

Windows x64 无签名驱动程序运行方法

张 智, 蔡皖东

(西北工业大学 计算机学院, 陕西 西安 710129)

摘 要: 主机监管系统利用过滤驱动程序对系统实现全面监管. 随着微软 64 位操作系统的推出, 要求驱动程序经过付费签名后才能正常运行. 由种种原因使驱动程序不能被签名时, 主机监管系统就不能在 WINDOWS 64 位操作系统上使用. 因此无签名驱动程序问题成为在 WINDOWS 64 位系统开发最普遍的问题之一, 会导致程序难移植、影响用户体验. 通过对 WINDOWS 64 位系统数字签名过程的逆向分析, 提出一种能一次性关闭系统数字签名验证机制的方法, 从而顺利加载运行未签名驱动程序.

关键词: 驱动程序验证; 数字签名; WINDOWS X64; 内核

中图分类号: TP316

文献标识码: A

文章编号: 1000-7180(2014)02-0101-05

Excution Method of Unsigned Driver on Windows x64

ZHANG Zhi, CAI Wan-dong

(College of Computer, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: Via filter drivers, host regulatory system manages computers perfectly. With the launch of Microsoft's 64-bit operating system, only signed drivers which cost money can be executed on windows x64. For some reasons drivers cannot be signed so that host regulatory system is impossible for windows x64. Unsigned driver problem becomes one of the most common problems, which lead to bad effect on portability or user experience, so it is urgent demand to execute unsigned drivers. In this paper, upon analysis of kernel assembler about verifying driver signature, we proposed a method to turn off the verification of driver signature. Then unsigned drivers can be executed.

Key words: driver verification; Driver signature; WINDOWS X64; kernel

1 引言

主机监管系统是指对主机特定的设备、操作、系统状态等方面进行监控以及管理的系统. 网络技术的发展为用户带来了极大便利性的同时, 也成为了最大的安全隐患之一.

主机监管系统的实现需要对系统掌握较高的控制权限, 通常由各种过滤驱动程序来完成功能. 过滤驱动程序是一种能向外围设备添加特定值或修改其操作的驱动程序. 常见的过滤驱动程序有文件过滤驱动程序、键盘过滤驱动程序、USB 过滤驱动程序等等. 它们可以控制文件的删改复制、截获键盘输入

内容、甚至修改 USB 存储设备中的内容. 在 WINDOWS 系统 32 位版本下, 通过过滤驱动程序实现的主机监管系统已经相对成熟, 并且达到了一定的效果. 随着 WINDOWS 系统 64 位版本的逐渐流行, 主机监管系统的移植遇到了一些新的问题. 微软官方规定在 WNDOWS 系统 64 位版本上只能正常加载运行经过数字签名的驱动程序. 主机监管系统必须依靠各种过滤驱动程序来保证对系统的控制权, 如果逐个为过滤驱动程序购买签名证书将会大大增加系统开发成本.

文献[1]中, 微软官方介绍了 VISTA 下运行驱动程序所需的签名流程以及在测试模式下利用测试

收稿日期: 2013-03-27; **修回日期:** 2013-05-21

基金项目: 西北工业大学研究生创业种子基金项目(Z2012141); 2013 年陕西省科学发展计划项目(2013K06-19)

签名测试驱动程序的方法,但是测试模式会在系统桌面上打印出“测试模式”的字样.文献[2]中详细分析了在系统启动过程加载内核时用于完整性检查验证的关键函数. SKYWING^[3-5]对 WINDOWS 系统 64 位版本动态内核保护 PATCHGUARD 的三个版本进行了详尽的分析.文献[6-7]不同程度地分析了 VISTA 的驱动程序签名验证过程.

上述文献并没有提供一种能在 WINDOWS 系统 64 位版本下不为系统带来副作用并无需任何形式签名的正常加载运行未签名驱动程序的方案.本文通过对 WINDOWS 64 位系统数字签名过程的逆向分析,提出一种能一次性关闭系统数字签名验证机制的方法,从而顺利加载运行未签名驱动程序.

2 相关概念

微软发布 WINDOWS X64 以来,安全机制主要有三部分组成:代码完整性验证、动态内核保护 PATCHGUARD 和强制驱动程序签名.三部分相辅相成,代码完整性检查保证了动态内核保护 PATCHGUARD 能正常启动;动态内核保护 PATCHGUARD 确保代码完整性验证(包括签名验证机制)不被修改;强制驱动程序签名要求屏蔽掉了大多数恶意驱动程序对系统的破坏.

2.1 代码完整性验证

代码完整性验证的目标在于通过验证保证系统二进制文件未被恶意代码篡改以及没有未签名的驱动程序在系统内核模式下运行来保护系统.

代码完整性验证机制随着 WINDOWS 系统开机而启动, WINLOAD. EXE 中的代码完整性验证机制验证内核、BOOT 启动驱动程序等的完整性.随后的启动阶段中,内存管理器对每项被加载到内核内存空间的文件进行代码完整性验证.通常,代码完整性验证机制主要对两大部分内容进行验证:完整性验证和驱动程序签名.

代码完整性验证机制保证了被加载运行的模块都是可信的,并且可信的模块不会被修改利用为恶意程序.但其属于一次性的验证机制,即当系统通过对某一模块的验证,并未检测到异常,在该模块正确的成功加载运行以后,系统不再负责对其进行代码完整性验证.

2.2 PATCHGUARD 动态内核保护

在 WINDOWS X64 系统中, PATCHGUARD 以其神秘而健壮的安全机制而著名.它最初出现在 WINDOWS VISTA X64 版本,用于禁止第三方对

系统内核进行补丁. PATCHGUARD 机制被隐藏在 NTOSKRNL. EXE 文件中,并在启动之后约 5~10 分钟间隔内周期性地检查系统关键结构和操作系统镜像文件,若发现所保护部分被修改, PATCHGUARD 将会触发系统蓝屏.在一段时间内, PATCHGUARD 成为了金钟罩般的保护层.

概括来说, PATCHGUARD 严谨的机制造成了它难调试、难分析、难破解的特点.

2.3 强制驱动程序签名

WINDOWS X64 系统依靠对内核模式代码进行数字签名来提升 WINDOWS 平台的安全性和稳定性.未签名的驱动程序在得到未签名的提示后,强制驱动程序签名是微软官方的一种市场策略,而代码完整性验证是一种实现手段.因此,对驱动程序签名的验证过程是包含在代码完整性检查机制中.

3 驱动程序签名验证机制解析

本文以 WINDOWS 7 X64 版本为例,通过逆向技术,从实际代码上分析驱动程序签名验证机制的整个过程.

3.1 ImgPValidateImageHash 函数

ImgPValidateImageHash 函数体存在于操作系统启动加载器 WINLOAD. EXE 中. WINLOAD. EXE 在启动过程中负责加载操作系统内核文件 NTOSKRNL. EXE 和 BOOT 方式启动的驱动程序,并对其进行完整性验证.一方面,通过计算二进制文件得到的校验码和 PE 文件头中的校验码对比来验证其是否被修改;另一方面验证包括操作系统的内核 NTOSKRNL. EXE 在内的二进制文件的签名是否正确.而正是由 ImgPValidateImageHash 函数检查二进制文件签名.

该函数首先调用 MinCrypL_CheckSignedFile 子函数检查文件是否被可信证书颁发机构正确签名,这个签名必须是一个由微软公司或者 VeriSign 公司提供的 CLASS 3 类的代码签名证书. CLASS 3 类证书是 CLASS 1 类证书的高安全子集部分,相对更适用于整合、发布过程.如果该子函数验证通过,则直接跳转至 ImgPValidateImageHash 函数返回模块,表示验证通过.

对于未通过 MinCrypL_CheckSignedFile 子函数验证的情况,函数将进入一个循环过程.在该循环过程中,每次从操作系统的签名目录文件夹 %SYSTEM32\CATROOT 中加载后缀名为. CAT 的签名目录文件,再由 MinCrypL_CheckImageHash 子

函数来验证二进制文件的校验值是否与签名目录文件相匹配。如果找到匹配的校验值,该子函数返回0,随即跳转到返回模块,否则返回 C0000428 错误码并继续下一次循环。

通过验证的文件,ImgpValidateImageHash 函数返回值为0;对于未签名的二进制文件,该函数返回值为 C0000428。如果返回其它错误代码,可以在微软官方帮助文档中查找对应的错误信息。

3.2 SepInitializeCodeIntegrity 函数

SepInitializeCodeIntegrity 函数体存在于操作系统内核文件 NTOSKRNL.EXE 中,负责在启动过程中完成代码完整性检查初始化工作。以下是启动过程中对该函数调用栈。

1. nt! SepInitializeCodeIntegrity
2. nt! SepInitializationPhase1+0x236
3. nt! SeInitSystem+0x31
4. nt! Phase1InitializationDiscard+0xc56
5. nt! Phase1Initialization+0x9
6. nt! PspSystemThreadStartup+0x5a
7. nt! KiThreadStartup+0x16

经过反汇编的分析 SepInitializeCodeIntegrity 函数体,其函数伪代码如下:

```
VOID SepInitializeCodeIntegrity()
{
    IF(不是 PE 模式)
        设置代码完整性检查开启标志;
    清零代码完整性回调函数表;
    IF(检查参数是否有效)
    {
        设置标志位(测试模式标志|取消完整性检查标志);
        CiInitialize();
    }
}
```

从函数体来看,SepInitializeCodeIntegrity 实际上是一个外壳函数,内部调用 CiInitialize 子函数来实现初始化。该子函数由 CI.DLL 文件导出,而 CI.DLL 是由操作系统内核文件加载的动态链接库,用于实现代码完整性验证相关的功能。该动态链接库内最重要的三个导出函数分别为:

```
CiInitialize()
CiValidateImageHeader()
CiValidateImageData()
```

而 CiInitialize 函数正是 SepInitializeCodeIntegrity 函数完成初始化工作的真正执行体。首先,它会进行对自身完整性进行检查,如果发现任何异

常则返回 C0000428 错误码;接着,将 CiValidateImageHeader 函数和 CiValidateImageData 函数在内存中的调用地址告诉操作系统内核,用于后面的完整性检查验证;最后,调用 PESet Phase1 Initialization 函数对 BOOT 启动列表上的每个驱动程序进行签名验证。

3.3 SeValidateImageHeader 函数

SeValidateImageHeader 也是一个外壳函数,它通过调用 CI.DLL 中的 CiValidateImageHeader 来实现功能。代码完整性被开启的状态下,当可执行程序被映射到内核存储空间时系统调用该函数对被加载文件的签名进行验证。同样的,如果通过签名验证,函数返回0;否则返回 C0000428 错误码表示签名验证失败。

经过逆向分析,CiValidateImageHeader 函数内部算法如下:

- (1)检查相关事件机制是否已经初始化;
- (2)分析被载入文件类型,以及获得文件大小;
- (3)从被载入文件映射到内存中的二进制区域中获得证书信息(例如,PE 文件头第 0xA8 偏移位置的值);
- (4)根据 PE 头信息分析文件区段数目,并循环每个区段,利用 HashpHashBytes 计算校验值;
- (5)检查所找到的证书信息是否有效,若无效则直接返回 C0000428。对比计算的出的校验值和文件签名证书中的校验值是否一致,若一致返回0,否则返回 C0000428 错误码。

3.4 耦合关系

构成 WINDOWS 操作系统 64 位版本内驱动程序签名验证机制的模块有多个部分。如图 1 所示,WINLOAD.EXE 在系统启动时触发了 LOAD 模式代码完整性检查确保各部分经过签名且完整未被修改。随着操作系统 NT 内核的加载和运行,系统对 KERNEL 模式代码完整性验证机制进行初始化,并在初始化过程中获得后期代码完整性验证所需要的函数地址。最后,完成动态内核完整性保护机制 PATCHGUARD 的初始化,由它动态周期性地对包括代码完整性验证代码在内的系统关键部分进行检查,以防系统被破坏。

换言之,单独在某一个模块内寻找方法通常会引发其他模块对本模块的验证失败而造成系统崩溃。例如,WINLOAD.EXE 未通过内核文件代码完整性检查时,系统无法启动并报告 C0000428 错误码;PATCHGUARD 检查到系统文件被修改时,则会

引发蓝屏. 因此, 要达到在 WINDOWS 系统 64 位版本上加载并运行未签名的驱动程序, 需要对上述各个方面进行综合考虑.

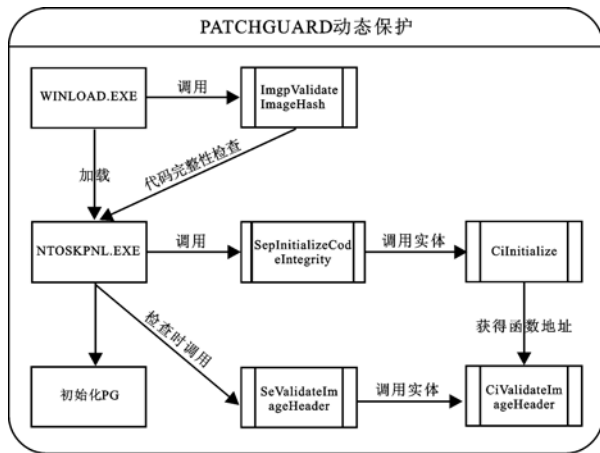


图 1 驱动程序签名验证机制

4 解决方案

本文提出一种不影响用户体验, 允许完全未签名的驱动程序(不需要测试签名)在 WINDOWS 系统 64 位版本上加载运行的解决方案.

(1)关闭 LOAD 模式验证. 该模式验证是在操作系统启动阶段由引导文件 WINLOAD. EXE 中 ImgpValidateImageHash 函数完成. 通过在 WINLOAD. EXE 中该函数定位后, 笔者直接对其二进制进行修改, 令该函数直接返回零.

(2)关闭 KERNEL 模式验证. 阻止代码完整性检查初始化工作是一种相对简单的方法. 只需对 NTOSKRNL. EXE 补丁, 令 SepInitializeCodeIntegrity 函数跳过实质性工作, 直接返回初始化完毕.

(3)关闭 PATCHGUARD. 按照 SKYWING 的文章详细介绍, 可以找到被他取名为 KiInitializePatchGuard 函数的隐藏函数. 通过补丁, 令该函数按安全模式初始化, 即不开启动态内核保护功能.

(4)校验值更新. WINLOAD. EXE 和 NTOSKRNL. EXE 文件被修改了以后, 其校验值产生变化, 无法和 PE 头部中记录的校验值正确匹配. 因此, 还应该将其 PE 头部中对应校验值修改为当前文件的校验值.

(5)设置系统选项. 除了二进制代码上对系统修改以外, 还需要利用 BCDEDIT 工具关闭系统的错误恢复和完整性检查选项.

通过上面的修改以后, 即可加载并运行未签名的驱动程序, 进而在 WINDOWS 系统 64 位版本中

实现主机监管系统来对计算机进行更好的监管.

5 实验结果及分析

本实验运行在 WINOWS 7 X64 旗舰版操作系统, 对各种系统修改情况进行试验和分析.

本文实验部分将分四种不同的情况对系统文件进行修改实验, 通过修改后系统的反应来分析修改结果. 为了保证比较的公平性, 实验在同一个操作系统版本的同一个虚拟机镜像中实验, 并且采用相同的测试驱动程序. 表 1 中显示了不同的修改情况造成的不同结果. 由此可验证, WINLOAD. EXE 中 ImgpValidateImageHash 函数主要在启动阶段对系统内核文件进行验证; SepInitializeCodeIntegrity 函数负责代码完整性验证的初始化; PATCHGUARD 动态验证系统文件是否被修改.

表 1 效果比较

	关闭 LOAD 模式验证	关闭 KERNEL 模式验证	关闭 PATCH GUARD	结果
实验一	✓			无效
实验二	✓	✓		蓝屏
实验三		✓	✓	无法启动
实验四	✓	✓	✓	成功

为了充分证明本方法的性能优越性, 需要设计大量的实验进行对比. 通过对不同加载运行驱动程序的方法进行对比, 本文主要从“测试模式”水印、开机垃圾信息和是否需要测试三个方面进行比较. 按表 2 显示, 若在水印处划勾, 表示该情况会在系统桌面角落显示“测试模式”字样的水印. 若在开机信息处划勾, 表示该情况在开机时会打印很多普通用户无关的系统信息. 若在测试签名处划勾, 表示该情况下运行驱动程序需要进行测试签名.

表 2 不同方法比较比较

方法	水印	开机信息	测试签名	其他
开机 F8		✓		需重复操作
修改启动选项		✓		无效
测试模式	✓		✓	影响用户体验
官方签名				需要花费
本文方案				推荐本方案

其中, “开机 F8”是指在开机时按 F8 键选择“禁用驱动程序签名强制”来取消强制驱动程序签名要求. “修改启动选项”是通过 BCDEDIT 命令行来设置关闭完整性检查. “测试模式”和“官方签名”请参照官方文档文献[1].

6 结束语

提出了通过修改系统关键函数达到加载无签名驱动程序的方法. 具有一次性、不影响用户体验、驱动程序完全无需签名的优点. 通过自主开发的驱动程序实现主机监管系统来对系统进行监控和管理更具有自主性,更能适用于需要高度安全性的环境.

为了进一步提高主机安全性,应对驱动程序的加载运行采用用户个性化的管理. 优秀的操作系统应该在需要的时候对驱动程序签名进行严格验证,在需要加载驱动程序时提供相应的免签名选项,从而实现用户对计算机真正的完全掌控.

参考文献:

- [1] 微软官方. Digital signatures for kernel modules on systems running windows vista [EB/OL]. [2013-04-20]. America: Microsoft, 2007. <http://www.microsoft.com/whdc/winlogo/drvsign/kmsigning.msp>
- [2] 韩卓,冉晓曼,吕文高. Windows 7 内核完整性验证机制研究[J]. 信息工程大学学报,2011,12(6):764-768.
- [3] Skape, Skywing. Bypassing patchguard on windows x64 [EB/OL]. [2013-04-20]. America: Uninformed, 2006. <http://uninformed.org/index.cgi?v=3&a=3&t=sumry>
- [4] Skywing. Subverting patch guard version 2 [EB/OL].

[2013-04-20]. America: Uninformed, 2007. <http://www.uninformed.org/?v=6&a=1&t=sumry>

- [5] Skywing. Patch guard reloaded [EB/OL]. [2013-04-20]. America: Uninformed, 2007. <http://uninformed.org/?v=8&a=5&t=sumry>.
- [6] 邵伟,何静. Vista 中信任链建立机制研究[J]. 计算机工程与设计,2009,30(15):3494-3497.
- [7] Matthew Conover. Assessment of Windows Vista Kernel-mode security [EB/OL]. [2013-04-20]. America: Symantec Corporation, 2006. http://www.symantec.com/avcenter/reference/Windows_Vista_Kernel_Mode_Security.pdf
- [8] Fyyre. Disable Patch guard & driver signing on X64 windows 7 + SP1 [EB/OL]. [2013-04-20]. America: binary modifications, 2012. http://fyyre.ivory-tower.de/projects/no_pg_ds_v3.rar.
- [9] 微软官方. Windows 7 code integrity (ci.dll) security policy [EB/OL]. [2013-04-20]. America: Microsoft, 2013. <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp1327.pdf>

作者简介:

张智男, (1988—), 硕士研究生. 研究方向为网络信息安全.

蔡晓东男, (1955—), 教授, 博士生导师. 研究方向为网络信息安全.

(上接第100页)

- [4] 罗天洪. 面向敏捷制造的制造执行系统调度策略研究[J]. 机床与液压, 2009(5):1-4.
- [5] 李应, 杨善林, 郑家强. 敏捷制造系统的基于 Agent 的混合调度[J]. 系统仿真学报, 2009, 21(12): 3763-3767.
- [6] Li P G, Li S X, Rao Y Q. Coalition formation and its application in planning for agile manufacturing cell[J]. Int J Adv Manuf Technol, 2004, 24(3-4): 298-305.

作者简介:



孔令夷男, (1977—), 副教授. 研究方向为敏捷制造.