内核是操作系统最核心的组成，而驱动在内核源码中可以占到70%的部分，是操作系统内核漏洞的主要来源。由于驱动程序十分依赖操作系统的其它部分，任何内核和驱动支持库接口的改变都会引发那些相关联的驱动代码的大数量调整。然而为了解决新的需求和改善性能，Linux内核演变的非常迅速，这些演变，使得一系列相关的设备驱动程序必须随着更新，但目前这种更新操作基本是手工的，十分费时费力还易于出错。为了改善这种现状，我们提出我们的研究目标：由A、B两版本Linux内核源码和A版本内核的设备驱动程序，分析A版设备驱动对A版内核的依赖关系，分析这些依赖关系在两版内核中的差异信息，根据差异信息通过计算机辅助方式构建起适合B版内核的B版设备驱动，然后在B版内核中编译B版设备驱动，并加载驱动模块进行验证，最后将这一研究过程最大化的实现自动化，进而开发出驱动自动更新的辅助工具。

国内外现状：

目前驱动源码的更新方式主要是程序员手工更新，根据内核修改日志和内核补丁文件获取内核差异信息，根据差异信息人工进行驱动代码的更新操作。现在尚未有工具可以完整实现设备驱动程序的自动更新。

已有的相关技术有内核差异分析技术、函数调用关系分析技术。内核差异分析技术有基于文本的和基于语法树的差异分析技术，用于获取内核差异信息，但功能不全面，并不能实现驱动的自动更新。函数调用关系分析技术是分析函数之间的调用关系，这项技术可以作为驱动更新的前期工作。

已有相关工具有差异应用工具Coccinelle工具，它最初被设计于解决Linux内核与驱动的协同进化问题即实现驱动的自动更新，认为应由Linux 内核开发人员使用语义补丁语言说明两个版本之间的变动，然后由Coccinelle 工具将这个补丁打到驱动上完成升级，但其设计理念并没有被linux内核接受，如今Coccinelle 转而专注于在Linux内核中发现并修复漏洞。

意义：

1、Linux内核演变十分迅速，若是设备驱动程序能自动更新，可以在很大程度上减少设备驱动程序随着内核变化产生的更新操作所耗费的时间。

2、设备驱动程序更新操作可能导致超过35%的代码修改。如此庞大数量的驱动代码，驱动代码的自动更新可以在一定程度上提高设备驱动程序更新的效率。

3、作为占据内核源码较大比重而且是操作系统内核漏洞主要来源的驱动代码，驱动程序的自动更新可以提高驱动代码的稳固性，进一步提高操作系统的安全和稳固。

研究内容：

1. 了解设备驱动程序的组成，分析A版的设备驱动程序，获取基本框架（LKM）、数据结构及功能函数。

2. 分析内核源码目录组织结构，提取内核目录文件差异信息。

3. 分析A版设备驱动程序对A版Linux内核的依赖关系，例如引用的头文件，使用的内核接口函数等。

4. 分析Linux的A版内核和B版内核，提取设备驱动程序依赖的内核接口原型等的差异信息。

5.构建目标设备驱动程序。根据提取的内核差异信息，在A版的设备驱动程序基础上构建目标版本的驱动程序。

参考文献：

[1]Yoann Padioleau, L. Lawall, Gilles Muller. Semantic Patches-Documenting and Automating Collateral Evolutions in Linux Device Drivers. 2007

[2]Yoann Padioleau, Julia L. Lawall, Gilles Muller. SmPL A Domain-Specific Languagefor Specifying Collateral Evolutionsin Linux Device Drivers. Electronic Notes in Theoretical Computer Science, 2007,166:47–62.

[3]Yoann Padioleau, René Rydhof Hansen, Julia Lawall, Gilles Muller. Towards Documenting and Automating Collateral Evolutions in Linux Device Drivers. https://hal.archives-ouvertes.fr. 2007.

[4]Julia L. Lawall, Gilles Muller, Nicolas Palix. Enforcing the Use of API Functions in Linux Code. ACM. 2009.

[5]Julio Sincero, Reinhard Tartler, Daniel Lohmann, Wolfgang Schröder-Preikschat. Efficient Extraction and Analysis of Preprocessor-Based Variability. ACM. 2010.

[6]Suman Saha, Julia Lawall, Gilles Muller. An approach to improving the structure of error-handling code in the linux kernel. 2011.

[7]Vladimir Itsykson, Alexey Zozulya. Automated program transformation for migration to new libraries. IEEE. 2011.

[8]Christian Kästner, Paolo G. Giarrusso, Tillmann Rendel, Thorsten Berger. Variability-aware parsing in the presence of lexical macros and conditional compilation. ACM. 2011.

[9]Julia Lawall. Coccinelle\_reducing the barriers to modularization in a large C code base. ACM. 2014.

[10]Davide Di Ruscio, Patrizio Pelliccione. Simulating upgrades of complex systems\_the case of free and open source software. Information and Software Technology. 2014, 56:438–462.

[11]Maelick Claes, Tom Mens, Roberto Di Cosmo, Jérôme Vouillon. A historical analysis of Debian package incompatibilities. IEEE. 2015.

[12]Luis R. Rodriguez, Julia Lawall. Increasing Automation in the Backporting of Linux Drivers Using Coccinelle. IEEE. 2015.

[13]Masatomo Hashimoto, Akira Mori, and Tomonori Izumida. A Comprehensive and Scalable Method for Analyzing Fine-Grained Source Code Change Patterns. IEEE. 2015.

[14]Julien Brunel, Damien Doligez, René Rydhof Hansen, Julia L. Lawall, Gilles Muller. A Foundation for Flow-Based Program Matching Using Temporal Logic and Model Checking. ACM. 2009.

[15]Hitoshi Araki, Shin Futagami, and Kaoru Nitoh. A non-stop updating technique for device driver programs on the IROS platform. IEEE. 1995.

[16]E. M. CLARKE, E. A. EMERSON, A. P. SISTLA. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. ACM Transactions on Programming Languages and Systems. 1986 , Vol. 8, No. 2, Pages 244-263.

[17]Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez,Martin Monperrus. Fine-grained and Accurate Source Code Differencing. ACM. 2014. [18]Jesper Andersen, Julia L. Lawall. Generic Patch Inference. IEEE. 2008.

[19]Hiroyuki Tanaka, Yoshinari Nomura, and Hideo Taniguchi. Run-time Updating of Network Device Drivers. IEEE. 2009.

[20]Cristiano Giuffrida, Anton Kuijsten, Andrew S. Tanenbaum. Safe and Automatic Live Update for Operating Systems. ACM. 2013.

[21]Yoann Padioleau, Julia L. Lawall, Gilles Muller. Understanding Collateral Evolution in Linux Device Drivers. ACM. 2006.

[22]Yoann Padioleau, Julia Lawall, René Rydhof Hansen, Gilles Muller. Documenting and Automating Collateral Evolutions in Linux Device Drivers. ACM. 2008.

[23]Olatunji Ruwase, Michael A. Kozuch, Phillip B. Gibbons, Todd C. Mowry. Guardrail: A High Fidelity Approach to Protecting Hardware Devices from Buggy Drivers. ACM. 2014.

[24]Jia-Ju Bai, Hu-Qiu Liu, Yu-Ping Wang, Shi-Min Hu. Runtime Checking for Paired Functions in Device Drivers. IEEE. 2014.

[25]李雪. 基于语法的源代码差异分析及与内核分析平台的集成. 2016.

[26]武祥晋. 基于语法树的源码差异分析工具设计与显卡驱动分析. 2015.

[27]颜世勋. 内核差异分析与展示. 2015.

[28]王欢. Linux 内核错误追溯系统的研究与设计. 2015.

[29]孙卫真，杜香燕，向勇等. 基于RTL的函数调用图生成工具CG-RTL.小型微型计算机系统. 2014, V01.35 No.3.

[30]黄双玲，黄章进，顾乃杰. 基于ＣＦＧ 的函数调用关系静态分析方法. 计算机系统应用. 2015,24(11).