Anti-Debug Technics

26 octobre 2011

This talk ?

- Only anti-debugging
- No anti-dump
- No Anti-VM
- No Anti-Emulator
- No Obfuscation

This talk ?

- Only anti-debugging
- No anti-dump
- No Anti-VM
- No Anti-Emulator
- No Obfuscation

Only win32

- Not interesting under *nux
- Most system use
- Most Softwares are not free
- Malwares
- Software Protection (include games)

PEB.BeingDebugged Flag

- Most basic (and obvious)
- PEB.BeingDebugged flag is set if a debugger is present
- fs :[0x30] points to PEB
- kernel32 !IsDebuggerPresent() check this flag

```
kd> dt nt!_TEB
   +0x000 NtTib                  : _NT_TIB
   +0x01c EnvironmentPointer: Ptr32 Void
   +0x020 ClientId               : _CLIENT_ID
   +0x028 ActiveRpcHandle   : Ptr32 Void
   +0x02c ThreadLocalStoragePointer : Ptr32 Void
   +0x030 ProcessEnvironmentBlock : Ptr32 _PEB
   ...
```

```
kd> dt nt!_PEB
   +0x000 InheritedAddressSpace : UChar
   +0x001 ReadImageFileExecOptions : UChar
   +0x002 BeingDebugged    : UChar
```

Solution

- Patch PEB.BeingDebugged
- Manually
- Script
- Plugin

NtGlobalFlags

- 0x0 is not debugged
- 0x70 is debugged
- FLG_HEAP_ENABLE_TAIL_CHECK (0x10)
- FLG_HEAP_ENABLE_FREE_CHECK (0x20)
- FLG_HEAP_VALIDATE_PARAMETERS (0x40)

```
kd> dt nt!_PEB
    ...
   +0x068 NtGlobalFlag     : Uint4B
    ...
```

Heap Flags

- PEB.NtGlobalFlags are set, so Heap Flags too
- Usually 0x2 (HEAP_GROWABLE) is not debugged
- 0x50000062 is debugged
- HEAP_TAIL_CHECKING_ENABLED (0x20)
- HEAP_FREE_CHECKING_ENABLED (0x40)

```
kd> dt nt!_PEB
    ...
   +0x018 ProcessHeap       : Ptr32 Void
    ...
kd> dt nt!_HEAP
   +0x000 Entry             : _HEAP_ENTRY
   +0x008 Signature         : Uint4B
   +0x00c Flags             : Uint4B
   +0x010 ForceFlags        : Uint4B
```

Solution

- Patch PEB.NtGLobalFlags
- Patch _HEAP
- Manually
- Script
- Plugin

DebugPort

- 0 is not debugged
- non - zero value is debugged
- ntdd !NtQueryInformationProcess (ProcessInformationClass : ProcessDebugPort)
- returns 0xFFFFFFFF is debugged
- return 0 is not debugged
- kernel32 !CheckRemoteDebuggerPresent() use ntdll !NtQueryInformationProcess

```
kd> dt nt!_EPROCESS
    ...
   +0x0bc DebugPort         : Ptr32 Void
    ...
```

Solution

- Manipulating return value of ntdll!NtQueryInformationProcess
- Hook in userland (but be careful with syscall)
- Driver and set DebugPort to 0

Debugger Interrupts

- Setup exception Handler
- Generate Single-Step / Breakpoint with INT1/INT3
- If exception handler is not invoked, process is debugged
- kernel32 !DebugBreak()

CONTEXT

- Contains current state of the thread
- GetThreadContext()
- THREAD_GET_CONTEXT
- Parameter of exception handler

```
kd> dt nt!_CONTEXT
   +0x000 ContextFlags     : Uint4B
   +0x004 Dr0              : Uint4B
   +0x008 Dr1              : Uint4B
   +0x00c Dr2              : Uint4B
   +0x010 Dr3              : Uint4B
   +0x014 Dr6              : Uint4B
   +0x018 Dr7              : Uint4B
    ...
```

Solution

- Configure your debugger
- In case of GetThreadContext / SetThreadContext, Hook

Side Effet

Obfuscation

Checksum

- Checksum critical part of code
- Check for CC opcode (int3) at entry of API

Solution

- Use Hardware breakpoint
- No wonder
- Reverse
- Find check routines

Little Disassembly Engine

- Is it a CC OpCode ?
- Instruction is ret ? (end scan)
- Instruction is ret n ? (end scan)
- Dissasemble a least 100 instructions ?

Redirection

- Alloc memory
- Copy instruction until conditional jump
- Copy instruction until relativ jump
- Copy instruction until ret

Solution

- Use Hardware Breakpoint
- Set breakpoint in Unicode Version of the API
- Break on native API
- LoadLibraryExW instead of LoadLibraryA

CreateFile, CloseHandle

- CreateFile on '.id', '.til', '.nam' (anti ida)
- CreateFile on its own executable
- CloseHandle on invalid handle
- Check if exception is reached

Solution

- Breakpoint and change value
- Hook
- Setup your debugger

Anti attach

- ntdll !DbgBreakpoint() called when a debugger want to attach
- Exception is raised (int3)
- VirtualProtect()
- Change opcode CC to C3 (ret)

Solution

- Attach to process
- Restore opcode
- Hook ntdll !DbgBreakpoint()

Hide From Debugger

- ntdll!NtSetInformationThread (Undocumented)
- ThreadHideFromDebugger
- No more exception passed to debugger
- KiTrape03->CommonDispatcherException ->KiDispatchException->DbgkForwardException

```
kd> dt nt!_ETHREAD
    ...
   +0x248 HideFromDebugger : Pos 2, 1 Bit
    ...
```

Solution

- Restore flag
- Hook DbgkForwardException

Thread Local Storage

- TLS Callback called before OEP
- Can check debuger presence before reach oep

Solution

- Learn how to conf your debugger
- You can see it inside IDA

Trap Flag

```
push offset @handler
push dword fs:[0]
;... junk code
pushfd
xor dword ptr [esp], 154h
popfd
mov eax, 1; there is something wrong
;....
@handler:
;continue execution
```

Solution

- Don't trace the code
- Put bp on exception handler

INT2D

- Int2D use by ntoskrnl.exe to play with DebugServices

```
push offset @handler
push dword fs:[0]
mov fs:[0], esp
;... junk code
db 02Dh
mov eax, 1 ; there is something wrong
;...
@handler:
;continue execution
```

Timing Checks

- CPU cycles are spent inside debugger
- Check time between operations
- RDTSC instruction (Read Time-Stamp Counter)
- kernel32 !GetTickCount()
- kernel32 !GetProcessTimes()

Solution

- Do not step, use breakpoint
- Hook GetTickCount()
- Time Stamp Disable Bit (TSD) in cr4
- RDTSC executed in privilege != 0, cause general protection
- Be careful with last solution :]

SeDebugPrivilege

- SeDebugPrivilege is disabled by default
- Debugger usually enable this privilege
- Debugged will inherit this access token
- Only administrator can grant this
- ntdll !OpenProcess CSRSS.exe (only accessible to SYSTEM)

Solution

- Patch ntdll !OpenProcess()

Parent Process

- Is my parent explorer.exe ?
- Probably is being debugged

Solution

- Patch kernel32 !Process32NextW()

Debugger Windows

- FindWindow() / FindWindowEx()
- “ollydbg” / ”WinDbgFrameClass” / “IDA” / ...

Solution

- Break on FindWindow() and fuck lpClassName
- Hook FindWindow()
- Patch your debugger

Debugger Process

- Enumerating all process
- PROCESSENTRY32.szExeFile
- "ollydbg.exe", "idaq.exe", "windbg.exe" ...

Solution

- Rename the executable
- Patch kernel32 !Process32NextW()
- Be careful with wmi stuff too

Debug Object

- ntdll !NtQueryObject()
- ObjectAllTypesInformation
- Prepare stuff
- Handle -1
- Check “DebugObject” in da list

```
typedef struct _OBJECT_TYPE_INFORMATION
{
        UNICODE_STRING TypeName;
        ULONG TotalNumberOfHandles;
        ULONG TotalNumberOfObjects;
} OBJECT_TYPE_INFORMATION, *POBJECT_TYPE_INFORMATION;
```

Device Drivers

- kernel32 !CreateFileA()
- Well-known device names (\\.\NTICE)

Solution

- Break on kernel32 !CreateFile()
- Hook

Questions ?