反病毒产品的兼容性问题白皮书

（2010 年 7 月修订版）

安天实验室

一、 反病毒产品兼容冲突问题概述

反病毒产品从最开始的行命令扫描工具发展至今、已经形成了带有文件、注册表、内存多种本地监控机制；浏览器、邮件客户端等多种保护环节；以及整合了主机防火墙、入侵检测机制、主动防御机制的综合型产品。而其核心价值早在上世纪末，就已经从最开始的静态的文件扫描检测，变成实时化的主机防护。而实时化的防护推动了反病毒产品使用了更多的服务、驱动等底层技术，运行于更接近系统内核的位置，这也为反病毒产品相互产生兼容性问题打下了伏笔。

随着操作系统的复杂化，威胁的不断离散化等影响，反病毒的监控保护点野日趋复杂，又加之反病毒厂商数量也在增加，而操作系统厂商并没提供比较标准的技术规范，因此反病毒产品互不兼容，发生共存冲突等带来的问题越来越多，并间接影响了用户对反病毒产品的信任。

目前已经发现并被报道过的兼容性问题包括：

- **造成系统崩溃和其他严重故障：**从 2000 年以来，根据公开报道，已经出现多起反病毒产品之间发生相互冲突，导致系统蓝屏、死锁等事件。
- **资源占用：**反病毒扫描、监控和其他防护机制，都会带来系统资源的占用。如病毒库的内存展开对内存的资源使用，文件监控、定时扫描可能导致更多的 I/O 开销、以及各种保护机制对 CPU 时间片的占用等等。这些对用户操作有一定影响，如系统延迟感、文件复制操作时间变长等等。而由于消息传递等机制，有可能在多种反病毒产品共存时，其对资源和时间的影响不是简单的线性叠加，而出现明显的性能恶化。
- **失效：**由于监控机制之间的冲突，多种监控机制共存时，有可能造成其中之一失效或者、部分机制双双失效的风险。上述后果，可以在兼容性测试中被浮现。
- **相互误报：**由于厂商之间互信互通机制尚不够通畅，以及少数恶意的“误报构造”攻击的存在，厂商之间出现相互误报的情况，也时常发现。由于被报警为病毒严重的影响用户的产品信心。因此，各厂商对被误报问题也均比较敏感。

但需要指出的是，绝大多数情况下，反病毒产品之间的兼容性冲突问题，都是技术与协调的问题，有其工作机理上的必然性，并往往具有一定的不可避免性。相关问题多数并不是由商业竞争引发的，商业竞争也不是反病毒产品兼容冲突问题的本质。本白皮书主要介绍当前反病毒产品冲突的主因，以澄清公众误解。

下文中涉及到安天和兄弟厂商产品所使用的技术点，均用于说明反病毒产品之间兼容性冲突的必然性，不是对实现水平进行评价。

©安天实验室 版权所有

二、 主要冲突点详细解析

实时监控、主动防御等技术的发展是一柄双刃剑，一方面随着监控点的增加应对新的安全威胁，一方面也使反病毒产品的稳定性遇到挑战，测试难度普遍加大。反病毒产品自身的稳定性压力都在呈现几何级数增长，相互之间的兼容则变得更加困难。以下从文件监控、防火墙、浏览器防护、主动防御四个主要技术点，描述导致反病毒产品自身的稳定性下降、与操作系统之间和相互之间出现严重的冲突问题的成因。

2.1 文件监控的潜在兼容性问题

文件监控是反病毒厂商普遍采用的技术手段，其主要机理是对文件的创建、读取、关闭等行为进行监控触发对文件病毒检测，以阻断病毒的执行和部分相关行为。

文件监控的实现方式主要有以下两大类型：

1. API 挂钩。根据钩挂 API 的层测不同我们又可以分为：

1) Ring3 文件监控。多采用 inline hook 的方式修改文件操作相关函数的前几个字节，跳转到自己的函数，然后对将要操作的文件和缓冲区进行检测。、

a) 由于实现跳转的方式有多种，所以不同软件的实现策略有不同，很难保证多个软件共存时前一软件修改后不影响到后一个软件，而且很可能导致相关进程**崩溃**。

b) 由于此类技术也多被恶意软件使用，一些安全软件不会在执行完自己的代码后执行修改前的自己，而是采用从原始文件中读取分析，执行的方式，导致多个安全软件同时监控一个 API 时，只有一个生效。

c) 出于性能考虑，不传递给原 API 处理而采用自己实现的代码完成该 API 的相应功能，那么也不会将相关信息传递给其他软件。

d) 部分软件在执行完自己的函数后会把修改后的字节还原，然后调用原系统 API 完成功能后再次修改、挂钩。两个使用此实现的软件同时工作则可能造成互相调用导致死锁，**程序没有响应**。

2) Ring0 文件监控。与 ring3 的情况类似，但是后果更加严重。

a) 由于 inline hook 不同实现的不兼容性很可能导致系统 API 无法正常工作，导致系统**蓝屏**。

例如：安天客户端产品（Antiy Ghostbusters 4 .0）早期版本采用大量使用 inline hook，后因与其他产品严重冲突放弃； 360 安全卫士中 inline hook 有几种不同实现方式共存的现象，以及与 360 安全浏览器有重合的监控点。

b) 挂钩之间相互调用，会导致死锁，**系统死机。**

c) 采用替换 SSDT 的方式，则只有自己的挂钩有效，**其他软件的挂钩均无效。**

例如：360 安全卫士等产品采用替换 SSDT 的方式，与其共存的安全软件通过标准方法获取的 SSDT 是无效的，导致其他安全软件功能出现缺失。

©安天实验室 版权所有

d) Vista 以后微软对内核进行保护，部分对内核函数的修改会导致**系统蓝屏。**

例如：多个产品在内测和对外版本都出现过对关键内核函数进行修改，但是并没有仔细判断版本，导致兼容性出现问题系统蓝屏。

2. 文件过滤驱动

1) File System Filter Drivers

2) File System Minifilter Drivers

过滤驱动的本质依然是挂钩，但微软为了方便厂商调用，对其进行了封装，属于官方的技术。此类技术本身已经力求稳定与兼容，多数都提供了一定的兼容性保障。微软自己的安全产品在驱动层面也多使用类似的方式。

处置方式	交给后续过滤函数继续过滤	直接交给下层函数处理	拒绝访问
兼容问题现象	扫描次数多、系统变卡、变慢	后续过滤不生效，实际上只有一个安全软件在保护主机。	后续过滤不生效，只有一个安全软件报告病毒。

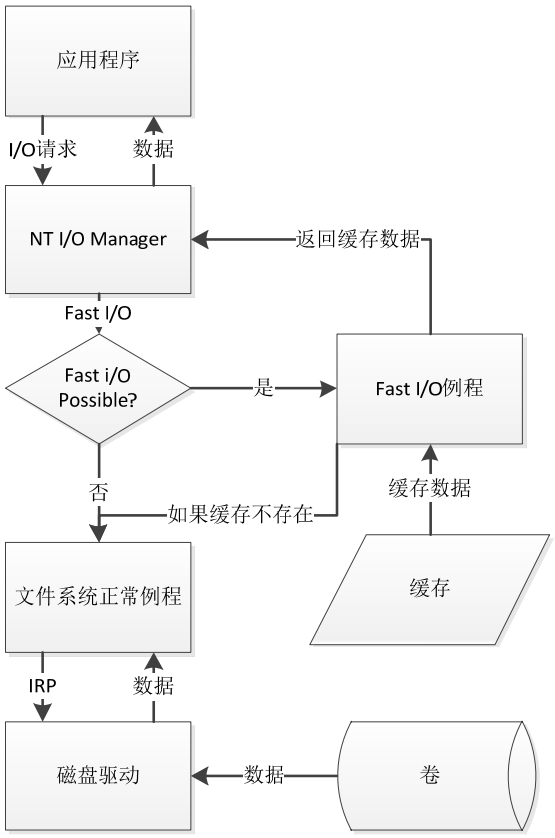
但是在安全产品的实际应用中，多层监控逐层传递会造成效率的降低，一些厂商在实现时会绕过后续的过滤驱动，直接将 IRP 请求发给下一层驱动完成相关的功能。

2.2 防火墙的潜在兼容性问题

目前反病毒产品普遍使用单机防火墙技术来过滤出入数据，应对来自网络的威胁。

目前防火墙的实现方式包括：

- TDI
- NDIS
- IP Filter
- Windows Filtering Platform
- Winsock Kernel

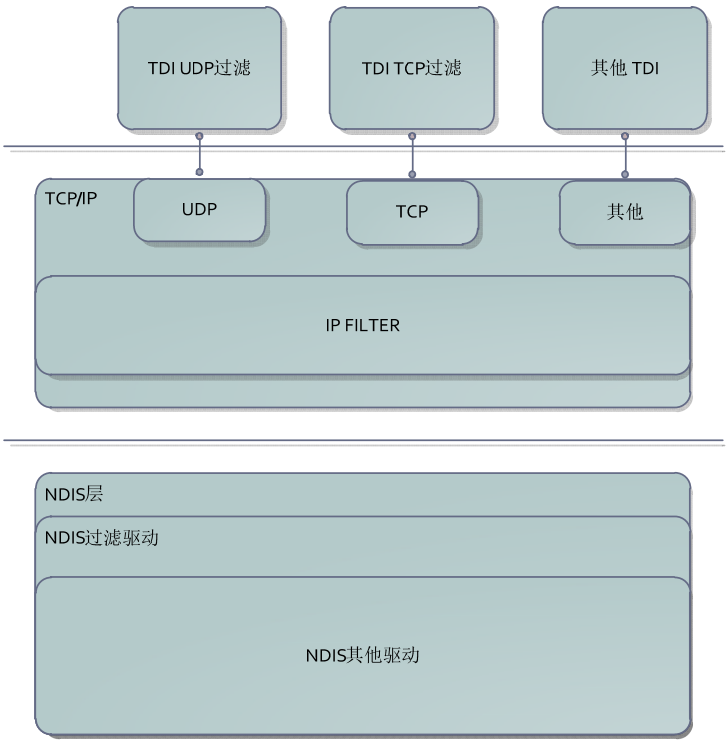


在 windows XP 为主的时期，多数防火墙选择前三者中的某一种或者某两种结合的方式。在实际应用的过程中，同样存在多个共存无法同时生效的现象。其原因有：

过滤时出于效率考虑，直接将允许通过的包交给下层驱动处理，而不是后续的防火墙驱动。

由于 NDIS 和 IP Filter 比 TDI 更接近底层，前两者实现的防火墙会对 TDI 层造成影响。

XP 自带的 IP Filter 存在谁先设置谁先生效的问题。安天盾防火墙与金山防火墙某版本同时使用该方式，如果同时安装在用户的机器中则出现**谁先启动谁先生效**的现象。



WFP 和 Winsock Kernel 是自 vista 之后引入的方式，兼容性相对而言更好，只要实现得当，一般没有兼容性的问题。特别是 WFP 针对防火墙和 IDS 类软件做了设计上的考虑，兼容性问题得到了较好的解决。这也说明，操作系统提供相对规范的接口是解决安全产品冲突的较好方式。

2.3 浏览器防护的潜在兼容性问题

浏览器防护多使用 ring3 API HOOK。下面结合网页防护类的主流产品分析其潜在的兼容性问题。

拦截点	金山网盾	360 网盾	锐甲	作用
BHO	BeforeNavigate	BeforeNavigate	BeforeNavigate	拦截网址导航
wsock32		recv		特征检测
Ws2_32		recv		特征检测
	WsaSend	send		拦截请求的 URL
Wininet	InternetOpenUrl			
	InternetQueryDataAvailable			
	InternetConnect	-	InternetConnect	拦截请求的 URL
	InternetOpenUrl	-	InternetOpenUrl	拦截请求的 URL
	InternetReadFile	-		拦截请求的 URL
	InternetQueryDataAvailable	-		拦截请求的 URL

	HttpOpenRequest	-	HttpOpenRequest	拦截请求的 URL
Vbscript.dll / jscript.dll	compile	ParseScriptText		对浏览器执行的脚本进行分析
Kernel32	CreateProcess(Internal)	CreateProcess(Internal)	CreateProcess(Internal)	
			CreateFile/Write/Read	文件格式溢出检测
	CopyFile	CopyFile(Ex)/MoveFile		
	LoadLibrary	-	LoadLibrary(Ex)	
			VirtualProtect	防止内存修改和溢出
	WinExec	-		
Ntdll	NtCreateProcess(Ex) / ZwCreateProcess(Ex)	-		
Oleaut32	SysAllocStringLen	-		
	SysAllocStringByteLen	-		
	CoRegisterClassObject			
	CoCreateInstance	CoCreateInstance	CoCreateInstance	CLSID 检测
	CoGetClassObject	CoGetClassObject	CoGetClassObject	CLSID 检测
Urlmon	UrlDownloadToFile	UrlDownloadToFile	UrlDownloadToFile	
	UrlDownloadToCacheFile	UrlDownloadToCacheFile	UrlDownloadToCacheFile	
Advapi32	RegQueryValueEx			
			CreateProcessAsUser	
Shell32			SHCreateProcessAsUserW	
			ShellExecute	
		ShellExecuteExW	ShellExecuteEx	
comdlg32			GetSaveFileName	
WinTrust			WinVerifyTrust(Ex)	数字签名分析
浏览器行为分析	进程创建	进程创建	进程创建	
	模块加载	-	模块加载	
	字符串分配(堆喷)	-		
	文件下载	文件下载	文件下载	
	文件拷贝	文件拷贝		
		ActiveX 对象创建	ActiveX 对象创建	

其他	内存占坑 + 保护		内存溢出防护+栈追溯	堆喷防护
			OnDocument Complete	搜索结果标识、部分钓鱼识别

表 1 主流安全防护软件监控点

使用 ring3 api hook 实现的浏览器保护，无可避免的要遇到和文件监控一样的问题。在实际对抗中，网马不断发展升级，已经开始将浏览器中的 ring3 hook 摘除，为了对抗此类威胁，主流软件多会采取不断检测自己的 hook 是否存在，发现不存在则重新将函数挂钩。造成多款软件争抢一个 API 的现象，**导致浏览器无响应或者响应缓慢**。特别是 CreateProcesss 相关的。

在浏览器防护软件进行行为分析的时候，由于其他浏览器防护软件的参与会对浏览器的行为造成一定影响，这些影响可能是对堆栈的影响（由于多了一层 Hook 实际调用的堆栈可能会增加一层），以某浏览器防护软件为例，其堆栈检测只会向上追溯 5 层，如果同时共存 3 个软件，那么其想要检测的那层调用，则会由原来的第 5 层变为第 7 层，原来的**检测方式就失效了**。

例如：360 和金山网盾共存时，网马在创建进程时 360 进行检测如果放行则会通过 360 的函数调用系统 API，进而再次调用金山的进程创建检测，此时金山追溯堆栈会发现是 360 的模块调用，而不是 js 脚本解释引擎调用，予以放行。

即使不考虑对堆栈的影响，由于不同浏览器防火软件都进行了拦截，对浏览器的行为也就产生了影响，造成相互的行为分析干扰，无法正确分析恶意行为，进而出现**漏报或者误报**的现象。

例如：金山网盾在脚本解释层根据特征检出了恶意脚本，构筑在其上层的其他防护的行为分析就无法发现该恶意脚本的行为。

再例如：360 网盾会不断检测监控点是否存在，而检测的机制是基于二进制比较的，如果发现其他软件进行了 HOOK，则用自己的进行替换，导致其他软件失效。

2.4 主动防御的潜在兼容性问题

目前反病毒产品也普遍采用了一些不依赖于内容规则的，综合的行为判断和阻断机制，一般通称为主动防御技术。

在多款软件共存的情况下，主动防御的兼容性问题尤为严重。主动防御主要可以分为两个方面来看待：

1. 自我保护

   a) 多款安全软件都保护自己的监控点不被修改，特别是使用了 inline hook 或者对系统设备、内核对象、SSDT 进行了 hook 的产品。类似安天 Atool 等 Rootkit 检查工具曾使用过替换进程 SSDT 保证自己的 hook 不被修改，但也导致使用 SSDT hook 的主动防御完全失效的副作用。360 安全卫士曾使用替换 SSDT 的技术对自己的监控点进行保护，导致与其共存的安全软件主动防御功能中关于 SSDT 的部分完全失效。

©安天实验室 版权所有

b) 一旦发现自己的监控点被修改，则会对监控点进行修复，也就是重新 hook。这就有可能导致多款安全软件反复争夺监控点，或者 hook 直接互相调用造成死锁，最终耗尽系统资源，导致机器**死机**。

c) 由于自我保护的存在，病毒感染安全软件后，依然会被安全软件的自我保护所保护，其他安全软件可能无法读取其内容进行检测，或者可以检测，但是**无法结束相应的进程**。

2. 恶意行为分析

a) 一款软件处于安全角度考虑，不调用被 hook 的原始 API 则会导致，其他软件在分析恶意行为时，无法检测到该行为，进而造成程序的行为序列发生改变，最终导致行为分析**误报或者漏报**。

b) 由于安全软件的某些行为和恶意软件具有相似性，例如：对 API 进行的某些 hook。在多款安全软件共存的情况下，很有可能一款安全软件被另一款报为病毒，这也是一种**误报**。

c) 为了保证自身进程的行为不被重复记录或者对行为分析产生影响，安全软件可能会对特殊 API 的调用参数含义进行修改，增加自身需要的标识，而这些标识可能会造成后续的监控产生不可预知的行为。最糟糕的情况就是导致**系统蓝屏**。

d) 对于 ring3 与 ring0 结合判断的主动防御，处于二者中间的其他安全软件对数据的修改可能会造成 ring0 缓冲区的溢出（例如：ring3 的 api 挂钩通知 ring0 的驱动，需要 26 字节实际数据可能由于中间沙箱软件的路径重定向被改为 27 字节或者更多）或者被截断，导致**系统蓝屏**或者**漏报**。

以上仅仅是主动防御潜在兼容性问题的一部分，由于主动防御技术本身的复杂度，在多款实用主动防御技术的安全软件共存的情况下兼容性问题就跟容易出现也更加严重，这里仅仅是说明的一部分，不是全部。

2.5 兼容性问题对反病毒厂商的影响

兼容性是反病毒厂商的核心困扰之一，由于反病毒使用大量的内核技术、驱动等，一旦出现兼容性的后果，其所造成的影响，要远严重于其他应用软件。同时反病毒产品的为了对抗病毒的一些自我防护机制，也会使问题的处置变得比较复杂。因此， 而多种反病毒软件之间冲突引发的问题要比其他软件冲突问题后果更加严重。

兼容性冲突问题使反病毒厂商技术支持的难度增大，由于环境的复杂性，问题变得更加难以排查和处理。

特别是企业版产品，厂商承担着更大的责任和支持义务，如果由于冲突性问题，带来问题，对于问题的责任、后果的认定等方面都带来较大压力。

同时，有的冲突问题由于需要厂商间相互沟通的才能解决，因此增加了支持压力，和解决用户问题的周期。

2.6 解决冲突问题的约定俗成规则

在实时监控技术在 windows 体系下刚刚成熟时，能够提供相关机制的厂商不多，监控点

也相对较少，因此在出现兼容冲突问题时，厂商之间可以相互改造和规避。后来由于安全厂商过多，而监控点也在不断增加，解决相关问题的复杂度已经超出了每个厂商的个体能力。

因此在经历了大量冲突事件后，部分主流厂商选择了从安装环节保证互斥的方法。其具体操作时，在安装时检测用户系统中是否有其他反病毒产品，如果有则提示用户可能的冲突后果，并提示用户卸载，如果用户不卸载，则明示用户共存后果，或者选择自身不予安装。但先被装载到主机上的反病毒产品，并不监控其他反病毒产品的安装行为，以及进行提示。这就是约定俗成的“后卸前”原则。

这种方法虽然影响到了一定用户让多种反病毒产品共存的意愿，但其形成了一个单向的逻辑，即后安装的有告知和卸载的主动权，先装入主机的接受用户的选择。

这个过程是由一定合理性的：

1、 通过互斥一定程度上解决了兼容性问题。
2、 保证了用户的知情权、选择权和自觉权。用户是在被告知后果的情况下，做出了符合个人意愿的选择。
3、 用户在后一个反病毒产品的体验中，如果觉得不如之前的软件，可以通过再安装前一个软件的方法，重新选择之前的产品。这确保了厂商之间的竞争基本上是用户体验和效果的竞争，而没有赖在用户机器上不走的情况。

但互斥性的方法，所带来的问题则也显而易见，那就是用户很难同时分享两个产品的保障。而这对于地下经济产业链所驱动的木马小众化、数量爆炸化的严峻安装形式，只靠任何一个厂商的力量，可能都很难保证用户安全，相关产品能够合理共存、机制互补更符合用户价值的最大化。

三、 目前的问题和我们的建议

需要指出的是，尽管反病毒厂商之间存在着长期的竞争，但主流厂商之间的技术层次的互通；技术资源（如病毒样本）的分享是一直存在的。加之管理部门、测评机构、CERT组织等的协调和努力，各厂商之间基本遵循了一些基础的原则。在历史上较长的时间内，尽管出现过各种商业摩擦，但一直都把兼容冲突作为期望回避的问题，而不是作为竞争的筹码和手段。有关问题的复杂化和扩大化是后期才出现的。

这说明任何没有明确行业规范为支撑的原则，都是脆弱的。在后卸前的用户自主选择链条中，只要有一家采用不良手段[如通过反人机工程的方法，在竞争对手产品安装过程中，给与多次提示，造成用户一次选择不当就无法安装、或者安装后失效等]，保护自己所谓装机率，则就会连带造成被拦截的厂商进行报复，从而导致规则被打破，形成恶性循环，最终导致通过恶意构造兼容性壁垒和陷阱变成普遍性的行为。因此，自然形成的事实标准需要立法过着成型行业规范保障，才能具有权威性。

反病毒产品作为主机安全的核心屏障，不仅要使用户远离安全威胁，也要保护用户的知情权和选择权。尊重用户卸载意愿，包括在保证稳定性基础上，尊重用户希望多个安全产品共存的愿望，不仅是用户权益的体现，也是厂商不怕被用户比较选择的自信心的体现。因为这个行业之所以前进，正是是用户拥有不断尝试和选择的机会，这是每个厂商的动力之源。

©安天实验室 版权所有

同时，我们需要意识到用户期望多安全产品共存，是当前严峻的安全形势所催生的真实需求，反病毒产品如果能提供更多的定制选择，如提供行命令的扫描工具、提供扫描界面和监控机制[甚至不同监控机制]的可定制安装，就能让用户在获得多个厂商的检测能力的基础上，较少的付出兼容性代价。如果安全厂商之间需要进一步强化其互通体制和配置策略，未来能够支持用户更灵活的定制多产品共存的监控策略，则更是一种更为理想的用户境界。

而主流操作系统厂商如果能对监控基础给出更为丰富的接口和定义相应的规范，包括保证公共安全接口统一嵌套化，则能进一步降低发生冲突的概率。

当然，我们希望用户理解，只要反病毒产品的实时监控和防御的使命存在，不同产品之间的兼容冲突问题就必然存在，不可能有终极解决之道。我们也希望传达给用户的信息是大量安全厂商所研发的充满不同个性的安全产品，是攻击者需要逾越的一个整体代价。只有八仙过海、各显岂能的厂商个性存在，才能保证网络整体安全水准。多个厂商的共存和竞争，是互联网安全的基础保证。

©安天实验室 版权所有