反病毒产品的误报问题白皮书

（2010.09 修订版）

安天实验室

近日，多起反病毒产品误报误杀的事件被媒体关注和报道，引发了一些公众的猜疑和恐慌。安天实验室本着客观公正的态度，发布本白皮书，期望通过阐述反病毒产品的工作原理、归纳和分析误报的形成原因，让公众对相关问题有所认识和理解，并澄清误解。

一、 误报的概念与特点

1.1 相关概念

反病毒产品：以判别和处置计算机病毒为主要功能的软件或硬件产品。

检测对象：提交给反病毒产品判断是否为病毒的一个确定性内容，其形态可以是文件、内存内容或者网络数据。

报警：反病毒产品认定一个监测对象是病毒或包含病毒，并提供病毒名称及相关信息的行为。

1.2 什么是误报？

误报，是指反病毒产品检查一个可以确定不是病毒的检测对象时，将其报警为病毒。

漏报和错报这两个概念与误报相似，经常被人们混为一谈。因此，同时给出它们的明确定义，以便区分：

漏报，是指反病毒产品检查一个可确定是病毒的检测对象时，没有报警。

错报，是指反病毒产品检查一个可确定是某一种病毒的检测对象时，将其报警为另一种病毒。

1.3 对误报的基本观点

在反病毒行业，尤其是工程实现方面，误报的问题比漏报和错报更加敏感。这是因为：误报既造成用户的心理恐慌，又可能对被误报产品产生不好的舆论影响，更是会直接导致误杀、使用户系统出现不可预期的后果。反病毒技术应以保障用户系统的可用性为前提，极低的误报率，是商用反病毒技术的基本要求。

工程界普遍以严谨的态度看待误报，对误报率的约束之严格，远远超过漏报与错报。例如，有非官方技术标准规定：反病毒产品的误报率不能超过万分之零点五，即对十万个不同的检测对象，最多允许五个误报。即便如此，无论是公众还是反病毒工作者，还是希望产品能达到更低的误报率。

学术界的视角与工程界有所不同。一些研究人员通过各种技术与方法（比如神经网络）来获得较高的未知病毒检出率，往往能得到令人鼓舞的结论。然而，除非根本性的重大技术突破，对未知病毒的高检出率一般都伴随着高误报率。大量优秀的成果，是以牺牲了误报率为代价的。

任何先进的检测手段，只要误报了一个系统关键文件或正常文件，就无法不加改动地直接用于真实系统。正因为如此，尽管学术研究的成绩显著，但难以用于真实的产品。实践中，反病毒产品如果能有超过 10%的未知病毒检出率，就可以认为是很不错的结果了。

反病毒工作者依然要努力探索新的病毒检测理论和方法。只有不断突破现状、开拓创新，才能实现真正的技术进步，推动行业的良性发展。

二、 误报的类型与产生环节

2.1 误报的技术性与争议性

前面已经介绍，误报即反病毒产品将非病毒对象判定为病毒。从技术角度，误报是产品中的问题。

但在现实生活中，误报也可能是一种争议。产生争议的原因在于：判断一个对象是否病毒（或恶意代码）的标准，不同的人有不同的看法。这种争议通常不是技术上的问题，也比技术上的误报要更加普遍和难以应对。

比较常见的一种标准争议是广告件（Adware）厂商与反病毒厂商之间经常发生的冲突和诉讼。

广告件厂商为了提高软件的装机量，往往采取一些比较极端的推广方式，比如：与其他软件的安装程序捆绑、不经用户确认强行安装、页面注入、通过蠕虫或僵尸网络分发、阻止用户卸载等。为了提高广告的点击率，通常频繁弹出广告、修改浏览器功能等。这些行为常常遭到反病毒产品的查杀，这触犯了广告件厂商的根本利益，从而引发各种冲突甚至法律诉讼。更有一些广告件厂商，以极端方式推广并取得满意的装机量之后，为了避免查杀，马上从软件中移除有争议的技术，并寻求公证机构的公证，最后以此为据，与反病毒厂商交涉。

另一种标准争议是对工具软件的定性。一些工具软件的作者为了规避法律责任，坚称自己所开发的软件不是后门、木马，而是远程管理工具。另一方面，从反病毒厂商的基本技术标准来看，远程管理工具与后门的根本区别在于：前者的被控制端对用户是可见的（例如，运行时在系统托盘中有明显的图标），并且安装的时候需要经过用户的确认同意。但是这些软件通常不满足上述两点。

还有一种标准争议是，反病毒产品除了查杀对本地主机有害的恶意文件，是否还应该查杀运行结果会危害其他主机的检测对象？例如，1998 年 Back Orifice 后门工具开始流行后，一些人认为反病毒产品不应该查杀计算机中用于控制受害主机的 BO 客户端（即控制端）程序。

如何判断一个误报是技术问题还是一种争议？简而言之，如果反病毒产品对检测样本的判定结果与厂商分析后的主观判定是一致的，就不能称其为技术问题。

比如，有的厂商迫于市场压力，不得不查杀一些并非病毒的样本，这就不能称之为误报。所谓市场压力，是有第三方机构使用一些所谓经典样本库对反病毒产品进行评测，虽然其中大部分样本不具有主流操作系统中的活性，甚至根本就不是病毒，但公众往往将检出率作为产品好坏的唯一评判标准，这就迫使反病毒厂商单独处置这些样本。

本文接下来讨论的误报，是指作为技术问题的误报。即：一个检测对象被反病毒产品报警了，但厂商经过分析确认，认为它实际上不是病毒。

2.2 产生误报的环节

误报的产生，涉及反病毒厂商对可疑样本处理流程中的每一个环节，包括：

- 样本分析与判定环节：厂商将一个非病毒样本误判为病毒，导致该文件（或同源文件）被误报；
- 特征提取环节：厂商在一个病毒样本上提取特征，但这个特征在其他正常对象上被匹配到；
- 引擎机理环节：厂商的查毒引擎使用了不完善的机理或实现方式，导致正常对象被误报；
- 未知病毒判定环节：厂商的未知病毒检测机制认为一个检测对象的风险超过报警阈值，而实际上它是一个正常对象。

接下来，我们对这些环节做进一步的详细分析。

三、 样本分析与判定环节

在这一环节，反病毒厂商需要判断可疑样本是否病毒。一旦将非病毒对象误判为病毒，无论所提取特征的质量，误报都不可避免。目前，恶意代码数量剧增，绝大部分可疑样本都通过分析系统被计算机自动化地分析和判定，而不是由病毒分析人员依据个人经验来分析决定，因此，这个环节产生的误判一直存在。

可能使分析系统误判的可疑样本包括：

3.1 被利用的第三方软件

病毒作者或攻击者常使用已有的第三方软件来实现一部分攻击，而不是自己重写实现类似功能的工具。例如，早在 Back Orifice 后门出现之前，就有攻击者修改商用软件，作为后门工具使用。

DIY 蠕虫的出现促长了这种趋势。由于各类工具的不断涌现，一些病毒作者发现，要 DIY 一个蠕虫变得很简单。不论是蠕虫的扫描部分、投放部分、升级部分、后门部分和跳板部分，都有很多工具可以直接使用（见表一），只需将它们组合起来，使用批处理命令或脚本代码来调度，就形成了一个蠕虫。

表 1 蠕虫经常使用的第三方软件

类型	典型代表	说明
网络服务	Serv-U 、 Small HTTP Server	利用这些网络服务程序开启 FTP 或 HTTP 服务。从而访问和下载被攻击主机上的文件，也可作为上传其他后门工具的途径。
行命令	PS 工具集	Sysinternals 工具集的一部分，例如：在远端系统执行本地程序的 psexec，被基于口令破解的 DIY 蠕虫用作投放模块；终止进程的 pskill，被用于对抗反病毒软件。
脚本解释	mIRC 客户端	mIRC 有专门的指令体系、强大的脚本执行能力，被一些病毒作者作为脚本解释器使用。
远程控制	Remote Administrator 、Remote Explorer	一些中小型商用远程控制工具，常被用作 DIY 蠕虫中的后门，典型代表是 Remote Administrator。尽管它运行后会在托盘显示图标、连接状态，但很多普通用户对此并不敏感。
代理服务	3proxy、CCProxy、HHProxy	代理服务工具，常被蠕虫用于制作跳板。

这些工具绝大部分是免费或商用的正常软件。但是，当病毒作者将它们组合成一个整体产生危害时，反病毒产品就难以区分到底是用户在正常使用，还是作为整体在执行。一旦反病毒产品将这些工具本身视为病毒，就产生了误报。

3.2 公共控件和驱动程序

开发软件时，程序员经常使用一些公共的控件或驱动程序。这种机制为开发提供了便捷，但也被病毒用来实现一些功能。

例如：在 Window 9x 时代，木马经常通过可执行文件绑定（ExeBind）的方式，搭载 winsock.ocx 控件，在运行后将其释放为独立文件，并利用它获得网络功能。

如果将病毒衍生的这些公共控件或驱动程序当成了病毒，并提取特征，就会导致使用了它们的其他正常程序被报警，产生误报。

3.3 开源软件

开源软件的情况与上述公共控件和驱动程序类似，但更容易为病毒作者所用。

例如，VNC 是一款著名的开源远程管理工具，其行为非常规范——运行后，在系统托盘显示图标；用户可以设定密码验证；可以正常关闭等。因为可以获得它的源码，一些病毒作者就对其进行修改，去掉这些对用户可见的功能，最终改造成为一个后门工具。

如果反病毒厂商对这类修改开源软件形成的病毒提取特征，很难避免不提取到原软件的代码之上，这将导致正常的开源软件也被报警，产生误报。

3.4 被修改的文件

有的病毒会修改系统文件或其他程序文件，以达到自己的目的。

例如，著名的蠕虫 Happy99 修改系统中的动态链接库文件 wsock32.dll，改动它的一些函数调用，使之调用蠕虫所带的另外一个链接库。这样，当系统程序通过 wsock32.dll 发送邮件时，蠕虫就会获得收件人地址，然后将自身作为附件发送到该地址，实现复制和传播。

同样，一旦厂商认为修改前的文件是病毒，或者在被修改过的文件上提取特征稍有不当，就会导致正常的原文件被误报。

四、 特征提取环节

在上一环节我们已经看到，即便没有发生误判，也有可能因为提取的特征质量不高、不能显著区分恶意代码和正常文件，从而产生误报。一个高质量的特征，既表达了代码的恶意行为，又不会在正常对象中被匹配。但是，与自动化分析和判定相对应，在反病毒厂商的后台处理流程中，大部分病毒样本的特征都通过一种或多种固定的算法来提取。面对一些特殊的情况，就有可能产生质量不高的特征。

4.1 感染式病毒

感染式病毒的特点是在正常的可执行文件上附着一段恶意代码，通过修改入口点或执行流程，获得执行权限。

对感染式病毒的特征提取得是否准确，取决于对这一段附加恶意代码的边界判定是否精确。如果分析系统或分析人员没有准确地找到恶意代码与宿主文件的边界，而将特征提取到宿主文件之中，不仅无法有效地查杀病毒，反而会导致对宿主文件的误报。

4.2 被加壳的文件

加壳是一种压缩或加密可执行文件的方法。可执行文件被加壳后，在运行时先由壳代码将其解压缩或解密到内存，然后开始自身的正常运行。恶意代码通常将加壳以躲避查杀，也有一些正常的软件为了压缩尺寸或保护自身而加壳。

当分析系统或反病毒产品不能脱掉一种壳时，通常从带壳的病毒文件上直接提取静态特征。一旦提取的位置正好位于壳代码之中，而不是被壳压缩或加密的恶意代码上，就会导致所有加了这种壳的文件都被反病毒产品报警，对其中的正常文件就发生了误报。

4.3 自解压文件

自解压文件常被蠕虫使用，尤其是前面提到的 DIY 蠕虫。在采用了众多工具软件和脚本程序拼装成一个 DIY 蠕虫后，为了将它们组装成单个文件，作者一般将其打包为一个自解压包，并设定其中一个程序在解压缩后自动运行。

与被加壳的文件类似，一旦特征提取到了自解压代码上，就会导致所有使用相同技术制作的自解压文件都会被报警。

从广义上看，自解压文件并不仅是 WinZIP 或 WinRAR 等常见软件生成地、能自动解压和运行的包裹，还包括其他几种情况：

1. 使用安装包制作工具，将多个文件打包为一个软件安装包。不少作者还将病毒与其

它软件捆绑到一起，制作成一个安装包，这样做更具有欺骗性。

2. 使用并不常见的打包软件。

3. 对生成的自解压文件进行修改，去掉一些识别特征，但不影响它的功能。目的是使自动分析系统误认为它不是包裹文件。

这些情况尤为容易导致不成熟的自动化分析处理系统产生误判，并提取出低质量的特征。

4.4 编译器的影响

使用不同编译器生成的病毒，编译器的特性也会对特征提取造成影响。

例如，有的编译器生成的可执行文件，入口点处就会是实际的恶意代码；也有的编译器生成的可执行文件，入口点处有一段较长的相同或相似代码。如果将这部分相同的入口代码提取成了特征，所有使用同一款编译器生成的文件都会产生误报。

针对主流的编译器，反病毒厂商都制定相应的一组特征提取原则和禁忌。但还是有一些不常见的编译器，反病毒厂商对其特点的了解并不一定足够。

最后，更复杂的一种情况是壳与编译器的组合。当一种壳无法被脱掉时，反病毒厂商往往会根据积累的分析经验，避开壳代码本身，到被加壳的原文件代码中提取特征。当原文件中有编译器生成的大段共用代码，这些代码段被加壳后还是会相同。如果特征被提取到了这种地方，依然会产生误报。

五、 引擎机理环节

在反病毒产品中，引擎负责将恶意代码特征与被检测对象进行匹配。使用什么算法进行匹配？对不同类型的对象如何选择匹配策略？这些对引擎的设计，同样有可能引发误报。

5.1 哈希碰撞

很多反病毒软件的特征库中并不存储特征码本身，而是特征码的哈希（hash）值。这样做有很多优点：

1. 将不定长的特征码，变成长度相同的哈希值，可以大幅减少特征库所占用的磁盘空间和内存空间，并在一定程度上提高引擎的执行速度；

2. 只存储哈希值，降低了特征被抄袭的可能；

3. 因为哈希算法是不可逆的，病毒作者无法拿到特征，从而提高了病毒反查杀的难度。

但是不恰当的哈希算法和实现方式也有可能产生误报。例如，最常用的哈希算法有 MD5、CRC32，其哈希值的长度分别为 128 位和 32 位。CRC32 的哈希值更短，虽然极大地减少了占用空间，但它的冲突域也小了很多，更容易发生哈希碰撞。如果正好有一段其他的代码与病毒特征的 CRC32 值相同，就会导致误报。

5.2 格式识别与预处理

在反病毒技术发展的早期，误报的一个重要原因是引擎的格式识别与预处理机制不够完备。比如一个病毒只感染 PE 格式文件，那就应该只对 PE 文件匹配它的特征。如果对一些图片、音乐文件也做匹配，就有可能发生特征碰撞，导致误报。

目前，反病毒引擎在匹配特征之前，先判断是否包裹文件，是则解包并递归分析；然后判断其类型是二进制可执行文件、一般二进制文件、脚本文件或 Office 文件等；接下来分别调用不同的模块，进行脱壳、预处理；最后，才进行特征匹配和脚本语法分析等工作。

5.3 脚本文件

与二进制文件相比，脚本程序没有严谨的结构。例如，在大多数脚本中，增减几个空格是不会对其运行产生影响的。此时，传统的特征匹配方法就有可能效果不佳。

因此，对于脚本文件，有的厂商采用语法分析和语义分析的方法单独处理，此时的情况有些类似于下面将要介绍的未知病毒检测。

此外，恶意脚本和正常脚本之间的界限也难以划清。一些关于批处理命令、注册表编辑的教程书籍，其附带的示例代码就经常被反病毒产品误报。

六、 未知病毒检测环节

前面的三个环节，都位于样本判别、特征提取、特征库升级、病毒扫描的传统处理流程中。未知病毒检测则有所不同，它完全依靠反病毒产品中内建的经验模型，判断检测对象是否有害。

未知病毒检测的机理，本质上是加权判定法。反病毒产品通过分析检测对象的一些静态特性（如 API 调用、所在目录等），或建立虚拟运行环境，监视其动态行为（下载文件、创建服务、增加启动项等）。根据预先定义的采集点是否被触发，设置不同标志，每个标志有一定的权值。最后，将所有标志的权值相加，就得到了对这个对象的风险评估值。如果超过了阈值，就对其报警。

在这一套机理的实现过程中，存在以下几个困难：

1. 标志采集点的选择：哪些特点或行为是敏感的、有风险的？一方面，这依赖于分析经验和统计规律；另一方面，对不同类型的对象，采集点并不相同。
2. 权值的设置：哪些标志是相对而言更重要的，到底有多重要？重要程度可以根据经验和统计来排行，但怎样将其精确量化？
3. 标志之间的关系：简单的累加是否合理？多个标志同时被置位是否有更深刻的含义？
4. 不同的策略：脚本文件与可执行文件在特点和行为上差异较大，如何设置相应的采集点和权值？如何进一步细分，形成相应的判断策略？

在这些方面，不恰当的实现很容易导致误报的频繁发生。但是，人们对已知病毒的误报率和对未知病毒的误报率有着完全一样的指标要求。因此，未知病毒检测的实施困难重重。

这一环节中，误报最常发生在一些系统辅助工具上，尤其是与系统底层有关的安全工具上。这类工具有太多的特性和行为与恶意代码近似，因此总被反病毒产品误报。

七、 降低误报的方法

前文介绍了误报产生的环节和原因，同时也提到了一些情况下降低误报的思路。现在，我们再来介绍一些目前已被实际使用的方法。

7.1 白名单测试

白名单机制是最原始也最有效的降低误报率的方法。反病毒厂商一般都要维护一个庞大的文件白名单库。

建立白名单库，一般需要搜集和建立三类资源：

1. 目前所有的可信软件与文件，及其所有发行版本；
2. 用能搜集到的所有加壳工具（包括不同版本）对各种编译器编译的正常程序加壳；
3. 用能搜集到的所有安装包制作工具（包括不同版本），制作出无毒的安装包。

目前，可信软件的数量和版本剧增，厂商对它们的搜集很难做到齐全和及时。随着库的增大，白名单测试的速度也会受到影响。因此，白名单机制并非彻底的解决方案。

7.2 开放β库测试

一些软件厂商对部分专业用户提供软件的 beta 版本，通过这些用户的协同测试，来发现和排除反病毒产品对它的误报。这些高水平用户承担了一定的误报风险，但也提前享受到最新版本软件的益处。

也有部分软件厂商会主动将产品发送到各反病毒厂商，以助其建立白名单库。

这些方法都会降低误报的可能，但其推动者是软件厂商本身，而非反病毒厂商。

7.3 辅助判定标志

在当前，恶意代码的结构和技术日趋复杂的情况下，代码本身不再是判定其是否病毒的唯一依据。更为准确地判定依据可能与具体环境有关。简而言之，就是追求判定标准的多元化。前面提到的文件格式识别，就是这种思想的一个实现。

例如，对那些有可能被病毒利用的第三方软件，可以采用“特征规则+辅助判定” 的方法。特征被匹配到后，引擎不是马上报警，而是搜集更多的相关信息，然后加载其他规则来综合判定。

实际上，这其中有已知检测技术与未知检测技术相结合的思想，但它比单纯的未知检测更加准确。

实践表明，对这些思路的合理运用，会在降低误报率上取得不错的效果。

7.4 选项检测

对于一些可能引起误报的敏感规则，以及对可疑文件的处理方式，提供开关接口，让用户自己予以选择。这也是从工作机理上避免误报、降低误杀所造成影响的一种方式。

7.5 提示性的入前缀/后缀标志

一些反病毒厂商对广告件（Adware）、色情件（Pornware）等类别，设置了类似于 not-a-virus 的命名前缀/后缀，表明厂商并未完全认定这是一个病毒，而把最终决定权和处置权交给了用户。这样，在产品的用户体验方面，取得了一定的效果。

7.6 更细腻的加权采集点

在未知病毒检测中，要降低加权判定法的误报率，除了前面提到的合理细分场景、建立不同的加权模型、选择恰当的权值等，还需要更加扎实细腻地建立采集点。细腻的采集点使得其他细分工作成为可能，从而使判定模型能灵活应对各种实际场景，做出更为精确地判断。

7.7 快捷高效的误报处理机制

最后，不论付出多少努力，上述方法也注定无法修正所有的误报问题。从整个反病毒体系来看，建立一个快捷高效的误报处理流程是非常重要的。它是弥补误报问题，将各种影响降至最低的最终保障。

八、 结束语

漏报和错报可能造成无法有效地保障用户安全，而误报则不仅干扰用户，还可能伤害被误报软件的作者。因此，误报问题一度成为反病毒技术的危机。为了解决误报问题，一些反病毒厂商在积极构建更广泛的基础支撑体制，也有的厂商在发布特征库的时候变得谨慎。在寻求更加有效可靠的病毒检测方法、为用户提供良好保障的道路上，我们一直在努力前进。

最后，我们愿以一句话与全体同路人共勉：我们需要更完备的体制和更先进的方法，同时我们也清楚，任何制度和方法都无法代替反病毒工程师的水平、耐心与责任感。