计算机病毒

理论与实验

这篇文章介绍了“计算机病毒”并且检验了它们导致计算机系统的广泛破坏的潜力。文章展示了基本的理论结果、病毒防御在大型系统中的不可行性、防御方案以及一些实验结果。

1. 前言

这篇文章解释了一个主要的计算机安全性问题，即病毒。病毒的有趣之处在于它能依附于其他的程序并使它们也成为病毒。考虑到现代计算机系统的广泛共享，携带着木马程序[1,20]的病毒的威胁越来越值得注意。尽管在阻止信息非法传播[4,7]的执行政策上有相当多的工作已经完成，并且许多系统已经采用它们来避免这种类型的攻击[12,19,21,22]，但是在如何保持信息进入一个领域而不导致破坏方面的工作却很少[5,18]。在计算机系统中有很多不同类型的信息路径，其中一些是合法授权的，另一些可能是隐秘的[18]，这常常被用户所忽略。在这篇文章中我们将忽略隐秘性的信息路径。

一般设施提供正确的保护方案[9]，但是它们依赖于只对正在进行的攻击能有效防御的安全性政策。就算是一些相当简单的防护系统也不能被证明是“安全的”[14]。防止拒绝服务需要停止程序的检测[11]。给系统中的信息流进行精确标记的问题已被证明是一个NP-完全问题。对于在用户之间的不可信信息的传播，进行某种方式的防卫，这已经被测试过了[25]，但是一般依赖能力来提高程序正确性，这也是一个众所周知的NP完全问题。

施乐公司的蠕虫程序已经说明，该程序可以通过网络繁殖，并且可以偶然地造成设备无法服务。在后来的变种中，一个“核心战争”的游戏中，能够让两个程序互相争斗。一些针对这一主题的，被匿名作者发布的其他的变种，是基于在程序之间进行的夜间游戏的上下文中的。术语病毒也被用于结合丰APL的作者的地方一般开始时调用每个函数反过来调用一个预处理器,以增加默认APL翻译。

一个广泛传播的安全问题的潜在威胁被分析了【15】，对于政府、金融、商业和学术设施的危害是严重的。另外，这些机构倾向于使用特别的保护机制来应对这些特有的威胁，而不是研究全面而彻底的理论上的保护机制【16】。当前的军队保护系统，很大程度上依赖隔离机制【3】；然而，新的系统开始使用多层的用法【17】。没有一个发布出来的被建议使用的系统可以定义或者部署一种方法，用来阻止一种病毒。

这篇文章中我们提出了一个防止计算机病毒的新问题。首先我们检测了病毒的感染特性，并且展示了共享信息的传递闭包可以被感染。当和木马一起使用的时候，可以导致设备广泛停止服务以及数据的非授权的操作。一些对计算机病毒的实验结果表明，病毒对于普通的和高安全性的操作系统都是一个强大的威胁。传播的途径、信息流的传递以及信息解释的一般性是防止计算机病毒的关键特性，这些特性将在下面一一解释。分析表明，有潜力阻止病毒攻击的系统具有有限的传递性和有限的共享性，或者完全不能共享也不具有一般的信息解释（图灵能力）。只有前一种情形对于现代社会具有研究的意义。一般来说，使用先验和运行时分析对病毒的防护的研究是不可判定的，并且没有防护，治愈是很难或者无法实现的。

我们检查了一些被提出来的对策，这些对策都针对特定的情景，来对病毒的特质进行就事论事的分析。限制传播系统被认为是值得期待的，但是精确的部署是很棘手的，并且模糊的政策一般来说会导致有用的系统随着时间越来越少。系统范围内的病毒抗体的使用也被测试了，但普遍来说，还是要依赖针对特定棘手问题的解决方案。

综上可知对计算机病毒是一个非常重要的研究领域，因为它和其他领域的潜在性应用。现代系统对于病毒攻击提供很少或者没有提供防护，并且当代唯一有效的“安全”措施就是隔离了。

2 计算机病毒

我们定义计算机“病毒”为一个程序，它可以复制自身来调整自己从而感染其他程序。通过感染特性，病毒可以通过计算机系统或者网络进行广泛传播，使用每一个用户的授权来感染他们的程序。每一个被感染的程序也可能会表现为病毒，从而使得感染扩散。

接下来的伪程序展示了一个病毒如何写成一个伪计算机语言。符号‘：=’表示定义，符号‘：’标注一个状态，符号‘；’用于分隔状态，符号‘=’用于赋值或者比较，符号‘~’表示否，符号‘{’和‘}’将状态序列放在一起，符号‘…’表示代码的不相关部分已经略去了。

示例病毒（V）（图1）通过搜寻没有“1234567”开头的可执行文件来寻找一种不可感染可执行文件（E），将V加入E，将它变成一个感染文件（I）。然后V检查一些触发条件和破坏是否是正确的。最后，V执行剩余的前缀程序。当用户尝试执行E，那么I会在它的部分执行；它会感染其他的文件并且把它当做E来执行。由于感染的轻微延迟异常，I在触发条件导致破坏之前被看做是E。我们注意到，病毒不需要预先考虑自己，也不会限制于每次使用的单个感染。

病毒的一个常见的误解是它的程序简单地通过网络传播。蠕虫程序，“核心战争”，和其他类似的项目做到这一点，但实际上没有人涉及到感染的部分。病毒的关键特性是它感染其他程序的能力，从而达到共享用户之间的传递闭包。例如，如果V感染用户的一个可执行文件（E），然后用户B也运行 E，那么V可能蔓延至用户B的文件。

应该指出，病毒不可以用于邪恶的目的或者是成为一个木马。例如，一个压缩病毒可以写来用于找到未受感染的可执行文件，在用户许可的情况下压缩它们，并且预先考虑自己而非它们。在执行时，受感染的程序进行解压，然后正常执行。因为它总是在执行服务之前寻求允许，所以这不是一个木马，但因为它有感染特性，所以它仍然是一个病毒。研究表明，在平均系统中，这种病毒可能节省由可执行文件占据的超过50%的空间。被感染的程序的表现将下降，这是因为它们被解压了，从而压缩病毒实现一个特定的时间与空间的权衡。病毒样本压缩可以编写如下，见图2。

这个程序（C）找到了一个未感染可执行文件（E），压缩它，并且突出显示C形成一个感染可执行文件（I）。然后将剩余的自己解压为一个临时文件并且可以正常执行。当运行I时，在将I解压成一个临时文件盒执行之前，它会寻找并压缩其他可执行文件。传播的效果是通过系统压缩可执行文件，并解压它们进行执行。用户将有重大延迟，因为他们的可执行文件是在解压后才运行的。

一个更具威胁性的例子中，我们假设通过以下的方法修改程序V，一是指定在一个给定的数据和时间进行触发，二是指定通过无限循环的方式进行破坏。在大多数现代系统的共享水平下，整个系统可能无法在指定的日期和时间条件下使用。可能需要大量的工作来消除这种病毒的破坏。这一修改如图3所示。

作为计算机病毒的一个类比，可以考虑一个可以100%感染的生物疾病，每当动物交流时就会传播，在给定的时刻立即杀死所有被感染的动物，并且在那个时刻之前没有检测到副作用。如果疾病的引入和见效之间存在一周的延迟，将很有可能只有少数偏远村庄能够幸存，并肯定会消灭绝大多数的现代社会。如果类似于这种类型的电脑病毒可以传遍世界所有的计算机，它可能会在很长的时间阻止大多数电脑的使用，并且在现代政府、金融、商业和学术机构中造成很大的破坏。

3 计算机病毒的预防

我们已经向读者介绍了病毒的概念和对于系统的实际病毒。已经种下潜在的毁灭性袭击的种子，所以适当的检查保护机制可以帮助抵御它。我们在这里研究计算机病毒的预防。

3.1 基本的限制

为了系统中的用户能够共享信息，必须存在一个信息路径可以让信息从一个用户传送到另一个用户。我们不区分用户和作为用户代理的程序，这是因为在任何计算机使用中程序总是作为用户的代理，而且我们忽略通过用户的秘密通道。假设有一个计算的图灵机模型，我们可以证明如果信息可以由用户通过图灵能力进行读取，那么它就可以被复制，复制后可以被视为一个图灵机磁带上的数据。

给定一个通用的系统，用户可以按照自己的希望使用自己具有的信息，并且传递这些他们认为合适的信息，可以清楚的看到，分享信息的能力是可传递的。也就是说，如果有一条从用户A到用户B的路径，还有一条从用户B到用户C的路径，那么在用户B有意或者不知情的合作下，存在从用户A到用户C之间的路径。

最后，可以用作数据和程序的信息之间没有本质的区别。这可以清楚的在翻译的情况下看出来，信息编辑成数据，并译作一个程序。实际上，信息只有在译出之后才有意义。

在一个信息可以被接受者译作一个程序的系统中，这种翻译可能导致如上所示的感染。如果存在分享，感染可以通过共享信息的解释进行传播。如果对信息流的传递性没有限制，那么从任何资源开始，信息可以到达信息流的传递闭包。共享、信息流的传递性和解释的普遍性使得在任何给定的资源的条件下，病毒可以传播到信息流的传递闭包。

显然，如果没有共享，就没有跨边界信息的传染，因此外部的信息不能解释，并且病毒不能在单个分区之外进行传播。这就是所谓的“隔离”。显然，系统中如果没有程序可以改变并且没有信息用于决策，那么系统不会被感染，这是因为感染需要解释信息的修正。我们称之为“固定一阶函数”系统。我们应该注意到，几乎任何具有实际效用的科学系统或开发环境需要解释的普遍性，如果我们希望从别人的工作中获益，那么隔离是不可接受的。然而，在有限的情况下可能存在病毒问题的解决办法。

3.2 分区模型

可以将限制信息流动的路径分为两类，一些将用户在传递条件下分割成适当的闭子集，另外一些则不进行分割。流量限制可以导致闭子集视为一个系统的分区并成孤立子系统。这让每一次感染限制在一个分区中。这是一个可行的预防手段，用于预防病毒在有限隔离下的完全占领，相当于给每个分区自己的计算机。

完整性模型[5]提供一种策略，它可用于在传递性条件下将系统分割成封闭子集。在Biba模型中，一个完整性层次与所有信息都相关联。严格的完整属性是Bell-LaPadula属性的双重叠加；在一个给定的完整性水平下，没有用户可以读取一个完整性水平更低的对象，也不能写出一个完整性水平更高的对象。在Biba最初的模型中，读取和执行访问之间是有区别的，但是在没有限制信息解释的普遍性的条件下不能执行，这是因为一个高完整性的程序可以编写一个低完整性的对象，并且对其进行复制，然后读取低完整性的输入并且产生低完整性的输出。

如果完整性模型和Bell-LaPadula模型共存，一种有限隔离可以通过传递性将空间划分为有限的闭子集。如果相同的划分用于机制(高完整性与高安全性)