UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Ramon Sousa Sarmento

MONITORAMENTO CONTÍNUO DE INTEGRIDADE PARA CONTÊINERES EM AMBIENTES LINUX

CAMPINA GRANDE - PB

2023

Ramon Sousa Sarmento

MONITORAMENTO CONTÍNUO DE INTEGRIDADE PARA CONTÊINERES EM AMBIENTES LINUX

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Reinaldo Cezar De Morais Gomes

CAMPINA GRANDE - PB

2023

Ramon Sousa Sarmento

Monitoramento contínuo de integridade para contêineres em ambientes Linux

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Reinaldo Cezar De Morais Gomes

Orientador – UASC/CEEI/UFCG

José Antão Beltrão Moura

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de JUNHO de 2023.

CAMPINA GRANDE - PB

RESUMO

Neste trabalho será discutido o uso crescente de containers na virtualização de ambientes e implantação de aplicações, enfatizando a importância de garantir a integridade e segurança destes containers. O trabalho apresenta uma solução que usa a arquitetura de medição de integridade do kernel do Linux e o chip Trusted Platform Module para avaliar a integridade de aplicativos e bibliotecas. O ímpeto desta pesquisa está na necessidade de proteger as aplicações em ambientes virtualizados e conteinerizados, garantindo sua confiabilidade e segurança.

A solução proposta neste estudo envolve um modelo cliente-servidor, no qual os containers são executados no cliente. Uma aplicação foi desenvolvida para monitorar continuamente a integridade desses contêineres. Quando um erro de integridade no contêiner é identificado, o servidor é notificado imediatamente. O servidor, munido dessas informações, decide a ação apropriada a ser tomada em resposta ao erro identificado. Este processo permite uma resposta rápida e eficaz a possíveis violações de integridade, aumentando assim a segurança geral do sistema.

Continuous Integrity Monitoring for Containers in Linux Environments

ABSTRACT

In this work, the increasing use of containers for virtualizing environments and deploying applications will be discussed, emphasizing the importance of ensuring the integrity and security of these containers. The work presents a solution that uses the integrity measurement architecture of the Linux kernel and the Trusted Platform Module chip to assess the integrity of applications and libraries. The motivation for this research lies in the need to protect applications in virtualized and containerized environments, ensuring their reliability and security.

The proposed solution in this study involves a client-server model, where the containers are executed on the client. An application has been developed to continuously monitor the integrity of these containers. When an integrity error in a container is identified, the server is immediately notified. Equipped with this information, the server decides on the appropriate action to be taken in response to the identified error. This process enables a quick and effective response to potential integrity violations, thus enhancing the overall system security.

Monitoramento contínuo de integridade para contêineres em ambientes Linux

Ramon Sousa Sarmento
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
ramon.sarmento@ccc.ufcg.edu.br

Orientador: Reinaldo Cezar De Morais Gomes
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
reinaldo@computação.ufcg.edu.br

Resumo

Neste trabalho será discutido o uso crescente de containers na virtualização de ambientes e implantação de aplicações, enfatizando a importância de garantir a integridade e segurança destes containers. O trabalho apresenta uma solução que usa a arquitetura de medição de integridade do kernel do Linux e o chip Trusted Platform Module para avaliar a integridade de aplicativos e bibliotecas. O ímpeto desta pesquisa está na necessidade de proteger as aplicações em ambientes virtualizados e conteinerizados, garantindo sua confiabilidade e segurança.

A solução proposta neste estudo envolve um modelo cliente-servidor, no qual os containers são executados no cliente. Uma aplicação foi desenvolvida para monitorar continuamente a integridade desses contêineres. Quando um erro de integridade no contêiner é identificado, o servidor é notificado imediatamente. O servidor, munido dessas informações, decide a ação apropriada a ser tomada em resposta ao erro identificado. Este processo permite uma resposta rápida e eficaz a possíveis violações de integridade, aumentando assim a segurança geral do sistema.

1. Introdução

Nos últimos anos, o uso de containers tem se popularizado entre desenvolvedores e empresas, pois proporcionam flexibilidade e facilidade em um ambiente virtualizado para desenvolvimento e implantação de aplicações, além de consumir menos recursos e reduzir os custos de hospedagem em nuvem.

No entanto, a natureza dinâmica dos contêineres e o fato de serem baseados em imagens baixadas de repositórios online podem levar a riscos de segurança, como modificação maliciosa do conteúdo de imagens ou aplicativos em uso, comprometendo a integridade e a confiabilidade do contêiner. Nesse contexto, é fundamental adotar soluções eficazes para monitorar a integridade dos contêineres e garantir a proteção de dados e aplicativos.

O kernel do Linux fornece uma solução para verificar a integridade dos aplicativos por meio do sistema Integrity Measurement Architecture (IMA). O IMA coleta hashes de binários, scripts executados e bibliotecas necessárias, armazenando-os em uma área segura da memória do kernel onde mesmo um usuário com privilégios de root não pode modificá-los ou excluí-los. Além disso, os hashes são incluídos no chip físico Trusted Platform Module (TPM), permitindo a verificação remota ou local da validade dos dados.

Este trabalho apresenta uma abordagem para monitoramento contínuo da integridade de contêineres em um ambiente Linux, explorando o uso de IMA e TPM para garantir a segurança e confiabilidade das aplicações executadas em contêiners.

2. Motivação

A crescente adoção de contêineres no desenvolvimento e implantação de aplicativos trouxe muitos benefícios em termos de flexibilidade, escalabilidade e economia de recursos. No entanto, essa tendência também gera preocupações significativas sobre a segurança e a integridade dos ambientes virtualizados.

A motivação deste trabalho se assenta na necessidade de garantir a confiabilidade do container e proteger as aplicações ali hospedadas. Ataques maliciosos, modificações não autorizadas e vulnerabilidades nos códigos e nas bibliotecas podem comprometer a integridade e a segurança do contêiner, afetando negativamente o desempenho e a confiabilidade do aplicativo.

Além disso, a complexidade dos sistemas de contêineres e a variedade de tecnologias envolvidas tornam a tarefa de garantir segurança mais desafiadora. Portanto, é importante desenvolver soluções eficazes para detectar e prevenir possíveis ameaças, garantindo a integridade dos contêineres e dos aplicativos nos quais eles são executados.

Atualmente não existem opções para realizar um monitoramento contínuo da integridade de containers, as soluções atuais se baseiam em uma verificação periódica, isto abre espaço para que uma aplicação considerada falha rodar livremente durante o longo intervalo de tempo entre uma verificação e outra. Além disso, há poucas opções de configuração sobre quais atitudes tomar quando uma aplicação falha for detectada.

Este trabalho visa mitigar essa deficiência, propondo uma solução de monitoramento contínuo. Enfrentar esse desafio ajudará a criar um ambiente de contêiner mais seguro e confiável, permitindo que desenvolvedores e organizações colham os benefícios dos contêineres, garantindo a segurança de seus dados, aplicativos e serviços.

3. Embasamento teórico / Trabalhos relacionados

Neste capítulo, são apresentados os conceitos básicos e trabalho relacionados que dão suporte a este trabalho, elas serão necessárias para entender como funciona e onde surge o problema a ser corrigido. A revisão deste documento ajuda a entender o contexto e as abordagens existentes na área, fornecendo uma base para a proposta e desenvolvimento da solução apresentada neste trabalho.

3.1 Integridade

A integridade representa um dos pilares fundamentais da segurança da informação, ao lado da confidencialidade e disponibilidade. Esse recurso refere-se à proteção de dados e aplicativos contra modificações não autorizadas, garantindo a consistência e precisão das informações ao longo do tempo.

Manter a integridade é essencial para garantir a confiabilidade e a segurança dos aplicativos executados em qualquer ambiente de computação. Violações de integridade podem levar a consequências graves, como vazamento de informações confidenciais e execução de códigos maliciosos.

Para garantir a integridade dos dados, é essencial ter fortes medidas de segurança. Isso pode incluir o monitoramento contínuo das atividades, aplicando políticas de acesso rígidas, validando os recursos de software usados, bem como reforçando o gerenciamento eficaz e desenvolvendo práticas de governança. Essas estratégias formam o núcleo de qualquer solução eficaz de segurança da informação.

3.2 Trusted Platform Module

O Trusted Platform Module (TPM) é um chip de hardware de criptografia incorporado em muitos dispositivos que fornece a capacidade de armazenar chaves com segurança e executar operações criptográficas, e também coleta medidas sobre os componentes físicos da máquina e do próprio kernel do sistema operacional. Sua principal função é estabelecer uma plataforma confiável para ancorar medidas de integridade, garantindo que os dados coletados sejam protegidos contra a adulteração. Além disso, o TPM permite a verificação remota ou local da validade dos dados que ele mantém, permitindo verificar a integridade dos aplicativos e sistemas em execução.

Portanto, o TPM desempenha um papel na melhoria da confiabilidade e segurança do ambiente de computação. A adoção do TPM, juntamente com abordagens de software e outras tecnologias de segurança, pode melhorar as soluções de proteção de dados e aplicativos em diferentes cenários e infraestruturas.

3.3 Integrity Measurement Architecture

A Integrity Measurement Architecture (IMA) é uma extensão do kernel Linux projetada para garantir a integridade de aplicativos em sistemas operacionais baseados em Linux. O IMA coleta hashes de binários, scripts executados e das bibliotecas que eles utilizam, fornecendo uma maneira eficiente de verificar a autenticidade e a integridade dos aplicativos em execução no sistema. Esses hashes coletados pelo IMA são armazenados em uma área segura da memória do kernel, protegendo-os de adulterações e acessos não autorizados.

Além disso, quando possível, o IMA utiliza o chip TPM para ancorar as medições de integridade, incluindo um nível adicional de confiabilidade e segurança ao processo. A combinação de IMA e TPM representa uma estratégia eficaz para proteger a integridade dos aplicativos em um ambiente Linux.

3.4 Container-IMA (Luo, Wu et al.)

A solução desenvolvida por Luo, Wu et al. [5], inclui uma extensão do IMA adaptada às características do contêiner, permitindo que seja realizada uma análise sobre a integridade de seus aplicativos. A extensão coleta e armazena métricas de integridade relacionadas a aplicativos e bibliotecas específicas para cada contêiner, se utilizando de soluções geradas pelo IMA, garantindo a relevância e precisão das informações obtidas.

A solução proposta também integra o TPM no processo de monitoramento de integridade, ancorando as métricas de integridade coletadas na memória segura do kernel e no chip TPM. Isso garante que as medições sejam protegidas contra modificações não autorizadas e possam ser verificadas remotamente ou localmente.

Luo, Wu et al. [5] abordou os desafios de aplicar o IMA em um ambiente de contêiner, criando uma solução capaz de lidar com as peculiaridades desse contexto. A extensão recomendada fornece uma maneira de garantir a integridade dos aplicativos executados em contêineres, fornecendo uma camada extra de segurança e confiabilidade para esses ambientes.

4. Projeto da Solução

Nesta seção, a solução proposta é descrita, focando nos componentes e como eles interagem entre si para garantir a confiabilidade e segurança dos containers no ambiente. A solução trata da integração de tecnologias com o container-IMA, além de abordar desafios específicos que surgem ao aplicá-las em um contexto de contêiner.

4.1 Arquitetura

A arquitetura do sistema proposto é projetada de forma modular e interconectada, permitindo a extensão e adaptação para diferentes ambientes para que a solução possa funcionar adequadamente em um cenário real, Nesta seção será discutido em mais detalhes o papel de cada componente.

4.1.1 IMA

O IMA é a base para toda a solução deste trabalho, pois é a fonte primária de informação de integridade, seu papel na solução é assegurar confiança que o kernel se encontra em um estado que inclui as modificações provenientes do Container-IMA, além de garantir a integridade das aplicações responsáveis pela gestão dos contêineres, e do próprio cliente desta solução.

4.1.2 TPM

Para que um atestador remoto possa de fato confiar nas informações fornecidas pelo IMA sobre o estado de integridade dos componentes e aplicações da máquina, é necessário confrontar estas informações com o valores registrados no TPM, assim, tendo certeza de que a máquina é confiável, garante-se que os dados e ações do cliente estão de acordo com o esperado.

4.1.3 Extensão do IMA para contêineres

Essa componente é referente a solução do conainer-IMA [5], ele é responsável por prover as informações sobre o hash de cada aplicação que está sendo executada em um container, em tempo real ele irá incluir em seus arquivos o que foi executado durante todo o ciclo de vida deste container.

4.1.4 Cliente

Esta é a aplicação que estará rodando na máquina onde os contêineres estão sendo executados, suas funções são.

4.1.4.1 Coletor

Continuamente o coletor irá observar as medições de integridade geradas pela extensão do IMA para contêineres, identificando quando uma nova aplicação foi executada, e enviando a informação sobre as novas adições para o analisador.

4.1.4.2 Analisador

Em posse dos dados enviados pelo coletor sobre a nova execução de uma aplicação dentro do container, o analisador irá confrontar seu valor de hash com uma lista de valores hash considerados confiáveis gerados previamente para a imagem base desse container, caso identifique alguma irregularidade como um valor de hash diferente da lista, ou que se trata de uma aplicação que não deveria estar presente neste container, o analisar irá repassar esta informação para o servidor deliberar sobre que ações serão tomadas.

4.1.4.3 Executor

Após receber do servidor a resposta sobre o que fazer com o container em relação à irregularidade encontrada, caso seja necessário, o executor é o componente no cliente que irá realizar a comunicação com o gerenciador de containers para cumprir a ação.

4.1.5 Servidor

O servidor é responsável por definir que ações tomar no caso de inconsistência em um container. ele irá se basear em ações previamente configuradas pelo usuário para solucionar cada tipo de irregularidade. Ele pode ser rodado remotamente para atender a múltiplos clientes, ou localmente visando atender o cliente que está rodando na máquina local.


```
{'containerID': 'fd79823cd4fea22fd91fa79b74d312a53d2a078893d7b23dc597eec2061795e4', 'notificate': 'ENTRY_NOT_FOUND', 'imageName': 'ubuntu:18.04', 'imageID': 'fd79823cd4fea22fd91fa79b7
4d312a53d2a078893d7b23dc597eec2061795e4', 'line': 'sha1:523252a830206329c80570ece9bcaf1d60574824 /lib/x86_64-linux-gnu/libdl-2.27.so\n'}
{'containerID': 'fd79823cd4fea22fd91fa79b74d312a53d2a078893d7b23dc597eec2061795e4', 'notificate': 'INVALID_HASH', 'imageName': 'ubuntu:18.04', 'imageID': 'fd79823cd4fea22fd91fa79b74d3
12a53d2a078893d7b23dc597eec2061795e4', 'line': 'sha1:46e93283ff53133360e02a73ae5b5ba375410855 /lib/x86_64-linux-gnu/libc-2.27.so\n'}
```


Figura 1


```
Action from server {"containerID":"fd79823cd4fea22fd91fa79b74d312a53d2a078893d7b23dc597eec2061795e4", "action":"REMOVE"}
Action from server {"containerID":"fd79823cd4fea22fd91fa79b74d312a53d2a078893d7b23dc597eec2061795e4", "action":"STOP"}
```


Figura 2

4.2 Comunicação

Para simplificar o processo de comunicação entre cliente e servidor em caso de inconsistências, é utilizado o protocolo gRPC, uma tecnologia de chamada de procedimento remoto (RPC) de alto desempenho desenvolvida pela Google, que, note-se, pode ser facilmente adaptada a outras alternativas. Pode ser necessário o uso de outras tecnologias de comunicação para atender a requisitos específicos, como compatibilidade com outras plataformas ou segurança adicional.

Na Figura 1 é mostrado um exemplo de mensagem gerado pelo cliente após a detecção de irregularidade na integridade de duas bibliotecas, a primeira biblioteca não foi encontrada na lista base, e a segunda apresentava um valor de hash diferente do valor presente na lista.

A Figura 2 se trata da resposta dada pelo servidor diante de cada notificação da Figura 1, ele indicou ações diferentes a serem executadas de acordo com o tipo de erro encontrado.

4.3 Configuração

A configuração da solução é feita via arquivo YAML, e permite a especificação de ações customizadas específicas para cada container ou imagem, bem como a configuração de opções padrão para cada erro.

Na Figura 3 é apresentado um exemplo do arquivo de configuração do servidor, neste caso as ações do servidor podem ser diferentes para dois containers com a mesma imagem, de acordo com o tipo de erro.

Figura 3

4.4 Gerador de valores base

Como parte desta pesquisa, foi desenvolvida uma aplicação para localizar todos os executáveis e bibliotecas contidas em um container. O objetivo deste aplicativo é criar uma lista com os valores base das aplicações em um determinado container ou imagem, que será utilizado pelo analisador do cliente para comparar com os valores das aplicações e bibliotecas executadas no container.

É necessário que o gerador de valores bases seja executado previamente em um estado em que se confia no container, se feito após o container estar em execução ou em um ambiente inseguro, pode acontecer dele já está com a aplicação maliciosa implantada no sistema e ela entrará na lista como um valor seguro.

5. Avaliação, com prova de conceito ou estudo de caso

Para avaliar a eficácia e a eficiência da solução proposta de monitoramento contínuo de integridade em contêineres em ambientes Linux, conduzimos uma prova de conceito e um estudo de caso, conforme descrito a seguir.

5.1 Prova de Conceito

A solução foi implementada em um ambiente controlado, empregando contêineres com aplicações e bibliotecas pré-selecionadas. Nesse contexto, verificou-se que a extensão do IMA para contêineres coletava as medições de integridade adequadamente e se a integração com o TPM ocorria conforme o esperado. Os resultados indicaram que a solução conseguiu identificar corretamente as aplicações e bibliotecas íntegras, além de detectar modificações maliciosas.

5.2 Estudo de Caso

O presente estudo de caso exemplifica um cenário em que um invasor malicioso foi capaz de modificar o executável de uma aplicação em execução dentro de um contêiner. Tal situação evidencia a necessidade de se monitorar continuamente a integridade dos contêineres, a fim de detectar e mitigar possíveis ameaças.

Suponha que um invasor malicioso tenha explorado uma vulnerabilidade na aplicação em execução dentro do contêiner para substituir o executável original por uma versão maliciosa. O invasor pode ter agido para obter acesso a dados confidenciais ou para executar atividades maliciosas dentro do ambiente do contêiner, sem a necessidade de ter acesso ao host do contêiner.

Ao utilizar uma solução de monitoramento contínuo de integridade, seria possível detectar a alteração do executável e tomar medidas para mitigar o ataque, como notificar um administrador ou desligar o container comprometido. Sem esse tipo de monitoramento, o invasor poderia executar suas atividades maliciosas sem ser detectado por um período indeterminado de tempo, o que poderia resultar em consequências graves para a segurança do sistema como um todo.

Em resumo, este estudo de caso destaca a importância de monitorar continuamente a integridade dos contêineres para garantir sua segurança e confiabilidade. A detecção precoce de ameaças pode permitir a implementação de medidas preventivas e a redução de possíveis danos causados por atividades maliciosas.

No caso descrito, a solução proposta foi capaz de detectar a alteração maliciosa do executável dentro do contêiner. Através do monitoramento contínuo das medições de integridade das aplicações em execução, a solução foi capaz de identificar a discrepância entre as medições originais e as atuais, alertando para a violação de integridade.

Ao detectar a violação, a solução foi capaz de bloquear o acesso do contêiner comprometido à rede, evitando que o invasor pudesse acessar ou comprometer outros recursos do ambiente. Além disso, a solução registrou a violação de integridade em um log, permitindo que os administradores do sistema identificassem e analisassem o incidente, tomando as medidas necessárias para recuperar a integridade do contêiner e evitar futuras violações.

Essa resposta rápida e efetiva demonstra a importância do monitoramento contínuo de integridade em ambientes virtualizados e contêineres, garantindo a segurança e a confiabilidade dos recursos e aplicações em execução. A solução proposta oferece uma base sólida para a proteção de ambientes virtualizados e contêineres contra ameaças externas e internas, contribuindo para a segurança e a eficiência do ambiente de TI.

6. Experiência e Lições Aprendidas

Durante o desenvolvimento do projeto, experiências foram acumuladas e importantes lições aprendidas, que são compartilhadas neste tópico.

6.1 Pesquisa

Durante a fase de pesquisa, um dos principais desafios foi a escassez de referências e literatura sobre o tema, o que exige maior esforço na busca de informações relevantes. Além disso, a complexidade técnica nas tecnologias envolvidas também geraram dificuldades, exigindo um estudo aprofundado de códigos e documentações sobre o tema.

6.2 Processo de desenvolvimento

A adoção do Scrum e sprints, auxiliou na organização e no andamento das atividades. Estabelecer metas claras e priorizar as tarefas a serem realizadas, bem como rever e ajustar o planejamento conforme as necessidades, foi crucial para a finalização do trabalho.

As experiências adquiridas durante o desenvolvimento do projeto e as lições aprendidas contribuíram para aprimorar habilidades e conhecimentos na área de segurança e de contêineres, além de auxiliar na identificação de oportunidades de melhoria e Implementações a serem enfrentadas no futuro.

7. Obstáculos

A solução proposta para monitoramento contínuo da integridade do contêiner em um ambiente Linux possui diversos obstáculos que precisam ser considerados antes da implantação. Um desses obstáculos é a necessidade de uma versão modificada do kernel atual.

Para implementar a solução, é necessário que esteja sendo utilizado uma versão específica do kernel do Linux que suporte a medição de integridade (IMA). Embora esse recurso seja comum em distribuições Linux modernas, a solução proposta requer uma versão modificada do kernel, com correções específicas para que o IMA funcione no contêiner.

Isso pode ser uma barreira para a implementação da solução, pois as modificações do kernel podem ser complexas e exigem conhecimento avançado de desenvolvimento de sistema operacional. Além disso, usar uma versão modificada do kernel pode não ser compatível com outros componentes do sistema e pode causar instabilidade ou incompatibilidade com outras soluções.

Outro obstáculo é o consumo de recursos computacionais que precisará ser alocado para que seja realizado esse monitoramento contínuo, o que pode comprometer a eficiência de aplicações rodando na mesma máquina, e aumentar mesmo que pouco o custo para manter o sistema.

8. Conclusão

O projeto responde à necessidade de garantir a integridade e segurança dos containers, uma vez que são amplamente utilizados para desenvolvimento e implantação de aplicativos. Durante este trabalho, tecnologias relacionadas como IMA, TPM e Containers foram estudadas e uma solução foi desenvolvida para integrar esses componentes para fornecer um sistema de monitoramento de integridade.

A solução proposta auxilia na verificação da integridade de aplicações rodando em containers, aumentando assim a confiabilidade e segurança destes ambientes. Durante a avaliação do projeto, verificou-se que a solução atendeu aos requisitos declarados e funcionou de forma eficaz. No entanto, também foram identificadas algumas limitações e oportunidades de melhoria. Os custos do sistema devido ao monitoramento contínuo e a necessidade de adaptação a ambientes com diferentes configurações de IMA e TPM são algumas das áreas que precisam de atenção em trabalhos futuros. Além disso, estender a solução para outros sistemas operacionais e ambientes virtualizados é uma forma promissora de ampliar o escopo do projeto.

Em suma, o trabalho apresentado neste documento contribuiu para o avanço do conhecimento na área de segurança e integridade de contêineres, fornecendo uma solução eficaz para monitorar a integridade de aplicações executadas nesses ambientes. As experiências e lições adquiridas durante o desenvolvimento do projeto servirão de base para futuras pesquisas e melhorias nesta área.

9. Referencias

1. Alyas, Tahir et al. “Container Performance and Vulnerability Management for Container Security Using Docker Engine.” *Security and Communication Networks* (2022): n. pag.
2. Benedictis, Marco de and Antonio Lioy. “Integrity verification of Docker containers for a lightweight cloud environment.” *Future Gener. Comput. Syst.* 97 (2019): 236-246.
3. Berger, Stefan et al. “vTPM: Virtualizing the Trusted Platform Module.” *USENIX Security Symposium* (2006).
4. Kabbe, Jon-Anders. “Security analysis of Docker containers in a production environment.”(2017).
5. Luo, Wu et al. “Container-IMA: A privacy-preserving Integrity Measurement Architecture for Containers.” *International Symposium on Recent Advances in Intrusion Detection* (2019).
6. Sailer, Reiner et al. “Design and Implementation of a TCG-based Integrity Measurement Architecture.” *USENIX Security Symposium* (2004).
7. Sultan, Sari et al. “Container Security: Issues, Challenges, and the Road Ahead.” *IEEE Access* 7 (2019): 52976-52996.
8. Trusted Computing Group. (2019). Trusted Platform Module (TPM) Summary. https://trustedcomputinggroup.org/work-groups/trusted-platform-module/ (acessado em: 10 Maio 2023)
9. *Trusted platform module - windows security*, *Windows Security | Microsoft Learn*. disponivel em: https://learn.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-top-node (acessado em: 15 Maio 2023).