Anais do Congresso Brasileiro de Iniciação Científica
(https://periodicos.unisanta.br/COB)

ISSN 2317-1316

ANÁLISE PRÁTICA DO SECURE BOOT EM MÁQUINAS LINUX UTILIZADAS EM LABORATÓRIO ACADÊMICO

PRACTICAL ANALYSIS OF SECURE BOOT ON LINUX MACHINES USED IN AN ACADEMIC LABORATORY

Bruno Elie Albanese Hayek, Enzo Benedetto Proença, Elieder Damasceno Sousa, Bruno da Silva Rodrigues, Fabio Kawaoka Takase

Universidade Presbiteriana Mackenzie, Faculdade de Computação e Informática, curso de Ciência da Computação
E-mail para contato: brunoelie.hayek@mackenzista.com.br

RESUMO – No contexto de boot por rede em endpoints, existem possibilidades de ataques que utilizam vulnerabilidades e malwares, tornando assim complexa a tarefa de segurança do processo de boot. Diante disso, uma análise sobre o mecanismo de segurança do firmware do computador torna-se necessária, com destaque a laboratórios acadêmicos que utilizam o PXE boot para seus endpoints Linux. O principal objetivo no estudo é examinar o funcionamento do UEFI Secure Boot e analisar o seu comportamento relativo a endpoints com Ubuntu e Debian, fornecendo uma conclusão a respeito do mecanismo. Nesta pesquisa foram analisados métodos qualitativos pelos quais o Secure Boot opera em Linux, adicionalmente foi implementado um binário para o UEFI, posteriormente aplicado e testado em um ambiente com PXE boot.

Palavras-chave: Secure Boot; Linux; UEFI; PXE boot; Segurança.

ABSTRACT – In the context of network booting on endpoints, there are attack possibilities that exploit vulnerabilities and malware, making the task of securing the boot process complex. In view of such context, an analysis of computer firmware security mechanisms becomes necessary, with a particular focus on academic laboratories that use PXE boot systems for their Linux endpoints. The main objective of this study is to examine how UEFI Secure Boot works, analyze its behavior on endpoints running Ubuntu and Debian and provide a conclusion concerning the studied mechanism. The study adopts a qualitative approach, evaluating the methods by which Secure Boot operates with Linux and additionally a UEFI binary was also implemented and later applied and tested in a PXE boot test environment.

Keywords: Secure Boot; Linux; UEFI; PXE boot; Security.

ISSN 2317-1316

1 INTRODUÇÃO

Laboratórios didáticos e de pesquisa em universidades tem seus equipamentos gerenciados pelos seus departamentos de tecnologia da informação, abordagem que garante a segurança de seus *endpoints* e equipamentos de rede. Nesse contexto em que o controle de acesso, governança e segurança são fundamentais, há uma predominância do uso do sistema operacional Windows, representando 66,14% dos sistemas operacionais no mercado, enquanto o Linux possui 2,94% do total de mercado (Statcounter Global Stats, 2025).

Considerando o perfil específico dos profissionais de tecnologia, universo em que muitos trabalharão com o desenvolvimento de software, cerca de 40% dos desenvolvedores de software utilizam o sistema operacional Linux (Stack Overflow, 2022). Há, portanto, uma necessidade de equiparar os recursos que suportam o uso seguro e eficiente do sistema operacional Linux em *endpoints* no ambiente acadêmico com os disponibilizados pela Microsoft no uso do Windows.

O ambiente acadêmico estabelece alguns desafios adicionais como o uso compartilhado de computadores por diversos estudantes, um contexto em que a reinicialização dos computadores para um estado inicial conhecido é importante. O foco deste trabalho está no estudo da segurança do primeiro passo do processo de configuração de *endpoints*, na automação da instalação do sistema operacional Linux utilizando os mecanismos de *boot* por rede local. Um exemplo de tal mecanismo é o PXE (*Preboot eXecution Environment*) *boot*, o qual permite um servidor de protocolo TFTP (*Trivial File Transfer Protocol)* com configuração DHCP (*Dynamic Host Configuration Protocol*) fornecer uma imagem e arquivos de configurações para o *endpoint*, concedendo assim arquivos suficientes para o processo de *boot* (Microsoft, 2025).

Em uma circunstância de um *boot* do sistema Linux, antes do servidor fornecer a imagem do sistema operacional, ele envia o *bootloader* ao *endpoint*, o qual servirá para carregar a imagem (Ruth e Richardson, 2014). O processo inteiro do PXE *boot* pode ocorrer de forma segura, porém o processo de verificação da autenticidade do *bootloader* ocorre no ambiente de UEFI (*Unified Extensible Firmware Interface*) Secure Boot (Wilkins et al, 2023). Logo, com base neste contexto, mostra-se uma oportunidade que manifesta a importância de uma avaliação acerca de um dos mecanismos de segurança do *firmware* do computador e seu funcionamento.

ISSN 2317-1316

Além dessa ocasião, observa-se que certas empresas estão embutindo *rootkits* não maliciosos que têm acesso diretamente ao *kernel*, como empresas de jogos com sistema *anti-cheat* (Dorner e Klausner, 2024). Portanto, tais mecanismos podem ser alterados por agentes maliciosos e computadores de usuários destes *softwares* podem ser comprometidos.

Um exemplo de uma situação real de segurança é a da empresa Riot Games, conhecida pelos jogos “League of Legends” e “Valorant”, que implementou um sistema *anti-cheat* proprietário denominado de “Vanguard”, o qual atua de maneira semelhante a um *rootkit* (Dorner e Klausner, 2024). Caso ele seja comprometido por um agente malicioso, jogadores poderão ser prejudicados, visto que esse sistema *anti-cheat* atua em nível de *kernel*.

Apesar de antivírus estarem fortalecendo o código e reduzindo vetores de ataques, os sistemas operacionais ainda não conseguem detectar *malwares* que são executados no processo de *booting* e que estão escondidos no *firmware*, por isso é imprescindível uma análise de ambos os componentes, sendo assim a única barreira que pode prevenir executáveis não autorizados pelo usuário (Wilkins e Richardson, 2019).

Este trabalho aborda padrões de segurança no sistema de *boot* em computadores com o mecanismo do UEFI Secure Boot, em que foi criado um programa autenticado pelo Secure Boot, fornecido pelo protocolo TFTP, e que solicita ao PXE *server* outro arquivo, comprovando a assinatura do Secure Boot e o funcionamento do protocolo de inicialização por rede PXE *boot*.

Ao final do estudo foi realizada uma ponderação sobre o Secure Boot, concluindo e indicando quais seriam as vantagens e desvantagens de ativá-lo, especialmente no contexto de *boot* por rede, empregando o PXE para *endpoints* Linux, com ênfase em um ambiente laboratorial acadêmico onde há a necessidade de uma instalação segura nos computadores.

Nas seções a seguir são descritos os materiais e métodos utilizados neste trabalho, apresentação dos resultados e suas discussões e, por fim, conclusões do estudo.

ISSN 2317-1316

2 MATERIAIS E MÉTODOS

Neste trabalho foi adotada uma abordagem experimental, iniciando com um estudo sobre trabalhos acadêmicos, documentos e notícias relacionadas ao processo de inicialização de *endpoints*, UEFI, Secure Boot e PXE, buscando a compreensão de conceitos e tecnologias em uso para a instalação de sistemas operacionais por meio do boot. Em seguida, experimentos foram definidos para testar, validar e aplicar uma solução em um laboratório acadêmico de pesquisa, o Laboratório de Estudos de Ambiente de Produção de Software (MackLEAPS) da Universidade Presbiteriana Mackenzie (UPM), no âmbito do Projeto Campus Living Lab.

2.1 PXE boot

O PXE boot é um protocolo criado pela Intel, que visa prover serviços em ambientes pré-*boot*, para que computadores possam obter programas e imagens de sistemas operacionais antes do *boot* de dispositivos locais, utilizando uma rede em comum (Microsoft, 2025). Neste processo, são utilizados o protocolo *Dynamic Host Configuration Protocol* (DHCP) e *Trivial File Transfer Protocol* (TFTP) em um servidor intitulado de *PXE server*, além de uma configuração no ambiente UEFI do *endpoint*, a qual permite a network *booting* (Intel Corporation, 1999).

O processo começa com o *endpoint*, também denominando *PXE client*, iniciando o protocolo ao enviar um *broadcast* de *DHCP Discover* para encontrar um servidor *DHCP* na rede. Nessa mensagem, além de solicitar informações relativas à configuração de rede, o *endpoint* informa que é um cliente *PXE*. Ao receber a mensagem o servidor de DHCP envia ao *endpoint* as configurações de rede, assim como uma lista de servidores apropriados para o boot disponíveis. A partir dessas informações, o *endpoint* identifica o servidor de boot e utiliza o protocolo TFTP para obter os arquivos necessários à inicialização pela rede. Ao final do carregamento, o *endpoint* requisita a um servidor web a imagem do sistema operacional, bem como outros arquivos, dando início ao processo de instalação do mesmo (Intel Corporation, 1999).

Em plataformas com UEFI, a configuração responsável por habilitar o PXE *boot* no *endpoint* está presente na *network stack* do *firmware* e o programa que fornece as instruções para a inicialização é o *network bootstrap program* (NBP), informado pelo servidor de DCHP ao cliente. Em ambientes com Windows, o arquivo fornecido pelo NBP é o EFI (Extensible Firmware

ISSN 2317-1316

Interface) “wdsmgfw.efi”, enquanto em ambientes Linux ele é disponibilizado por *bootloaders* compatíveis com UEFI, como o GRUB (Ruth e Richardson, 2014).

2.2 UEFI

O UEFI é uma especificação que define como o *firmware* deve interagir com o sistema operacional. Esse padrão determina a interface do *firmware* em computadores de diferentes fabricantes, portanto há vários tipos de implementações, porém todas seguem esse mesmo padrão estabelecido pelo UEFI Forum (Wilkins et al, 2023).

A especificação UEFI foi proposta pela Intel, com o objetivo de substituir o *software* de *boot* da Legacy BIOS, pois novos processadores mais potentes estavam surgindo e o modelo de arquitetura estava se modernizando (Wilkins et al, 2023). Atualmente o modelo UEFI está presente na maioria dos computadores com CPUs Intel e fabricantes de processadores ARM estão também migrando para essa mudança (Matrosov et al, 2019).

2.3 UEFI Secure Boot

O Secure Boot é responsável por garantir que os programas executados durante o *boot* sejam autenticados, fazendo com que um agente malicioso não consiga trocar programas legítimos e autenticados dessa cadeia de inicialização por programas não autenticados (Hagl, 2021).

Seu objetivo é prevenir qualquer execução de código não autorizado em ambientes de pré-boot. O uso de tal mecanismo é frequente em sistemas embarcados, aparelhos mobile e computadores com UEFI. Há três principais formas de implementação do Secure Boot: Secure Boot no sistema operacional, definido por ser implementado no nível do bootloader; UEFI Secure Boot, o qual é implementado no firmware do UEFI; e por fim há o Platform Secure Boot, caracterizado por ser junto ao hardware (Matrosov, 2019). O estudo concentra-se na implementação do UEFI Secure Boot.

Para criar um padrão de confiança no binário UEFI, faz-se necessário um mecanismo para conferir se o binário foi assinado com uma chave autorizada, pois um agente malicioso poderia criar uma chave customizada para assinar um binário malicioso e passar na validação do Secure

ISSN 2317-1316

Boot. Devido a isso, uma chave pública que é utilizada para a validação da assinatura, deve ser verificada com uma chave privada confiável (Matrosov, 2019).

O banco de dados "db" contém uma lista de certificados de chaves públicas confiáveis e autorizadas para autenticar as assinaturas. Além desses certificados, o "db" também contém *hashes* de binários autorizados a execução independentemente da assinatura (Matrosov, 2019).

Outro banco de dados de importância para o mecanismo é o "dbx", denominado de Revoked Signature Database, que armazena certificados de chaves públicas e *hashes* de binários de UEFI proibidos de serem executados no processo de *boot*, por possuirem vulnerabilidades conhecidas. Assim, no processo de autenticar a assinatura de um executável UEFI, ele verifica se a assinatura é válida, caso ela não seja e não possui uma *hash* no "db", ele bloqueia a execução. Se o executável possuir uma assinatura válida, não possuir um *hash* no "dbx" e possuir a identificação de seu autor no "db", ele poderá ser executado, contudo, se ele possuir uma *hash* no "dbx", ele será bloqueado (Matrosov, 2019).

Os certificados dos bancos de dados "db" e "dbx" são assinados por uma autoridade central denominada de Certificate Authority (CA), chefiada pela Microsoft, a qual permite que empresas enviem seus binários para a Microsoft CA UEFI assinar e incluir um certificado referente ao binário no "db" (Wilkins e Richardson, 2019).

Foram coletadas informações acerca do uso, da configuração e comportamento do Secure Boot, em sites como Debian wiki, Ubuntu wiki e Learn Microsoft. Devido ao funcionamento diferente do Secure Boot em cada distribuição do Linux, os *sites* de documentação dessas distribuições, foram consultadas para uma descrição mais detalhada (Matrosov et al, 2019).

2.4 Seleção das distribuições para uso nos *endpoints*

Durante a pesquisa sobre as distribuições Linux que seriam utilizadas, optou-se por aquelas que fossem *open-source* onde o Ubuntu e Debian destacaram-se. Portanto, ambas tiveram o funcionamento, configuração e comportamento do Secure Boot estudados.

O Ubuntu foi escolhido devido à sua grande popularidade, versatilidade, facilidade de uso e customização como distribuição Linux em *endpoints* de usuários, sendo amplamente adotado por

ISSN 2317-1316

usuários com diferentes níveis de conhecimento em Linux. O Debian foi escolhido para o estudo por sua ampla adoção e estabilidade em infraestruturas de servidores, bem como sua ampla utilização em ambientes de desenvolvimento, providenciando assim um ecossistema ideal para simulações (Legay et al, 2021).

2.5 Secure Boot no Ubuntu e Debian

Para a integração do Secure Boot com o Linux, foi criado o carregador UEFI "shim", que possui a função de atuar como intermediário entre o UEFI e o sistema operacional. O pacote "shim" é um binário EFI, programado pelo time da Red Hat, que abre e executa outra aplicação utilizando chamadas padrões do EFI, que carregam a imagem e a iniciam (Red Hat Bootloader Team, 2025). Ele é assinado pela Microsoft e possui chaves para assinatura do Debian e Ubuntu, permitindo assim a comunidade Linux assinar os próprios binários sem a necessidade da autenticação por parte da Microsoft. Portanto, os autenticadores de certificados como a Microsoft só precisam assinar o shim e não todos os outros programas das distribuições, facilitando assim o uso do Secure Boot em plataformas *open-source* (Debian Project, 2025).

A funcionalidade que desempenha o papel de permitir que usuários assinem suas próprias chaves é o *Machine Owner Key* (*MOK*), que também permite o gerenciamento de chaves de outros programas da distribuição do Linux. Com o *MOK* e "mokutil", o comando para gerenciar as chaves, é possível assinar os próprios binários customizados e adicioná-los ao banco de dados com certificados de programas autorizados. Porém, só se pode confirmar as chaves do *MOK* no período de *boot*, pois assim previne-se que um possível *malware* no nível de usuário assine seu programa malicioso e por sequência seja autorizado no Secure Boot (Debian Project, 2025).

Outro serviço que visa integrar a segurança do *firmware* com o Linux é o Linux Vendor Firmware Service (LVFS). Trata-se de um serviço *web* que pode ser usado por fabricantes de *hardware* para publicar atualizações e versões diferentes de *firmware*, permitindo que os usuários possam realizar a transferência e a instalação dessas atualizações de forma segura. Assim, junto ao *daemon* "fwupd", os usuários podem realizar tais ações, tanto utilizando interfaces gráficas, como usando o próprio terminal e comandos como "fwupd" e "fwupdmgr" (Hughes, 2016).

2.6 Especificação e montagem de ambiente para realização dos testes

ISSN 2317-1316

Para a simulação do ambiente UEFI com Secure Boot foi utilizado o virtualizador e emulador de máquina QEMU (*Quick Emulator*), o qual passou a ser usado como um emulador do ambiente de *boot*. Junto ao QEMU, utilizou-se o Tianocore EFI Development Kit II (EDK II), que desempenha o papel de ambiente de desenvolvimento e implementação do UEFI. Adicionalmente, o Open Virtual Machine Firmware (OVMF) foi aplicado, sendo o emulador do *firmware* UEFI.

Ademais, utilizou-se um ambiente real para a simulação de instalação com o PXE *boot* e Secure Boot devidamente configurado. Para tal, empregou-se um servidor previamente configurado no MackLeaps, bem como um computador habilitado para inicialização via rede e com as configurações do Secure Boot atualizadas.

No ambiente simulado de UEFI com Secure Boot anteriormente proposto, foi programado um binário do UEFI para servir de prova de conceito sobre o mecanismo estudado, utilizando a documentação e padrão UEFI. Durante o processo, o UEFI foi configurado para permitir a execução de um aplicativo que não faz parte do banco de dados de certificados oficialmente assinados, para isso, foram geradas chaves para autenticar o binário e garantir a sua execução.

O aplicativo desenvolvido, inicialmente realiza a solicitação de um arquivo definido no código-fonte, ao PXE *server*, utilizando um protocolo PXE da própria especificação UEFI. Após o recebimento, o arquivo é armazenado localmente, demonstrando de forma prática, a viabilidade de execução de um aplicativo EFI assinado em um ambiente PXE com Secure Boot habilitado.

3 RESULTADOS E DISCUSSÃO

3.1 PXE *boot* e UEFI Secure Boot

3.1.1 Ambiente Virtual

Para a simulação em ambiente virtualizado, configurou-se a máquina *host* para atuar como PXE *server* e uma máquina virtual utilizando o *hipervisor* QEMU e *firmware* OVMF, com suporte ao UEFI Secure Boot, para implementar o PXE *client*.

No PXE *server* foi empregado o “dnsmasq” para desempenhar o papel de DHCP e TFTP, fornecendo os arquivos “grubx64.efi”, correspondente ao *bootloader* GRUB devidamente assinado

ISSN 2317-1316

e autenticado, e o "shimx64.efi", utilizado para viabilizar o processo do Secure Boot. Além disso, foram disponibilizados os arquivos "vmlinuz" e o "initrd", que contém, respectivamente, o *kernel* do Linux compactado e o *initial ramdisk*. Complementarmente, um *web server* HTTP nginx foi utilizado para fornecer o arquivo "filesystem.squashfs", responsável por providenciar a estrutura dos arquivos.

Nesta configuração, foram realizados testes com o shim da RedHat em máquina virtual para evitar a possibilidade de inutilização do dispositivo (*bricking*).

Um aplicativo UEFI em C foi criado para ser assinado e importado no banco de dados db, utilizando as ferramentas do "mokutil", "sbsign", "efitools", "openssl" e "keyutils" (Red Hat Inc., 2025). A principal funcionalidade desta aplicação é efetuar uma transferência de um arquivo através do TFTP, utilizando protocolo PXE Base Code, um dos serviços de rede fornecidos pelo *firmware* no processo pré-*boot* (UEFI Forum, 2022).

Antes do processo de assinatura do binário, foi preciso criar o ambiente hierárquico de *chain of trust* do Secure Boot na máquina virtual do QEMU com OVMF, em que foram geradas as chaves PK, KEK e o banco de dados db (Matrosov et al, 2019). Assim, a chave PK autenticou a chave KEK que por consequência autenticou o banco de dados db, conforme a geração de chaves e certificados, conversão para o formato *EFI Signature List* (ESL), autenticação e posterior processo de cadastro dos três elementos no *firmware* do EDK II. Ao final, o binário UEFI feito para testes foi assinado utilizando o comando "sbsign", que usou a chave privada do db para estabelecer a autenticidade do binário (Red Hat Inc., 2025).

Inicialmente o binário é enviado via TFTP pelo "dnsmasq" do PXE *server* em seguida ele é executado no PXE *client*. No começo da execução, ele localiza o protocolo PXE Base Code, o qual abstrai os protocolos DHCP, ARP e TFTP em um única API padronizada dos módulos de *firmware* (UEFI Forum, 2022). Após isso, a operação de leitura do *multicast* TFTP é inicializada e posteriormente há uma alocação de memória para o *download*. Ao final, é feita a transferência do arquivo, o qual nos nossos testes foi especificado como o pacote shim.

Os testes de inicialização e instalação do Linux a partir da rede com o Secure Boot habilitado foram bem-sucedidos, e a possibilidade de assinar pacotes de software de forma independente foi

ISSN 2317-1316

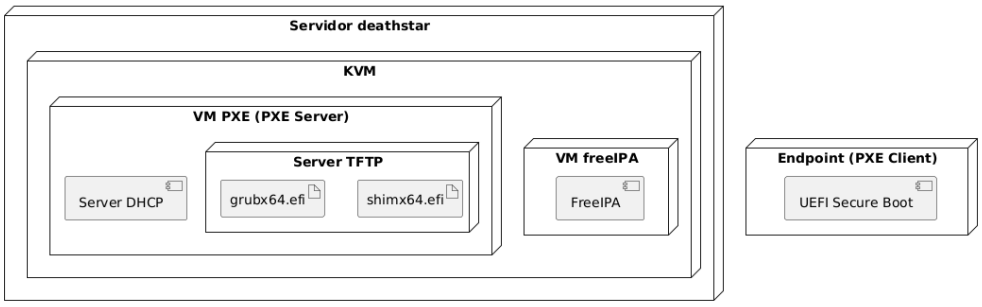
verificada experimentalmente, esta autonomia ajuda a superar as limitações do processo centralizado de gerenciamento de chaves sob o cuidado da Microsoft. Os resultados positivos permitiram que testes no ambiente real pudessem ser conduzidos de forma segura.

3.1.2 Ambiente Real

Na implementação em ambiente real, usufruiu-se de um ambiente de PXE elaborado no MackLeaps, do qual possuí um servidor denominado de "deathstar", utilizando o Kernel-based Virtual Machine (KVM), contendo duas máquinas virtuais, uma para agir como PXE server e outra com o software de gerenciamento de identidade FreeIPA, como apresentado na Figura 1.

De modo análogo ao procedimento feito no ambiente virtual, empregou-se um servidor TFTP, também gerenciado pelo "dnsmasq", para o fornecimento do arquivo "shimx64.efi" ao *endpoint*, o qual corresponde a um computador com UEFI Secure Boot e com a opção de *boot* por rede habilitados. Em sequência, o "shimx64.efi" solicita o *bootloader* assinado "grubx64.efi", igualmente fornecido pelo servidor TFTP, e posteriormente uma imagem é fornecida por intermédio de um servidor HTTP nginx.

Figura 1: Diagrama do ambiente real com PXE boot – Fonte: autor.



A solução implementada funcionou no ambiente virtual e está em uso no MackLEAPS.

3.2 Desvantagens e limites impostos pelo Secure Boot

Em contraste às proteções oferecidas pelo Secure Boot, há limitações impostas afetam o sistema Linux. Uma delas é a ativação do modo *kernel lockdown*, em que uma função bloqueia o acesso a uma imagem do *kernel* em execução, para proteger modificações não autorizadas da sua imagem e prevenir acesso a informações criptografadas da sua memória (Kerrisk, 2024).

ISSN 2317-1316

Como consequência do *kernel lockdown*, certas características do próprio *kernel* são desabilitadas, como a manipulação de arquivos e serviços que afetam diretamente a imagem do *kernel* e a customização de métodos ACPI (Kerrisk, 2024).

Um problema relacionado aos certificados oficiais do Secure Boot é a lentidão das autoridades responsáveis pela assinatura dos binários, com destaque à avaliação e resposta às submissões de *softwares.* Ademais, pode-se citar que existe a possibilidade de tais autoridades concederem assinaturas e certificados a *bootkits* disfarçados ou a binários que contenham vulnerabilidades conhecidas. Esse tipo de falha poderia ser negligenciada ou não identificada, pelos processos de avaliação dessas entidades, comprometendo a segurança do ecossistema (National Security Agency, 2019).

Ao utilizar o MOK e mokutil para assinar binários no Secure Boot, surge também uma desvantagem por parte dos administradores de sistemas, a qual é a constante necessidade de criação, manutenção e atualização de *scripts*, com o propósito de verificar e assinar novos executáveis fornecidos tanto por novas atualizações dos fabricantes e vendedores do *hardware* como também por binários customizados pela empresa ou instituição, para assim manter a segurança e funcionalidade dos *endpoints* (National Security Agency, 2019).

4 CONCLUSÃO

Ao estudar o funcionamento do Secure Boot e sua integração com o Linux e *softwares* Open-Source, foi possível extrair especificações técnicas, casos e fatos que contribuíram para conclusão do estudo. A análise permitiu esclarecer o contexto prático da aplicação dessas tecnologias, utilizando o PXE *boot* em conjunto ao UEFI Secure Boot para assim, elaborar um ambiente seguro, integro e automatizado para o uso de *endpoints* com Linux em um laboratório acadêmico de pesquisa. O uso do SHIM, proposto pela Redhat foi testado e validado, dando maior autonomia para a operação não depender da atualização de chaves UEFI por parte da Microsoft, permitindo concluir que o Secure Boot pode ser integrado junto a ambientes com Linux sem impedimentos operacionais. É importante ressaltar que este trabalho teve um escopo limitado, sendo aplicado a

ISSN 2317-1316

um conjunto pequeno de dispositivos de hardware e limitado a duas distribuições do Linux, Ubuntu e Debian.

5 DECLARAÇÃO DE USO DE IA

Não foram utilizadas ferramentas de IA no desenvolvimento desse artigo.

6 REFERÊNCIAS

CERT Coordination Center. Howyar Reloader UEFI bootloader vulnerable to unsigned software execution. 2025. Disponível em: <https://www.kb.cert.org/vuls/id/529659>.

Debian Project. Secure Boot – Debian Wiki. 2025. Disponível em: <https://wiki.debian.org/SecureBoot#What_is_UEFI_Secure_Boot>.

Dorner C., Klausner L. D. If It Looks Like a Rootkit and Deceives Like a Rootkit: A Critical Examination of Kernel-Level Anti-Cheat Systems. In: Proc. of the 19th Int. Conf. on Availability, Reliability and Security. Vienna Austria: ACM, 2024. p. 1–11. Disponível em: <https://dl.acm.org/doi/10.1145/3664476.3670433>.

Hagl J., Mann O., Pirker M. Securing the Linux Boot Process: From Start to Finish: In: Proc. of the 7th Int. Conference on Information Systems Security and Privacy. Sankt Pölten, Áustria: SCITEPRESS – Science and Technology Pub. , 2021. p. 604–610. Disponível em: <https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010313906040610>.

Hughes R. LVFS — Developers. 2015. Disponível em: <https://fwupd.org/lvfs/docs/developers>.

Hughes R. Linux Vendor Firmware Service - Introduction. 2016. Disponível em: <https://lvfs.readthedocs.io/en/latest/intro.html>.

Intel Corporation. Preboot Execution Environment (PXE) Specification – Version 2.1. [S.l.], 1999. 10–15 p. Disponível em: <https://web.archive.org/web/20131102003141/http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf>.

ISSN 2317-1316

Kerrisk M. kernel_lockdown(7). 2024. Disponível em: <https://man7.org/linux/man-pages/man7/kernel_lockdown.7.html>.

Legay D., Decan A., Mens T. "A Quantitative Assessment of Package Freshness in Linux Distributions," 2021 IEEE/ACM 4th Int. Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal), Madrid, Spain, 2021, pp. 9-16, doi: 10.1109/SoHeal52568.2021.00008.

Matrosov A., Rodionov E., Bratus S. Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats. San Francisco, CA: No Starch Press, 2019. ISBN 9781593277161 9781593278830.

Microsoft. Understand PXE boot in Configuration Manager. Microsoft, 2025. Disponível em: <https://learn.microsoft.com/en-us/troubleshoot/mem/configmgr/os-deployment/understand-pxe-boot>.

National Security Agency. CSI – Boot Security Modes and Recommendations. [s.l.], 2019. 1-4 p. Disponível em: <https://www.nsa.gov/portals/75/documents/what-we-do/cybersecurity-professional-resources/csi-boot-security-modes-and-recommendations.pdf>.

Red Hat Bootloader Team. shim – README. 2025. Disponível em: <https://github.com/rhboot/shim/blob/main/README.md>.

Red Hat Inc. Signing a kernel and modules for Secure Boot – Chapter 3. [S.l.], 2025. Disponível em: <https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/signing-a-kernel-and-modules-for-secure-boot_managing-monitoring-and-updating-the-kernel>.

Ruth L., Richardson B. UEFI PXE Boot Performance Analysis. [S.l.], 2014. 1–4 p. Disponível em: <https://web.archive.org/web/20140808044632/https://uefidk.com/sites/defaultfiles/Intel_UEFI_PXE_Boot_Performance_Analysis.pdf>.

Stack Overflow. Developer Survey 2022. 2022. Disponível em: <https://survey.stackoverflow.co/2022/#section-most-popular-technologies-operating-system>.

Statcounter Global Stats. Desktop Operating System Market Share Worldwide. 2025. Disponível em: <https://gs.statcounter.com/os-market-share/desktop/worldwide/>.

ISSN 2317-1316

UEFI Forum. Network Protocols — SNP, PXE, BIS and HTTP Boot: Section 24.3 PXE Base Code Protocol. [S.l.], 2022. Disponível em: <https://uefi.org/specs/UEFI/2.10/24_Network_Protocols_SNP_PXE_BIS.html#pxe-base-code-protocol>.

UEFI Forum. UEFI Specification 2.10 - Introduction. 2022. Disponível em: <https://uefi.org/specs/UEFI/2.10/01_Introduction.html>.

Wilkins R., Mullen B., Lewis T., Wei D., Keown W., Zimmer V. Decoding UEFI Firmware: Unraveling the Intricacies of System Firmware, its Ecosystem and Supply Chain. [S.l.], 2023. 1–12 p. Disponível em: <https://uefi.org/sites/default/files/resources/What%20is%20UEFI-Aug31-2023-Final.pdf>.

Wilkins R., Richardson B. UEFI Secure Boot in Modern Computer Security Solutions. [S.l.], 2019. 1–10 p. Disponível em: <https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2019.pdf>.