


Chapter 6

Trusting Critical Open Source Components: The Linux Case Study

Marcelo Schmitt

Universidade de Sao Paulo, Brazil

Paulo Meirelles

 <https://orcid.org/0000-0002-8923-2814>
Universidade de Sao Paulo, Brazil

ABSTRACT

Device drivers are an elementary part of the Linux kernel and comprise roughly 2/3 of the project's lines of code. Even though the fraction of device driver code in a conventional operating system (OS) can vary, some of these components are essential for system functioning. In addition, the Linux kernel is used in a wide range of applications, from cloud service providers to embedded systems and supercomputers. If GNU/Linux systems should be trustworthy to justify running them in those environments, then testing the kernel is fundamental. However, since device drivers are designed to interface with hardware, conventional test approaches may not suit the occasions when devices are unavailable at test time. This raises the question: How are device drivers tested?

DOI: 10.4018/978-1-6684-4785-7.ch006

INTRODUCTION

Trusting Critical FOSS Components

The present chapter revolves around the theme of dependability in Free and Open Source Software (FOSS) systems, with a specific focus on Linux. The study highlights the significance of the FOSS infrastructure that underpins a substantial portion of the software and online systems we rely on for both business and personal purposes. The examination centers on Linux device drivers, serving as a representative case study to raise questions about the level of trust that can be placed in the FOSS ecosystem. The material conducts an exhaustive analysis of the software testing methodologies utilized by the Linux project to assess the quality and reliability of device drivers. This discussion sheds light on the transparency inherent in the FOSS model, as it offers a unique opportunity to inspect and scrutinize the internal workings of the software. This openness stands in contrast to the limited ability to examine proprietary products that drive mission-critical systems, which lack similar transparency.

Linux, as an enduring project, forms the foundation of a substantial portion of today's modern computational infrastructure, a fact that underscores its importance and widespread adoption. The success of Linux is attributed, in part, to the implementation of innovative business models that facilitate collaboration between individuals and organizations with diverse perspectives. This collaborative environment enables stakeholders to leverage shared resources while being actively encouraged to contribute back to the community through money investments and technological enhancements. The chapter presents a valuable exploration of the factors contributing to the positive reputation and resilience of the Linux ecosystem, underscoring the significance of effective strategies that foster cooperation and mutual support within the FOSS community.

Device Drivers

Device drivers are an elementary part of the Linux kernel and comprise roughly 2/3¹ of the project's lines of code. Even though the fraction of device driver code in a conventional operating system (OS) can vary, some of these components are essential for system functioning. In addition, the Linux kernel is used in a wide range of applications, from cloud service providers to embedded systems and supercomputers (Corbet and Kroah-Hartman, 2017). If GNU/Linux systems should be trustworthy to justify running them in those environments, then testing the kernel is fundamental. However, since device drivers are designed to interface with hardware, conventional test approaches may not suit the occasions when devices are unavailable at test time. This bares the question: how are device drivers tested?

23 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/trusting-critical-open-source-components/326642

Related Content

Incorporating Free/Open-Source Data and Tools in Software Engineering Education

Liguo Yu, David R. Surma and Hossein Hakimzadeh (2015). *Open Source Technology: Concepts, Methodologies, Tools, and Applications* (pp. 381-391). www.irma-international.org/chapter/incorporating-freeopen-source-data-and-tools-in-software-engineering-education/120926

Governance and the Open Source Repository

R. Todd Stephens (2007). *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives* (pp. 480-493). www.irma-international.org/chapter/governance-open-source-repository/21210

Open Source for Higher Conventional and Open Education in India

Ramesh C. Sharma (2015). *Open Source Technology: Concepts, Methodologies, Tools, and Applications* (pp. 1247-1264). www.irma-international.org/chapter/open-source-for-higher-conventional-and-open-education-in-india/120967

Assessing Quality of Mobile Applications Based on a Hybrid MCDM Approach

Puneet Kumar Aggarwal, P.S. Grover and Laxmi Ahuja (2019). *International Journal of Open Source Software and Processes* (pp. 51-63). www.irma-international.org/article/assessing-quality-of-mobile-applications-based-on-a-hybrid-mcdm-approach/238010

Software Reliability Prediction Using Cuckoo Search Optimization, Empirical Mode Decomposition, and ARIMA Model: CS-EEMD-ARIMA Based SRGM

Ankur Choudhary and Anurag Singh Baghel (2016). *International Journal of Open Source Software and Processes* (pp. 39-54). www.irma-international.org/article/software-reliability-prediction-using-cuckoo-search-optimization-empirical-mode-decomposition-and-arima-model/182783