DESAFIOS NO PROVISIONAMENTO DE SERVIDORES LINUX COM SECURE BOOT HABILITADO: IMPACTO NAS RESTRIÇÕES DE ASSINATURA DE MÓDULOS DE KERNEL

Gustavo Behnck Cardoso, Flávio Oliveira Monteiro, Luiz Thiago Gonçalves Cassab, Matheus Henrique Gonçalves Rodrigues, Irapuan Rodrigues de Oliveira Filho, Luiz Eduardo Camargo Aranha Schiavo.

Universidade do Vale do Paraíba/Faculdade de Engenharias Arquitetura e Urbanismo, Avenida Shishima Hifumi, 2911, Urbanova - 12244-000 - São José dos Campos-SP, Brasil, gustavobehnckcardoso@gmail.com, flaviomonteiro2013@gmail.com, ltcassab@gmail.com, matheus.rodrigues2704@gmail.com, irapuan.rodrigues@gmail.com, eduschiavo@gmail.com.

Resumo

O *Secure Boot*, uma tecnologia de segurança da especificação UEFI, foi projetado para garantir que apenas softwares autenticados sejam executados durante a inicialização do sistema. Embora essencial para prevenir ameaças como malware, sua implementação em servidores Linux, especialmente em *datacenters* e *clusters* HPC, enfrenta desafios complexos. A necessidade de assinatura digital de módulos de kernel impede o carregamento de módulos não assinados, comprometendo a operação de hardware especializado, como GPUs. Em ambientes distribuídos, a gestão de certificados para centenas ou milhares de servidores adiciona complexidade, aumentando o risco de falhas operacionais. Este artigo investiga os obstáculos técnicos do uso de *Secure Boot* em ambientes massivos, simulando provisionamento de servidores em Fedora 40 e avaliando os desafios de assinatura e distribuição de módulos. Soluções como automação de processos de certificação e uso de infraestruturas de chave pública internas são discutidas como alternativas para mitigar esses problemas e garantir tanto segurança quanto operacionalidade em ambientes críticos.

Palavras-chave: Secure Boot. Linux. Provisionamento de Servidores. Módulos de Kernel. UEFI. Datacenter. HPC.

Área do Conhecimento: Engenharia de Computação.

Introdução

O *Secure Boot* é uma tecnologia de segurança incorporada à especificação UEFI (*Unified Extensible Firmware Interface*), destinada a garantir que apenas software autenticado e confiável seja executado durante o processo de inicialização de um sistema (Smith, 2024). Desenvolvido como resposta às crescentes ameaças de *malware* e *rootkits*, *o Secure Boot* tem sido amplamente adotado em diversas plataformas, desde dispositivos de consumo até servidores empresariais (Bishop, 2004).

Apesar de seu propósito de reforçar a segurança, a implementação do *Secure Boot* em sistemas Linux, particularmente em servidores, apresenta uma série de desafios complexos. Em ambientes de produção como *datacenters* e *clusters* HPC (*High Performance Computing*), onde flexibilidade, escalabilidade e automação são cruciais, as exigências de assinatura digital de módulos de *kernel* impõem barreiras significativas. Módulos essenciais para o funcionamento de hardware específico ou para a implementação de funcionalidades avançadas precisam ser assinados digitalmente para serem carregados com *o Secure Boot* habilitado. A ausência de uma assinatura válida impede o carregamento do módulo, comprometendo a operação do sistema (Red Hat, 2024a).

Além disso, em ambientes distribuídos com centenas ou milhares de servidores, o processo de assinatura e distribuição de certificados para cada módulo de *kernel* é tecnicamente complexo e suscetível a falhas, resultando em possíveis paradas operacionais significativas. A necessidade de atualizações frequentes de hardware e software em ambientes HPC adiciona uma camada extra de dificuldade ao gerenciamento desses sistemas sob as restrições do *Secure Boot* (Anderson, 2020).

Este artigo investiga as dificuldades práticas e técnicas no provisionamento de servidores Linux com *Secure Boot* habilitado, com foco especial em ambientes distribuídos (HPC, *Datacenters, etc.*). A

análise inclui uma revisão das limitações impostas pelo *Secure Boot*, exemplos práticos observados em *clusters* HPC e uma discussão sobre alternativas e soluções para mitigar os problemas encontrados. O objetivo é fornecer uma visão abrangente dos desafios enfrentados por administradores de sistemas e engenheiros de infraestrutura ao lidar com a assinatura de módulos de *kernel* em ambientes críticos, oferecendo recomendações para melhorar a operacionalidade sem comprometer a segurança.

Metodologia

Para investigar os desafios do provisionamento de servidores Linux com *Secure Boot* habilitado, foi adotada uma abordagem prática e experimental em um ambiente controlado. A metodologia deste estudo incluiu as seguintes etapas:

1. Configuração do Ambiente de Testes: O ambiente de testes foi configurado utilizando um *hypervisor* baseado em libvirt e KVM (*Kernel-based Virtual Machine*), ambos executando o Fedora 40 como sistema operacional. O Fedora 40 foi escolhido devido ao seu suporte robusto para UEFI e *Secure Boot*, além de ser uma distribuição amplamente utilizada em ambientes de produção (Love, 2010). Tanto o *hypervisor* quanto as máquinas virtuais (VMs) estavam rodando Fedora 40, simulando condições reais de um servidor de produção. As VMs foram configuradas para utilizar UEFI com *Secure Boot* habilitado, conforme recomendado por Red Hat (2024b).

O hardware subjacente ao *hypervisor* incluía processador com suporte a virtualização por hardware (VT-x/AMD-V), 20 GB de RAM (*Random access memory*) e armazenamento SSD (*Solid State Drive*), garantindo a capacidade necessária para suportar múltiplas VMs, conforme descrito por Garrett (2012).

2. Compilação e Assinatura de um Módulo de *Kernel* "*Dummy*": Para testar e validar o processo de assinatura e carregamento de módulos de *kernel* sob o *Secure Boot*, foi compilado um módulo de *kernel* "*dummy*", seguindo as práticas descritas por Red Hat (2024b). Este módulo, sem funcionalidades específicas, serviu como um artefato de teste. Após a compilação, o módulo "*dummy*" foi assinado digitalmente utilizando uma infraestrutura de chave pública (PKI) interna. O processo de assinatura incluiu a criação de chaves criptográficas, a geração de certificados e a inserção desses certificados no banco de dados de chaves da UEFI (Smith, 2024; Wilkins; Richardson, 2013).

3. Provisionamento e Validação em Ambientes Distribuídos: As VMs Fedora 40 foram provisionadas pela rede utilizando PXE (*Preboot Execution Environment*). As imagens de sistema utilizadas no provisionamento incluíam o módulo "*dummy*" devidamente assinado. A validação foi realizada para garantir que as VMs inicializassem corretamente e que o módulo "*dummy*" fosse carregado sem falhas, comprovando o sucesso do processo de assinatura e a conformidade com os requisitos do *Secure Boot* (Trudel-Lapierre, 2017).

4. Teste de Escalabilidade: Para simular um ambiente de produção massivamente distribuído, o teste de escalabilidade foi repetido em 5 VMs Fedora 40. O processo de provisionamento foi realizado em múltiplas instâncias para avaliar a escalabilidade da solução adotada e identificar possíveis pontos de falha. A ênfase foi colocada na distribuição e validação dos certificados de assinatura, bem como na resposta do sistema em caso de falha ao carregar o módulo "*dummy*" (Palau, 2011).

5. Análise de *Logs* e Diagnóstico de Problemas: Após a conclusão dos testes de provisionamento, os logs de inicialização das 5 VMs Fedora 40 foram analisados detalhadamente para identificar qualquer erro relacionado à assinatura e ao carregamento do módulo "*dummy*". A análise incluiu a verificação de mensagens do GRUB2 (*GRand Unified Bootloader*) e do sistema operacional, além da investigação de falhas de inicialização ou mau funcionamento potencial relacionado ao módulo (Bishop, 2004).

A configuração do *hypervisor* e das VMs, ambos rodando Fedora 40, juntamente com a assinatura e validação do módulo de *kernel* "*dummy*", permite um exame detalhado das complexidades envolvidas no uso do *Secure Boot* em ambientes Linux virtualizados e distribuídos. As dificuldades enfrentadas durante o processo foram documentadas e analisadas, fornecendo uma base sólida para as discussões e recomendações subsequentes apresentadas neste artigo.

Resultados

Os resultados indicam que, além da complexidade do processo de assinatura de módulos de *kernel*, o processo de *enroll* de certificados em ambientes remotos e distribuídos, como *datacenters* e *clusters*

HPC, apresenta desafios significativos. Tais ambientes requerem que o certificado de assinatura seja distribuído e instalado em centenas ou milhares de máquinas, o que pode ser logisticamente complexo e suscetível a falhas (Anderson, 2020).

Carregamento via GRUB2 e consequências da falha na importação de módulos não assinados: Quando o *Secure Boot* está habilitado, o GRUB2 desempenha um papel crucial no processo de inicialização, garantindo que todos os módulos do *kernel* que ele carrega sejam devidamente assinados (Smith, 2023). Se um módulo de *kernel* não estiver assinado ou se a assinatura não puder ser verificada, o GRUB2 impedirá o carregamento do módulo. Isso significa que o dispositivo associado ao módulo não funcionará corretamente, comprometendo a operação do servidor.

Por exemplo, em um cluster HPC que depende de GPUs (*Graphics Processing Units*) para processamento paralelo, os *drivers* proprietários das empresas NVIDIA ou AMD – líderes do mercado de GPU global (Shilov, 2024) – precisam ser carregados para que as GPUs possam ser utilizadas. Se esses *drivers* não estiverem assinados corretamente, as GPUs não serão inicializadas, resultando em falhas nas tarefas de computação que dependem desse hardware. Isso pode causar interrupções significativas em ambientes onde a continuidade e a eficiência do processamento são críticas (Love, 2010).

Exemplo de *cluster* HPC com provisionamento pela rede: Em um ambiente de *cluster* HPC, os servidores são frequentemente provisionados pela rede usando técnicas como PXE para inicializar sistemas a partir de uma imagem centralizada. Quando o *Secure Boo*t está habilitado, todos os módulos de *kernel* incluídos na imagem precisam ser assinados, incluindo *drivers* de rede, armazenamento e qualquer outro hardware específico utilizado pelos nós do cluster.

Um dos desafios nesse cenário é garantir que todas as imagens usadas no provisionamento incluam apenas módulos devidamente assinados. Qualquer módulo que não seja assinado impedirá que o servidor inicialize corretamente, resultando em falhas na implantação ou na operação do *cluster*. O gerenciamento desse processo em um ambiente distribuído pode ser extremamente complexo, especialmente se o hardware ou o *kernel* precisar ser atualizado com frequência (Red Hat, 2024a).

Discussão

A implementação do *Secure Boot* em ambientes distribuídos adiciona complexidades operacionais significativas. A necessidade de distribuir certificados de assinatura de forma segura e eficiente em um *datacenter* ou *cluster* HPC pode representar um desafio operacional substancial. Além disso, o tempo e o esforço necessários para assinar e realizar o *enroll* certificados para módulos de *kernel* compilados a partir do código-fonte, como os *drivers* da NVIDIA e AMD, podem levar a atrasos na implantação e atualização de sistemas (Barrett; Byrnes, 2003).

No contexto de *datacenters*, onde a automação é essencial para o gerenciamento eficiente, o processo manual de assinatura e *enroll* de certificados para *Secure Boot* pode se tornar um gargalo. Soluções como a implementação de PKIs internas e o uso de scripts automatizados para gerenciamento de certificados podem mitigar parte desses desafios, mas não eliminam a complexidade inerente ao processo (Palau, 2011).

Além disso, a utilização de chaves personalizadas, em vez das chaves padrão fornecidas pelo fabricante do hardware, oferece maior flexibilidade no gerenciamento de módulos assinados. No entanto, essa abordagem requer um processo rigoroso de gerenciamento de chaves para garantir a segurança e a integridade do sistema (Smith, 2024).

Conclusão

Conclui-se que, embora o *Secure Boot* ofereça uma camada adicional de segurança, sua implementação atual em servidores Linux, especialmente em ambientes remotos e distribuídos como *datacenters* e *clusters* HPC, apresenta desafios significativos. A complexidade do processo de *enroll* de certificados e a assinatura de módulos de *kernel*, especialmente para *drivers* proprietários como os da NVIDIA e AMD demandam uma abordagem cuidadosa e frequentemente customizada.

É essencial que os administradores de sistemas considerem cuidadosamente as implicações do *Secure Boot* e estejam preparados para implementar soluções que garantam tanto a segurança quanto a operacionalidade dos servidores. A adoção de infraestruturas de chave pública internas, automação de processos de assinatura e gerenciamento rigoroso de chaves são medidas recomendadas para

mitigar os desafios identificados. Além disso, a consideração de alternativas como desativação seletiva do *Secure Boot* ou uso de chaves personalizadas deve ser avaliada com base nas necessidades específicas de segurança e operacionais de cada ambiente

Referências

ANDERSON, R. **Security Engineering: A Guide to Building Dependable Distributed Systems**. Wiley, 2020. Cap 5, p. 194. ISBN 978-1119642787.

BARRETT, R. E. S. D. J.; BYRNES, R. G. **Linux Security Cookbook: Security Tools & Techniques**. O'Reilly Media, 2003. Cap 4, p. 78-87. ISBN 978-0596003913.

BISHOP, M. A. **Introduction to Computer Security**. Addison-Wesley Professional, 2004. Cap.17, p. 365-369. Cap 22, p. 456-458. ISBN 978-0321247445.

GARETT, M. **The Security of Secure boot**. 2012. Disponível em: https://mjg59.dreamwidth.org/12897.html. Acesso em: 22 set. 2024.

LOVE, R. **Linux Kernel Development**. Pearson Education, 2010. Cap 2, p. 14. Cap 17, p. 248-338. ISBN 978-0-672-32946-3.

PALAU, V. T. **White Paper: Secure Boot impact on Linux**. 2011. Disponível em: https://canonical.com/blog/white-paper-secure-boot-impact-on-linux. Acesso em: 22 set. 2024.

Red Hat. **Security hardening**. 2024. Disponível em: https://docs.redhat.com/en-us/documentation/red_hat_enterprise_linux/8/pdf/security_hardening/Red_Hat_Enterprise_Linux-8-Security_hardening-en-US.pdf. Acesso em: 22 set. 2024.

Red Hat. **Signing a kernel and modules for Secure Boot**. 2024. Disponível em: https://docs.redhat.com/en-us/documentation/red_hat_enterprise_linux/8/pdf/managing_monitoring_and_updating_the_kernel/Red_Hat_Enterprise_Linux-8-Managing_monitoring_and_updating_the_kernel-en-US.pdf. Acesso em: 22 set. 2024.

SHILOV, A. **Nvidia's grasp of desktop GPU market balloons to 88% - AMD has just 12%, Intel negligible, says JPR**. Tom's hardware, 2024. Disponível em: https://www.tomshardware.com/pc-components/gpus/nvidias-grasp-of-desktop-gpu-market-balloons-to-88-amd-has-just-12-intel-negligible-says-jpr. Acesso em: 22 set. 2024.

SMITH, R. **Managing EFI Boot Loaders for Linux: Controlling Secure Boot**. 2023. Disponível em: https://www.rodsbooks.com/efi-bootloaders/controlling-sb.html. Acesso em: 22 set. 2024.

SMITH, R. **Managing EFI Boot Loaders for Linux: Dealing with Secure Boot**. 2024. Disponível em: https://www.rodsbooks.com/efi-bootloaders/secureboot.html. Acesso em: 22 set. 2024.

TRUDEL-LAPIERRE, M. **How to sign things for Secure Boot**. Ubuntu, 2017. Disponível em: https://ubuntu.com/blog/how-to-sign-things-for-secure-boot. Acesso em: 22 set. 2024.

UEFI Forum, Inc. **Unified Extensible Firmware Interface (UEFI) Specification 2.10**. 2022. Disponível em: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_10_Aug29.pdf. Acesso em: 22 set. 2024.

WILKINS, R.; RICHARDSON, B. **UEFI Secure Boot in Modern Computer Security Solutions**. 2013. Disponível em: https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf. Acesso em: 22 set. 2024.