0040F000   Commit    Read          Image     1000
00410000   Commit    Read+Write    Image     4000
00414000   Commit    Read          Image     6C000
                        ...
74110000   Commit    Read          Image     1000
74111000   Reserve                 Image     F000
74120000   Commit    Execute+Read  Image     64000
                        ...
7FFE1000   Reserve                 Private   F000
```

lpMaximumApplicationAddress 0x7FFF0000

Part 2. 모방은 창조의 어머니

Find Accessing Address #Debugger/Hardware Breakpoint #x86_64

Setting Hardware Breakpoint

Enumerate Thread
kernel32.GetContextThread
kernel32.SetContextThread

Enumerate Thread x86/x64

kernel32.CreateToolhelp32Snapshot
kernel32.Thread32First
kernel32.Thread32Next

ntdll.NtQuerySystemInformation
SystemProcessInformation
SYSTEM_INFORMATION_BLOCK
SYSTEM_THREAD_INFORMATION

EVENT LOOP

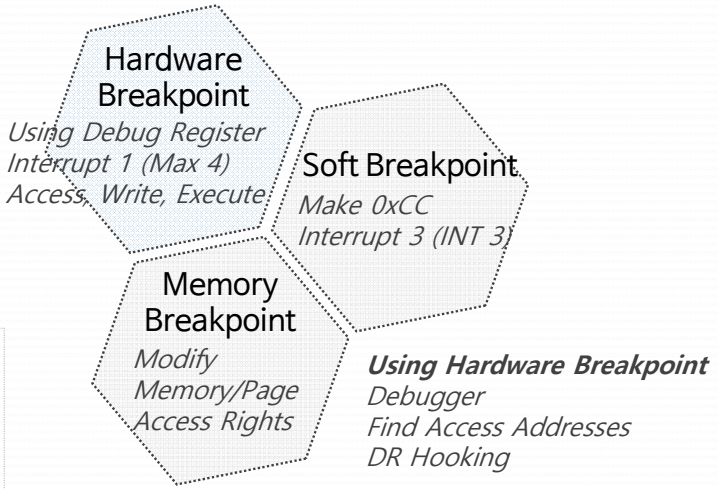
Kernel32.WaitForDebugEvent
ExceptionHandler
kernel32.ContinueDebugEvent

Find Accessing Addresses

Setting Hardware Breakpoint
kernel32.DebugActiveProcess
EVENT LOOP
kernel32.DebugActiveProcessStop

Exception Handler

```
if debug_event.DebugEventCode == EXCEPTION_DEBUG_EVENT:
    exception_code = debug_event.u.Exception.ExceptionRecord.ExceptionCode
    if exception_code == EXCEPTION_SINGLE_STEP:
        exception_single_step_handler ()
```



Part 2. 모방은 창조의 어머니

DLL Injection

Find Process

kernel32.CreateTool32SnapShot
kernel32.Process32First
kernel32.Process32Next

ntdll. NtQuerySystemInformation
SystemProcessInformation
PROCESS_INFORMATION_BLOCK

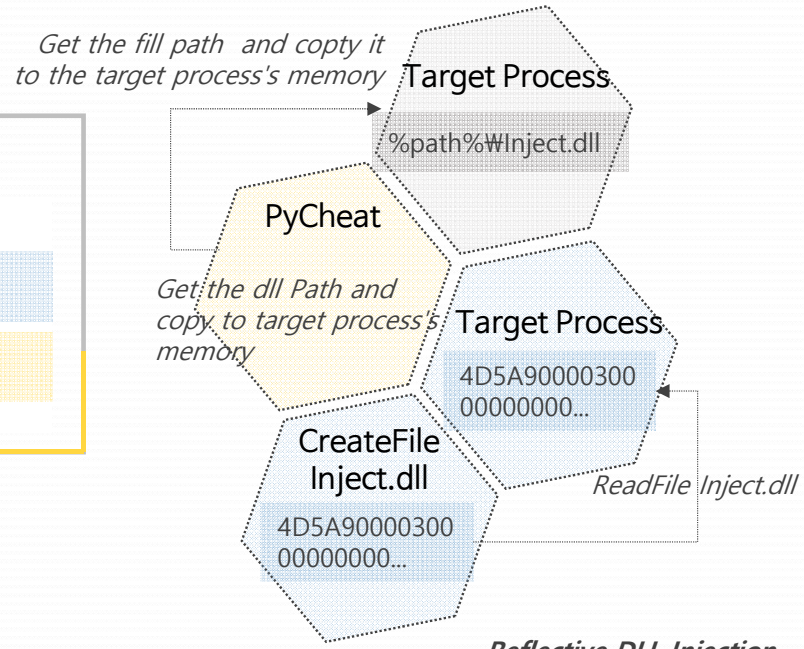
DLL Injection

kernel32.VirtualAllocEx
kernel32.WriteProcessMemory
kernel32.GetModuleHandleA
kernel32.GetProcAddress
kernel32.CreateRemoteThread

kernel32.CrateFileA
kernel32.GetFileSize
kernel32.VirtualAlloc
Kernel32.ReadFile
kernel32.WirteProcessMemory
kernel32.GetThreadContext
kernel32.SetThreadContext

Auto DLL Injection

- Find Process
- DLL Injection

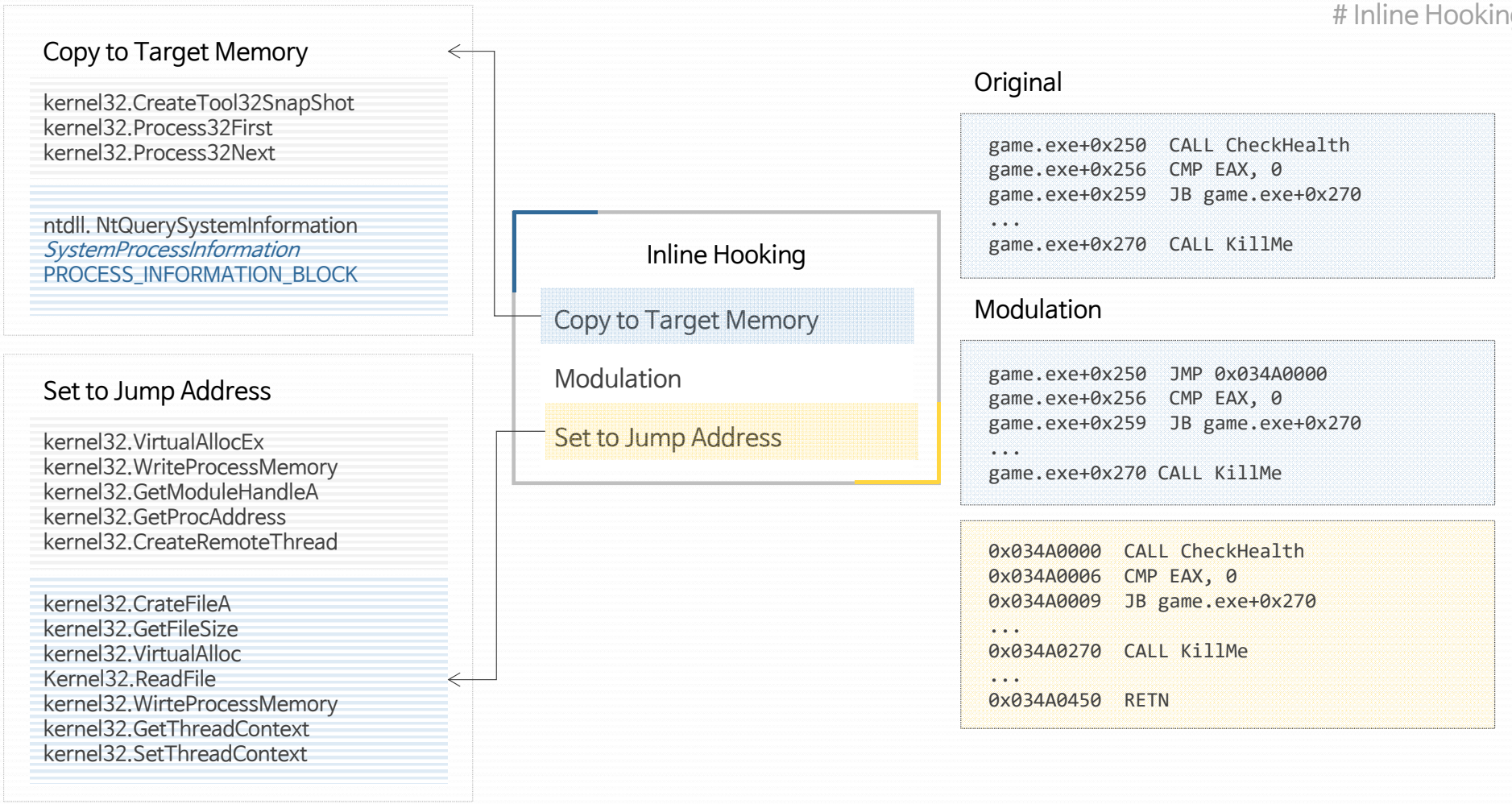


Reflective DLL Injection

Read DLL contents into Local process memory, then copy contexts into the target process's memory

Part 2. 목마른 사람이 우물을 판다

Inline Hooking



Part 2. 목마른 사람이 우물을 판다

MultiClient

Enumerate Handle

ntdll.NtQuerySystemInformation

SYSTEM_HANDLE_INFORMATION

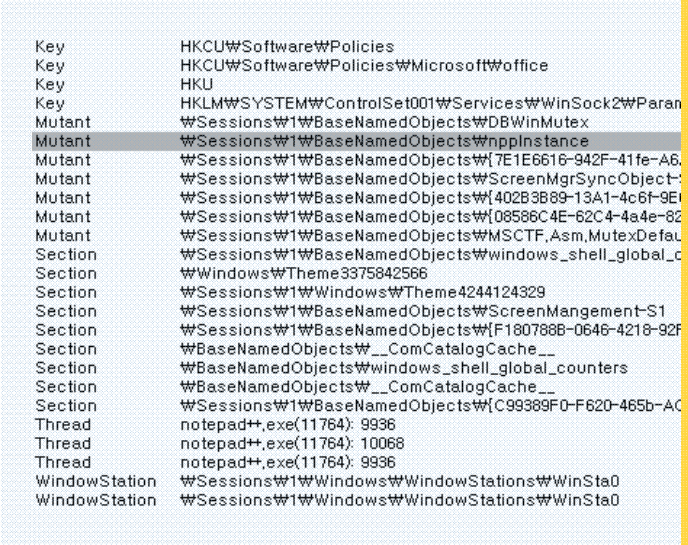
kernel32. DuplicateHandle

ntdll.NtQueryObject

OBJECT_BASIC_INFORMATION

OBJECT_TYPE_INFROMATION_TAG

ntdll.NtDuplicateObject



Part 2. 삽질 어워드 Best

Enumerate Modules #Enumerate DLL's and Symbols

x86

kernel32.CreateToolhelp32Snapshot
kernel32.Module32First
kernel32.Module32Next

ntdll.NtQueryInformationProcess
ProcessBasicInformation
PROCESS_INFORMATION_BLOCK32
PEB32
kernel32.ReadProcessMemory
PEB_LDR_DATA32
LDR_DATA_TABLE_ENTRY32

Enumerate Handle

x64

ntdll.NtQueryInformationProcess
ProcessBasicInformation
PROCESS_INFORMATION_BLOCK64
PEB64
kernel32.ReadProcessMemory
PEB_LDR_DATA64
LDR_DATA_TABLE_ENTRY64

wow64

ntdll.NtQueryInformationProcess
ProcessBasicInformation / ProcessWow64Information
PROCESS_INFORMATION_BLOCK 32/64
PEB 32/64
kernel32.ReadProcessMemory
PEB_LDR_DATA 32/64
LDR_DATA_TABLE_ENTRY 32/64

```
kd> dt nt!_EPROCESS fffffadfe71edc20
   +0x000 Pcb              : _KPROCESS
   +0x0b8 ProcessLock      : _EX_PUSH_LOCK
   +0x0c0 CreateTime       : _LARGE_INTEGER 0x1d1f2c7`e7a34268
   +0x0c8 ExitTime         : _LARGE_INTEGER 0x0
   /* ... */
   +0x2a8 Wow64Process     : 0xfffffadf`e5d79a90 _WOW64_PROCESS
   +0x2b0 ActiveThreads    : 2
   +0x2b4 GrantedAccess    : 0x1f0fff
   +0x2b8 DefaultHardErrorProcessing : 0x8004
   +0x2bc LastThreadExitStatus : 0n0
   +0x2c0 Peb              : 0x00000000`7efdf000 _PEB
```

Part 3. 만든 것은 써먹어야 제맛

#1 취약 지점 찾기

PyCheat 로 취약 지점 찾아보기

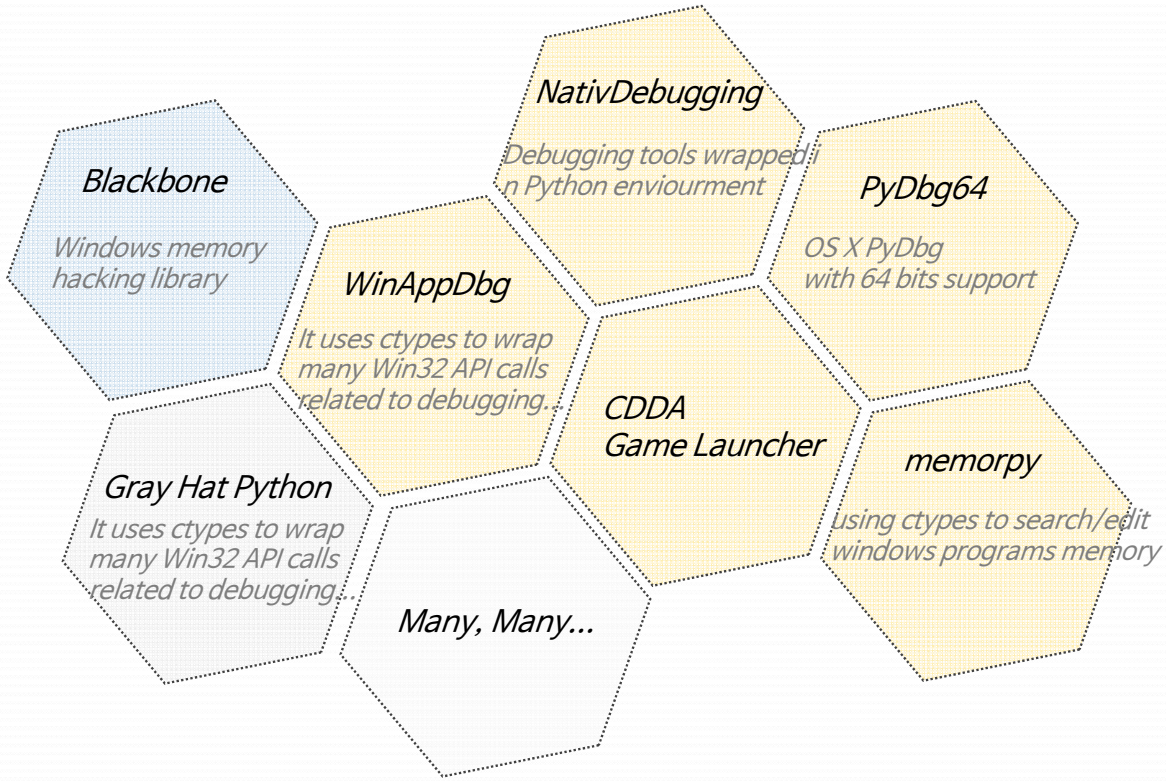
Part 3. 만든 것은 써먹어야 제맛

#2 해킹툴 제작하기

해킹툴 제작하기

Part 4. 후기

Reference #그래서, 좀 더 나아졌는가?



Part 4. 후기

결론 # CheatEngine 과 비교를 한다면? #자문자답의 시간

Speaker Info

끝 # GitHub에 올리러 총총 # 첫 업로드 떨림

김학수 (hakbaby92@gmail.com)