基于差异分析的隐蔽恶意代码检测*)

曹 跃 梁 晓 李毅超 何子昂

(电子科技大学计算机科学与工程学院网络攻防实验室 成都 610054)

摘 要 隐蔽性恶意程序 Rootkit 通过篡改系统内核代码与指令,导致操作系统返回虚假的关键系统信息,从而逃避管理员和主机型安全工具的检查。通过分析 Rootkit 技术的实现原理,包括进程、TCP 端口、注册表和文件的隐藏技术,提出了基于差异分析的隐藏行为检测技术。该技术将可信任的系统信息与不可信任的系统信息进行比较,从而获得被隐藏的信息。最终实现了相应的原型系统。与特征码扫描法相比,该检测方法检测在未知和变形 Rootkit 方面具有明显优势。

关键词 入侵检测,恶意程序,Rootkit,隐蔽

Variance Analysis Based Stealthy Malicious Code Detection

CAO Yue LIANG Xiao LI Yi-Chao HE Zi-Ang

(Laboratory of Network Attack & Defense, School of Computer Science and Engineering, UEST of China, Chengdu 610054)

Abstract Stealthy malicious rootkits evade inspection of administrators and host-based security detection tools by modifying operating system kernel programs and instructions, then bring unreal pivotal information to system and security utilities. With the analysis of the malicious code hidden technology, we present a stealthy malicious code detection technology base on the analysis of the differences. This technology compares the trusty system information with untrusty ones, and regards the differences as hidden information. Finally we establish a trusted cooperation detection model with its prototype. Compared with signature scanning method, our method is demonstrated much superior on detecting unknown and metamorphous rootkits, which gets authentic detection results.

Keywords Intrusion detection, Malware, Rootkit, Stealth

1 引言

随着计算机网络技术的迅速发展,全世界的人们面临着大量的网络安全隐患。拒绝服务(DoS)攻击阻止被攻击的服务器为合法用户提供正常的服务。而远程主机的非法访问与控制是另一类极具隐蔽性的攻击方式,入侵者使用木马程序提供的隐蔽通道来获取对被控制主机非法的访问控制,窃取被攻击主机及相邻网络的敏感信息。

基于网络(Network-Based)的入侵检测技术[1]为管理员提供对恶意程序网络行为的监控;而基于主机(Host-Based)的入侵检测技术[2]主要提供在主机级别上对恶意程序的安全检测。虽然大量的主机型安全检测技术得到广泛应用,如完整性检测[3]、特征码扫描[4]等,但在检测未知变形恶意程序和新兴隐蔽恶意程序方面,效果皆不理想。随着网络攻防技术的发展,入侵者增强了恶意程序的隐蔽能力,使其能轻易逃避现有安全工具的检测,这类隐蔽恶意程序称之为 Rootkit[5]。Rootkit 是一类长期存在于计算机系统中而不被发现的程序和代码集。

本文试图通过对恶意代码实现机制、反检测技术和隐藏技术的研究,探索新的恶意代码检测技术。通过分析现有的商用恶意代码检测系统的原理及其缺陷,研究恶意代码的隐藏技术。提出针对使用隐藏技术的恶意代码的检测方法,根据上述检测原理实现了 Aigis 检测系统。

2 Rootkit 技术

2.1 进程隐藏

为逃避恶意代码检测系统的扫描,现在恶意代码大都使用 Rootkit 技术来隐藏进程,包括:

(1)拦截 SDT 实现进程隐藏

Windows 操作系统提供 API 函数 EnumProcesses 和 CreateToolhelp32Snapshot 系列函数遍历当前进程。这些函数最终将调用由 ntdll. dll 导出的 NtQuerySystemInformation 函数。通过此函数可以得到进程信息、线程信息、句柄信息、加载模块信息等。通过修改 SDT 表中 NtQuerySystemInformaion 的表项,使之指向一个新的隐藏程序添加的例程 MyNtQuerySystemInformaion,该函数再调用原函数获得进程的相关信息,并过滤需要隐藏的信息即可。

(2)修改 ActiveProcessLink 实现进程隐藏

NtQuerySystemInformation 函数将调用未被系统导出的函数 ExpGetProcessInformation 来获得进程信息。而 ExpGetProcessInformation 则是通过遍历一个双向循环链表来实现其功能的,该链表就是 Active Process Link。系统将所有的进程对象链接到 Active Process Link 上,该链表由一个未导出内核全局变量 PsActiveProcessHead,通过遍历 Active Process Link 可以获得所有的进程信息,NtQuerySystemInformation 函数最终遍历该链表返回进程相关的信息。如果将某一个 EPROCESS 对象从 PsActiveLink 链表上摘下来,NtQuerySystemInformation 将无法检测到该进程。因此,通过调用 PsLookupProcessByProcessId 例程得到指向指定 ProcessId 的 EPROCESS 对象的指针,然后调用 RemoveEntryList 将结构从链表上摘下。

2.2 端口隐藏

很多恶意代码都采用 TCP 协议进行网络通讯,工具 netstat 和 fport 都可以列举当前建立的所有连接。netstat 调用

*)基金项目:国家科技基础条件平台工作基金资助项目(2003DIA7J051)。曹 跃 助教,硕士生,研究方向为计算机系统安全、计算机网络安全。

API 函数 SnmpExtensionQuery 实现对 TCP 端口的查询功能，该函数的调用情况由 ntdll. dll 导出的 NtDeviceIoControlFile 函数，其中参数 FileHandle 是\Device\Tcp 设备对象对应的文件对象句柄，IoControlCode 是一个内部的命令号，告诉 TCP 设备对象，将查询 TCP 连接情况，返回的端口信息将被送入输出缓存。

恶意代码常常使用内嵌拦截(inline-hook)由 ntdll. dll 导出的 NtDeviceIoControlFile 函数来实现端口隐藏。Hook 是一种改变程序流程的技术，在进程隐藏中使用的修改 SDT 的方法即是一种 hook 技术。此外还可以通过修改模块导入表，虚函数表，修改指令等方法来实现 hook。

2.3 文件隐藏

很多恶意代码开始使用文件隐藏技术，使检测程序找不到恶意代码的文件。恶意代码主要使用以下方法实现文件隐藏：

(1)内嵌拦截 FindFirstFile 和 FindNextFile 实现隐藏

Windows 提供 API 函数 FindFirstFile 和 FindNextFile 来遍历一个文件夹中的文件，采用与前述隐藏 TCP 端口类似的方法在 FindFirstFile 和 FindNextFile 函数上进行内嵌拦截即可实现文件隐藏。

(2)拦截 SDT 实现文件隐藏

与前面的进程隐藏类似，FindFirstFile 和 FindNextFile 最终也将调用由 ntdll. dll 导出的函数 NtQueryDirectoryFile。该函数实现对文件和文件夹的遍历，遍历得到的信息将以数组的形式存放在 FileInformation 缓冲区中。

恶意代码通过修改 SDT 中 NtQueryDirectoryFile 函数的地址就可以在文件查询时获取控制权，然后调用原始的 NtQueryDirectoryFile 例程遍历文件，当发现遍历到的文件中含有需要隐藏的文件时，修改 FileInformation 删除相应的信息。

2.4 注册表隐藏

注册表查看工具通常调用 Windows API 的函数 RegEnumKeyEx 和 RegEnumValue 来列举注册表键和键值。这两个 API 函数将分别调用 NtEnumerateKey 和 NtEnumerateValueKey 来实现其功能。其中 KeyInformation 和 KeyValueInformation 分别为这两个函数返回的键和键值的信息，通过拦截这两个函数修改其返回的信息就可以实现注册表的隐藏。恶意代码通常都使用拦截 SDT 的方法实现对注册表的隐藏。

3 隐蔽行为检测

3.1 基于差异分析的检测技术

通过前面的分析，可以发现隐藏技术包括两种手段：直接修改数据或者修改代码。隐藏的最终目的是让上层的查询函数得到虚假的查询结果。因此如果能从某种途径获得真实的系统信息，将之与上层的可能被修改了的信息进行对比，如果发现有不同的地方，则存在隐藏行为。根据前面的分析，提出了基于差异分析的存在性检测技术，如图 1 所示。

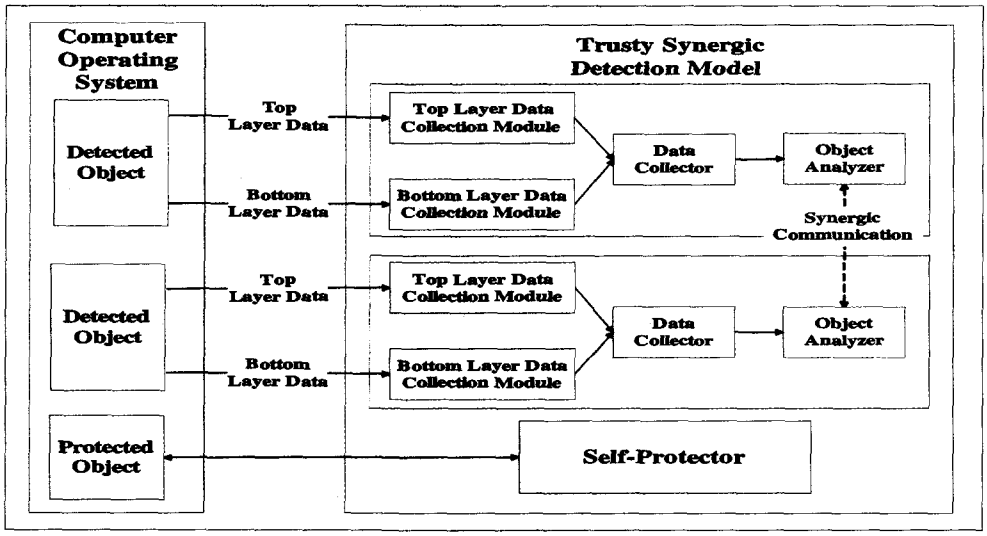

图 1 基于差异分析的隐蔽恶意代码检测模型

其思路与步骤如下：

(1)普通应用程序的方式，从用户态上层调用系统 API 对系统进行扫描。假设此时系统已经有恶意代码并使用了隐藏技术，上层调用返回的信息会在某个地方被修改，获得不可信任的数据。我们将本次扫描结果进行转储，设为不可信任系统信息。

(2)将操作系统底层很难被恶意程序修改的函数或数据称为可信任层，从可信任层获取相关的系统信息是真实可靠的。将从可信任层扫描获取的信息进行转储，设为可信任系统信息。

(3)将两次扫描转储后的系统快照，输入到一个对象分析器中进行比较。一旦两种扫描方式的结果有所不同，则基本可以断定，高层扫描时的结果经过了恶意代码的修改。

3.2 获取不可信任信息

用户程序可以通过调用由 psapi. dll 导出的 EnumProcess 函数获取当前运行的所有进程的 ID。用户程序调用由 Iphlpapi. dll 导出的 GetTcpTable 函数可以获得当前的 TCP 端口连接信息。通过调用 Advapi32. dll 导出的 RegEnumKey 和 RegEnumValue，用户程序可以获得注册表键和键值的相关信息。通过调用 FindFirstFile 和 FindNextFile，用户程序可以遍历并获得指定目录中的所有文件信息。

3.3 获取可信任信息

基于差异分析的行为检测技术的关键是如何获取可信任的数据。获取系统信息主要有两种手段：调用系统查询例程或者直接读取系统数据。系统提供的查询例程最终仍然是通过读取系统数据完成查询的，而直接读取的数据显然可以避开使用拦截技术进行信息隐藏。

(1)可信进程信息

为了保证系统的安全性，应用程序不能直接访问内核对象，而必须通过句柄访问。句柄是一个与进程相关的不透明32位值。Windows为每一个进程维护一个句柄表，句柄值是在句柄表中的索引，在不同的进程中同一个对象的句柄值并不相同。PspCidTable是一个特殊的全局句柄表，它不输入任何进程，其所对应的句柄项也很特殊，就是进程的ID。因此，调用PsLookupProcessByProcessId的过程实际上就是一个进程ID号映射到对象指针的过程。可以看出，通过读取PspCidTable句柄表中的有效值进行映射就可以实现对当前所有进程的映射，这种方法将不会受到前面提到的进程隐藏技术的影响，从而获得可信的进程信息。

(2)可信端口信息

由于当前的恶意代码通常都是通过拦截NtDeviceIoControlFile函数实现隐藏的，通过直接调用内核态ntoskrnl.exe中的NtDeviceIoControlFile函数来读取的文件相关信息是未被修改且可信的。但是拦截SDT时，指向NtDeviceIoControlFile的表项已经被修改了。要获得真正ntoskrnl.exe中的NtDeviceIoControlFile函数，可以通过直接读取ntoskrnl.exe文件中的SDT表项来完成。由于恶意代码可能采用内嵌拦截内核态中ntoskrnl.exe的NtDeviceIoControlFile函数，只是从文件中读取该函数的地址仍然不能保证不被拦截。为防止内嵌拦截，使用了代码覆盖技术。

(3)可信文件信息

和进程、TCP端口等查询一样，对文件的查询也会通过调用ntdll.dll导出NtQueryDirectoryFile进入内核态，查询SDT并调用相应的例程处理。不同的是，由于支持多种文件系统，内核态的处理例程会根据文件系统的不同，调用不同的驱动程序完成具体的文件查询。对于NTFS文件系统，会调用ntfs.sys；对FAT32文件系统会调用fastfat.sys。通过直接调用文件系统驱动程序ntfs.sys或fastfat.sys获取的文件信息是可信的。

(4)可信注册表信息

由于注册表中的信息要求在系统重启后仍然有效，也就不可能保存在内存中。所以，注册表信息只能保存在磁盘的文件上，这种文件被称为Hive文件。在Windows系统中，可以通过使用API RegSaveKey/RegSaveKeyEx函数来将指定键下的注册表信息转储为Hive格式的文件。当然，这些文件是通过API调用来实现的，这也就给恶意程序留下了可能的漏洞，虽然现在还没有发现恶意程序采用该机制，但从理论上说，恶意程序可以通过拦截API来避免被dump出来以逃避检测。在获取了注册表Hive文件的基础上，可以实现对它们进行完整的检测。在此，借鉴了NT Registry Hive Access Library，通过其对Hive文件的分析，另外设计了一个实用的Hive文件访问接口，利用该接口可以用来分析Hive文件存储内容，获取其中信息。

4 实现与测试

4.1 系统实现

目前，我们已经在Windows平台上实现了基于差异分析的Rootkit检测原型系统Aigis，系统设计的重点集中在不可信数据采集器、可信数据采集器和对象分析器三部分。分析器主要包括对系统进程、文件、注册表和网络通信隐蔽行为的分析检测。该原型系统在Windows 2000 Server平台上开发完成，使用VC++6.0作为开发工具，并结合Windows驱动程序开发包(Windows DDK)实现底层的开发功能。

4.2 测试结果

我们选择了Windows XP Professional版本作为测试的操作系统平台。测试的安全检测工具包含反病毒软件、木马检测软件与Aigis检测系统，它们为eTrust、Norton、Macfree和Anti-Trojan Shield。同时，选择了三个具有代表性的Rootkit程序Hacker Defender、Vanquish和Fu，以及一个未公开的Rootkit程序Sh4d0ws作为测试的对象，它们分别基于不同的实现机制。在此基础上，还选用了两款加壳变形工具PECompact和UPX，以改变Rootkit程序原始二进制文件的特征码，实现变形检测。将每个Rootkit及其相应的变形版本安装于测试操作系统上，然后分别使用所列安全检测软件对系统进行扫描检测，并将扫描结果汇总于表1中所示。

表1 Aigis与反恶意代码软件的Rootkit检测结果对比

	eTrust	Norton	Macfree	Anti-Trojan Shield	Aigis
Hacker Defender	√	√	√	×	√
Vanquish	×	×	×	×	√
Fu	√	√	×	√	√
Sh4d0ws	×	×	×	×	√
UPX+Hacker Defender	×	×	√	×	√
PECompact+Vanquish	×	×	×	×	√
PECompact+Fu	×	√	×	√	√

基于以上测试结果，可以看出Aigis检测系统能够检测出所有类型的Rootkit程序，而现有的反病毒反木马检测工具则显示出了巨大的缺陷，面对新兴的隐蔽型恶意程序Rootkit及其变形版本的攻击，她们失去了以往的效果。

结论 本文的研究工作对基于Windows系统的恶意代码检测技术进行了有益的探索，采用本文所提出的技术实现了一个具有鲜明特色的恶意代码检测系统Aigis，该工具在某些性能上明显优于现有商业恶意代码检测系统。

随着信息技术和网络技术的进一步发展，新的恶意代码将不断涌现，结合目前恶意代码的信息安全的发展趋势，我们认为以下工作有待进一步研究。由于现有恶意代码主要依靠网络传播，恶意代码的防治只依靠主机的检测是远远不够的，还需要结合网络入侵检测、审计等措施才能有效确保系统的安全。恶意代码的入侵很大程度上是由于系统存在漏洞，对于Windows这样的未开源系统来说，对未知漏洞的发掘和对已知漏洞的保护工作对于防范恶意代码显得尤为重要。

致谢 感谢国家科技部科研院所专项基金委对本项课题的大力资助。同时，本文工作得到实验室研究生刘颖、黄沾、崔甲等给予的大力支持与帮助，对此表示衷心的感谢！

参考文献

1 Mukherjee B, Heberlein L T, Levitt K N, et al. Network intrusion detection. IEEE Network, 1994, 8(3):26～41

2 Leu F Y, Yang T Y. A host-based real-time intrusion detection system with data mining and forensic techniques. In: IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, Taiwan, 2003

3 Fiskiran A M, Lee R B. Runtime execution monitoring (REM) to detect and prevent malicious code execution. In: Proceedings of the IEEE International Conference on Computer Design (ICCD' 04). San Jose, CA, 2004

4 Deng P S, Wang J H, Shieh W G, et al. Intelligent Automatic Malicious Code Signatures Extraction. In: IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, Taiwan, 2003

5 Hoglund G, Butler J. Rootkits: Subverting the Windows Kernel. Boston: Addison-Wesley, 2005