Fortalecendo a Confiança em vTPMs: Mecanismo de Ancoragem Baseado em Integridade

Marcela Tassyany[1], **Ronaldo Medeiros**[1], **Ramon Sarmento**[1], **Reinaldo Gomes**[1]

[1]Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos, Campina Grande, Brasil

{marcela,ronaldomedeiros,ramon.sarmento}@lsd.ufcg.edu.br

{reinaldo}@computacao.ufcg.edu.br

Abstract. *This work introduces a new mechanism for secure anchoring of virtual TPMs. The proposed approach focuses on fully protecting the states of vTPMs, which collectively represent all essential resources and data of a vTPM. The central objective of the solution is to establish a robust hardware-based proof of integrity, thus facilitating the verification of a vTPM by a remote attester. We implemented a prototype of the solution and conducted evaluations to analyze the overhead imposed on the system. The results show that the proposed approach does not compromise vTPM performance and introduces a slight operational overhead of 4 percentage points for CPU and 0.05 percentage points for system memory.*

Resumo. *Este trabalho introduz um novo mecanismo para a ancoragem segura de TPMs virtuais. A abordagem proposta foca na proteção integral dos estados dos vTPMs, que coletivamente representam todos os recursos e dados essenciais de um vTPM. O objetivo central da solução é estabelecer uma prova robusta de integridade baseada em hardware, facilitando assim a verificação de um vTPM por um atestador remoto. Implementamos um protótipo da solução e conduzimos avaliações a fim de analisar a sobrecarga imposta ao sistema. Os resultados evidenciam que a abordagem proposta não compromete o desempenho do vTPM e introduz uma leve sobrecarga operacional de 4 pontos percentuais para CPU e 0,05 pontos percentuais para memória do sistema.*

1. Introdução

Com o avanço da Computação em Nuvem, a virtualização tornou-se ubíqua e, simultaneamente, emergiu como alvo significativo para invasores. Um dos princípios essenciais da segurança de software consiste na premissa de que uma aplicação pode ser alterada ou configurada exclusivamente por entidades autorizadas [Kucab et al. 2021]. Nesse cenário, a avaliação da integridade em máquinas virtuais é uma questão premente [Du et al. 2018].

Alguns métodos existentes para medir a integridade em sistemas de computação incorporam a tecnologia de Computação Confiável, seguindo uma cadeia de confiança estabelecida pelo TPM (*Trusted Platform Module*). O TPM, um co-processador criptográfico integrado à maioria dos servidores atuais, é reconhecido por sua

combinação de custo acessível e segurança. Esse chip possibilita a atestação remota através de assinaturas digitais dos *hashes* criptográficos de componentes de software [Arthur and Challener 2015]. Nesse contexto, atestação remota implica em afirmar que um software, firmware ou hardware é genuíno ou correto.

No entanto, os TPMs de hardware são reconhecidos por possuírem recursos limitados, como uma quantidade reduzida de NVRAM (*Non-Volatile Random Access Memory*), baixo desempenho de entrada e saída, e um número restrito de PCRs (*Platform Configuration Registers*) [Arthur and Challener 2015]. Portanto, em ambientes de Computação em Nuvem, nos quais várias máquinas virtuais (VMs) são frequentemente executadas em uma única máquina física e podem ser migradas entre essas máquinas, torna-se ineficiente e complexo utilizar um único TPM de hardware para oferecer serviços de segurança a todas as VMs [Perez et al. 2006, De Benedictis et al. 2024].

Com o intuito de abordar essa questão, surgiu a proposta do vTPM, um TPM virtual destinado a proporcionar uma raiz de confiança para máquinas virtuais. Ao invés de depender de uma NVRAM física, o vTPM utiliza uma abstração da NVRAM na forma de um arquivo para o armazenamento de chaves secretas. Isso possibilita, de maneira similar ao TPM físico, o armazenamento de credenciais, a execução de operações criptográficas e a medição da integridade [Perez et al. 2006].

Os vTPMs têm uma ampla aplicação em plataformas de nuvem comerciais, como VMware Cloud [VMware 2018], Google Cloud [Cloud 2023], Microsoft Azure Cloud [Microsoft 2023] e Amazon AWS [Amazon Web Services]. Eles desempenham um papel crucial ao oferecer serviços de raiz de confiança e segurança para máquinas virtuais [Wang et al. 2023]. Entretanto, por ser uma solução baseada em software, o vTPM não provê as mesmas garantias de segurança que o TPM de hardware.

Nesse sentido, a comunidade científica tem buscado propor soluções que visem fornecer uma maior segurança para os TPMs virtuais. A maioria desses trabalhos concentram-se na ancoragem do vTPM no TPM físico da plataforma [Perez et al. 2006, De Benedictis et al. 2024], na utilização de Autoridades Certificadoras (AC) independentes de hardware [Perez et al. 2006, Pecholt and Wessel 2022] e na utilização de TEEs (*Trusted Execution Environment*) [Wang et al. 2023, Sun et al. 2018]. A maior parte das propostas que objetivam fornecer maiores garantias de segurança ao vTPM, concentram-se sobretudo em aspectos de confidencialidade.

Neste trabalho é proposta uma nova abordagem para ancoragem segura de TPMs virtuais. A proposta fundamenta-se na proteção dos estados desses vTPMs, que, quando considerados em conjunto, abrangem a totalidade dos recursos e dados críticos de um vTPM. Nossa solução visa gerar uma evidência de integridade baseada em hardware, permitindo que um vTPM seja atestado por um verificador remoto. No melhor do nosso conhecimento, a utilização de uma abordagem com tais propósitos e características ainda não foi explorada. Além disso, os resultados sugerem que a solução sugerida não compromete o desempenho do sistema, gerando um acréscimo de, em média, 0,05 pontos percentuais para memória e 4 pontos percentuais para CPU.

O restante deste trabalho está organizado da seguinte forma: nas Seção 2 são apresentados os conceitos básicos, úteis para a compreensão da proposta. Na Seção 3 são discutidos alguns trabalhos relacionados. Na Seção 4 é descrito o mecanismo proposto.

Na Seção 5 são apresentados os principais resultados obtidos. Na Seção 6 o trabalho é finalizado com um resumo das principais conclusões e propostas de trabalhos futuros.

2. Verificação de Integridade com TPM

Incorporado nas placas-mãe de uma variedade de servidores comerciais, o chip TPM desempenha funções essenciais, incluindo criptografia, assinatura digital, cálculo de funções de hash e armazenamento seguro para pequenas quantidades de dados [Wang et al. 2023, Narayanan et al. 2023]. Para entender melhor a solução proposta, é crucial consolidar o entendimento de como a implementação do TPM contribui na construção de uma cadeia de confiança, permitindo a verificação da integridade e a produção de relatórios confiáveis sobre um ambiente de execução específico.

O TPM dispõe de 24 Registradores de Configuração de Plataforma (PCRs, do inglês *Platform Configuration Register*). Os PCRs são destinados à preservação de medições críticas, derivadas mediante a aplicação de algoritmos de *Hashing* a diversos tipos de conteúdo, que podem variar desde o binário de um executável até arquivos de configuração. É importante notar que cada medição não está diretamente associada a um PCR específico. Para incorporar uma medição, o TPM realiza uma operação de *hash* estendido sobre o valor atual do PCR correspondente. A título de exemplo, se o PCR 10 for estendido com uma medição M, então o novo valor do PCR 10 é calculado da seguinte forma: $PCR10_{novo} = hash(PCR10_{atual} + M)$ [Tassyany et al. 2021].

Em geral, sistemas operacionais que priorizam a verificação da integridade de seu ambiente de execução geram relatórios de medição, os quais são anexados ao TPM por meio da operação de *hash* estendido. No contexto dos sistemas Linux, é utilizado o IMA (*Integrity Measurement Architecture*). Nesse cenário, as medições do *firmware* e de todas as configurações de inicialização são estendidas nos PCRs 0-7 durante o processo de inicialização. Posteriormente, as medições das bibliotecas, módulos de Kernel, arquivos de configuração e aplicações executadas são estendidas no PCR 10.

Nesse sentido, o Kernel do Sistema Operacional, por meio do IMA, mantém um relatório com a medição de todos os binários executados e sumariza o histórico dessas medições no PCR 10. Dessa forma, torna-se viável utilizar essas informações para verificar se alguma aplicação foi indevidamente modificada por terceiros não autorizados ou se alguma aplicação maliciosa foi executada.

Durante uma atestação remota, após o processo de verificação de autenticidade do TPM [Arthur and Challener 2015], o atestador remoto recebe tanto o relatório de medições quanto o relatório dos PCRs. O relatório de PCRs é assinado por uma Chave de Atestação (*Attestation Key*), permitindo ao mesmo verificar a integridade desse relatório e a autenticidade do seu emissor. Ao confiar no relatório de PCRs, o atestador remoto pode utilizá-lo para verificar a integridade do relatório de medições.

Normalmente, os dados do relatório de medições são sumarizados e comparados ao PCR 10 (no caso do IMA). Se os valores correspondem, é possível estabelecer confiança no relatório de medições e avaliar a partir dele se alguma aplicação foi modificada indevidamente ou se alguma aplicação maliciosa foi executada.

3. Trabalhos Relacionados

O principal trabalho a fundamentar a ideia da virtualização do TPM foi o proposto por Perez et al. (2006), estabelecendo as bases e as principais vantagens e desafios a se trabalhar com essa abordagem. O trabalho foi pioneiro ao propor a ideia e apresentar uma arquitetura para criação de instâncias virtuais de TPM que se adaptam ao ciclo de vida dinâmico das máquinas virtuais e que podem se relacionar diretamente com TPM físico da plataforma onde essas VMs são hospedadas.

Perez et al. (2006) apresentam quatro abordagens distintas para estabelecer confiança em instâncias de vTPM. Duas dessas abordagens envolvem a ancoragem do vTPM ao TPM físico, através da conexão da Chave de Atestação ou da Chave de Endosso (EK) do vTPM com a Chave de Atestação do TPM físico. As outras duas alternativas sugeridas consistem na criação de uma autoridade certificadora independente de hardware para a Chave de Endosso ou no emprego de um co-processador seguro como substituto do TPM físico. Os trabalhos subsequentes, em sua maioria, concentraram-se em propor esquemas similares, utilizando abordagens variadas.

A proposta de Pecholt e Wessel (2022) apresenta um módulo chamado CoCoTPM, destinado a oferecer uma plataforma confiável para Computação Confidencial, disponibilizando funcionalidades do TPM para máquinas virtuais. Os autores advogam por um modelo de gerenciamento centralizado de instâncias vTPM. Dentro dessa estrutura, é considerado um *driver* nas VMs, denominado CoCoTPMdriver, que estabelece comunicação com o TPM em software (CoCoTPM), utilizando o Hipervisor como intermediário. A confiança no CoCoTPM é estabelecida por meio da implementação de um certificado autoassinado, que é utilizado para assinar a Chave de Endosso e desempenha o papel de uma sub-autoridade certificadora. Conjuntamente, o estado permanente do CoCoTPM é criptografado e armazenado no host que hospeda as VMs.

Benedictis et al. (2024) apresentam uma abordagem para vinculação do TPM físico ao vTPM. A solução concentra-se em utilizar o TPM físico para criptografar as chaves primárias, a criação de objetos filhos e as sementes primárias do vTPM. Os autores propõem um mecanismo destinado a resguardar o estado permanente quando a instância é interrompida, baseado na criação de uma chave primária no vTPM para sua criptografia.

O eTPM, conforme apresentado por Sun et al. (2018) , e sua evolução, o svTPM, proposto por Wang et al. (2023), oferecem métodos para proteger o estado persistente do vTPM através de mecanismos de selagem e criptografia fornecidos pelo *Intel Software Guard Extensions* (SGX). Ambas as soluções empregam o SGX para gerenciar as instâncias do vTPM. No contexto de estabelecimento de confiança, o eTPM utiliza o TPM físico para reforçar a confiabilidade da plataforma subjacente e sugere o emprego da CPU para verificar a inicialização do eTPM. Em contrapartida, o svTPM introduz um mecanismo baseado no esquema de atestação remota do SGX, conhecido como Intel DCAP.

Os trabalhos supramencionados propõem contribuições significativas. No entanto, podemos elencar alguns desafios. Em primeiro lugar, a utilização de certificados autoassinados não oferece o mesmo nível de segurança proporcionado pelo uso de uma AC confiável. No contexto do TPM físico, a EK é assinada por uma AC de confiança durante o processo de fabricação do dispositivo, o que resulta em garantias de segurança mais robustas [Arthur and Challener 2015]. Em segundo lugar, a utilização de criptografia não

impede alguns tipos de ataque de atualidade e integridade do estado permanente, como ataques de *Rollback* [Wang et al. 2023, Narayanan et al. 2023].

Em terceiro lugar, é importante considerar que, apesar das tecnologias de ambiente de execução confiável, como o SGX, oferecerem garantias de segurança mais robustas, existem algumas limitações de desempenho que precisam ser levadas em consideração. Um exemplo é a restrição de memória do SGX, que é de aproximadamente 93 MB. Isso limita o número de instâncias de TPM que podem ser operadas simultaneamente, conforme destacado por Wang et al. (2023). Embora a utilização da paginação EPC do SGX possa permitir um número maior de instâncias, isso pode resultar em uma sobrecarga adicional. Além disso, o uso do SGX em si pode reduzir o desempenho geral do sistema. Estudos realizados por Jha et al. (2022) indicam que a execução do SWTPM (Software TPM Emulator) [Berger 2024] dentro do SGX pode aumentar o tempo de execução em até cinco vezes.

A maior parte da literatura enfoca na proteção e ancoragem do vTPM, com especial atenção à segurança de suas chaves, abordagens que podem inclusive complementar a abordagem proposta neste artigo. No entanto, é crucial considerar a integridade dos estados do vTPM como um alicerce de importância fundamental. Ter a capacidade de monitorar e validar a confiabilidade de um vTPM a partir de sua integridade, respaldada por uma raiz de confiança estabelecida em hardware, confere uma robustez adicional em termos de segurança. No melhor do nosso conhecimento, não foram identificados estudos que investiguem essa perspectiva.

4. Mecanismo de Suporte a Ancoragem Segura de TPMs Virtuais

O vTPM é composto por dois estados distintos: o estado permanente e o estado volátil. O estado permanente retém informações específicas de um vTPM particular, como chaves persistentes e sementes que são armazenadas na NVRAM. Em contraste, o estado volátil abrange informações que são redefinidas ao reiniciar uma máquina, como os valores do PCR. Ambos os estados, quando combinados, formam o estado atual de um vTPM.

A solução proposta neste artigo parte da hipótese de que a confiança em um vTPM está intrinsecamente ligada à integridade dos seus estados permanente e volátil. A relação entre os principais componentes da solução pode ser visualizada na Figura 1. Nossa abordagem propõe a criação de uma cadeia de confiança, com sua raiz estabelecida no TPM de hardware.

Nesse sentido, nossa solução considera que cada máquina hospedeira poderá dispor de N VMs de Usuário, cada uma delas dispondo de um vTPM. Conjuntamente, consideramos a adoção de uma VM de Gerenciamento. A VM de Gerenciamento também dispõe de sua própria instância de vTPM. Sua principal função é coordenar a ancoragem dos vTPMs de usuário em seu próprio vTPM. Todos os vTPMs são instanciados a partir do Emulador de Software TPM. Para simplificação, denominaremos o vTPM da VM de Usuário como vTPMu e o vTPM da VM de Gerenciamento como vTPMg.

Empregar uma VM de gerenciamento possibilita a integração da solução com uma das principais arquiteturas utilizadas em *clusters* contemporâneos [Lowe 2019], além de viabilizar a implementação de mecanismos destinados a gerenciar a sobrecarga no TPM em decorrência da ancoragem dos vTPMs e das limitações intrínsecas ao hardware do chip.

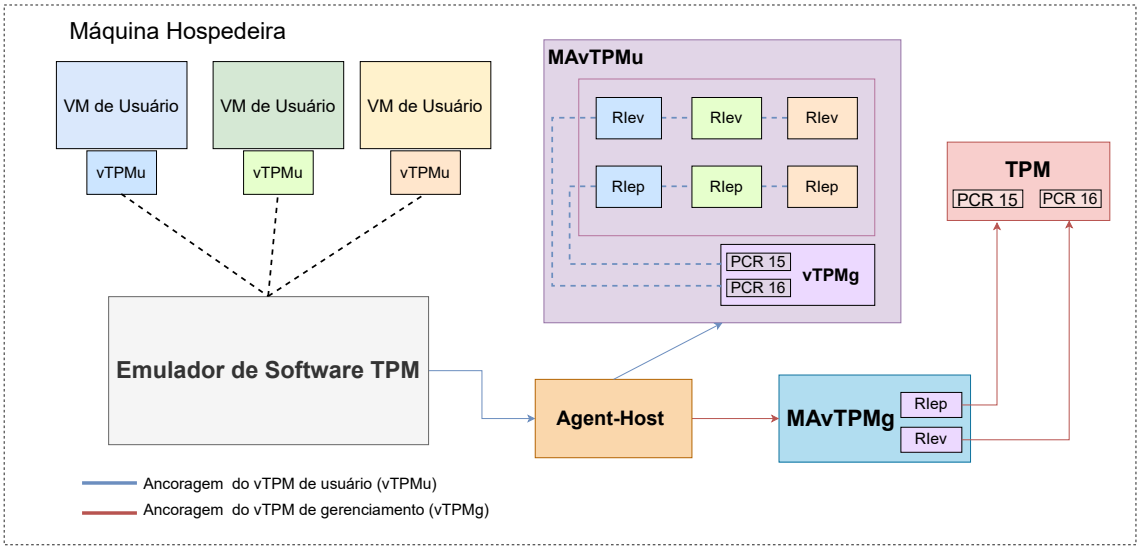

Figura 1. Arquitetura do Mecanismo de Suporte a Ancoragem Segura de TPMs Virtuais

Adicionalmente, para consolidar a cadeia de confiança, nossa abordagem propõe ancorar o vTPMg no chip TPM. Essa estrutura estabelece uma relação de confiança, resultando em uma redução da sobrecarga na raiz de confiança, que passa a lidar apenas com a ancoragem do vTPMg. Dessa forma, ao confiarmos no TPM, podemos verificar a integridade do vTPMg e, por último, a integridade do vTPMu.

Tanto no caso do vTPMg quanto no vTPMu, a ancoragem referida está relacionada à medição e ao mapeamento das alterações nos estados dos vTPMs. Mecanismos de avaliação de integridade, como o IMA [Luo et al. 2019], possibilitam a verificação da integridade de arquivos críticos no sistema. Sob essa mesma perspectiva, a abordagem proposta envolve a realização de medições nos estados do vTPM toda vez que ocorrem mudanças.

Funcionalmente, toda solicitação legítima deve passar pelo Emulador de Software TPM. Dessa forma, o Emulador é o único componente autorizado a realizar alterações nos estados dos vTPMs. Portanto, nossa abordagem confere ao Emulador a responsabilidade de identificar modificações nos estados e efetuar as medições necessárias. É importante destacar que a solução proposta pressupõe a prévia realização do processo de atestação remota do *host* que hospeda as VMs (e, por conseguinte, os vTPMs). Assim, assegura-se a manutenção da integridade das aplicações em execução no mesmo.

Sempre que o Emulador detecta uma alteração de estado, o Agent-Host é acionado. O Agent-Host é encarregado de processar as alterações relatadas pelo Emulador, discernindo se elas se originam de um vTPMu ou vTPMg. Nossa abordagem considera que o estado permanente é alterado quando há uma modificação na NVRAM do vTPM e que o estado volátil é alterado quando um ou mais PCRs são modificados.

Se a modificação decorre do estado permanente, o Emulador mede o arquivo NVRAM do vTPM e envia esses dados para o Agent-Host. Em caso de alterações no estado volátil, indicando modificações nos PCRs do vTPM, o Emulador informa ao Agent-Host os detalhes sobre o número e o valor atual do PCR afetado. Ademais, cada conjunto

de informações enviado ao Agent-Host inclui a especificação do estado relevante (volátil ou permanente) e a identificação do vTPM relacionado. Com isso, o Agent-Host avalia a origem da medição, determinando se ela vem de um vTPMu ou vTPMg.

Quando a medição é associada a um vTPMu, o Agent-Host a encaminha para um Módulo de Ancoragem de vTPMu (MAvTPMu). O MAvTPMu é um módulo de Kernel em execução na VM de Gerenciamento, responsável por receber as informações de medição por meio de um socket netlink e realizar a ancoragem dos estados do vTPMu no vTPMg. Da mesma forma, quando o Agent-Host verifica que a medição advém de um vTPMg, ele a encaminha para um Módulo de Ancoragem de vTPMg (MAvTPMg). O MAvTPMg é um módulo de Kernel em operação na máquina hospedeira, responsável pela ancoragem dos estados do vTPMg no TPM. Tanto o MAvTPMu quanto o MAvTPMg possuem algumas semelhanças e diferenças em sua operação. Nas seções subsequentes, será detalhado o processo de ancoragem realizado por cada um desses módulos.

4.1. Processo de Ancoragem do vTPMu no vTPMg realizado pelo MAvTPMu

Para cada vTPMu, o MAvTPMu cria uma estrutura de Registro de Integridade (RI) para o estado volátil (RIev) e para o estado permanente (RIep). No RIev e RIep ocorre a ancoragem das medições do estado volátil e estado permanente, respectivamente. Sempre que uma nova medição é recebida, o módulo do Kernel identifica a qual vTPM e estado ela pertence, executando, então, uma operação de hash estendido no RI correspondente. Para associar os RIs ao vTPMg, os RIeps de cada vTPMu são estendidos no PCR 15 do vTPMg (gPCR 15), enquanto os RIevs de cada vTPMu são estendidos no PCR 16 do vTPMg (gPCR 16).

Nesse sentido, ao receber uma medição proveniente do estado permanente, o MAvTPMu armazena a informação no seu respectivo RIep. Nesse cenário, o Módulo sempre mantém armazenado o valor mais recente da medição do estado permanente de um vTPMu em seu RIep correspondente. Quando um novo RIep é criado (no caso de uma nova instância vTPMu) ou um dos RIeps são alterados, todos os RIeps são sumarizados e estendidos no gPCR 15. Além disso, para viabilizar a recomputação do valor corrente do gPCR 15, o MAvTPMu também retém seu valor anterior. Dessa forma, se o valor anterior do gPCR 15 estendido com o valor sumarizado de todos os RIeps corresponder ao valor atual do gPCR 15, é possível confiar nos RIeps.

Uma vez que os RIeps são considerados confiáveis, eles podem ser utilizados para avaliar a integridade do estado permanente. Se o resultado coincidir com o valor corrente do gPCR 15, isso significa que os valores dos RIeps são confiáveis. Assim, torna-se possível validar a integridade do estado permanente de um vTPMu específico ao comparar o hash de sua NVRAM com o valor do RIep correspondente. Se esses valores corresponderem, conclui-se que o estado permanente está íntegro.

Quando a medição é proveniente do estado volátil, o MAvTPMu estabelece uma estrutura que registra o valor mais recente de cada um dos 24 PCRs de cada vTPMu (uPCR). A título de exemplo, se o MAvTPMu armazena o valor Z para o uPCR 10 e recebe a indicação do Agent-Host de que o uPCR 10 foi alterado para o valor H, então o valor Z será substituído por H. Dessa maneira, o MAvTPMu mantém um registro dos valores esperados para os uPCRs de cada VM de Usuário. Desse modo a ancorar essas informações, a cada nova alteração nos valores dos uPCRs, o MAvTPMu realiza o hash

estendido do valor sumarizado dos uPCRs no seu respectivo RIev. Isso significa que o RIev de cada vTPM armazenará o valor sumarizado dos seus respectivos 24 uPCRs.

Semelhante ao estado permanente, quando um dos RIev é criado ou modificado, todos os RIevs são sumarizados e estendidos no gPCR 16. Adicionalmente, assim como para os RIep, o valor anterior do gPCR 16 também é armazenado para permitir a recomputação do gPCR 16 durante o processo de verificação de integridade dos RIev.

Portanto, após a verificação da integridade dos RIev, é possível realizar a atestação do estado volátil do vTPMu de uma VM específica. Para isso, é efetuado o hash estendido entre os 24 valores de uPCR registrados pelo módulo. Se o valor corresponder ao RIev daquele vTPMu, significa que é possível confiar nos valores registrados pelo módulo. Nesse cenário, ao solicitar um *quote* do vTPMu correspondente, espera-se que os valores de cada um de seus uPCRs sejam equivalentes aos registrados pelo MAvTPMu para que o estado volátil seja considerado íntegro.

4.2. Processo de Ancoragem do vTPMg no TPM realizado pelo MAvTPMg

O MAvTPMg opera de maneira análoga ao MAvTPMu, entretanto, faz uso exclusivamente de um RIev e um RIep. Isso se deve ao fato de estarmos considerando a ancoragem de apenas um vTPM: o vTPMg. Dessa forma, ao receber uma medição proveniente do estado permanente, o MAvTPMg armazena a informação no RIep e realiza a operação de hash estendido dessa informação no PCR 15 do TPM. Nesse cenário, o módulo sempre retém o valor mais recente da medição do estado permanente no RIep.

Além disso, assim como no MAvTPMu, para viabilizar a recomputação do valor corrente do PCR 15, o MAvTPMg também retém seu valor anterior. Durante um processo de verificação de integridade do estado permanente, o valor anterior do PCR 15 é estendido com o valor da RIep. Se o resultado coincidir com o valor atual do PCR 15, isso indica que é possível confiar no RIep, ou seja, na medição registrada pelo MAvTPMg. Por último, deve-se comparar o RIep com o hash do arquivo da NVRAM. Caso os valores correspondam, o estado permanente do vTPMg é considerado íntegro.

Igualmente ao MAvTPMu, para o estado volátil, o MAvTPMg estabelece uma estrutura que registra o valor mais recente de cada um dos 24 gPCRs do vTPMg. Desse modo, a cada nova alteração nos valores dos gPCRs, o MAvTPMg sumariza os valores no RIev e realiza o hash estendido do valor sumarizado no PCR 16 do TPM físico. De maneira análoga ao estado permanente, também é necessário armazenar o valor anterior do PCR 16. Dessa forma, durante a verificação de integridade, o valor anterior do PCR 16 é estendido com o valor do RIev. Caso o resultado corresponda ao valor atual do PCR 16, isso indica que é possível confiar no valor do RIev.

Se o RIev for considerado íntegro, é possível utilizá-lo para validar a integridade dos registros dos gPCRs feitos pelo MAvTPMg. Se o hash estendido entre todos os valores dos 24 gPCRs armazenados pelo MAvTPMg corresponder ao valor do RIev, significa que é possível confiar nos registros dos gPCRs. Consequentemente, durante um processo de atestação de integridade, é possível solicitar um *quote* do vTPMg e compará-lo com os valores de referência armazenados pelo MAvTPMg. Caso os valores correspondam, isso indica que é possível confiar na integridade do estado volátil.

Isso posto, se o TPM físico for validado legítimo no processo de verificação de

autenticidade e os estados do vTPMg ancorados nele forem confirmados como íntegros, é seguro confiar no vTPMg. Dessa forma, caso um vTPMu ancorado no vTPMg tenha seus estados considerados íntegros, isso implica que é possível confiar no vTPMu.

Vale ressaltar que, o MAvTPMg e o MAvTPMu armazenam os valores de medição para estados permanentes e voláteis em arquivos que compartilham as mesmas características e propriedades de segurança do arquivo gerado pelo IMA em sistemas Linux tradicionais. Estes arquivos são mantidos em uma área protegida, empregando o *sysfs* como sistema de arquivos. Isso possibilita a realização de verificações de integridade, incluindo a atestação remota.

5. Vulnerabilidades do vTPM

Uma das principais considerações ao utilizar um vTPM diz respeito à proteção do estado permanente, ou seja, da NVRAM. Diferentemente do TPM convencional, que incorpora fisicamente a NVRAM no hardware, o vTPM geralmente utiliza um arquivo em disco para simular a NVRAM [Narayanan et al. 2023]. A proteção das informações confidenciais armazenadas nesse arquivo é de extrema importância, dada sua natureza sensível.

Recentemente, a CVE (*Common Vulnerabilities and Exposures*) divulgou uma vulnerabilidade crítica associada ao estado permanente do SWTPM, um emulador de vTPM em código aberto amplamente reconhecido e utilizado [Jha et al. 2022, Pecholt and Wessel 2022]. Na referida vulnerabilidade, um atacante é capaz de sobrescrever o arquivo do estado permanente por meio de um ataque de ligação simbólica [CVE 2023a].

Outro ataque similar frequentemente discutido na literatura especializada é o de *Rollback* [Wang et al. 2023, Narayanan et al. 2023]. Neste cenário, o atacante prepara uma cópia do estado permanente do sistema antecipadamente e, em um momento posterior, substitui esse estado no lugar da versão armazenada. A principal consequência desse ataque é a falta de garantia de que o estado persistente esteja sempre atualizado, pois o invasor tem a capacidade de trocar o arquivo da NVRAM por uma versão anterior.

Há pouca informação a respeito da implementação do vTPM nas plataformas comerciais, o que torna difícil a compreensão das garantias de segurança fornecidas. Contudo, é importante notar que a maioria dessas implementações adere à especificação e implementação de referência do TPM 2.0, estabelecida pela TCG (*Trusted Computing Group*) [Cloud 2023, Microsoft Azure , Amazon Web Services]. Isso implica que vulnerabilidades identificadas na especificação do TPM 2.0 tendem a impactar todas as implementações baseadas nela.

Um exemplo relevante é a vulnerabilidade divulgada pelo CVE no código de referência da TCG relacionada a gravação fora dos limites [CVE 2023b]. Esta falha permite que um atacante insira até 2 bytes extras em um comando, possibilitando alterações não autorizadas em áreas específicas da memória. Essa vulnerabilidade é crítica tendo em vista que os valores dos PCRs, que representam o estado volátil, são armazenados em memória. Portanto, se a falha de segurança impactar a área de memória onde os dados dos PCRs estão armazenados, a integridade do estado volátil do sistema é comprometida.

Todas as vulnerabilidades identificadas colocam em risco a integridade do vTPM. Nossa solução é capaz de detectar não só as violações decorrentes das vulnerabilida-

des mencionadas, mas também outras que possam comprometer a integridade do vTPM. Considerando que o objetivo principal do vTPM é estabelecer uma base confiável, nossa solução assegura a validade dessa raiz de confiança baseada na sua integridade.

6. Avaliação Experimental

Nesta seção, detalhamos a metodologia utilizada para avaliar o desempenho da abordagem proposta e discutimos os resultados obtidos. O ambiente de testes foi estabelecido em um computador equipado com um processador Intel i9-12900 e 32 GB de RAM. Utilizamos o NPCT75x TPM 2.0 como nosso modelo de TPM de hardware. O sistema operacional foi o Ubuntu 20.04, operando com a versão 4.2.1 do QEMU. Para os testes, configuramos a VM de Gerenciamento executando Ubuntu 22.04, com 4 vCPUs e 4 GB de memória principal, e dez VMs de Usuário operando com OpenSUSE 5.14, 1 vCPU e 1,5 GB de memória principal. Cada VM dispõe de um vTPM emulado pelo SWTPM 0.8.1, usando a biblioteca libtpms versão 0.9.6.

A solução foi implementada no SWTPM, um Emulador de Software de código aberto e de ampla utilização. Conjuntamente, os componentes de ancoragem da solução foram incorporados em módulos de Kernel. Além disso, esses módulos são responsáveis por manter os arquivos que contêm os valores de medição dos estados, essenciais para a atestação do vTPM. Esses arquivos são armazenados em uma área segura, de maneira similar aos arquivos de medição produzidos pelo IMA.

O propósito do experimento consiste em avaliar a sobrecarga introduzida pela nossa solução. Dado que a abordagem proposta realiza modificações no Emulador, consideramos importante avaliar se a abordagem impacta o desempenho do ponto de vista do usuário que interage com o vTPM. Para isso, contemplamos duas métricas: o *Tempo para Adicionar uma Chave (TAC)* e o *Tempo para Hash Estendido (THE)*. Essas métricas acarretam alterações, respectivamente, no estado permanente e volátil.

A métrica *TAC* está relacionada ao período necessário para que o vTPM incorpore uma chave criada em sua NVRAM. A execução dessa operação envolve o uso do comando *tpm2_evictcontrol* [Ubuntu Manpage 2019a]. Para mensurar essa métrica, registramos o intervalo de tempo entre o início e o término da execução desse comando. Durante o experimento executamos o comando 100 vezes em cada uma das VMs simultaneamente.

Por outro lado, o *THE* refere-se à duração necessária para realizar o hash estendido no PCR de um vTPM. Utilizamos o comando *tpm2_pcrextend* [Ubuntu Manpage 2019b] para realizar essa operação. Devido ao tempo de execução bastante reduzido de um único *extend*, para esta métrica, medimos o tempo total decorrido durante a execução de dez *extends*. Este experimento foi replicado 10 vezes simultaneamente em cada uma das dez VMs em execução.

Na Figura 2 é possível observar a comparação entre a distribuição dos dados coletados para o *TAC* quando o vTPM é operado com nossa abordagem e quando é executado na versão padrão do SWTPM, que não inclui nossas modificações. Os resultados dos experimentos sugerem que a implementação de nossa solução não impõe sobrecarga no processo de adição de chaves à NVRAM. Tal análise é um indício de que a abordagem proposta não interfere no desempenho da interação com o estado permanente da perspectiva do usuário do vTPM.

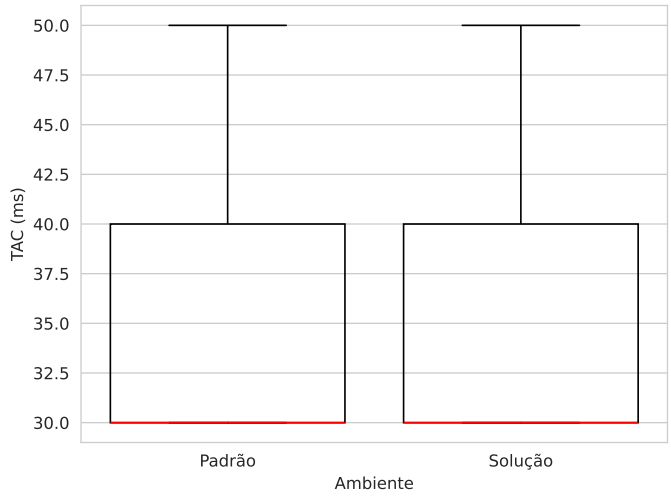

Figura 2. Tempo para Adição de Chaves (TAC) à NVRAM quando utilizado o ambiente padrão e o ambiente com a solução proposta

Semelhantemente, na Figura 3 é possível comparar o THE em cenários com e sem a implementação da nossa solução. A análise revela uma sobreposição nas distribuições, portanto, não é possível afirmar que a adoção da nossa metodologia interfere no tempo necessário para a operação de *hash* estendido. Esses achados indicam preliminarmente que nossa abordagem não exerce impacto adverso no desempenho das operações que envolvem interações com o estado volátil.

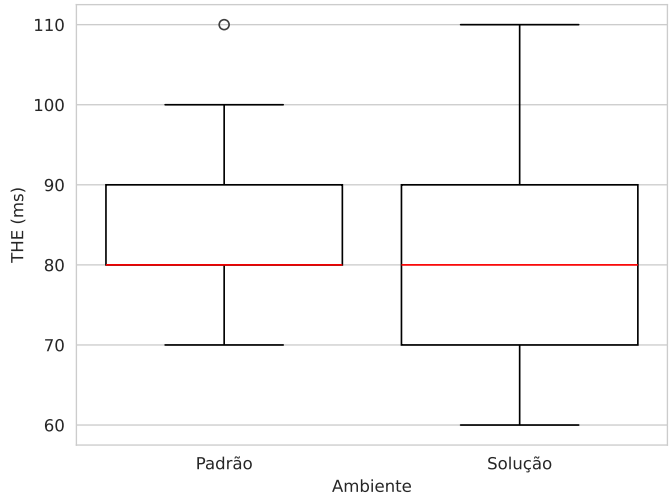

Figura 3. Tempo para Hash Estendido (THE) quando utilizado o ambiente padrão e o ambiente com a solução proposta

Concomitantemente à avaliação do TAC e THE, realizamos medições da utilização de CPU e Memória com o objetivo de analisar a carga imposta pela nossa solução nesses recursos. Na Figura 4, é possível verificar que a implementação da abordagem proposta resultou em um aumento na utilização da CPU. Este comportamento é previsível, dada a execução dos componentes adicionais necessários à operação da solução, como o Agent-Host e o MAvTPMg. No entanto, tanto para o estado permanente quanto

para o estado volátil, o crescimento foi, em média, de aproximadamente 4 pontos percentuais. Isso sugere que, embora nossa abordagem gere um aumento no consumo de recursos, esse incremento pode ser considerado moderado e, possivelmente, aceitável dentro do contexto aplicado.

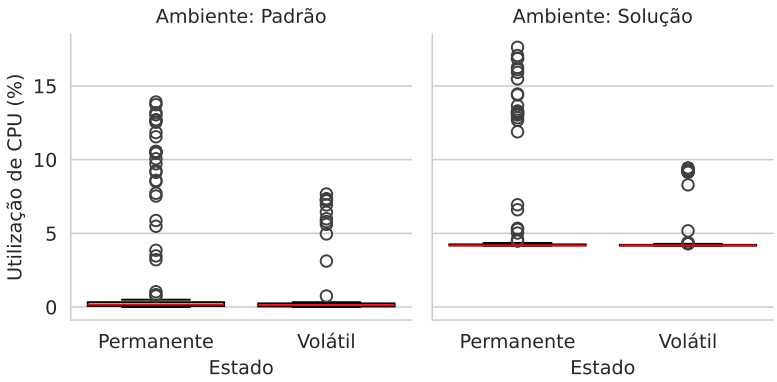

Figura 4. Percentual de Utilização de CPU durante as interações do estado permanente e volátil para os cenários padrão e com a solução proposta

Além disso, na Figura 5, a distribuição dos dados coletados revela um discreto aumento na utilização de memória. Em média, tanto para o estado permanente quanto para o estado volátil, observa-se um crescimento de aproximadamente 0,05 pontos percentuais. Esses resultados sugerem que a solução proposta tem um impacto mínimo na utilização de memória.

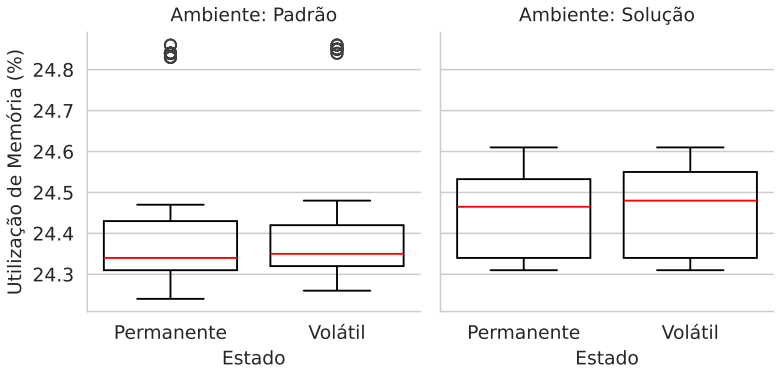

Figura 5. Percentual de Utilização de Memória durante as interações do estado permanente e volátil para os cenários padrão e com a solução proposta

7. Conclusão

Este trabalho apresenta uma nova abordagem para ancoragem de vTPMs baseado em sua integridade. A solução proposta se baseia em evidências de integridade dos estados permanente e volátil. A raiz de confiança para assegurar essas evidências é construída a partir de um módulo de computação confiável: o TPM. Contudo, reconhecendo as limitações de desempenho inerentes ao chip TPM, nossa arquitetura emprega uma VM de gerenciamento para coordenar a ancoragem dos vTPMs dos usuários em seu próprio vTPMg. Neste arranjo, somente o vTPM de gerenciamento (vTPMg) é diretamente ancorado ao

chip TPM. Tal abordagem visa minimizar a sobrecarga operacional no chip. Por meio desta cadeia de confiança estabelecida, um vTPM é considerado seguro e confiável se sua integridade for validada. Isso assegura que as operações estão sendo executadas conforme o previsto e que não ocorreram alterações indevidas.

Os resultados das avaliações sugerem que as alterações realizados no emulador não interferem no tempo de resposta das interações do usuário com o vTPM e seus respectivos estados. Ademais, a análise conduzida revela que a abordagem proposta adicionou um custo adicional de, em média, 4 pontos percentuais no uso CPU e 0,05 pontos percentuais no consumo de memória. Tais resultados preliminares demonstram que a abordagem proposta não compromete o desempenho do sistema, e que os incrementos observados são justificáveis, considerando os benefícios trazidos pela implementação proposta.

Para trabalhos futuros, sugere-se aprimorar a solução atual focando no desenvolvimento de suporte para snapshots. Isso permitiria a reversão segura do vTPM a um estado anterior, aumentando a proteção contra ataques de reversão. Outro aspecto importante é a expansão da solução para viabilizar a migração. Isso envolveria a transferência segura das informações de ancoragem para a máquina hospedeira de destino, em conjunto com a VM e o vTPM. Ademais, é relevante ampliar o escopo de experimentação, abrangendo ambientes mais robustos e investigando a utilização de outros emuladores de vTPM. Por último, considera-se relevante explorar a integração dessa abordagem com outros esquemas de ancoragem baseados em confidencialidade, já propostos na literatura científica.

Referências

Amazon Web Services. AWS Nitro Enclaves - Nitro TPM. `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/nitrotpm.html`. Accessed on: March 28, 2024.

Arthur, W. and Challener, D. (2015). *A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security*. Springer Nature.

Berger, S. (Acessado em 9 de janeiro de 2024). swtpm - software-based tpm emulator. `https://github.com/stefanberger/swtpm`.

Cloud, G. (2023). Virtual trusted platform module for shielded vms: security in plaintext. Acesso em: 18 de janeiro de 2024.

CVE (2023a). Cve-2020-28407. `https://www.cve.org/CVERecord?id=CVE-2020-28407`. Acessado em 9 de janeiro de 2024.

CVE (2023b). Cve-2023-1017. `https://www.cve.org/CVERecord?id=CVE-2023-1017`. Acessado em 9 de janeiro de 2024.

De Benedictis, M., Jacquin, L., Pedone, I., Atzeni, A., and Lioy, A. (2024). A novel architecture to virtualise a hardware-bound trusted platform module. *Future Generation Computer Systems*, 150:21–36.

Du, R., Pan, W., and Tian, J. (2018). Dynamic integrity measurement model based on vtpm. *China Communications*, 15(2):88–99.

Jha, D. N., Lenton, G., Asker, J., Blundell, D., and Wallom, D. (2022). Trusted platform module-based privacy in the public cloud: Challenges and future perspective. *IT Professional*, 24(3):81–87.

Kucab, M., Boryło, P., and Chołda, P. (2021). Remote attestation and integrity measurements with intel sgx for virtual machines. *Computers & Security*, 106:102300.

Lowe, Piscaer, G. M. C. T. (2019). The gorilla guide to hyperconvergence infrastructure foundations.

Luo, W., Shen, Q., Xia, Y., and Wu, Z. (2019). Container-ima: a privacy-preserving integrity measurement architecture for containers. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, pages 487–500.

Microsoft (2023). Azure attestation - overview. `https://learn.microsoft.com/en-us/azure/attestation/overview`. Acesso em: 18 de janeiro de 2024.

Microsoft Azure. Trusted launch for azure virtual machines. `https://learn.microsoft.com/en-us/azure/virtual-machines/trusted-launch`. Accessed on: March 28, 2024.

Narayanan, V., Carvalho, C., Ruocco, A., Almási, G., Bottomley, J., Ye, M., Feldman-Fitzthum, T., Buono, D., Franke, H., and Burtsev, A. (2023). Remote attestation of sev-snp confidential vms using e-vtpms. *arXiv preprint arXiv:2303.16463*.

Pecholt, J. and Wessel, S. (2022). Cocotpm: Trusted platform modules for virtual machines in confidential computing environments. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 989–998.

Perez, R., Sailer, R., van Doorn, L., et al. (2006). vtpm: virtualizing the trusted platform module. In *Proc. 15th Conf. on USENIX Security Symposium*, pages 305–320.

Sun, H., He, R., Zhang, Y., Wang, R., Ip, W. H., and Yung, K. L. (2018). etpm: A trusted cloud platform enclave tpm scheme based on intel sgx technology. *Sensors*, 18(11):3807.

Tassyany, M., Sarmento, R., Falcao, E., Gomes, R., and Brito, A. (2021). Um mecanismo de aprovisionamento de identidades para microsserviços baseado na integridade do ambiente de execução. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 714–727. SBC.

Ubuntu Manpage (2019a). tpm2_evictcontrol - manages entries in the tpm2 persistent storage (eps) index. `https://manpages.ubuntu.com/manpages/focal/man1/tpm2_evictcontrol.1.html`. Acessado em 17 de janeiro de 2024.

Ubuntu Manpage (2019b). tpm2_pcrextend - extend pcr values. `https://manpages.ubuntu.com/manpages/jammy/man1/tpm2_pcrextend.1.html`. Acessado em 17 de janeiro de 2024.

VMware (2018). vsphere 6.7 virtual trusted platform modules. `https://blogs.vmware.com/vsphere/2018/05/vsphere-6-7-virtual-trusted-platform-modules.html`. Acesso em: 18 de janeiro de 2024.

Wang, J., Wang, J., Fan, C., Yan, F., Cheng, Y., Zhang, Y., Zhang, W., Yang, M., and Hu, H. (2023). Svtpm: Sgx-based virtual trusted platform modules for cloud computing. *IEEE Transactions on Cloud Computing*.