



# 点击图标运行程序的机制、安全问题及对策

杨程涛<sup>1</sup>、秦子茗<sup>2</sup>、刘迅<sup>3</sup>

**摘要：**本文先通过对软件运行时系统产生的变化进行一个时序的介绍，使学者对个中过程有一个大致的概念。再对各个时序可能出现的安全问题进行一个简单举例论述，最后，给出一些问题的解决方案，以供学者参考。

**引言：**随着信息化的加深，越来越多的软件被开发和使用。而运行软件的第一步便是点击软件图标，在这一步中我们电脑的系统发生了什么变化，其中存在哪些安全问题，又有哪些方案去解决。对这三个问题进行深入的了解，有助于软件安全人员在软件运行的第一步就做好安全的把关。

## 1 点击图标后系统发生的变化

首先，对我们常见的图标进行一个分类。大致分为两类，一类是快捷方式 （图标左下角有一个蓝色箭头的）；而另一类为可执行文件 （常见的是.exe 文件）。虽然二者在图标上差别不大，并且实际双击后系统变化极为相似，但是就因为二者细微的差别使得快捷方式会多出一类安全问题。并且，我们需要明确一点，程序的运行永远都是通过可执行文件来实现。

### 1.1 查找目标

这是快捷方式不同于可执行文件的一步，可以说是多出来的一步。当我们双击在我们双击存在桌面的快捷方式时，会先通过预先保存在快捷方式的地址找到可执行文件在硬盘中的位置，也就是我们所谓的“目标文件”。这时就会出现一个问题：如果我们把保存的地址修改了会发生什么？这一问题将在第二部分中展开。

在这之后，之前分的两类图标所发生的事情将完全一致，如果保存的地址没错的话。

### 1.2 检测并查找地址

当双击可执行文件的图标时，就会向操作系统发出一个请求——我想要运行一个程序。操作系统接受到请求后，就会去磁盘上找到程序的相关信息，检测它的类型是不是可执行文件，同时通过程序首部信息确定代码和数据在可执行文件中的位置并且计算出对应的磁盘块地址。

### 1.3 创建进程

---

<sup>1</sup> 负责第一部分以及摘要、引言部分

<sup>2</sup> 负责第二部分

<sup>3</sup> 负责第三部分以及结语部分

操作系统做完第 2 步之后，就会创建一个进程，并且将可执行文件映射到该进程结构，意思就是，这个进程负责执行这个程序。

#### 1.4 分配和读入内存

操作系统会分配足够的内存，并且将全部代码读入内存中。

#### 1.5 执行代码

读入内存的代码从第一行开始运行，执行相关的函数。在执行程序的过程中会调用各种软硬件，比如显示器、网络、窗口系统等。

#### 1.6 结束程序

在上述步骤完成后，程序的运行就会结束，并显示运行的结果。

## 2 程序运行过程中可能存在的安全问题

### 2.1 程序是攻击主体

当我们点开图标时，或许正在运行的并非正常程序而是作为攻击主体的恶意程序。而包含恶意代码的程序往往会突破权限的边界，非法读写本地文件，或在从网络上下载攻击载荷，从而带来安全问题。

#### 2.1.1 突破权限的限制

恶意程序可能非法突破系统的权限审核从而达到修改快捷方式，篡改用户数据，窃取网络流量等目的。

数年前，一款名为 Godless 的 Android 恶意应用被发现，它可以伪装成为正常的软件上线谷歌市场等应用市场，并且对于低于 Android 5.1 的系统版本获取 root 权限。此外，Godless 更新的变种版本还能绕过 Google Play 等应用市场的安全检查，一旦该应用完成 ROOT 之后将很难从手机被卸载。Google Play 多款应用中发现了 Godless 恶意代码，它可以隐藏与手电筒应用，Wi-Fi 应用或者是一些热门游戏。一些应用虽然没受到感染，但是恶意版本却内置了和正常应用一样的开发者证书，这样导致的结果就是用户安装纯净了应用，但是也会在用户不知情的情况下自动升级至恶意版本。

#### 2.1.2 非法处理本地文件

恶意程序在运行时可能将含有机密信息的本机文件拷贝发送至指定的服务器上从而造成用户信息的泄露。如 Vidar 病毒，能够实现搜索特定文档，从浏览器历史记录（包括 tor 浏览器）和加密数字货币钱包中窃取信息，窃取即时会话消息，获取用户计算机状态快照等。

其次，另一类程序会对本机文件进行加密，从而导致文件损坏或者无法使用。2017 年席卷全球的勒索病毒 WannaCry 会将电脑中所有的文件加密并锁定然后释放说明文档对用户进行敲诈勒索。文件使用随机生成的 AES 密钥进行加密，

AESKEY 通过 RSA 加密后保存在生成文件的文件头中。敲诈者会根据文件类型、大小、路径等来判断文件对用户的重要性，进而选择对重要文件的先行进行加密并擦写原文件，对认为不太重要的文件，仅作删除处理。尽管用户可通过支付赎金来获得恢复密钥，但部分文件可能永久丢失。

## 2.2 程序是攻击客体

在程序运行时，程序本身或系统环境也可成为被攻击的客体，由此引发的安全问题同样值得我们思考。

### 2.2.1 快捷方式被篡改

当我们从开始菜单打开 chrome 浏览器时，有时会发现主页变成了 hao123(见图 1)，而这些网页通常被广告充斥着。这是因为一些流氓软件会更改菜单的快捷方式来改变我们打开的页面，进而赚取收益。比如臭名昭著的 2345 网址。在用户通过一些安全性低的软件园下载软件时，往往会安装到捆绑 2345 的被篡改软件。这些软件的体积一般较官网下载的应用程序更小，而来源网站通常会宣传“安全无毒”，“纯净下载”等。用户需要提高防范意识，养成官网下载的习惯。

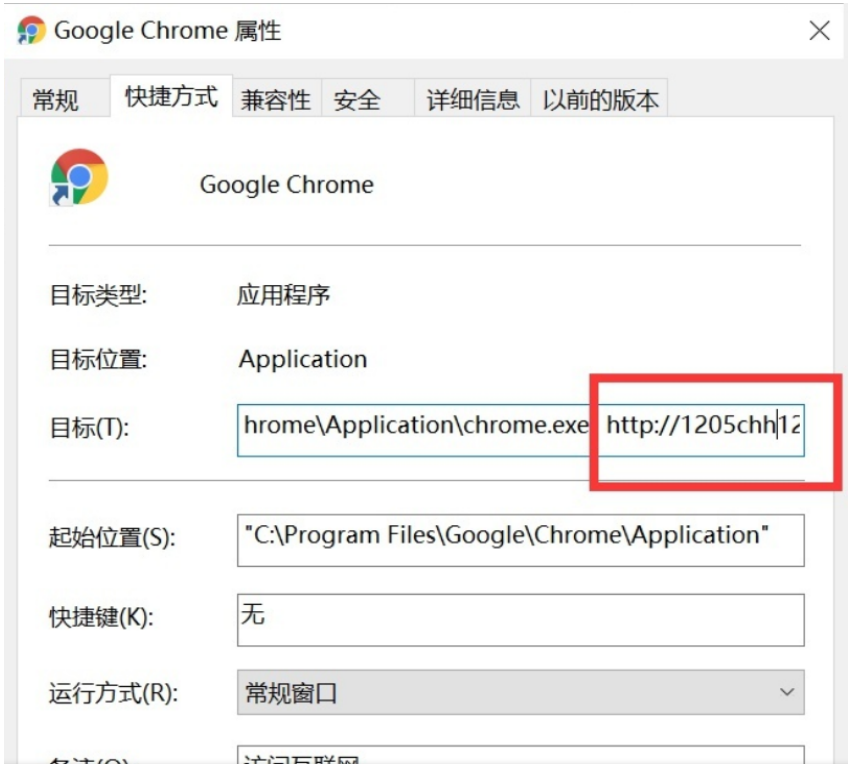


图 1: Chrome 主页被篡改

### 2.2.2 木马：恶意代码嵌入

木马病毒是指隐藏在正常程序中的一段具有特殊功能的恶意代码，是具备破坏和删除文件、发送密码、记录键盘和攻击 Dos 等特殊功能的后门程序。木马病毒的发作要在用户的机器里运行客户端程序。一旦发作，就可定时地发送该用户的隐私到木马程序指定的地址，同时内置可进入该用户电脑的端口；部分木马

程序运行后攻击者可以远程控制程序来进行记录目标计算机各种账户密码、获取系统信息并篡改系统设置、限制计算机功能、远程文件操作和注册表操作等。

木马病毒的传播方式多种多样：电子邮件、短信中的附带链接中隐藏病毒；各种不知名来源的未知程序可能包含木马病毒；各种系统和软件的漏洞也会被黑客利用从而投放病毒；甚至当用户点击并打开了包含木马病毒的网页，也会成为受害者。

### 2.2.3 内存管理的漏洞

原则上，所有软件都实现了自己的缓冲区管理机制，包括分配和释放存储数据的缓冲区的协议。然而，软件应用程序由来自不同组织部门软件工程师，在软件生命周期中不同的阶段共同开发和修改，自然而然会有漏洞产生，并可能对内存管理产生严重影响。

一个关于内存管理错误的例子是谷歌最近宣布推荐用户更新 Chrome 浏览器（CVE-2019-5786）。这是一个“使用释放后内存”（UAF）错误，该错误会导致用户的计算机文件泄露。这种内存管理问题发生在相对复杂的控制流中并且跨越函数边界。最重要的是，引入漏洞的函数是一个开源 API。简单来说，该错误是由特定时间窗口引起的，在错误时间窗口内，攻击者可以获取指向敏感数据的指针。这将带来用户敏感信息泄露，财产安全等受到威胁的风险。

### 2.2.4 可执行文件受 DLL 劫持攻击

在 Windows 中，许多应用程序并不是一个完整的可执行文件，它们被分割成一些相对独立的动态链接库，即 DLL 文件，放置于系统中当我们执行某一个程序的时候，相应的 DLL 文件就会被调用。一个应用程序可使用多个 DLL 文件，一个 DLL 文件也可能被不同的应用程序使用，这样的 DLL 文件被称为共享 DLL 文件。

而 DLL 劫持是将自己的 DLL 模块插入到别人的进程中运行，从而达到攻击的目的。比如，通过 C 语言编写一个游戏要加载的系统 DLL 文件(lpik.dll)，其中假 DLL 包含劫持功能和作弊功能且拥有相同的导出函数，将放入游戏相同目录下，游戏打开时会自动加载该假 DLL，使游戏直接包含作弊功能。而近日英国安全人员的研究也表明，近 300 个 Windows 10 可执行文件易受 DLL 劫持攻击。通过该种攻击方法，攻击者能利用其中某些 EXE 提升执行且完全绕过 UAC。同时使合法的 Windows 可执行文件加载受攻击者控制的任意（通常具有恶意意图的） DLL。

## 3 安全问题的对策

从双击图标到程序被运行，这个过程暗藏许多潜在的安全问题。从主动被动的角度入手，可以把安全问题分为两类：第一类情况为程序是攻击主体，也即我们运行的程序事实上是恶意代码，例如误点击了一个来路不明的可执行文件；第

二类情况为程序是攻击客体,换言之运行的程序由于或自身或外部的因素出现安全问题,例如被嵌入了木马程序。

本章节主要围绕主客体两方面的情形,开展安全问题对策的讨论。

### 3.1 程序是攻击主体

假如运行的程序本身是恶意的,那么安全的系统应该保证双击执行后,程序并不能正常运行、实现其既定的目的,也就是阻止恶意行为。应对此类安全问题,从防御对象上,可以分为代码段、代码行为和数据内容。

#### 3.1.1 审核代码段

首先不难想到的是一种方法是,利用已知的恶意代码片段,在实际运行程序之前,预先审核程序。假如匹配到了已知的恶意代码片段,那么立即终止程序运行,并且向用户发出报错。这也正是杀毒软件的“病毒库查杀”原理。

这种方法简单易行,但缺点同样明显。恶意代码变体众多,新的攻击方式层出不穷,为了完整记录现有的恶意代码,需要维护庞大的数据库。而为了应对代码混淆,查杀过程还需要多次匹配。与此同时,对用户而言,一方面需要花费网络资源,时刻保持病毒库更新,费时费力;另一方面这种方式效果有限,只能防御已有的攻击,对新兴的恶意代码无能为力。

#### 3.1.2 审核代码行为

行为级的审核可以有效减少代码级审核的冗余,同时更有能力应对未知的恶意代码。设想一个计算器程序突然开始录制你的屏幕并且向网络上传大量数据,即便不审核代码片段也能大概率肯定这个程序是攻击主体。行为审核有两种思路:其一是设立白名单,给定授权范围,程序只能在限定权限中运行;其二是设立黑名单,禁止程序进行特定操作。这一种对策中,可以在运行前预先向用户确认程序的权限,例如能否访问摄像头、能否读取通讯录、能否修改某些目录下的内容,并且在运行过程中审核程序是否存在异常操作。

对程序行为的审核,能够允许授权操作、禁止非法操作,这一种结合白名单、黑名单的行为审核,更能够减少误报率、有效阻止攻击主体的恶意行为。

#### 3.1.3 访问控制

设想在某种情况下,运行的程序通过了运行前的审核,并在运行中表现的和一个正常程序没有差别,简单的向网络传递、收发数据。但下载的内容事实上是被伪装成数据的可执行代码,这意味着攻击载荷被隐藏在网络中,因而本地运行的程序通过了审核。载荷注入后,绕过了对程序的一切审核,进而在本地对内存、硬盘开展攻击。

这提示我们,不仅要关心正在运行的程序,还要关心可能被非法读写的内存和数据。

针对内存,访问控制的主要对策是 DEP 技术。DEP 对内存可读、可写、可执行的划分,防止原本标记为数据的内容被执行,从而绕开对运行程序的审核。

针对硬盘数据,应对的方式是类似的。通过对文件分用户分为可读、可写等

权限，可以避免攻击主体非法的读取和写入。假如文件突然被程序试图读写，并且对程序的审核并没有报错，那么对文件本身的访问控制就成了防御攻击主体的最后一道屏障。

## 3.2 程序是攻击客体

所运行的程序，同样有可能是出现安全问题的客体。从双击图标到运行的过程中，还可能出现软件被攻击或是软件由于自身原因导致安全问题的情况。

### 3.2.1 程序完整性校验

无论是前文中提到的快捷方式被篡改，还是软件被植入木马，本质上都是文件经过了非授权的修改。相应的，从双击图标到运行的过程前，我们可以对程序进行完整性的校验。如果检测到了异常的被修改情况，就暂停运行程序并报出错误。

### 3.2.2 加强软件开发

针对例如 Chrome 中的内存管理漏洞，是操作系统无法判定的异常行为，因此成为了可被利用的 **vulnerability**。对于此类造成安全问题的成因，相应的对策是在软件开发时加强测试盒审查。

## 3.3 额外想法

程序运行的过程，相当大的部分可以简化为 **CPU** 和存储设备的通信问题。通信问题的安全问题在软件运行中同样存在。例如信道被窃听甚至劫持、中间人攻击等等问题。软件运行和通信两者之间的共通性安全问题有待挖掘。

**结语：**本文从“双击图标打开程序”入手，分析了用户端视角简单的一步背后系统有着怎么样的变化，这一些变化中有着哪些安全问题，这些安全问题又有哪些相应的解决方案。对系统安全的认识在一步步探索中不断深化。而对安全问题攻防博弈双方的分析，更为研究者提供了更多的攻防思考角度。