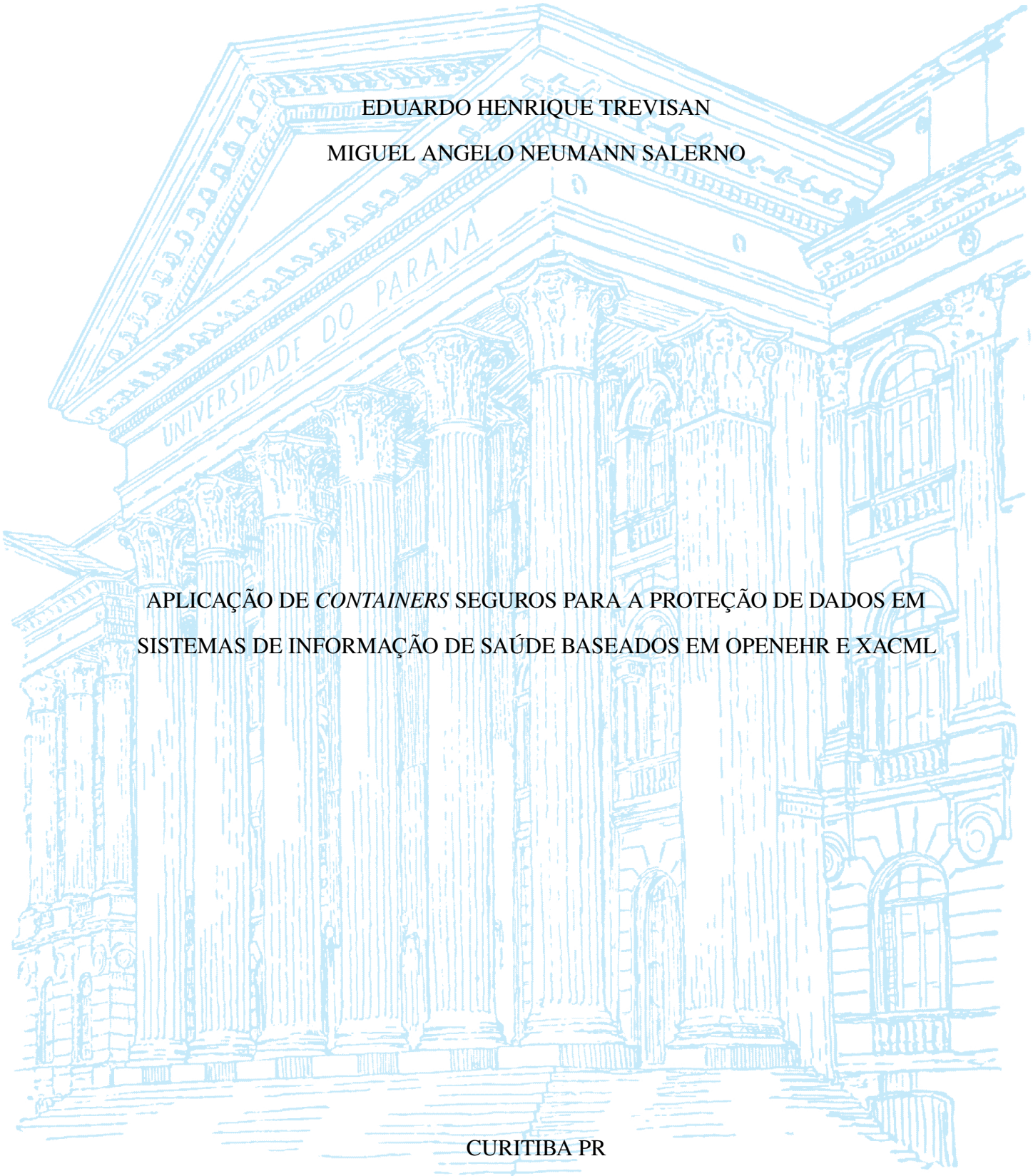
UNIVERSIDADE FEDERAL DO PARANÁ

EDUARDO HENRIQUE TREVISAN

MIGUEL ANGELO NEUMANN SALERNO



APLICAÇÃO DE *CONTAINERS* SEGUROS PARA A PROTEÇÃO DE DADOS EM SISTEMAS DE INFORMAÇÃO DE SAÚDE BASEADOS EM OPENEHR E XACML

CURITIBA PR

2022

EDUARDO HENRIQUE TREVISAN
MIGUEL ANGELO NEUMANN SALERNO

APLICAÇÃO DE *CONTAINERS* SEGUROS PARA A PROTEÇÃO DE DADOS EM SISTEMAS DE INFORMAÇÃO DE SAÚDE BASEADOS EM OPENEHR E XACML

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação e Informática Biomédica, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Carlos Alberto Maziero.

Coorientador: Thayse Marques Solis.

CURITIBA PR

2022

RESUMO

Com a demanda pela interoperabilidade de sistemas de informação de saúde aumentando, o nível de segurança dos sistemas de *Electronic Health Records* (EHR) precisa ser aperfeiçoado para estabelecer a integridade e a confidencialidade dos dados. É preciso pensar em formas de lidar com sistemas de dados médicos de forma a minimizar os riscos no caso de comprometimento de sistemas integrantes a redes de interoperabilidade. Neste trabalho é apresentada uma proposta para aprimorar a segurança de sistemas de informação em saúde baseados em OpenEHR, um conjunto de especificações e modelos clínicos utilizados como referência para criar padrões de dados e construir soluções de interoperabilidade para sistemas de informação em saúde. As tecnologias *Secure Container Environment* (SCONE) e *Software Guard Extensions* (SGX) são utilizadas para garantir a segurança e confiabilidade de dados sensíveis em caso de comprometimento de sistemas com controle de acesso em *Extensible Access Control Markup Language* (XACML). Com este estudo são identificados componentes do sistema de controle de acesso XACML que lidam com dados sensíveis de atributos médicos e podem ser protegidos por um enclave SGX. Com a encapsulamento destes componentes é possível atestar a validade, segurança e integridade dos dados, assim aprimorando o nível de segurança de sistemas de dados médicos em redes de interoperabilidade. Uma prova de conceito foi implementada para demonstrar que a utilização *containers* seguros SGX tem viabilidade no contexto de aplicações de dados médicos e que o impacto de desempenho com a aplicação das tecnologias apresentadas é pouco significativo.

Palavras-chave: OpenEHR. XACML. SGX. SCONE.

ABSTRACT

With a high demand for healthcare information systems interoperability it is necessary to improve Eletronic Health Records (EHR) level of security, estabilishing integrity and confientiality to medical data. New methods of dealing with medical data need to be developed in order to minimize risks when systems that are part of interoperability networks are compromised. In this work a proposal is presented to improve the security of health information systems based on OpenEHR, a set of specifications and clinical models used as a reference to create data standards and build interoperability solutions for health information systems. The Secure Container Environment (SCONE) and Software Guard Extensions (SGX) technologies are used to ensure the security and reliability of sensitive data in the event systems with access control in Extensible Access Control Markup Language (XACML) are compromised. With this study, components of the XACML access control system that deal with sensitive medical data attributes and can be protected by an SGX enclave are identified. With the encapsulation of these components, it is possible to attest to the validity, security and integrity of the data, thus improving the level of security of medical data systems in interoperability networks. A proof of concept was implemented to demonstrate that the use of SGX secure containers is viable in the context of medical data applications and that the performance impact with the application of the technologies presented is not significant.

Keywords: OpenEHR. XACML. SGX. SCONE.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ABAC	*Attribute-Based Access Control*
API	*Application Programming Interface*
AQL	*Archetype Query Language*
CKM	*Clinical Knowledge Manager*
DAC	*Discretionary Access Control*
EHR	*Electronic Health Record*
EPC	*Enclave Page Cache*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MAC	*Mandatory Access Control*
PAP	*Policy Administration Point*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PIP	*Policy Information Point*
PRP	*Policy Retrieval Point*
RBAC	*Role-Based Access Control*
REST	*Representational State Transfer*
SCONE	*Secure Container Environment*
SEHS	*Secure Electronic Health System*
SGX	*Software Guard Extensions*
XACML	*eXtensible Access Control Markup Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A interoperabilidade de sistemas de *Electronic Health Records* (EHR) tem ganhado importância para instituições de saúde. A consolidação de dados do histórico de saúde de um paciente a partir de instituições distintas pode melhorar consideravelmente a qualidade de seu tratamento médico. Um problema que dificulta essa integração é a diferença entre as implementações dos sistemas de saúde.

Algumas propostas de modelos e ferramentas que facilitam a comunicação entre sistemas distintos estão surgindo constantemente, como a iniciativa do Canadá e da Estônia de desenvolver a interoperabilidade de EHRs em âmbito nacional, o uso de EHRs em atendimento ambulatorial na África subsaariana e o sistema de Hong Kong que permite que as informações do paciente sejam compartilhadas em tempo real com clínicas e hospitais públicos e privados (Evans, 2016).

Para garantir a segurança na interoperabilidade entre sistemas de EHR, uma das etapas é analisar a arquitetura dos sistemas de controle de acesso e como estes sistemas se comunicam. Se uma instituição externa quer acessar recursos privados é importante utilizar sistemas robustos de identificação e autorização, assim como garantir que a troca de informações não sofra interferência de agentes externos (Kruse et al., 2017).

A segurança das informações de um sistema que integra uma rede de interoperabilidade também é um fator que deve ser considerado, ações devem ser tomadas para reduzir a quantidade de dados comprometidos da rede no caso de um ataque a um único sistema. Estes sistemas podem estar vulneráveis a ataques privilegiados, isto é, um ataque no qual são comprometidos recursos privilegiados do sistema como o *kernel*, *hipervisor* ou BIOS (Coppolino et al., 2018). Este tipo de ataque é o foco da proposta deste trabalho. Para que a informação esteja de fato segura, a segurança e a confiabilidade desses dados são necessárias nesse contexto de sistemas de saúde.

O trabalho está organizado em seções, contando com esta introdução (1), uma fundamentação teórica descrevendo os conceitos mais importantes para a compreensão do estudo (2), uma visão geral de alguns artigos de trabalhos relacionados (3), a explicação da proposta apresentada (4), o detalhamento da implementação do sistema proposto (5), a descrição dos experimentos realizados (6) e a conclusão do estudo (7).

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta a fundamentação teórica para direcionar a pesquisa de acordo com a problemática analisada. Para isso foram abordados conceitos como: OpenEHR, EHRbase, controle de acesso, ABAC, *Role-Based Access Control* (RBAC), *eXtensible Access Control Markup Language* (XACML), SGX e SCONE.

2.1 OPENEHR

O OpenEHR é um conjunto de especificações, modelos clínicos e artefatos de *software* que são utilizados como referência para criar padrões de dados e construir soluções de interoperabilidade para sistemas de informação em saúde (OpenEHR Foundation, 2022e). A organização do OpenEHR é dividida em quatro programas principais: especificações, modelagem clínica, *software* e educação, conforme apresentado abaixo.

- **Programa de especificações**: desenvolve e mantém as especificações técnicas do OpenEHR.
- **Programa de modelagem clínica**: desenvolve e publica arquétipos, modelos e diretrizes no contexto da saúde.
- **Programa de *software***: apoia o desenvolvimento de *software* com base nas especificações do OpenEHR.
- **Programa de educação**: a partir de um currículo básico, realiza o credenciamento de profissionais de treinamento e educação em OpenEHR (ainda não disponível).

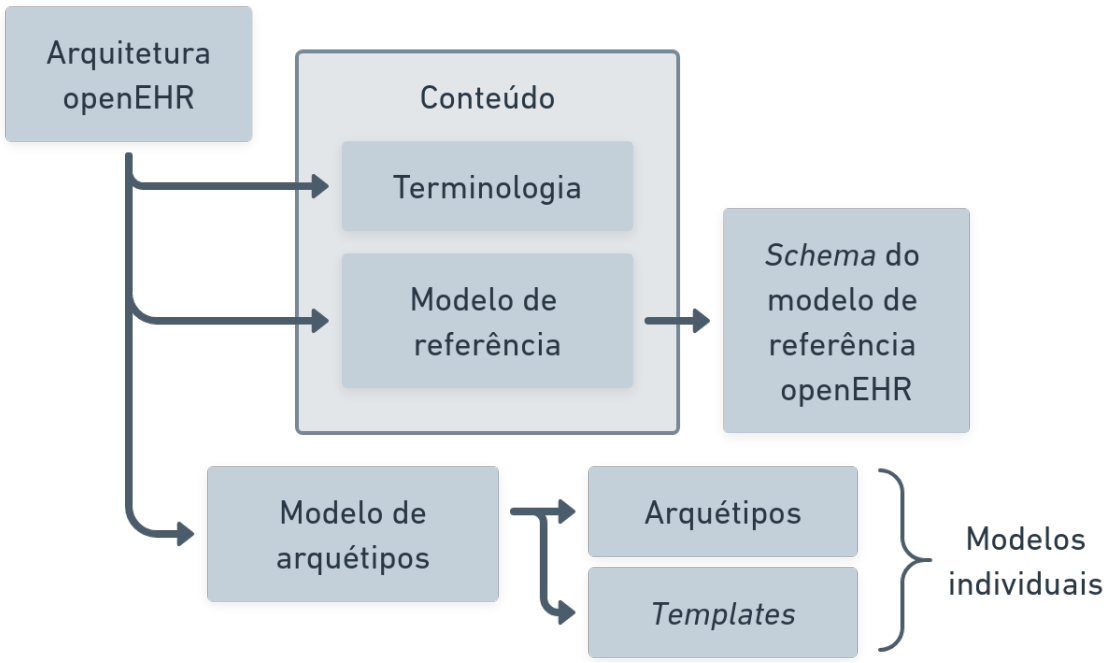


Figura 2.1: Arquitetura do OpenEHR. Fonte: adaptado de (Meredith, 2020).

A Figura 2.1 representa a arquitetura do OpenEHR de forma simplificada. Nesse diagrama fica evidente que o conteúdo de governança composto pela terminologia, que armazena as definições dos termos, e pelo modelo de referência de informação, que define como os dados devem ser utilizados, está separado da modelagem de arquétipos. Essa independência permite que novos arquétipos e *templates* sejam criados sem que alterações sejam feitas na forma como os dados são utilizados (OpenEHR Foundation, 2022d).

2.1.1 Arquétipos

Os arquétipos são uma definição genérica, formal e estruturada de uma informação a nível de domínio. Um arquétipo deve ter suas informações detalhadas da forma mais completa e abrangente possível para ser aplicável a diversas situações. Essa informação padronizada será posteriormente utilizada para construir modelos (*templates*) de uma forma mais específica (Bacelar e Correia, 2015). A Figura 2.2 mostra um exemplo de arquétipo de pressão sanguínea, no qual apenas alguns campos estão sendo apresentados, como eventos, descrição, dados e estado do paciente.

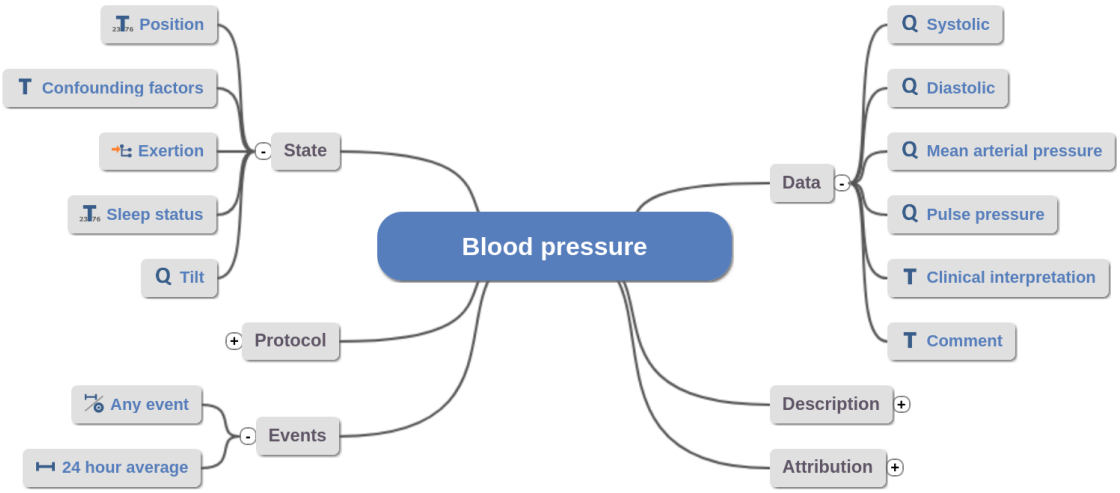


Figura 2.2: Exemplo de arquétipo com informações de pressão sanguínea. Fonte: (OpenEHR Foundation, 2022d).

2.1.2 *Templates*

Os arquétipos fornecem as informações de forma genérica e abrangente. Para aplicar essas informações em uma situação específica, é possível que apenas um subconjunto dos dados seja necessário. Essa modelagem específica representa o *template*, no qual só é representado o conteúdo que será utilizado na aplicação. Na Figura 2.3 é possível observar que um mesmo arquétipo pode fornecer dados para diferentes *templates*, que utilizam dados distintos (OpenEHR Foundation, 2022b). Nesse caso, a lista de arquétipos (ao centro) fornece dados para *templates* de *check-up* diabético e cardiologia clínica, que usam dados semelhantes.

2.1.3 *Compositions*

Uma *composition* no OpenEHR é um registro de informações sobre um determinado evento ou estado de saúde, estruturada usando arquétipos como o documento clínico padrão. A *composition* pode ser pensada como um recipiente para um ou mais itens de informação, conhecidos como objetos de entrada. Essas informações compõem uma visão do estado e da

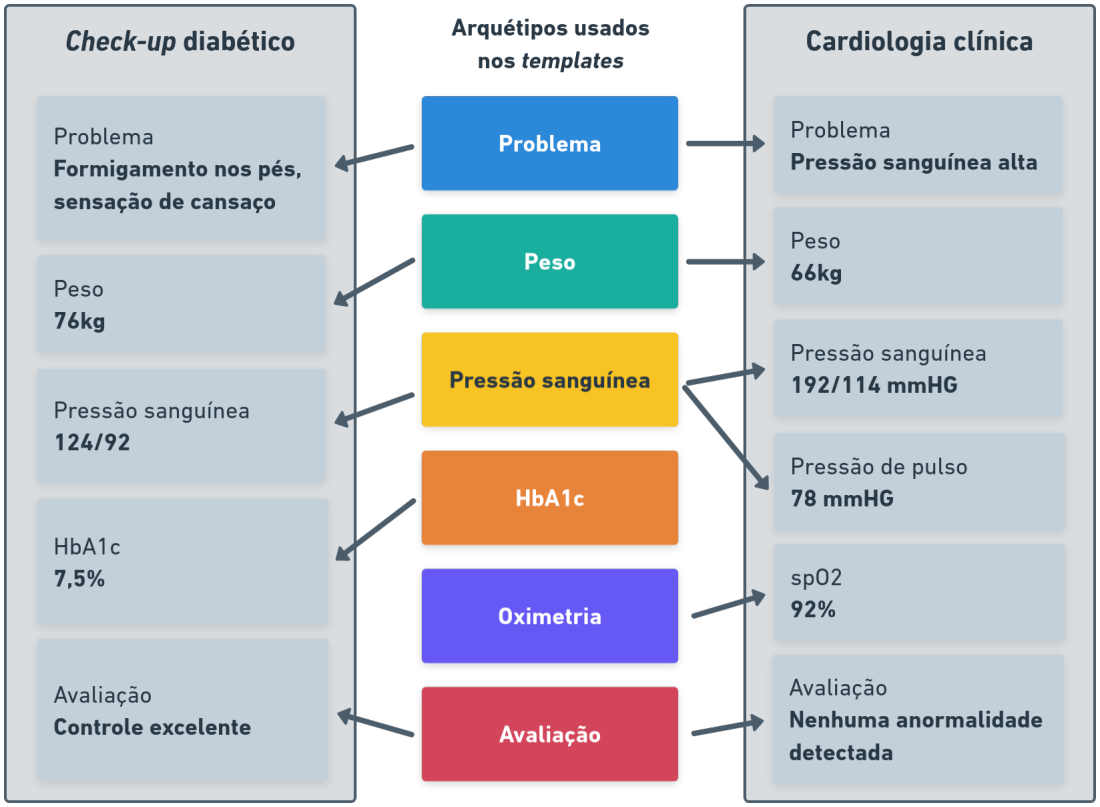


Figura 2.3: Exemplo de *template*. Fonte: adaptado de (FreshEHR Clinical Informatics, 2018).

situação paciente ao longo do tempo, à medida que novas informações vão sendo adicionadas (OpenEHR Foundation, 2022c).

As *compositions* são usadas para representar uma ampla gama de informações em um EHR, como por exemplo documentos clínicos, resumos de alta e notas de progresso, alergias, demografia, histórico familiar, pedidos e resultados, imunizações, medicamentos, encontros, imagens etc (OpenEHR Foundation, 2022c).

2.1.4 *Clinical Knowledge Manager*

O *Clinical Knowledge Manager* (CKM) é um repositório de arquétipos e *templates*, onde é possível consultar essas informações e aplicá-las nos sistemas. Também funciona como uma plataforma na qual é possível editar, atualizar e alterar as definições de um arquétipo ou *template* para obter melhorias. Por meio do CKM, sistemas distintos são capazes de usar os mesmos arquétipos, facilitando a troca de dados entre eles (Bacelar e Correia, 2015).

2.1.5 EHRbase

O OpenEHR permite que cada sistema tenha a aplicação separada do conjunto de dados, que está centralizado no CKM. Assim, a interoperabilidade se dá em dois níveis: primeiramente pela aderência ao modelo de referência do OpenEHR, e em segundo nível, pelo uso dos mesmos arquétipos entre os sistemas. (EHRbase, 2022).

O EHRbase é uma implementação *backend* do OpenEHR. Dessa forma, os dados podem ser acessados por meio de uma API (*Application Programming Interface*) REST (*Representational*

State Transfer) padronizada que usa a *Archetype Query Language* (AQL), linguagem criada para representar consultas em repositórios de arquétipos (OpenEHR Foundation, 2022a).

Pelo EHRbase é possível fazer requisições GET, POST, PUT e DELETE, usando tipos de dados como XML e JSON. A API também conta com funcionalidades de administração, métricas e monitoramento, validação de dados, controle de acesso, entre outros (EHRbase, 2022).

2.2 CONTROLE DE ACESSO

Controle de acesso é um campo da segurança da computação que engloba a identificação, autenticação e autorização de sujeitos, ações e objetos. O propósito do controle de acesso é proteger dados, serviços, aplicações executáveis ou qualquer outro tipo de objetos de operações não autorizadas (Chung et al., 2019).

A autorização define os direitos de acesso aos objetos e a realização de operações por sujeitos. Leitura, escrita e execução são exemplos de operações que um sujeito pode ser autorizado a realizar. Por outro lado, a autenticação é o ato de verificar se o sujeito que está realizando uma operação é realmente quem ele diz ser (Chung et al., 2019).

Dentre os mais importantes modelos de controle de acesso podemos citar o *Discretionary Access Control* (DAC), o *Mandatory Access Control* (MAC), o *Role-Based Access Control* (RBAC) e o *Attribute-Based Access Control* (ABAC). O DAC é um tipo de controle de acesso no qual o proprietário de um objeto decide quem tem permissão para acessá-lo e quais são os seus privilégios. No modelo baseado em MAC, um administrador central controla as políticas de segurança, que não podem ser sobrepostas ou substituídas. Os modelos ABAC e RBAC são mais relevantes para o escopo deste trabalho e serão descritos com mais detalhes nas próximas seções (Chung et al., 2019).

2.2.1 ABAC

O modelo *Attribute-Based Access Control* (ABAC) define que as requisições de sujeitos para executar operações em objetos são permitidas ou negadas baseadas nos atributos dos objetos, atributos dos sujeitos, condições do ambiente e um conjunto de políticas que são especificadas para esses atributos e condições (Chung et al., 2019).

Os atributos são as características do sujeito, objeto ou as condições do ambiente. Um sujeito é um usuário, uma aplicação ou dispositivo que realiza as requisições. Os objetos são recursos do sistema cujo acesso é gerenciado pelo próprio sistema, como arquivos, serviços e redes. Por fim, as condições de ambiente são características contextuais alheias ao sistema, como o horário, dia e localização (Chung et al., 2019).

Uma das vantagens de sistemas modelados em ABAC é não serem limitados pela identidade de um sujeito para a tomada de decisões. O ABAC fornece uma maior flexibilidade na construção de políticas do que outros métodos de controle de acesso como o RBAC, evitando a necessidade de autorizações explícitas para sujeitos e objetos individuais e a necessidade de conhecimento prévio de todos os papéis específicos presentes no sistema (Chung et al., 2019).

2.2.2 RBAC

O modelo *Role-Based Access Control* (RBAC) utiliza papéis predefinidos aos quais os sujeitos são atribuídos. Cada papel recebe um conjunto de privilégios que são herdados pelos sujeitos que têm esse papel. Quando uma requisição de acesso é realizada por um sujeito, o mecanismo de controle de acesso avalia o papel atribuído ao sujeito que fez a requisição e o

conjunto de operações que esse papel é autorizado a fazer. O acesso é permitido se o papel em questão contém a operação requisitada em seu conjunto (Sandhu et al., 1996).

O RBAC pode ser descrito como uma especialização do ABAC, utilizando os papéis como o atributo central do modelo. Apesar de ser menos flexível, a implementação e manutenção de um modelo RBAC pode ser mais simples, considerando que as relações entre papeis e permissões podem ser predefinidas, a atribuição de um usuário a um papel é fácil e contempla grande parte das necessidades comuns de controle de acesso (Chung et al., 2019).

2.2.3 XACML

A *Extensible Access Control Markup Language* (XACML) é uma linguagem de políticas e requisições/respostas de decisões de controle de acesso baseada em XML. Ela tem a função de descrever os requerimentos para o controle de acesso de recursos. A linguagem de requisições/respostas é usada para formar mensagens que perguntam se uma requisição deve ser permitida ou não. Esse modelo é um sistema de controle de acesso baseado em atributos (ABAC), mas um sistema baseado em papéis (RBAC) também pode ser implementado como uma extensão do ABAC (OASIS, 2010).

O XACML é útil para fornecer uma terminologia comum para implementações de controle de acesso e promover a interoperabilidade entre sistemas. Este modelo funciona de forma distribuída, permite o uso de atributos arbitrários em políticas e é fortemente extensível (OASIS, 2010).

O Modelo da linguagem de políticas é formado por três principais componentes: *Rule*, *Policy* e *Policy set*.

- ***Rule***, ou regra, é o principal elemento da política. Ela é formada por um alvo, efeitos, condições, obrigações e conselhos.
- ***Policy***, ou política, é formada por um alvo, um algoritmo de combinação de regras, um conjunto de regras, expressões de obrigações e conselhos.
- ***Policy set***, ou conjunto de políticas, é formado por um alvo, um algoritmo de combinação de políticas, um conjunto de políticas, expressões de obrigações e conselhos.

Neste modelo, como ilustrado na Figura 2.4, quando um sujeito quer realizar uma ação sobre um recurso ele fará uma requisição à entidade responsável, chamada de *Policy Enforcement Point* (PEP). O PEP tem a função de construir uma requisição baseada nos atributos do sujeito, o recurso a ser acessado e a ação a ser realizada. Esta requisição é enviada para um *Policy Decision Point* (PDP), que analisará a requisição baseada nas políticas que se aplicam. O PDP enviará uma resposta informando se o acesso foi permitido ou negado. O *Policy Retrieval Point* (PRP) guarda dados XACML de políticas para serem avaliados pelo PDP (OASIS, 2010).

Para analisar a requisição, o PDP pode consultar O *Policy Information Point* (PIP), que guarda os atributos descritivos não presentes na requisição. O *Policy Administration Point* (PAP) gerencia o funcionamento do PDP e do PIP (OASIS, 2010).

2.3 SGX

As *Software Guard Extensions* (SGX) são extensões da arquitetura de processadores Intel que fornecem integridade e confiabilidade para a execução de programas sensíveis em ambientes vulneráveis. O foco é lidar com incertezas envolvendo *software* de alto privilégio, como o *kernel* (Costan e Devadas, 2016).

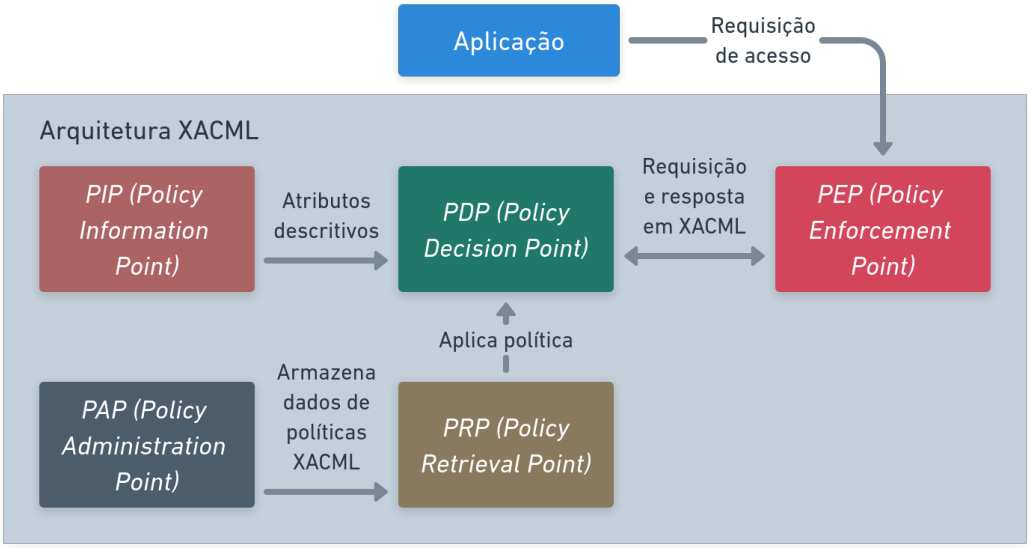


Figura 2.4: Arquitetura XACML. Fonte: os Autores (2022).

Através de recursos de *hardware* confiável, é possível executar, com maior seguraça, *software* sensível em máquinas remotas. O recurso *Attestation* é utilizado pelo SGX para provar para o usuário que o mesmo está se comunicando com um *software* específico executando em um *container* seguro sobre um *hardware* confiável (Costan e Devadas, 2016). Assim, é possível garantir integridade e confidencialidade das aplicações mesmo no caso de comprometimento do sistema operacional ou da BIOS, na máquina em que estão sendo executadas.

Os ambientes de execução confiáveis disponibilizados para as aplicações pelo SGX são chamados de enclaves. Esses ambientes utilizam uma região protegida da memória chamada *Enclave Page Cache* (EPC), onde os dados e o código da aplicação são armazenados. Processos que são executados fora do enclave não podem acessar a memória de enclaves, mas processos contidos em enclaves podem ter acesso à memória desprotegida, sendo responsabilidade do processo verificar a integridade destes dados (Arnautov et al., 2016).

O impacto na performance é inevitável quando um processo é executado em um enclave, já que este código deve executar em uma nova *thread* isolada dos processos não protegidos, além da necessidade de novas verificações sobre os dados e a criptografia das linhas de *cache* (Arnautov et al., 2016).

2.4 SCONE

O mecanismo de *Secure CONtainer Environment* (SCONE) usa o recurso SGX de processadores *Intel* em *containers Docker* para proteger processos executando no *container* contra ataques externos (Arnautov et al., 2016).

Os *containers* fornecem um maior desempenho do que máquinas virtuais, porém têm um isolamento mais fraco, pois utilizam virtualização à nível do sistema operacional, tornando mais fácil o comprometimento do sistema por um ataque externo (Arnautov et al., 2016). O SCONE fornece um método para aprimorar o isolamento e a segurança de aplicações utilizando a infraestrutura de *containers* para Linux já existente.

O objetivo do SCONE é criar um mecanismo que protege a confidencialidade e integridade da memória, código e entrada e saída de arquivos externos e rede contra atacantes

potencialmente privilegiados (Arnautov et al., 2016). Outro objetivo é integrar *containers* seguros em ambientes *Docker* existentes, sem esforço adicional por parte da administração do ambiente.

3 TRABALHOS RELACIONADOS

Modelos de controle de acesso baseados em XACML são flexíveis para a implementação de políticas complexas, podem ser distribuídos em múltiplos serviços e facilitam a interoperabilidade entre sistemas distintos (De la Rosa Algarín et al., 2016). Estas características se alinham com as necessidades de sistemas de dados médicos que buscam a interoperabilidade. Foram buscadas propostas que implementam modelos em XACML no contexto de sistemas de dados médicos que mostram as possibilidades de uso destes sistemas e sua viabilidade.

No artigo *Control Model for XML-Based Electronic Health Record System* (Seol et al., 2018) é proposto um modelo EHR focado em segurança e dividido em duas etapas. A primeira etapa é o controle de acesso ABAC construído com XACML e a segunda etapa é a segurança XML, que consiste em uma criptografia parcial e uma assinatura digital para a proteção dos dados sensíveis de documentos de pacientes, diminuindo o risco de expor esses dados. A assinatura digital garante que o conteúdo não foi falsificado ou alterado durante o envio e o recebimento dos dados.

O artigo *Standardized access control mechanisms for protecting ISO 13606 based electronic health record systems* (Calvillo-Arbizu et al., 2014) apresenta um mecanismo baseado em XACML que cumpre os princípios dos padrões ISO 13606. O modelo utiliza um motor de inferência para realizar as decisões automaticamente baseado nos padrões da ISO 13606-4. Para aumentar a interoperabilidade em cenários distintos, os atributos são definidos como parte de uma ontologia e as políticas de controle de acesso são transformadas em um formato semântico.

O artigo *Securing XML with Role Based Access Control: Case Study in Health Care* (De la Rosa Algarín et al., 2016) apresenta um modelo de controle de acesso baseado em papeis (RBAC) utilizando XACML para promover interoperabilidade de políticas entre sistemas. O modelo apresentado e técnicas analisadas são recomendadas para adoção de sistemas comerciais atuais em busca de maior interoperabilidade.

Estes artigos apresentam modelos com objetivos similares, a ideia é apresentar um modelo seguro que providencie interoperabilidade entre sistemas distintos. Mas existem significativamente menos recursos disponíveis sobre a segurança e confiabilidade de dados nestes modelos em caso de comprometimento de um sistema participante da integração.

No artigo *Exploiting new CPU Extensions for Secure Exchange of eHealth Data at the EU level* (Coppolino et al., 2018) o foco são possíveis ataques privilegiados, onde existe o comprometimento de recursos a nível do sistema operacional nos serviços que lidam com dados médicos. No contexto do *OpenNCP*, um modelo da União Europeia para interoperabilidade entre sistemas EHR, o objetivo é utilizar a extensão SGX para proteger os dados clínicos que são trocados entre nós da infraestrutura.

O modelo proposto utiliza o SGX para executar as operações sensíveis de uma aplicação de dados médicos dentro de um enclave seguro e estabelecer canais de comunicação seguros através do mecanismo de atestação remota do SGX, o que permite garantir a confiabilidade de um software em execução dentro de um enclave. A solução apresentada é avaliada em termos de desempenho e é concluído que não há um impacto significativo para volumes de dados realísticos (Coppolino et al., 2018).

Uma comparação entre os estudos de trabalhos relacionados pode ser observada na Figura 3.1, na qual podem ser verificados quais recursos são utilizados em cada estudo. Somente (Coppolino et al., 2018) não utiliza o XACML no desenvolvimento de seu trabalho, ao mesmo

Estudo	ABAC	RBAC	XACML	SGX	SCONE
(Calvillo-Arbizu et al., 2014)	x	-	x	-	-
(De la Rosa Algarín et al., 2016)	-	x	x	-	-
(Seol et al., 2018)	x	-	x	-	-
(Coppolino et al., 2018)	-	-	-	x	-
Modelo proposto	**x**	-	**x**	**x**	**x**

Tabela 3.1: Comparativo entre estudos de trabalhos relacionados. Fonte: os autores (2022)

tempo em que é o único ao integrar o SGX em seu modelo. Os outros estudos procuram apresentar um modelo de controle de acesso, seja por ABAC ou RBAC.

O modelo proposto busca combinar o XACML, o SGX e o SCONE para obter uma integração entre esses recursos, com a interoperabilidade que o XACML pode oferecer e a segurança do SGX com o SCONE.

4 PROPOSTA

Existem trabalhos que propõe modelos de controle de acesso e interoperabilidade entre sistemas de saúde, garantindo a segurança e confiabilidade dos dados quando em trânsito entre sistemas. Entretanto, poucos trabalhos são referentes à proteção de dados durante a execução de processos em plataformas de *hardware* não confiável (Coppolino et al., 2018).

Em sistemas distribuídos, especialmente em componentes que contém interfaces para comunicação com usuários e aplicações, existe a possibilidade da atuação de agentes maliciosos. Se um destes componentes for comprometido, as informações sensíveis que são tratadas nele também serão comprometidas (Coppolino et al., 2018). Informações que antes eram isoladas em sistemas específicos agora podem transitar por outros sistemas desconhecidos.

No trabalho de (Coppolino et al., 2018) o contexto é a interoperabilidade de sistemas de saúde entre países da União Europeia utilizando o OpenNCP, um *framework* análogo ao OpenEHR. Em sua análise, foi possível observar que, apesar de existir uma garantia de confiabilidade e segurança das informações que são trocadas entre instituições, o tratamento de dados pelo sistema de controle de acesso pode expor os dados para ataques internos. Um usuário malicioso poderia ganhar acesso à máquina *host* e, ao escalar privilégios, poderia ter acesso total ao sistema e alterar o conteúdo de documentos clínicos.

A conclusão do autor é que utilizando mecanismos de segurança providenciados pelo SGX é possível encapsular as etapas do controle de acesso nas quais as informações sensíveis são expostas. No contexto de sistemas implementados com o OpenEHR, é possível que o tratamento dos dados também esteja vulnerável a ataques desse tipo, e a aplicação do SGX pode ser uma solução para este problema.

O trabalho de (Seol et al., 2018) propõe um sistema de controle de acesso em XACML baseado em atributos para sistemas OpenEHR. Neste sistema é possível ver os componentes pertencentes ao XACML em um sistema distribuído. De acordo com o autor, a maior parte do controle de acesso está contida em um servidor seguro (bem protegido e não acessível ao público em geral), com exceção do PEP *(Policy Enforcement Point)* e do componente de segurança XML, que é a etapa responsável por realizar a criptografia dos dados a serem enviados para o usuário. Antes desta etapa, as informações do servidor de dados médicos são manipuladas pelo PEP e podem estar vulneráveis a ataques internos.

O servidor XACML é considerado seguro e disponibiliza uma interface segura para comunicação com o PEP. O serviço que contém o PEP tem várias funcionalidades que requerem a manipulação de dados sensíveis. Primeiro, um usuário ou instituição pede acesso para realizar uma ação em um recurso. Esta requisição em si já é sensível, especialmente se utilizando um sistema baseado em ABAC, pois contém mais informações em forma de atributos. O PEP, então, envia uma requisição em XML com os dados do sujeito e do recurso para o PDP do servidor XACML e recebe uma resposta. Se a resposta for positiva é este mesmo serviço que executará a ação nos recursos, que podem ser registros médicos ou outros dados sensíveis.

O artigo de (Seol et al., 2018) propõe uma etapa de segurança XML no serviço que contém o PEP. Esta etapa realiza uma criptografia nos dados sensíveis que serão enviadas a um sujeito, o que aumenta o nível de segurança e confiabilidade das informações quando em transição à um agente externo. Entretanto, como apontado no artigo de (Coppolino et al., 2018), o serviço análogo ao PEP é vulnerável a ataques internos, onde o servidor é comprometido por um agente adversário com acesso administrativo ao sistema, especialmente considerando que este serviço é exposto à requisições externas através de uma API.

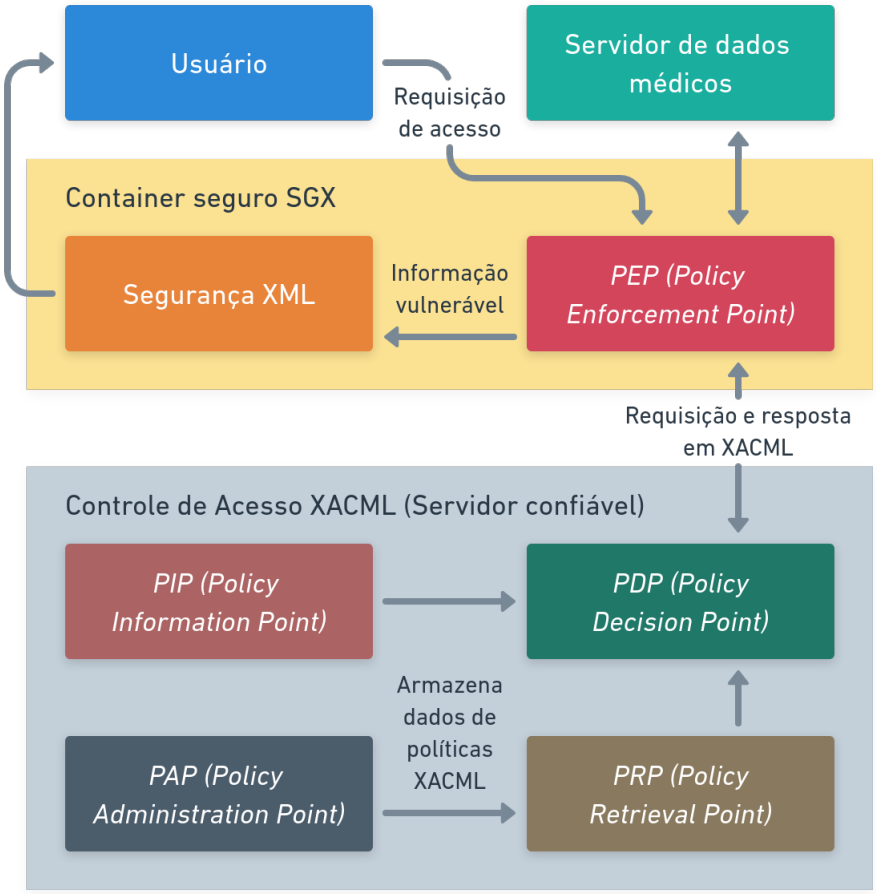


Figura 4.1: Diagrama da proposta. Fonte: adaptado de (Seol et al., 2018).

O comprometimento do PEP pode colocar em risco todos os dados sensíveis à qual a aplicação tem acesso, portanto, a execução do PEP e do componente de segurança XML em um enclave seguro pode mitigar os efeitos de um ataque interno, porém, introduzindo um custo de desempenho no sistema. Este custo foi abordado no artigo de (Coppolino et al., 2018), mas a introdução de um componente de criptografia de dados dentro do enclave tem o potencial de diminuir ainda mais o desempenho do serviço.

Considerando a possibilidade de comprometimento de recursos internos ao sistema, como servidores responsáveis pelo controle de acesso, o objetivo geral deste trabalho é aplicar o conceito de *containers* seguros utilizando as tecnologias *SCONE* e SGX para reproduzir a proposta de (Coppolino et al., 2018) no contexto de sistemas OpenEHR. O modelo de (Seol et al., 2018) fornece uma boa arquitetura para o estudo da aplicação desta proposta.

Para alcançar o objetivo geral deste trabalho, os seguintes objetivos específicos se mostram necessários:

- Implementar um sistema de controle de acesso em XACML para sistemas OpenEHR baseado no modelo de (Seol et al., 2018) como prova de conceito;
- Identificar neste sistema as etapas onde os dados estão vulneráveis a ataques internos, onde há o comprometimento do *kernel* do servidor da aplicação de dados médicos;
- Integrar o uso de *containers* seguros para a execução das operações que lidam com dados médicos sensíveis;

- Analisar o impacto no desempenho do tempo de resposta sobre as operações do modelo de controle de acesso;

O resultado esperado deste trabalho é a identificação de etapas vulneráveis no modelo de (Seol et al., 2018) e a criação de uma solução viável para estas vulnerabilidades com *containers* seguros.

5 DETALHAMENTO

Esta seção contém o detalhamento da implementação da prova de conceito, assim como do servidor de dados médicos e do servidor de controle de acesso.

O objetivo desta prova de conceito é construir um ambiente mínimo que suporte a aplicação do PEP. Esta aplicação serve para se comunicar com o servidor de dados médicos, o servidor de controle de acesso e disponibilizar uma API para as requisições do serviço, como especificado na Figura 4.1.

O PEP pode ser executado normalmente ou dentro de um enclave SGX por meio de um *container* SCONE. Todos os demais serviços (dados médicos e o restante do controle de acesso) são executados fora de enclaves. Apesar destes componentes serem considerados seguros na arquitetura, seu desempenho e segurança são independentes do componente sendo avaliado. Em uma aplicação real estes componentes podem ser múltiplos e pertencentes à terceiros. Sendo assim, a forma como a segurança deles é atingida não faz parte do escopo deste trabalho.

5.1 ARQUITETURA DO SISTEMA

O sistema funciona com a integração e comunicação de três componentes principais, a aplicação SEHS (*Secure Electronic Health System*), o servidor de dados médicos EHRbase e o servidor de controle de acesso baseado em XACML WSO2.

O SEHS contém o componente de segurança XML e o PEP. Ele disponibiliza uma interface gráfica de usuário para auxiliar no gerenciamento de usuários e de dados médicos para os testes de desempenho e também uma API utilizada para os testes de desempenho que faz uso de todos os componentes críticos da arquitetura.

A Figura 5.1 apresenta a arquitetura do sistema implementado e o fluxo das requisições feitas pelo usuário. Primeiramente o usuário faz uma requisição para executar uma ação no SEHS (1), assim, o PEP consulta o servidor WSO2, que vai comparar a requisição feita com as políticas definidas no PAP e avaliar se esse usuário tem as permissões necessárias para acessar esse recurso (2). O servidor WSO2 envia a resposta ao PEP (3) que vai agir de acordo com essa resposta. Caso a permissão seja concedida, o PEP segue com a requisição para o EHRbase (4) que retorna a resposta dessa requisição para o PEP (5). O PEP então encaminha os dados para serem criptografados (6) e posteriormente enviados para o usuário (7). Caso a permissão seja negada, o PEP interrompe o fluxo da requisição para o EHRbase e imediatamente retorna um erro para o usuário (4 - tracejado).

5.2 EHRBASE

O módulo de dados médicos está representado como uma instância do EHRbase. O EHRbase foi escolhido como servidor de dados médicos por ser uma implementação de código aberto das especificações do OpenEHR para sistemas que utilizam EHRs. Sendo assim, a versão do EHRbase utilizada foi a última disponível no DockerHub, um serviço em nuvem para a criação e compartilhamento de imagens de *containers*.

Com o servidor do EHRbase disponível, é possível enviar requisições para essa API e realizar ações sobre os dados. Por padrão, o EHRbase já tem rotas específicas para a recuperação ou inserção de dados por meio de requisições HTTP e também tem disponível uma rota para executar consultas na base de dados usando a linguagem AQL, que faz uma requisição POST

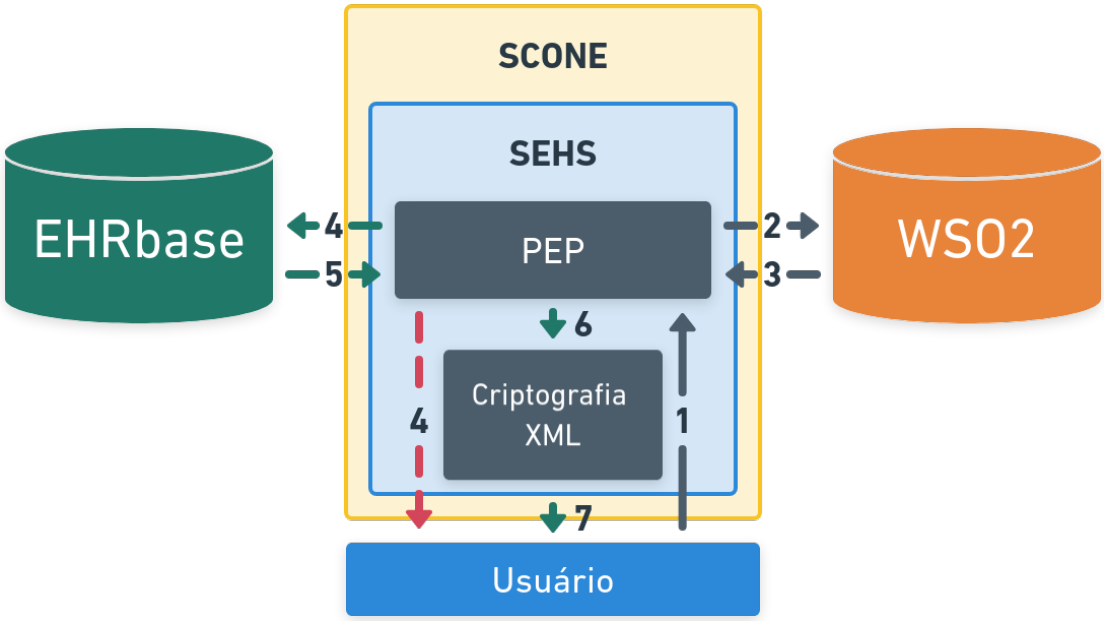


Figura 5.1: Arquitetura do sistema. Fonte: os autores (2022).

com a consulta a ser realizada, retornando o resultado dessa consulta em formato JSON. As rotas acessadas pelo SEHS podem ser observadas na Tabela 5.1.

Requisição	**Ação**	**Rota**
POST	Criar EHR	`/ehr`
POST	Carregar template	`/definition/template/adl1.4`
GET	Listar templates	`/definition/template/adl1.4`
GET	Criar um exemplo de composition para um template específico	`/definition/template/adl1.4/{template_id}/example`
POST	Criar composition	`/ehr/{ehr_id}/composition`
DELETE	Excluir composition	`/ehr/{ehr_id}/composition/{preceding_version_uid}`
POST	Executar consulta em linguagem AQL	`/query/aql`

Tabela 5.1: Rotas acessadas no servidor EHRbase. Fonte: os autores (2022).

Com essas rotas é possível realizar as ações básicas na aplicação SEHS e os dados são armazenados em um banco Postgres modificado especificamente para funcionar de acordo com as especificações do EHRbase, também disponibilizado pelo DockerHub. Esse banco de dados armazena todos os dados criados por meio dessas ações, assim como qualquer configuração criada para essa aplicação em especial.

O EHRbase possui um usuário comum e um usuário administrador. No SEHS, o usuário comum foi utilizado. As credenciais desse usuário são adicionadas ao cabeçalho de cada requisição feita para a API do EHRbase como uma autenticação do tipo *Basic Auth*, que requer um usuário e uma senha.

O *template* de *Vital signs* (sinais vitais) foi obtido pelo CKM em formato XML e foi adicionado à essa instância do EHRbase como exemplo de *template* a ser utilizado no sistema. Esse é o *template* que será usado para criar as *compositions* por um usuário que acessa o SEHS.

Ele é composto por vários arquétipos contendo os dados mais comuns de pressão sanguínea, massa corporal, altura, peso, pulsação cardíaca, respiração e oximetria de pulso. Um breve segmento desse *template* pode ser observado na Listagem 5.1.

```
<template>
    ...
    <archetype_id>
        <value>openEHR-EHR-OBSERVATION.body_temperature.v2</value>
    </archetype_id>
    < term_definitions  code="at0000">
      <items  id=" description "> A measurement of the  body temperature ,  which is  a
       surrogate  for  the  core  body temperature  of  the  individual .  </items>
      <items  id=" text ">Body temperature</items>
    </ term_definitions >
    < term_definitions  code="at0004">
      <items  id=" description ">The measured temperature .</items>
      <items  id=" text ">Temperature</items>
    </ term_definitions >
    ...
</template>
```

Listagem 5.1: Segmento do *template* de sinais vitais.

A Figura 5.2 mostra a relação entre os componentes EHR, *template* e *composition*. Com o *template* disponível no servidor do EHRbase, é possível criar *compositions* usando esse *template* como base e vinculá-las a EHRs existentes.

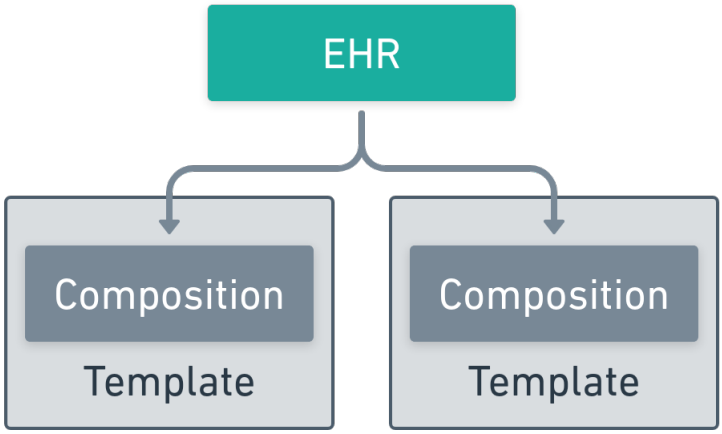


Figura 5.2: Relação entre EHRs, *templates* e *compositions*. Fonte: os autores (2022).

5.3 WSO2

O servidor de controle de acesso baseado em XACML utilizado foi a plataforma de código aberto WSO2 *identity server*. Esta plataforma foi escolhida porque implementa todos os recursos necessários para um sistema de controle XACML, conforme mostrado na seção 2.2.3, com exceção do PEP. Ela disponibiliza uma API web para que o PEP possa se comunicar com o PDP por meio de requisições XML. O WSO2 disponibiliza o PAP (*Policy Administration Point*) como uma interface web, permitindo o gerenciamento das políticas com maior facilidade. Além disso, ambos os sistemas RBAC e ABAC podem ser adotados. A estrutura de uma requisição XACML tem os seguintes componentes:

- **Sujeito** (*Subject*):
  - Identifica quem está fazendo a requisição
  - Ex: Médico, Enfermeiro
- **Recurso** (*Resource*):
  - Identifica qual é o alvo da requisição
  - Ex: Sinais vitais
- **Ação** (*Action*):
  - Identifica o que se quer fazer com o recurso
  - Ex: listar, modificar, criar
- **Ambiente** (*Environment*):
  - Identifica os atributos atuais da requisição
  - Ex: tempo, situação de emergência

Foram criadas políticas simplificadas para representar apenas os diferentes tipos de operações que serão testadas, conforme pode ser observado na Tabela 5.2. Requisições simples e com menos condições para o PDP resultam em menor tempo de processamento no servidor XACML e não causam diferença no tempo de execução do SEHS. Como o escopo é avaliar apenas o desempenho do SEHS, estas requisições mais simples foram utilizadas, reduzindo a carga de processamento dos servidores XACML e EHRbase.

Para controlar o acesso aos recursos do SEHS, foram criados três níveis de acesso de usuário, que podem ser definidos no momento do cadastro. Também foram criadas duas políticas de acesso para os recursos do servidor de dados médicos. Uma controla as permissões para as ações sobre os EHRs e outra controla as permissões para as ações sobre as *compositions*. A tabela 5.2 mostra as permissões por nível de acesso dos usuários.

Nível de acesso	**Permissões**
Nível 1	Listar EHRs
Nível 2	Nível 1 + Criar EHRs e listar *compositions*
Nível 3	Nível 2 + Criar e excluir *compositions*

Tabela 5.2: Níveis de permissões e ações do usuário. Fonte: os autores (2022)

A Listagem 5.2 contém um exemplo de requisição XACML que é enviado pelo PEP do SEHS para o PDP do servidor de controle de acesso. Podemos observar que os recursos, sujeitos e ações estão visíveis nas linhas 5, 13 e 21. Estas informações podem ser sensíveis, especialmente em sistemas ABAC, já que revelam os atributos relacionados a esse usuário. Portanto se demonstra necessário proteger os componentes que manipulam as requisições XACML dentro do sistema.

```
<Request xmlns=" urn:oasis:names:tc:xacml:3 .0:core:schema:wd-17" CombinedDecision="false"
    ReturnPolicyIdList =" false ">
  < Attributes  Category=" urn:oasis:names:tc:xacml:3 .0 :attribute - category:resource ">
    < Attribute  AttributeId =" urn:oasis:names:tc:xacml:1 .0 :resource:resource -id"
        IncludeInResult =" false ">
      < AttributeValue  DataType="http: // www.w3.org/2001/XMLSchema#string">
        resource
      </ AttributeValue >
    </ Attribute >
  </ Attributes >
  < Attributes  Category=" urn:oasis:names:tc:xacml:1 .0 :subject - category:access - subject ">
    < Attribute  AttributeId =" urn:oasis:names:tc:xacml:1 .0 :subject:subject -id"
        IncludeInResult =" false ">
      < AttributeValue  DataType="http: // www.w3.org/2001/XMLSchema#string">
        subject
      </ AttributeValue >
    </ Attribute >
  </ Attributes >
  < Attributes  Category=" urn:oasis:names:tc:xacml:3 .0 :attribute - category:action ">
    < Attribute  AttributeId =" urn:oasis:names:tc:xacml:1 .0 :action:action -id"
        IncludeInResult =" false ">
      < AttributeValue  DataType="http: // www.w3.org/2001/XMLSchema#string">
        action
      </ AttributeValue >
    </ Attribute >
  </ Attributes >
</Request>
```

Listagem 5.2: Modelo de requisição XACML utilizado para o controle de acesso.

5.4 SEHS

A prova de conceito foi implementada usando o *framework* Flask, baseado em Python, para criar um *front-end* que consome as APIs de dados médicos e de controle de acesso. Esse sistema consiste em um conjunto de páginas que permite a visualização dos dados médicos de acordo com o nível de acesso do usuário.

Para acessar as páginas de dados, um cadastro é realizado para obter credenciais do usuário como nome, e-mail, senha e nível de acesso. O sistema usa uma base de dados SQLite para armazenar os dados desse cadastro e a modelagem da tabela de usuários pode ser observada na Figura 5.3.

As páginas que possuem ações a serem realizadas são protegidas e requerem que o usuário faça o *login* para poder acessá-las. Com o usuário autenticado, é possível acessar essas páginas e realizar interações com o servidor de dados médicos. As páginas contém as seguintes ações disponíveis para os usuários: listagem de EHRs, criação de EHRs, listagem de *compositions*, criação de *compositions* e exclusão de *compositions*, conforme apresentado na Tabela 5.2. As Figura 5.4 mostra as páginas de listagem de EHRs (à esquerda) e de listagem de *compositions* (à direita) de um determinado EHR.

Ao realizar uma requisição e a permissão ser negada, um aviso de alerta é mostrado para o usuário, explicitando que ele não tem permissão para realizar essa ação sobre esse recurso. Caso a permissão seja concedida, o fluxo da requisição é concluído, podendo também apresentar um aviso de sucesso para o usuário dependendo da ação, conforme a Figura 5.5, que tem como

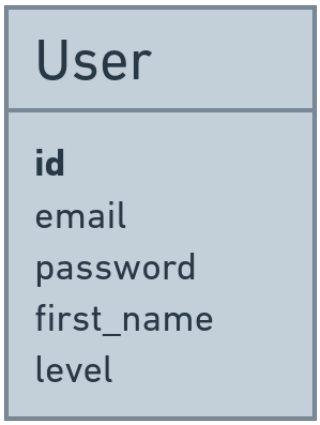


Figura 5.3: Esquema do usuário no banco de dados do SEHS. Fonte: os autores (2022).

exemplo as mensagens de sucesso (em verde) e de erro (em vermelho) para a ação de criar *compositions*.

5.4.1 PEP

O PEP, ou *Policy Enforcement Point*, é o componente do XACML responsável por executar as decisões de controle de acesso. Este componente é implementado no SEHS e é o principal componente que deve ser protegido pelo enclave.

A razão pela qual esse componente deve ser protegido é a natureza dos dados que devem ser processados por ele. Requisições em um sistema ABAC contém atributos com informações sensíveis de usuários que devem ser processados pelo PEP. Estas informações são enviadas para o PDP do servidor de controle de acesso e uma resposta é recebida. Baseado nesta resposta,

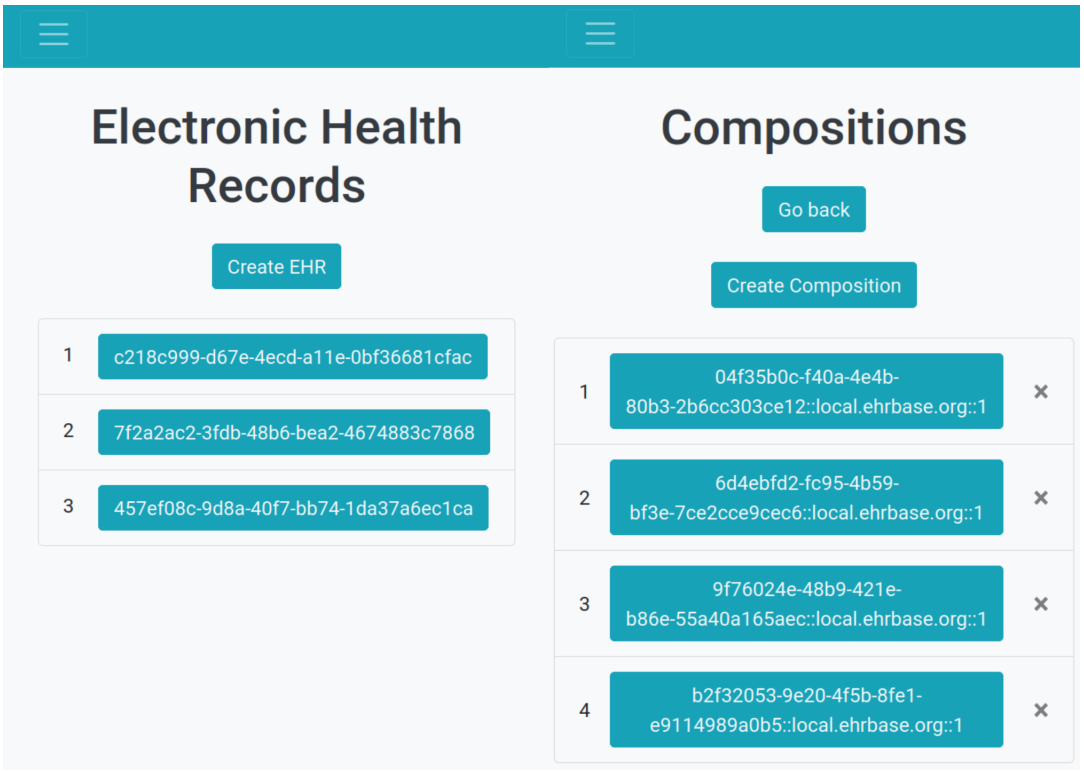


Figura 5.4: Página de listagem de EHRs e de *compositions*. Fonte: os autores (2022).

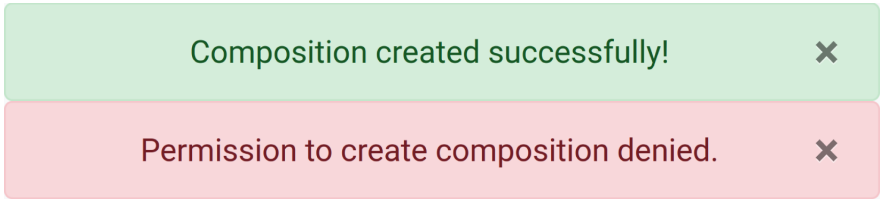


Figura 5.5: Mensagens mostradas ao usuário. Fonte: os autores (2022).

também é dever do PEP requisitar e transferir o recurso requisitado. Nesse caso, os recursos acessados são dados sensíveis de saúde. Por isso é importante procurar formas de garantir maiores níveis de segurança em sua execução.

No SEHS, o PEP é implementado nas *views* da API. Uma requisição XACML é enviada ao servidor WSO2 e os recursos só serão consultados caso a permissão seja recebida.

5.4.2 Criptografia XML

O componente de criptografia XML, também chamado de segurança XML, tem o papel de receber os recursos processados pelo PEP e criptografá-los antes de seu envio ao usuário.

Este componente é importante para garantir a segurança dos recursos fora do sistema, porém também é possível observar que ele deve ser incluído no enclave, por lidar com dados sensíveis dentro da sistema.

Ao incluir este componente no enclave é possível substituí-lo pelos recursos de criptografia SGX para garantir a integridade e realizar a atestação dos componentes XML das requisições. Porém, em casos onde os dados devem transitar por instituições intermediárias ainda é necessário implementar uma camada de criptografia para a comunicação direta com o usuário final. Para o propósito da prova de conceito, nestas situações os dados são criptografados pelo algoritmo AES com uma chave de 128 bits.

5.4.3 API

O *front-end* do SEHS, apesar de útil para o gerenciamento dos recursos de saúde, não deve fazer parte do enclave em uma aplicação real, já que serve somente para a visualização dos dados e interação com o usuário. A interface utilizada nos testes é uma API web que recebe requisições em XML.

O usuário da API, no contexto de interoperabilidade entre instituições de saúde, são aplicações de terceiros, ou uma aplicação interna que é separada do enclave.

5.5 SCONE

O SCONE fornece uma forma fácil de tornar uma aplicação existente executável em um enclave de memória SGX. Usando um *container Docker* como base, uma imagem curada específica para aplicações Python baseada na distribuição *Alpine Linux* foi utilizada. Quando o SEHS é executado fora do enclave para o propósito de testes, a mesma imagem *Docker* foi usada, com poucas modificações.

Ao construir a imagem do *container*, o código do SEHS é copiado sem alterações. Apenas o SEHS foi incluído na imagem do SCONE. Portanto, os recursos protegidos pelo enclave SGX são o PEP, o componente de segurança XML e a API.

6 EXPERIMENTAÇÃO E VALIDAÇÃO

A finalidade da experimentação é analisar qual é o impacto de desempenho no SEHS quando executado dentro de um *container* seguro. O objetivo é determinar se o encapsulamento desse sistema com o componente de segurança XML e o PEP em um enclave SGX tem viabilidade em uma aplicação real.

Para realizar os testes foram separadas três operações principais: criar *compositions*, listar *compositions* e excluir *compositions*. Todas estas operações são compostas por uma requisição ao servidor de controle de acesso, seguida de uma requisição ao servidor de dados médicos. A etapa de criptografia XML é executada no SEHS na estrutura de dados XML que será enviada como resposta ao usuário.

O tempo de cada requisição é calculado internamente pelo sistema. O tempo de execução da operação em segundos é enviado junto à resposta da API. Para calcular o tempo médio das operações e reduzir a variância dos resultados, 1.000 requisições foram realizadas e a média dos valores pode ser observada na tabela 6.1.

6.1 *HARDWARE*

Os testes foram realizados em uma máquina que continha as seguintes configurações de *hardware*:

- **CPU**: Intel® Core™ i5-8250U @ 3.40GHz
  - 4 núcleos
  - 2 *threads* por núcleo
  - Tamanho da linha de cache: 64B
  - Suporte para SGX
- **RAM**: 7720.39 MB
- **Kernel**: Linux 5.16.12-arch1-1

A execução dos testes foi realizada sem o uso de interface gráfica e com o uso mínimo de outros serviços no sistema. As funcionalidades de controle de frequência da CPU foram desativadas na BIOS.

6.2 RESULTADOS

Na tabela 6.1 é possível observar que o acréscimo de tempo gerado pelo *container* seguro é proporcionalmente pouco significante. A exceção é a operação de criptografia XML que é cerca de 1,7 vezes mais custosa. Ainda assim, este custo é pequeno quando comparado ao tempo de execução íntegro de uma operação.

Operação	Tempo médio sem SCONE	Tempo médio com SCONE
Criar *composition*	0.401s	0.472s
Listar *compositions*	0.164s	0.186s
Excluir *composition*	0.166s	0.203s
Criptografia XML	0.027s	0.046s

Tabela 6.1: Comparação de tempo médio em segundos das operações. Fonte: os autores (2022).

Analisando os resultados podemos inferir que o tempo das requisições aos servidores de dados médicos e de controle de acesso dominam o tempo total das operações. O custo da criptografia XML aumenta consideravelmente, porém, essa operação ocorre apenas dentro do enclave, não adicionando um custo de comunicação ainda maior. Como o custo das operações no enclave é baixo, a queda de desempenho total também é baixa.

6.3 LIMITAÇÕES

Os testes realizados mostram o tempo decorrido de uma única requisição ao servidor por vez. Na realidade, problemas de desempenho são exacerbados quando um grande volume de requisições é realizado simultaneamente. É possível que a proporção de trabalho do SEHS em relação aos demais serviços seja invertida em situações de muito tráfego. Para descartar esta possibilidade, será preciso realizar mais estudos.

6.4 ANÁLISE DE SEGURANÇA

O modelo propõe a inclusão de componentes em um enclave SGX. Estes componentes se tornam protegidos contra ataques privilegiados enquanto estão em execução, o que aumenta consideravelmente a segurança dos dados sendo processados. Porém somente esta mudança não garante completamente a segurança do sistema.

Um sistema completo deve conter mecanismos para garantir que apenas o código original possa ser carregado no enclave. Além disso, as mensagens enviadas pelo SEHS devem conter uma assinatura que comprovem a autenticidade do sistema em execução no servidor seguro.

É possível que um sistema que implemente estes mecanismos tenha limitações de desempenho não previstas nos testes.

7 CONCLUSÃO

A necessidade de uma interoperabilidade de dados entre instituições de saúde traz muitos desafios. Padrões como o OpenEHR oferecem um ótimo caminho para o desenvolvimento de novas aplicações que satisfazem essa necessidade. Porém, a interoperabilidade também implica em mais desafios de segurança, especialmente com a sensibilidade de dados de saúde (Evans, 2016).

Aplicações de dados médicos que têm a capacidade de atestar a validade e segurança de seus dados fornecem maior confiabilidade para implementação de sistemas de interoperabilidade (Kruse et al., 2017). Tecnologias de *hardware* seguro como o SGX podem ser aplicadas com o propósito de fornecer esta confiabilidade.

Sistemas de controle de acesso também fazem parte do desafio de integrar sistemas de dados médicos, mas não são abordados por especificações como o OpenEHR, que se limita aos dados de saúde. Também é possível perceber que a troca de informações de atributos em sistemas ABAC imputam em um novo desafio de segurança.

Neste trabalho foi estudada a possibilidade de encapsular componentes que lidam com dados de saúde e de controle de acesso em um enclave SGX para garantir a segurança destes dados no caso de um eventual comprometimento do servidor. A arquitetura é baseada no trabalho de (Seol et al., 2018), com o uso de XACML para o controle de acesso e um componente de segurança XML.

Com este estudo foi verificado que é possível incluir o *Policy Enforcement Point* de uma aplicação de controle de acesso XACML para dados médicos em um enclave sem impactos muito significativos de desempenho. Para aplicações de dados médicos que lidam com maiores volumes de dados, estudos com sistemas mais robustos são necessários para identificar possíveis novos gargalos. Neste trabalho foi possível inferir que a utilização de enclaves SGX para a proteção de dados em contexto de interoperabilidade de sistemas médicos tem viabilidade e há justificativa para realizar estudos mais compreensivos na área.

REFERÊNCIAS

Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'Keeffe, D., Stillwell, M. L., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P. e Fetzer, C. (2016). SCONE: Secure linux containers with intel SGX. Em *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, páginas 689–703, Savannah, GA. USENIX Association.

Bacelar, G. e Correia, R. (2015). Manual de introdução à norma OpenEHR. *OpenEHR Academy*.

Calvillo-Arbizu, J., Román-Martínez, I. e Roa-Romero, L. M. (2014). Standardized access control mechanisms for protecting iso 13606-based electronic health record systems. Em *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, páginas 539–542.

Chung, Ferraiolo, D., Kuhn, D., Schnitzer, A., Sandlin, K., Miller, R. e Scarfone, K. (2019). Guide to attribute based access control (abac) definition and considerations.

Coppolino, L., D'Antonio, S., Mazzeo, G., Romano, L. e Sgaglione, L. (2018). Exploiting new cpu extensions for secure exchange of ehealth data at the EU level. Em *2018 14th European dependable computing conference (EDCC)*, páginas 17–24. IEEE.

Costan, V. e Devadas, S. (2016). Intel sgx explained. *Cryptology ePrint Archive*.

De la Rosa Algarín, A., Demurjian, S. A., Ziminski, T. B., Sánchez, Y. K. R. e Kuykendall, R. (2016). Securing XML with role-based access control: Case study in health care. Em *E-Health and Telemedicine: Concepts, Methodologies, Tools, and Applications*, páginas 487–522. IGI Global.

EHRbase (2022). What is EHRbase? `https://ehrbase.org/about-ehrbase/`. Acessado em 02/05/2022.

Evans, R. S. (2016). Electronic health records: then, now, and in the future. *Yearbook of medical informatics*, 25(S 01):S48–S61.

FreshEHR Clinical Informatics (2018). OpenEHR templates in detail. `https://www.slideshare.net/freshehr/2-1-openehr-templates-in-detail-110661738`. Acessado em 01/05/2022.

Kruse, C. S., Smith, B., Vanderlinden, H. e Nealand, A. (2017). Security techniques for the electronic health records. *Journal of medical systems*, 41(8):1–9.

Meredith, J. (2020). Defining OpenEHR... simply! `https://www.archetextur.es/defining-openehr-simply/`. Acessado em 01/05/2022.

OASIS (2010). *eXtensible Access Control Markup Language (XACML) Version 3.0 - Committee Specification 01*.

OpenEHR Foundation (2022a). Archetype query language (AQL). `https://specifications.openehr.org/releases/QUERY/latest/AQL.html`. Acessado em 02/05/2022.

OpenEHR Foundation (2022b). Clinical models program. https://openehr.org/programs/clinicalmodels/. Acessado em 01/05/2022.

OpenEHR Foundation (2022c). Ehr information model. https://specifications.openehr.org/releases/RM/latest/ehr.html#_compositions. Acessado em 07/09/2022.

OpenEHR Foundation (2022d). OpenEHR clinical knowledge manager. https://ckm.openehr.org/ckm/archetypes/1013.1.3574/mindmap. Acessado em 11/05/2022.

OpenEHR Foundation (2022e). What is OpenEHR? https://openehr.org/about/what_is_openehr. Acessado em 01/05/2022.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L. e Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Seol, K., Kim, Y.-G., Lee, E., Seo, Y.-D. e Baik, D.-K. (2018). Privacy-preserving attribute-based access control model for xml-based electronic health record system. *IEEE Access*, 6:9114–9128.