Windows Rootkit 分析与检测综合方法

陈伟东[1,2]，张 力[1]
(1.清华大学 软件学院，北京 100084；
2.国家企业信息化应用支撑软件工程技术研究中心，湖北 武汉 430074)

摘 要：分析了已知的Rootkit 检测方法，并根据上述检测方法提出了一种基于采用比较核心模块查证内存代码段的完整性和检测隐藏驱动的方法。该方法能够检测大部分应用层Rootkit 和内核Rootkit.

关键词：Windows Rootkit；内核对象；系统服务描述符表；进程；端口；文件
中图分类号：TP311.1 文献标识码：B

Synthetic approach for Windows Rootkit analysis and detection

CHEN Wei Dong[1,2], ZHANG Li[1]
(1.School of Software ,Tsinghua University, Beijing 100084, China;
2.State Enterprise Informationization Application Support Software Engineering Research Center,Wuhan 430074,China)

Abstract: This paper analyses the methods of Rootkits detections of nowdays, brings a method of kernel modules compared,checks memory sections integration and hidden driver detection.This method can detect most application layer Rootkits and kernel Rootkits.

Key words: Windows Rootkit; kernel object; system service description table; process; port; file.

计算机网络和应用日益普及，针对计算机系统的攻击趋于隐蔽和复杂化，在各种攻击方法中，Rootkit 能够持久地在计算机中寄存，无法被普通用户检测。多数Rootkit 可以隐藏文件和目录，Rootkit 的其他特性通常用于远程访问和窃听。Rootkit 具备下述功能：(1)窃取重要信息：通过监控键盘击键以及网络数据，窃取用户口令以及网络银行密码等隐私信息；(2)Rootkit 隐藏的后门可以使攻击者维持控制权；(3)隐藏攻击痕迹，隐藏与Rootkit 相关的文件、进程、通信链接和系统日志等攻击痕迹；(4)欺骗检测工具，使检测工具无法发现系统的异常。病毒可以通过 Rootkit 提供的隐蔽通道植入系统，可使用 Rootkit 隐藏自身。还可以为蠕虫传播、拒绝服务等攻击提供攻击跳板。

Rootkit 可以分为应用级 Rootkit 和内核级 Rootkit 2 种类型。应用级 Rootkit 通过修改或替换系统现有的二进制程序和系统工具，以及运行时刻的内存补丁技术来达到隐藏自身的目的；内核级 Rootkit 通过修改系统底层内核来隐藏与 Rootkit 相关的文件、进程、通信等信息。

内核级 Rootkit 涉及操作系统的底层内核。操作系统内核作为操作系统最重要部分，完成了文件系统、进程调度、系统调用、存储管理等功能。内核级 Rootkit 修改操作系统的内核，例如中断调用表、系统调用表、文件系统等内容。由于系统内核在操作系统最底层，一旦内核受到 Rootkit 攻击，应用层的程序从内核获取的信息将不可靠，第三方的应用层检测工具无法发现Rootkit。Rootkit 软件正成为信息安全领域最大的挑战之一，越来越多的研究人员开始研究 Rootkit 技术。本文从 Rootkit 涉及的技术原理和检测技术谈起，通过对系统文件的完整性、程序代码段的完整性、IAT/EAT、SDT、IRP 表的内容、隐藏驱动程序的查找、检测隐藏文件、注册表键以及进程等方面提出了对 Rootkit 的检测方法并予以实现。

1 Windows Rootkit 的背景和概念

Windows Rootkit 分为 User mode 和 Kernel mode 2 种，对于一个 Win32 API 调用，是从应用层程序调用内核的系统服务，在这一过程中，Rootkit 可以修改任何一处，以控制特定的系统调用的返回结果。

Windows Rootkit 的主要功能是隐藏踪迹，它可以实现进程隐藏、文件隐藏、目录隐藏、驱动程序隐藏、注册表项隐藏、服务隐藏、端口隐藏等功能。同时，Windows Rootkit 也会附带上一些远程控制工具，使用秘密通道与目标主机建立连接，进行各种远程控制。

在多数情况下，内核 Rootkit 只是简单的改写系统描述符表 SDT(Service Descriptor Table)，用它们自己的函数替换原始的 API。用户模式下的 Windows Rootkit 必须在每个进程中运行一次 Rootkit，然后才能拦截到所有希望截获的系统调用。Rootkit 传统工作方法是：修改各种内核数据，包括 Hook DLL 函数,修改 DLL 函数、Hook 系统描述符表、系统服务表 SST 和 KiServiceTable 某些项，Hook 中断描述表 IDT 中的 2Eh 中断入口点，以及修改内核代码。

2 Windows Rootkit 的技术和原理

2.1 环 0 级及操作系统模型

Intel X86 系列芯片用环概念来实施访问控制。环包括 4 个级别：环 0 是最高权限，环 3 是最低权限，Winodws 操作系统的所有内核代码在环 0 级别上运行。在内核中运行的 Rootkit 工作在环 0 级别。用户模式工作在环 3 级别。CPU 负责跟踪为软件代码和内存分配环的情况，并在各环之间实施访问控制。通常，每个软件程序都会获得一个环编号，它不能访问任何具有更小编号的环。例如，环 3 的程序不允许访问环 0 程序，试图访问将会导致 CPU 产生异常中断。除了内存访问限制，某些 CPU 指令是具有特权的，只能在环 0 级使用。以下 X86 指令只允许在环 0 级使用：Cli-停止中断的处理；Sti-启动中断的处理；In-从硬件端口读取数据，Out-将数据写入硬件端口。

Windows 的内核模式组件包含以下部分：

(1)Windows 执行体

包含基本的操作系统服务，如内存管理、进程和线程管理、安全性、I/O 管理、网络和跨进程通信；

(2)Windows 内核

由线程调度、中断、异常分发、多处理器同步等构成；

(3)设备驱动程序

包括硬件设备驱动程序、文件系统和网络驱动程序；

(4)硬件驱动层 HAL(Hardware Abstraction Layer)

把内核、设备驱动程序和 Windows 执行体的其余部分与平台无关的硬件差异隔离。

2.2 Windows 系统内核对象分析

Windows 内核对象是操作系统分配的一个内存块，只能由内核访问和管理。应用程序只能通过句柄的方式调用 Windows 提供的一系列内核对象操作接口，且只有句柄使用权，无修改权。

2.2.1 中断描述符表

全局描述符表 GDT(Global Descriptor Table):存有整个系统的段描述符合各个进程共用的段描述符、如数据段、代码段、堆栈段等，最大表长 64 KB；

局部描述符表 LDT(Local Descriptor Table):该表中存有某个进程专用的段描述表，最大表长 4 GB；

段描述符(Segment Descriptor)：该结构用来描述一个段的基地址、段的类型，对段操作的限制等等；

段选择符(Segment Selector)：该结构是 GDT 或 LDT 中的索引；

中断描述表 IDT(Interrupt Describer Table):该结构中存有中断门描述符、陷阱门描述符和任务门描述符。

2.2.2 HOOK 系统服务描述符表

WINNT 内核中有 2 个全局变量：KeServiceDescriptor Table、KeServiceDescriptorTableShadow。KeService DescriptorTable 变量指向一个数组，存储着 ntoskrnl.exe 模块中的系统函数。

用户进程在调用 NTDLL.DLL 中的函数时，NTDLL.DLL 将调用转到系统服务描述符表中。

```
Typedef struct _SERVICE_DESCRIPTOR_TABLE
{
SYSTEM_SERVICE_TABLE    ntoskrnl;  //ntoskrnl.exe
SYSTEM_SERVICE_TABLE  win32k;  //win32.sys
SYSTEM_SERVICE_TABLE  Table3;  //not used
SYSTEM_SERVICE_TABLE  Table3;  //not used
} SERVICE_DESCRIPTOR_TABLE,
*PSERVICE_DESCRIPTOR_TABLE,
**PPSERVICE_DESCRIPTOR_TABLE;
```

系统服务分配表其实由 4 张表组成，SYSTEM_SERVICE_TABLE 表结构如下：

```
Typedef struct _SYSTEM_SERVICE_TABLE
{
  PNTPROC   ServiceTable;    //真正的函数表
  PDWORD    CounterTable;    //使用次数(调试有用)
```

```
    DWORD     SerivceLimit;     //服务表函数个数
    PBYTE     ArgumentTable;    //参数表
}SYSTEM_SERVICE_TABLE, *PSYSTEM_SERVICE_TABLE, **PSYSTEM_SERVICE_TABLE;
```

几个宏有助于 Hook 住 SSDT 表。

SYSTEMSERVICE 宏采用 ntoskrnl.exe 导出的 Zw* 函数地址，并返回相应的 Nt* 函数在 SSDT 中的地址。Nt* 函数是私有函数，其地址列于 SSDT 中。Zw* 函数是由内核为使用设备驱动程序和其他内核组件而导出的函数。

SYSCALL_INDEX 宏采用 Zw* 函数地址并返回它在 SSDT 中相应的索引号。内核中所用 Zw* 函数都是以操作码 mov eax，ULONG 起始，ULONG 是系统调用在 SSDT 中的索引号。通过将该函数的第二个字节看作 ULONG 类型，这些宏能得到该函数的索引号。

```
#define  Hook_SYSCALL(_Function, _Hook, _Orig)\
_Orig  =(PVOID)InterlockedExchange((PLONG)\
&MappedSystemCallTable
```

使用 Hook NtQuerySystemInformation 来隐藏进程和文件夹。

2.2.3 Hook 中断描述符表 IDT

中断描述符表 IDT 用于处理中断，IDT 指定了如何处理按键、页面错误或用户请求 SSDT 时触发的中断。SIDT 指令在内存中为每个 CPU 寻找 IDT，返回 IDTINFO 结构地址。

IDT 中每一项都具有自己的结构，长度为 64 位。IDTENTRY 结构中的 LowOffset 和 HiOffset 组成了中断处理程序的地址。

```
Typedef  struct
{
WORD      LowOffset;
WORD      selector;
BYTE      unused_lo;
Unsigned  char      unused_hi: 5; //stored  Type ?
Unsigned  char      DPL: 2;       //vector is present
WORD                HiOffset;
} IDTENTRY;
```

下面的 HookInterrupts 函数声明了一个全局 DWORD，存储实际的 INT2E 函数处理程序 KiSystem Service。将 NT_SYSTEM_SERVICE_INT 定义为 0x2E。这是 IDT 中将要钩住的索引。如下代码将 IDT 中的实际项替换为包含 Hook 地址的 IDTENTRY。

```
DWORD  KiRealSystemServiceISR_Ptr;      //The real INT 2E  handler
#define   NT_SYSTEM_SERVICE_INT          0x2e
Int            HookInterrupts()
{
  IDTINFO          idt_info;
  IDTENTRY*        idt_entries;
  IDTENTRY*        int2e_entry;
  __asm {
    Sidt           idt_info;
  }
```

已在 IDT 中安装了 Hook，就可检测或阻止使用系统调用的进程。系统调用编号存储在 EAX 寄存器中。通过调用 PsGetCurrentProcess 函数可以获得指向当前 EPROCESS 指针。

Hook SYSENTER：较新版本 Windows 系统不再用 INT 2E 或通过 IDT 请求系统调用表中的服务。而是使用 fast call method。在这种方法中，NTDLL 向 EAX 寄存器中加载被请求服务的系统调用号，向 EDX 寄存器中加载当前堆栈指针 ESP，然后发出 Intel 指令 SYSENTER。可在驱动中读取 IA32_SYSENTER_EIP 值，将其存储在一个全局变量中，然后将钩子地址填充到该寄存器中。

Hook 设备驱动程序对象中主要的 I/O 请求报文函数表：在驱动程序加载时，初始化一个函数指针表，包含了各种类型 I/O 请求报文(IRP)处理函数地址，例如可以 Hook 处理文件系统写操作或 TCP 查询的函数，IRP 和特定驱动程序依赖于期望完成的目标。例如可以钩住处理文件系统写操作或 TCP 查询的函数。在准备隐藏网络端口的使用情况时，首先要在内存中找到驱动程序对象。例如 TCPIP.SYS 及与之相关的设备对象 \\DEVICE\\TCP。内核提供的 IoGetDeviceObjectPointer 函数能返回相应的文件和设备对象。设备对象包含一个驱动程序对象指针，保存目标函数表。Rootkit 将要 Hook 的函数指针的旧值保存下来。使用 Interlocked Exchange 函数交换函数地址指针。可以 Hook IRP_MJ_DEVICE_CONTROL 。在 TCPIP.SYS 中安装了钩子后，就可以在 HookedDeviceControl 函数中接收 IRP。

2.2.4 Windows Rootkit 技术分析总结

(1) IDT/Sysenter hook 技术

Windows 系统下一个系统调用中断产生后，将系统调用服务号放进 eax 寄存器，然后读取中断描述符寄存器值，找到中断描述符表，根据系统调用中断号(2e)为索引找到 SSDT(系统服务描述表)表基址，在这个流程中中断描述符寄存器值可以用 Rootkit 技术修改成用户自

己定义的函数，所以当中断产生时，用户函数都会被调用；

(2) SSDT Hook 技术

系统服务描述符表示一个系统调用函数表，每一个表项存储对应函数在内存中的地址。Rootkit 也可以在此实现 Hook；

(3) Inline Hook 技术

Inline Hook 直接修改函数在内存中前几个字节；

(4) IAT/EAT Hook 技术

Windows 系统下 PE 文件有输入表和输出表，PE 文件加载到内存中后，需要将一些 dll 文件也加载到内存中，PE 文件需要调用这些 dll 定义函数，即为 PE 文件的输入表(IAT)与 dll 文件的输出表(EAT)。Windows PE 文件有固定格式，可根据该格式找到 PE 文件的输入表和输出表，保存函数虚拟地址以及加载到内存中的地址的变量，这个变量值可被修改，Rootkit 在此刻实现 IAT 表和 EAT 表的 Hook。

Windows 驱动中有一张 MAJOR_FUNCTION 表，存储驱动 MAJOR_FUNCTION 地址，Rootkit 技术能实现 hook MAJOR_FUNCTION 表。直接操作内核对象：Windows 系统是以对象(object)为操作单位的，Rootkit 技术关注的是 Windows 系统中关键数据结构(如进程 EPROCESS 结构与线程 ETHREAD 结构)、链表(如进程、线程链表与 CPU 调度链表)等，Rootkit 通过操作这些数据结构达到目的。例如进程隐藏等。

(5) 通信隐藏技术

一些 Windows Rootkit 使用了十分隐蔽的通信技术，利用各种类型的 ICMP 数据包进行通信、利用 UDP 协议进行通信及利用一些特殊类型的 TCP 数据包进行通信等。还有一些 Rootkit 使用了隐蔽通道技术。与目标主机中正常的网络服务程序共用相同的知名端口。当 Windows Rootkit 收到网络上发到此共用端口的数据包时，只处理带有自己特殊标记的数据包，其他数据包转发给正常的网络服务程序处理。

Netstat 及类似工具，如 fport、Tcpview 等都通过调用 Iphlpapi.dll 中的 API 获得端口列表，而 Iphlpapi.dll 的 API 最终会调用 ZwDeviceIoControlFile 函数，向设备对象 Device\Tcp 发送控制代码 IOCTL_TCP_QUERY_INFORMATION_EX 获得各种信息。因此，只要 Hook ZwDeviceIoControlFile 函数，就可以控制这些端口了。

3 Rootkit 现有检测技术

对已知 Rootkit 签名检测先要对已知的 Rootkit 分析，找到特点。Rootkit 检测工具可以利用一个签名对系统关键区域进行模式匹配，若存在已知的 Rootkit，则可以很快检测到。基于异常行为的 Rootkit 检测，首先建立目标系统及用户正常活动的标准模型，基于此标准模型对系统和用户实际活动(如进程文件访问、网络访问等)进行审计，判定用户行为是否对系统构成威胁。例如 Vice 的检测方法，判断 SSDT 表每一个表项所指向的地址是否在 NTOSKRNL.EXE 模块范围内。因为 SSDT 表函数都是在 NTOSKRNL.exe 模块内实现的。基于交叉视图 Rootkit 检测，Windows 系统下 Rootkit 藏身分成几个特定段，交叉视图检测对同一个任务通过获取任意 2 个点信息比较，判断两种信息是否一样，是则正常，否则认为是 Rootkit。例如利用 API 列举文件信息，一种是应用层 API，另一种是内核的系统服务函数。若两种信息不一样，则认为应用层 API 被 Hook。如 RootKit Reavealer 就是采用此方法。基于内存完整性 Rootkit 检测原理是：在系统中存有一个标准文件，当检测时将当前文件与标准文件匹配，若匹配则正常，否则判定系统受 Rootkit 攻击。VICE 分析用户模式应用和操作系统核心。VICE 检查函数指针在 SSDT 中的位置。在用户模式，VICE 检查每个应用程序地址空间，在每个 DLL 中查找 IAT hooks。进程隐藏检测方法，IceSword 进程隐藏检测方法是遍历 PspCidTable 表，可以检测出大多数利用 rootkit 技术隐藏的进程，但是现在新一代的进程隐藏技术又出现了。全局变量 PspCidTable 是一个 HANDLE_TABLE 结构的句柄表，保存在本系统中。PatchFinder 检测技术利用了 EPA 技术，通过分析系统中执行某些关键函数时所使用的指令个数的变化情况，发现系统中的 Rootkit，但是检测结果并不稳定，虚警的可能性很高。

常见的 Rootkit 隐藏和检测方法是 Hook NtQuerySystemInformation( )。Hack defender 采用了此种方法，直接遍历 ActiveProcessLinks 枚举进程，可以检测 Hook NtQuerySystemInformation。从 ActiveProcessLinks 上摘除自身，利用线程调用链表检测隐藏进程(KiWaitLnListHead、KiWaitOutListthread，KiDispatcher ReadyListhead)，可以检测 Hook NtQuerySystemInformation( )，可以检测从 ActiveProcessLinks 上摘除自身。Hook SwapContext( )，进程结束后，操作系统并不总能把相应 EPROCESS 从链表上摘掉 EPROCESS 其他成员已经破坏，MmProcessLinks 会枚举出完整的 EPROCESS 映象。EPA 可执行路径分析检测，内核级木马的一种影响是它通常改变一个正常程序的执行路径，通过将自身插入到程序执行体中间，Rootkit 修改内存或 Hook 会造成多余的指令被执行。当一个修改的函数和一个正常函数相比

时，指令执行数量不同则 Rootkit 存在。执行路径分析技术利用了上述原理，可以检测多种内核模式和用户模式下的木马，如果入侵者修改了内核函数隐藏了一些对象，则内核对象的执行路径将改变，调用一些典型的系统和库函数时，系统将运行一些多余指令。EPA 通过检查某些关键系统函数执行时所用的指令个数，与实现记录过的数据进行比较，若发现两者间有明显差异，即可断定这些函数执行了多余代码，即被 Hook 了。代码字节段是只读的，应用程序应该不会修改此字段。因此只需检查重要系统 DLL 和系统驱动程序(核心模式)在内存和相应的磁盘中的 PE 文件是否相同即可。

4 本文采用的 Rootkit 检测方法

大多数 Rootkit 都有一个共同的特点，即会对内存进行修改。因此本文主要采用基于内存扫描的检测方法，以检测出更改内存的 Windows 木马，对于隐藏驱动程序的检测，以遍历 PsLoadedModuleList 链表的方法进行。PsLoadedModuleList 是 Windows 加载的所用内核模块构成的链表的表头，利用它可以枚举所用这些模块的信息。进而可以列举出隐藏的驱动。

检测 SDT 修正的方法 KiServiceTable 是.text 的一部分。如果在 KiServiceTable 中作了修改，那么它是目标地址，实际的 SDT 可能被重定位(如 Kaspersky AV 所用的)。

Windows Rootkit 对内存区的修改有如下几种：

(1)用户模式：Rootkit 修改用户进程的用户地址空间中的内容，主要包括某些模块的代码区块(CodeSection)；

(2)内核模式：Rootkit 修改内核地址空间中的内容。包括各种内核模块的输出函数表(Export Table)、系统服务地址表、系统服务描述符表、各种内核模块的代码区块(Code Section)、各种特殊指针，内核模式 Windows Rootkit 修改内核地址空间中的内容，这部分内容是整个操作系统共用的，内核模式 Rootkit 只对内核地址空间修改一次，就可以持续地控制操作系统的行为。Windows Rootkit 木马修改区域通常是系统服务地址表、系统服务描述符表、内核模块的输出函数、内核模块代码区块以及各种内核数据结构。

本文采用比较核心模块的方法，查证内存代码段的完整性。程序分为驱动层和应用层。驱动层的功能包括：得到中断分配表的基地址和长度、SYSENTER 的寄存器 CS、ESP、EIP 的值和系统服务描述符表(SDT)。系统服务描述符表的具体结构如上段所述。还有对虚拟内存的读写操作。检测驱动程序采用遍历 PsLoadedModuleList 进行。

检查的内容包括：系统文件的完整性、程序代码段的完整性、IAT/EAT，SDT，IRP 表的内容、隐藏驱动程序的查找。检测隐藏文件，注册表键、进程等。首先得到核心模块的列表，使用 NtQuerySystemInformaiton。然后得到用户模式当前进程的 dll 模块列表，使用 EnumProcessModules().对于内核模块，可调用 ZwQuerySystemInformation 查找 SSDT 和 IRP 处理程序列表。列出所有模块，调用 ZwQuerySystemInformation 并指定 SystemModuleInformation 一类信息。它返回一个已知模块的相关信息。结构如下：

```
Typedef struct _MODULE_INFO {
    DWORD    d_Reserved1;
    DWORD    d_Reserved2;
    PVOID    p_Base;
    DWORD    d_Size;
    DWORD    d_Flags;
    WORD     w_Index;
    WORD     w_Rank;
    WORD     w_LoadCount;
    WORD     w_NameOffset;
    BYTE     a_bPath[MAXIMUM_FILENAME_
LENGTH];
}MODULE_INFO, *PMODULE_INFO, **PPMODULE_
INFO;
```

查找内核 Hook，函数查找名称为 ntoskrnl.exe 的一项。发现该项后，初始化一个包含该模块起始和结束地址的全局变量。该信息用于在 SSDT 中查找超出 ntoskrnl.exe 范围之外的地址。通过 DEVICE_IOCONTROL 在核心层和应用层通信，得到核心层的 SDT、IDT 和 SYSENTER 值，然后在应用层与相应的核心模块端比较。

```
Bool   detectKernel( )
{
   if(rtdect.kerAgent->currKiServiceTblStart != rtdect.
kerAgent->origKiServiceTblStart)
      TRACE0("WARNING:Service Table redirection
detected\n");
   verifyKerIDT( );
   verifyKerMSRS( );
   …
}
```

检测隐藏的进程：调用 ntoskrnl.exe 中的 SwapContext

(下转第 18 页)

（上接第14页）

函数将当前运行线程的上下文与重新执行线程的上下文进行交换。调用了 SwapContext 后，EDI 寄存器中值是下一个要交换进入的线程的指针，ESI 寄存器中的值是要交换出去的当前线程指针。对于这种检测方法，将 SwapContext 的前导替换为指向 detour 函数的 5 字节无条件跳转指令。

内核驱动检测通过 ZwQuerySystemInformation 的 SystemModuleInformation 功能号枚举内核驱动。通过打开目录对象进行枚举，或通过枚举 IoDriverObjectType 和 IoDeviceObjectType 对象类型进行查找枚举；接着可以通过查找 PsLoadedModuleList 对该链进行下枚举。关于 SSDT Hook 的检测，通过定位 ntoskrnl.exe 磁盘文件里 KeServiceDescriptorTable 与内存中的 KeServiceDescriptorTable 即可对各个服务函数进行比较；通过检测内存空间中重要内容的完整性来查找 Rootkit；通过检测 PE 文件中的输入地址表完整性检测 Hook 函数；通过系统服务地址表检测 Hook 函数；通过遍历 PsLoadModuleList 来检测隐藏驱动。

随着计算机网络的普及，用户在计算机上的重要资料越多，越容易成为被攻击的对象。Rootkit 由于其自身特点难以检测，本文针对此问题提出了一种检测方案，研究了 Windows 系统的一些内核技术。

由于 Rootkit 技术的发展，Rootkit 检测也趋于复杂化。只有综合采用多种检测方法，才有可能发现隐藏的多种 Rootkit。因此，需要对 Windows 系统内核深入地研究，找出更好的 Rootkit 检测方法。

参考文献

[1] GREG H, JAMES B.Rootkits: subverting the windows kernel[Z]. USA: Addison Wesley Professional, 2005.

[2] VICE-Catch hookers.www.rootkit.com.2008.

[3] 齐琪.基于内存完整性的木马检测技术研究[D].华中科技大学, 2006: 1208-1219.

[4] JOANNA R.Rootkit detection on windows systems[EB/OL].www.invisiblethings.org.2008.

(收稿日期：2009-01-20)