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

ARIEL ROQUE INÁCIO LUZ

SEML: UM PROCESSO PARA EXECUÇÃO SEGURA DE APRENDIZAGEM DE MÁQUINA EM NUVEM

CAMPINA GRANDE - PB

2023

ARIEL ROQUE INÁCIO LUZ

SEML: UM PROCESSO PARA EXECUÇÃO SEGURA DE APRENDIZAGEM DE MÁQUINA EM NUVEM

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

Orientador: Professor Dr. Andrey Elísio Monteiro Brito.

CAMPINA GRANDE - PB

2023

ARIEL ROQUE INÁCIO LUZ

SEML:UM PROCESSO PARA EXECUÇÃO SEGURA DE APRENDIZAGEM DE MÁQUINA EM NUVEM

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

BANCA EXAMINADORA:

Professor Dr. Andrey Elísio Monteiro Brito

Orientador – UASC/CEEI/UFCG

Professor Dr. Cláudio de Souza Baptista

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de fevereiro de 2023.

CAMPINA GRANDE - PB

RESUMO (ABSTRACT)

Machine learning algorithms are crucial to the development of software used in our daily lives. In the face of the large amount of data and the growing need for computational data to train these algorithms, modern computing has migrated to the cloud. On the other hand, there are concerns regarding the security of the data that travel in this transmission medium. This work aims to develop a process that ensures the confidentiality of machine learning applications using reliable execution technologies.

SEML: um processo para execução segura de aprendizado de máquina em nuvem

Ariel Roque Inácio Luz
Universidade Federal de Campina Grande - UFCG
Campina Grande, Brasil
ariel.luz@ccc.ufcg.edu.br

Andrey Brito
Universidade Federal de Campina Grande - UFCG
Campina Grande, Brasil
andrey@computacao.ufcg.edu.br

RESUMO

Algoritmos de aprendizado de máquina são cruciais para o desenvolvimento dos softwares utilizados em nosso cotidiano. Frente à grande quantidade de dados e a necessidade crescente de recursos computacionais para treinar esses algoritmos, a computação moderna tem migrado para a nuvem. Por outro lado, existem preocupações a respeito da segurança dos dados que trafegam nesse meio. Este trabalho tem como objetivo o desenvolvimento de um processo que garanta confidencialidade de aplicações de aprendizado de máquina utilizando tecnologias de execução confiável.

KEYWORDS

machine learning, confidential computing, security

1 INTRODUÇÃO

Os algoritmos de aprendizado de máquina (em inglês, *machine learning* - ML) estão cada vez mais presentes no cotidiano das pessoas, desde a recomendação de séries [2], como para identificação de doenças malignas (p.ex., câncer de mama [16]). Para que isso fosse possível, é necessário uma grande quantidade de informações de maneira que os dados sejam treinados e possam oferecer acurácia nos resultados.

Em meio a necessidade de recursos computacionais para o treinamento desses algoritmos, tornou-se crescente a demanda por computação em nuvem por oferecer tantos recursos (processamento, armazenamento e serviços) quanto forem necessários sem a necessidade de preocupação com a infraestrutura utilizada. Tendência essa que só aumenta e que de acordo com uma pesquisa realizada pela Gartner [6], onde estima-se que mais de 85% das empresas usarão computação em nuvem até 2025.

Por outro lado, a utilização de recursos em nuvem levanta preocupações a respeito da confidencialidade e privacidade das informações que circulam nesse ambiente, principalmente, no tratamento dos dados sensíveis no âmbito de conformidade com a legislação de países, como a Lei Geral Proteção dos Dados Pessoais (LGPD) no Brasil e a *General Data Protection Regulation* (GDPR) na Europa.

Para lidar com esses problemas, desenvolveu-se algumas abordagens do estado da arte para a proteção dos dados, tais como: privacidade diferencial (do inglês, *differential privacy* – DF), encriptação homomórfica (do inglês, *homomorphic encryption* – HE) e aprendizado federado (do inglês, *federated learning* – FL).

A privacidade diferencial é uma técnica que consiste em adicionar ruído aos dados a fim de anonimizar as informações. Seu ponto negativo reside na redução da utilidade dos dados ou da acurácia devido ao ruído introduzido pelo processo. Enquanto isso, a encriptação homomórfica visa realizar operações matemáticas em dados criptografados sem a necessidade de decifrá-los, essa técnica esbarra no número limitado de operações que podem ser realizadas ou no alto custo de processamento quando esquemas HE-completos são utilizados. Alternativamente, o aprendizado federado permite que as partes treinem modelos de ML e depois agreguem os modelos parciais colaborativamente, sem o compartilhamento dos dados. No entanto, essa abordagem possui brechas que podem ser exploradas para extrair informações sensíveis do modelo, como demonstrado em trabalhos recentes [5, 10, 17].

Neste trabalho, nós propomos o Secure Enviroment for Machine Learning (SEML), um processo para execução segura de aprendizado de máquina em nuvem utilizando um ambiente de execução confiável (do inglês, *Trusted Execution Enviroment* – TEE). TEEs têm sido um grande candidato para permitir a preservação da confidencialidade e integridade na nuvem, visto que, possibilitam a execução protegida de código em ambientes inseguros, trazendo acurácia comparáveis a versões desprotegidas ao preço de um aumento tolerável no processamento.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 apresentamos conceitos e tecnologias utilizadas. Os principais trabalhos relacionados são apresentados na Seção 3. O modelo de ameaça que assumimos é detalhado na Seção 4. Na Seção 5 apresentamos o SEML, sua arquitetura e seu fluxo de execução. Aspectos de segurança do SEML são analisados na Seção 6. Na Seção 7 descrevemos os experimentos realizados junto dos resultados e discussões. Por fim, na Seção 8, sumarizamos as conquistas atingidas e os próximos passos.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são introduzidos alguns conceitos e tecnologias importantes para entender o contexto do trabalho apresentado.

2.1 Aprendizado de Máquina

O aprendizado de máquina é uma área da inteligência artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática [11]. As técnicas de ML podem ser divididas, de maneira geral, em aprendizado supervisionado e não supervisionado.

O aprendizado supervisionado é quando o modelo busca saídas para os dados de entrada, para auxiliá-lo são utilizados pares de dados entrada/saída já conhecidos para que o algoritmo seja treinado [12]. Em contraste, no aprendizado não supervisionado não há conhecimento prévio sobre a saída esperada e nem rótulos que permitam identificar o que é cada informação, cabe ao algoritmo agrupar os dados de acordo com semelhanças, encontrando estruturas e padrões nos dados fornecidos.

2.2 Scikit-learn

É uma das bibliotecas de aprendizagem de máquina mais populares para a linguagem de programação Python. O nome da biblioteca vem do termo "SciKit" (SciPy Toolkit), uma extensão da biblioteca SciPy desenvolvida por terceiros.

A Scikit-learn foi criada para ser fácil de utilizar e atender a demanda de desenvolvedores de todos os níveis, dentre os algoritmos suportados estão: aprendizado supervisionado, não supervisionado e dimensionamento de dados. Através de uma extensa documentação e uma comunidade ativa, a biblioteca torna-se confiável para ser aplicada em variados projetos de ML.

2.3 Kubernetes

Com a demanda crescente para a utilização de contêineres, foram desenvolvidas soluções que pudessem gerenciá-los de forma facilitada em ambientes de produção. Essas ferramentas são chamadas de orquestradores e têm como objetivo automatizar tarefas de implantação, atualização, controle de acesso e proporcionar algumas funcionalidades como escalabilidade, tolerância a falhas e balanceamento de carga [1].

Kubernetes é um orquestrador de contêineres de código aberto criado em 2015 pelo Google através das experiências obtidas com os orquestradores internos da companhia chamados de Borg e Omega [4]. Desde então vem sendo utilizado como ferramenta padrão do mercado para ambientes em produção graças a suas poderosas abstrações que permitem facilitar a implantação em nuvem.

A ferramenta conta ainda com a capacidade de replicar contêineres a fim de aumentar a disponibilidade da aplicação hospedada. Quando um contêiner falha, o Kubernetes elimina o contêiner faltoso e instancia outro a partir de uma imagem [13].

2.4 Rclone

É um software de código aberto inspirado no Rsync[1] e escrito em Go que permite gerenciar arquivos em armazenamentos de nuvem de forma fácil. Rclone[2] oferece as funcionalidades de copiar, mover e sincronizar arquivos na nuvem com e sem criptografia utilizando a linha de comando ou uma interface gráfica ainda experimental. Ele pode ser usado tanto para a sincronização entre máquinas locais e a nuvem, como entre diferentes provedores de nuvem, suportando mais de 40 provedores de armazenamento, incluindo Amazon S3, Google Cloud Storage, Microsoft Azure Blob Storage e Nextcloud.

Quando os arquivos são criptografados, a criptografia é executada localmente na máquina que executa o comando. A ferramenta utiliza o módulo Crypto que utiliza o algoritmo NaCl SecretBox, baseado no XSalsa20 cipher e Poly1305, para proteger os dados. De forma a garantir a privacidade da informação, é possível encriptar o diretório e nome do arquivo a ser submetido.

[1]https://rsync.samba.org/
[2]https://rclone.org/

2.5 MinIO

MinIO é um software de código aberto para armazenamento de alta performance de objetos não estruturados, sendo ideal para aprendizado de máquina, análise de dados e outras áreas com cargas de trabalho intensas. É compatível com o Amazon S3, podendo ser usada como alternativa a solução da Amazon ou como um *gateway* para o S3.

A solução é fácil de ser instalada, estando presente em uma variedade de sistemas operacionais e é compatível com gerenciadores de contêiners como Kubernetes e Docker Swarm, contribuindo para melhor integração na nuvem. Além disso, o MinIO suporta TLS mútuo (mTLS) e oferece duas opções para encriptaçao dos dados, uma usando chave secreta provida por um cliente S3 ou uma chave secreta gerenciada por um KMS (do inglês, *Key Management Service*).

2.6 Intel SGX

Intel Software Guard eXtensions (SGX) consiste em um TEE equipado em processadores Intel a partir da arquitetura Skylake. TEEs garantem que todo o código executado fora do TEE, o que inclui o sistema operacional e o hipervisor, não comprometam a integridade de aplicações executadas dentro do enclave.

A tecnologia Intel SGX permite as aplicações serem executadas em regiões protegidas pela CPU (*enclaves*). Para realizar isso, a tecnologia reserva uma memória segura chamada de EPC (do inglês, *Enclave Page Cache*) para o código e os dados presentes no enclave, impedindo que qualquer software de fora acesse essa memória.

Atualmente, os processadores Intel mais comuns suportam no máximo 128 MB de memória EPC, das quais apenas 93.5 MB é utilizável pelos enclaves SGX. Versões mais recentes de processadores suportam até 1 TB de EPC, no entanto, ainda são pouco comuns. De modo a suportar aplicações que necessitam de grandes quantidades de memória, o kernel Linux permite um mecanismo de paginação para a troca de paginas entre o EPC e a memória regular. Esse mecanismo de paginação é custoso (cerca de 40 mil ciclos), pois requer chamadas do sistema operacional, cópia de páginas, atualizações para árvores de integridade e metadados, etc. [15]

Para desenvolver aplicações compatíveis com SGX, a Intel oferece o SGX Linux SDK. De acordo com essa documentação[3] , consiste em uma coleção de APIs, bibliotecas, documentações, códigos de exemplo e ferramentas para construir aplicações SGX em C/C++, o que dificulta desenvolvedores de outras linguagens criarem aplicações compatíveis.

[3]https://01.org/intel-software-guard-extensions

2.7 SCONE Framework

Secure CONtainer Enviroment (SCONE) é um arcabouço que permite que aplicações encapsuladas em contêineres possam ser migradas de forma transparente para executar dentro dos enclaves SGX. Isso possibilita contornar o problema da restrição do SGX Linux SDK discutida acima e reduz consideravelmente a curva de aprendizado para a adequação do código.

SCONE suporta uma variedade de linguagens de programação (como Python, Go, Java, R, Fortran) e conta com um conjunto de ferramentas, incluindo desde um compilador GCC modificado a bibliotecas do sistema, como `glibc` ou `must-libc`, para compilar aplicações existentes e executá-las em um enclave SGX.

```
name: sessao-exemplo
version: 0.3
services:
  - name: ola-mundo
    mrenclaves: [41f0117a3c62966b48ef6e2388b5fe7ff]
    image_name: ola-mundo
    command: python3 main.py
    pwd: /
images: ola-mundo
  - name:
    volumes:
      - name: volume-protegido
        path: /data
    injection_files:
      - path: /main.py
        content: |
          print("Ola, mundo SCONE")
```

Código 1: Exemplo de arquivo sessão SCONE

Somado a isso, o arcabouço suporta a funcionalidade de atestação remota. A atestação remota é um processo que permite que um sistema remoto determine o nível de confiança e integridade da plataforma, isso possibilita qualificar os mecanismos de proteção de segurança de hardware e software que um sistema inclui para proteger a confidencialidade, disponibilidade e integridade dos dados [8].

Para a atestação remota utiliza-se o componente CAS (do inglês, *Configuration and Atestation Service*) e o componente LAS (do Inglês, *Local Attestation Service*). O CAS é responsável por prover segredos e configurações de forma segura para a aplicação. Já o LAS é o responsável por atestar localmente o enclave, gerando um documento que é entregue ao CAS para ser checado, caso as informações estejam em conformidade, o CAS entrega os segredos e configurações.

Toda as aplicações novas são registradas no CAS por meio de sua API através de manifestos no formato YAML. A identidade do enclave esperada é descrita (MREnclave), e segredos e configurações são definidos de forma que estes segredos são entregues somente se o enclave tiver a identidade esperada e a plataforma (incluindo o processador) tiver as propriedades esperadas (p. ex., versão do firmware). Esses manifestos são chamados de arquivos de sessão (Código 1).

Outra funcionalidade oferecida pelo SCONE é atestação e criptografia do sistema de arquivos através do SCONE FSPF (do Inglês, *File System Protection File*) e volumes. O FSPF permite ao desenvolvedor criar regiões do sistema de arquivos que podem ser autenticadas e criptografadas. Uma das vantagens é que o gerenciamento de chaves e o processo de criptografia são feitos pelo *runtime* do SCONE, com o auxílio do CAS, o que torna a criptografia transparente para as aplicações. Volumes por sua vez, permitem definir regiões criptografadas do sistema de arquivos que podem ser montadas no sistema de arquivos de aplicações SCONE. Um volume (*volumes*, no arquivo de sessão acima) pode ser compartilhado entre várias aplicações, ou mesmo importado por aplicações definidas em sessões diferentes. [3]

3 TRABALHOS RELACIONADOS

Existem alguns trabalhos na literatura que focam na execução protegida de aplicações de aprendizado de máquina utilizando TEEs. Dentre esses trabalhos, podemos destacar:

- **Privacy-Preserving Machine Learning in Untrusted Clouds Made Simple** [14] que utiliza a biblioteca open-source Graphene aliada a tecnologia Intel SGX para execução protegida da biblioteca de ML PyTorch.
- **Plinius: Secure and Persistent Machine Learning** [18] se intitula como o primeiro framework ML seguro que integra Intel SGX com memória persistente (Persistent Memory - PM) para tolerância de falhas. Para atingir o objetivo proposto, os autores desenvolveram uma implementação SGX da biblioteca Romulus[4] para gerenciar a memória persistente e outra do framework ML Darknet.
- **Chiron: Privacy-preserving Machine Learning as a Service** [7] é um framework ML que utiliza Intel SGX juntamente com Ryon Sandbox para proteger o código a ser utilizado e controlar todas interações com o ambiente externo. Por desempenho, utiliza o Theano framework como ferramenta de aprendizado de máquina.
- **TensorSCONE: A Secure TensorFlow Framework using Intel SGX** [9] consiste em uma implementação da biblioteca ML TensorFlow para execução em enclaves SGX através do framework SCONE.

Com relação aos trabalhos acima apresentados, podemos destacar pontos fortes como as funcionalidades de tolerância a falhas, a utilização de bibliotecas de código aberto como a Graphene para a construção da solução e, por fim, a execução transparente possibilitada pelo framework SCONE que reduz a complexidade para execução de ambientes seguros.

Por outro lado, essas mesmas soluções apresentam arquiteturas engessadas a uma determinada ferramenta de ML. Isso obriga os usuários a adaptarem suas aplicações nativas a essas ferramentas para obterem a devida proteção, o que muitas vezes se torna proibitivo a depender do custo para a reescrita.

4 MODELO DE AMEAÇA

O modelo de ameaça proposto considera que o atacante pode ser um administrador curioso/malicioso ou um invasor que assumiu que o controle do sistema operacional e/ou hipervisor adicionando rotinas no sistema. Nesse contexto, o atacante poderia tentar manipular as aplicações a serem executadas, bem como, obter os dados sensíveis através da leitura do disco ou mesmo analisando a memória principal.

Somado a isso, consideramos que o atacante pode interceptar as requisições trocadas entre as aplicações SGX e o armazenamento em nuvem. No entanto, assumimos que o adversário não pode quebrar as criptografias simétricas e assimétricas utilizadas nas conexões, que obedecem as recomendações mais recentes sobre algoritmos e tamanhos de chave.

Assumimos que os processadores com Intel SGX, o código executado dentro do enclave e o conjunto de ferramentas fornecido pelo framework SCONE são livre de bugs. Por fim, ataques de canal

[4] https://github.com/anonymous-xh/sgx-romulus

lateral e ataques físicos ao processador estão fora do escopo deste trabalho assim como estão fora do modelo de ameaça do Intel SGX. Estratégias para mitigação de ataques de canais laterais podem ser aplicadas de forma ortogonal.

Em nossa solução, a arquitetura proposta deve lidar com um ambiente adverso, considerando que a infraestrutura pode estar comprometida e cujos operadores estão interessados em comprometer também a integridade das aplicações ou a privacidade dos dados.

5 SOLUÇÃO

Nesta seção, é apresentado o SEML, sua arquitetura e fluxo de execução.

5.1 Visão Geral

O Secure Enviroment for Machine Learning consiste em um processo criado para permitir a execução protegida de aprendizado de máquina em nuvem independente de ferramenta ML através de um TEE baseando em hardware.

Em nosso trabalho, escolhemos a tecnologia Intel SGX como TEE devido a sua confiabilidade e crescente adoção pela industria. Somado a isso, dada a dificuldade natural de portar aplicações para executarem em enclaves SGX, escolheu-se o framework SCONE por oferecer uma camada que permite usufruir das funcionalidades do Intel SGX com suporte a um grande número de linguagens de programação. Esta combinação possibilita ao SEML oferecer uma solução que seja integrável com outros frameworks de ML disponíveis no mercado.

5.2 Arquitetura

O design do SEML tem como aspecto central reduzir o esforço dos usuários compatibilizarem suas aplicações de ML existentes para executarem de forma segura na nuvem. A arquitetura da ferramenta é mostrada na Figura 1 e possui os seguintes componentes:

- **Rclone SGX** é a versão SGX distribuida pelo SCONE do software Rclone que é responsável por sincronizar os dados de forma criptografada de e para o armazenamento em nuvem. No primeiro momento, sincroniza o *dataset* a ser treinado na nuvem e permite que seja obtido pela ferramenta de aprendizado de máquina. Por fim, após o modelo ser gerado, sincroniza o modelo em nuvem para que possa ser obtido pelos usuários.
- **Armazenamento em Nuvem** é responsável por armazenar os dados de treinamento e o modelo produzido pelo SEML ML Provider.
- **SEML ML Provider** é o software portado para Intel SGX utilizando SCONE que executa na nuvem toda a computação de aprendizado de máquina de forma protegida. Dada a natureza dinâmica da arquitetura do SEML, é possível utilizar frameworks de ML utilizados no mercado, tais como TensorFlow, Scikit-learn e PyTorch, sem dificuldades.
- **SEML Model Runner** é o software portabilizado para Intel SGX utilizando SCONE que permite os usuários executarem o modelo gerado de forma protegida em suas máquinas.
- **SCONE Volume** é o componente SCONE responsável por prover um armazenamento criptografado em disco. Todo processo de criptografia é transparente para a aplicação, isto é, a aplicação não sabe que a região é criptografada, pois a *runtime* SCONE criptografa e descriptografa os dados de forma transparente para a aplicação autorizada.

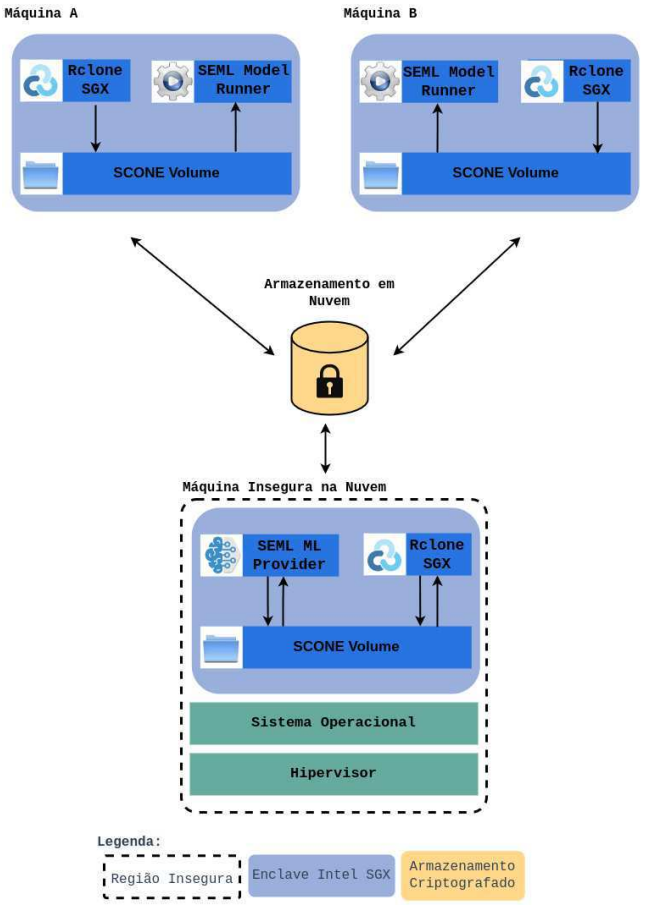


Figura 1: Arquitetura do SEML.

5.3 Fluxo de Execução

Nesta seção, descrevemos o fluxo de execução do SEML (Figura 3) através de duas partes interessadas (usuário A e usuário B) que desejam criar o melhor modelo ML possível e, portanto, decidem combinar os dados de forma segura para a produção de um modelo usando recursos de computação em nuvem. Devido a estratégias de negócios e compliance, nenhuma das partes devem ter acesso aos dados do outro, cabendo ao software protegido acessar as informações confidenciais e realizar a tarefa de forma independente.

Antes de executarmos os componentes do SEML, é preciso a criação de sessões SCONE entre as partes (Figura 2). As sessões serão responsáveis por guardar segredos (p. ex., chaves do armazenamento em nuvem), compartilhar volumes e determinar o código a ser executado pelas aplicações SCONE. Ambas as partes devem concordar com as sessões previamente, antes do inicio da execução. Para isso, as sessões podem ser divididas em partes públicas que definem, por exemplo, o código que será executado e o local onde os

dados e modelos serão armazenados, e, partes privadas que definem os segredos, como as chaves usadas na criptografia dos dados.

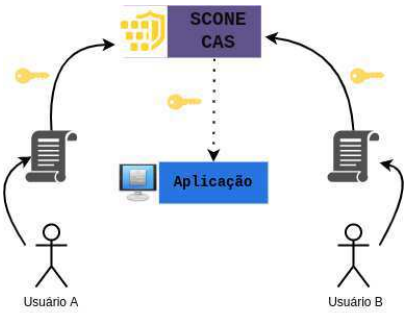


Figura 2: Criação das sessões SCONE

Nos passos ① a ④, detalhados na Figura 3, cada uma das partes executa a aplicação Rclone SGX em suas respectivas máquinas. A aplicação será responsável por enviar de forma criptografada os *datasets* para o armazenamento em nuvem escolhido na sessão. Após os dados serem enviados para o armazenamento em nuvem, o operador da máquina na nuvem executa a aplicação Rclone SGX que será responsável por baixar os *datasets* de ambas as partes e armazená-los em um SCONE Volume (passos ⑤ a ⑦). No passo ⑧, o operador executará o SEML ML Provider que obterá os dados baixados (passo ⑨) e produzirá o modelo especificado pelas partes. Esse modelo será guardado também no volume protegido (passo ⑩). Com o modelo construído, o operador executará novamente a aplicação Rclone SGX, dessa vez, para enviar o modelo produzido para o armazenamento em nuvem (passos ⑪ a ⑬). Por fim, cada uma das partes pode baixar o modelo (passos ⑭ a ⑯ e ⑲ a ㉑) e executar uma aplicação que use o modelo de forma segura através do SEML Model Runner (passos ⑰ a ⑱ e ㉒ a ㉓).

6 ANÁLISE DE SEGURANÇA

Com relação a potencial ameaça a segurança do sistema, surgem questionamentos quanto aos 3 atores envolvidos: usuários A e B, operador e provedor da nuvem. Através desses questionamentos conduziu-se a análise de segurança.

(1) Sobre os usuários:
- Conseguem acessar os dados do outro usuário?
- Conseguem modificar as informações das sessões acordadas ao longo do tempo?

(2) Sobre o operador da nuvem:
- Consegue acessar os dados dos usuários?
- Consegue modificar a execução das aplicações?

(3) Sobre o provedor da nuvem:
- Consegue acessar as informações do processamento do modelo que trafegam na memória principal?
- Consegue interceptar a comunicação entre as aplicações e o armazenamento em nuvem ou o SCONE CAS?

Quanto ao ①, os usuários não tem acesso aos dados do outro usuário, pois cada uma das informações é encapsulada utilizando uma sessão SCONE. Somado a isso, as sessões criadas são imutáveis, isto é, não podem ser atualizadas a fim de evitar que uma das partes

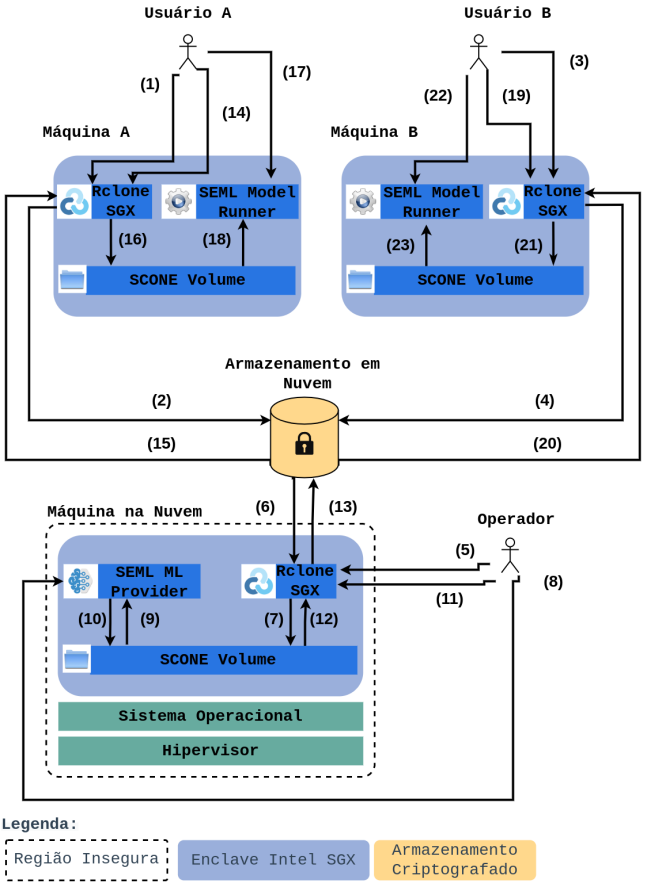


Figura 3: Fluxo de Execução do SEML.

rompa o compromisso acordado. Assim, caso os usuários A e B tenham concordado que apenas um código específico possa usar aquele modelo resultante, os usuários não têm a liberdade de alterar o código que utilizaria o modelo (e, portanto, não poderiam exportar dados ou modelos).

Ja em ②, todas as informações são armazenadas em um SCONE Volume criptografado com chaves geradas pelo SCONE CAS, o que impossibilita acessos não autorizados aos dados em repouso. As aplicação SCONE são verificadas através do mecanismo de atestação remota, qualquer alteração do código implica em mudanças no identificador do enclave (MREnclave) definido e, portanto, não são autorizados a executarem e receberem os segredos que permitem acesso aos dados.

Para ③, a tecnologia Intel SGX aloca uma porção protegida da memória principal para realizar as operações, impedindo *dump* de memória e outras tentativas correlatas de extração de informações. Todas as informações que trafegam (chaves e dados) são criptografados por padrão através de TLS.

7 AVALIAÇÃO

Nesta seção, apresentaremos a configuração utilizada para os experimentos, os cenários que queremos avaliar, seus resultados e discussões.

Os experimentos foram planejados para avaliar o desempenho entre o SEML utilizando a biblioteca Scikit-learn com SCONE e o mesmo fluxo de execução de maneira insegura, isto é, sem Intel SGX. Foi utilizado o *dataset* Iris, que consiste em 50 amostras de cada uma dos 3 tipos de Iris (Iris setosa, Iris virginica e Iris versicolor) com características do comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala, o que permite realizarmos experimentos com modelos para classificação e regressão.

De maneira a entender o comportamento com cargas de dados mais volumosas e manter as mesmas características do *dataset* escolhido, optou-se por gerar uma versão 100 vezes maior que a original (150 linhas) contendo 15000 linhas a partir de sucessivas cópias.

Adotaremos as seguintes convenções para os experimentos:

- **Software nativo** é a aplicação que executa sem enclaves SGX;
- **Software confidencial** é a aplicação portada pelo SCONE para executar em enclaves SGX;
- **Dataset pequeno (P)** é a versão original do Iris *dataset* com 150 linhas;
- **Dataset grande (G)** é a versão expandida do Iris *dataset* obtido atraves de sucessivas cópias, totalizando 15000 linhas.

7.1 Configuração do Experimento

Para realizar os experimentos foram utilizadas três máquinas virtuais. A primeira é uma máquina virtual provisionada por uma nuvem OpenStack, ela será utilizada pelos usuários para enviarem os dados e executarem o modelo ao término do processo. A máquina é equipada com processador Intel® Xeon® CPU E3-1280 v6 de 3.90 GHz, 8 CPUs virtuais, 16 GB de memória principal e 100 GB de armazenamento.

A segunda máquina virtual também é provisionada em uma nuvem OpenStack e será utilizada como armazenamento em nuvem para os dados criptografados rodando MinIO. A maquina é equipada com um processador Intel® Xeon® CPU X5550 de 2.67 GHz, 2 CPUs virtuais, 4 GB de memória principal e 80 GB de armazenamento.

Finalmente, a terceira máquina pertence a um cluster Kubernetes provisionado na Azure que será utilizado para o processamento dos dados e geração do modelo de aprendizagem de máquina. O cluster é equipado com processador Intel® Xeon® Platinum 8370C CPU de 2.80 GHz, 2 CPUs virtuais e 16 GB de memória principal.

Apenas a primeira e terceira máquina contam com suporte a Intel SGX. Quando executado em enclaves, o Scikit-learn é configurado com 2 GB de memória *heap*.

7.2 Questões e Cenários

A fim de avaliarmos o desempenho do SEML foram propostos as seguintes questões a serem entendidas e seus respectivos cenários. Cada um dos cenários foi executado 20 vezes.

1) Quando protegemos a biblioteca Scikit-learn com SCONE, qual o impacto no consumo de recursos?

Para responder essa pergunta, queremos entender como o consumo de recursos é impactado através dos algoritmos utilizados e o tamanho do *dataset* nas execuções nativas e confidenciais. A Tabela 1 apresenta os fatores e níveis dos 8 cenários.

Scikit-learn	Algoritmos	Dataset
Nativo	Classificação e Regressão	P e G
Confidencial	Classificação e Regressão	P e G

Tabela 1: Cenários para consumo de recursos do Scikit-learn.

2) Quando realizamos o fluxo de execução do SEML, qual o impacto no tempo sentido pelo usuário em relacão ao mesmo processo de forma desprotegida?

Para responder essa pergunta, queremos entender como o tempo é impactado através da execução pelo usuário do fluxo completo do SEML com seus componentes, dos algoritmos utilizados e o tamanho do *dataset* nas execuções nativas e confidenciais. A Tabela 2 apresenta os fatores e níveis dos 8 cenários.

Componentes	Algoritmos	Dataset
Nativo	Classificação e Regressão	P e G
Confidencial	Classificação e Regressão	P e G

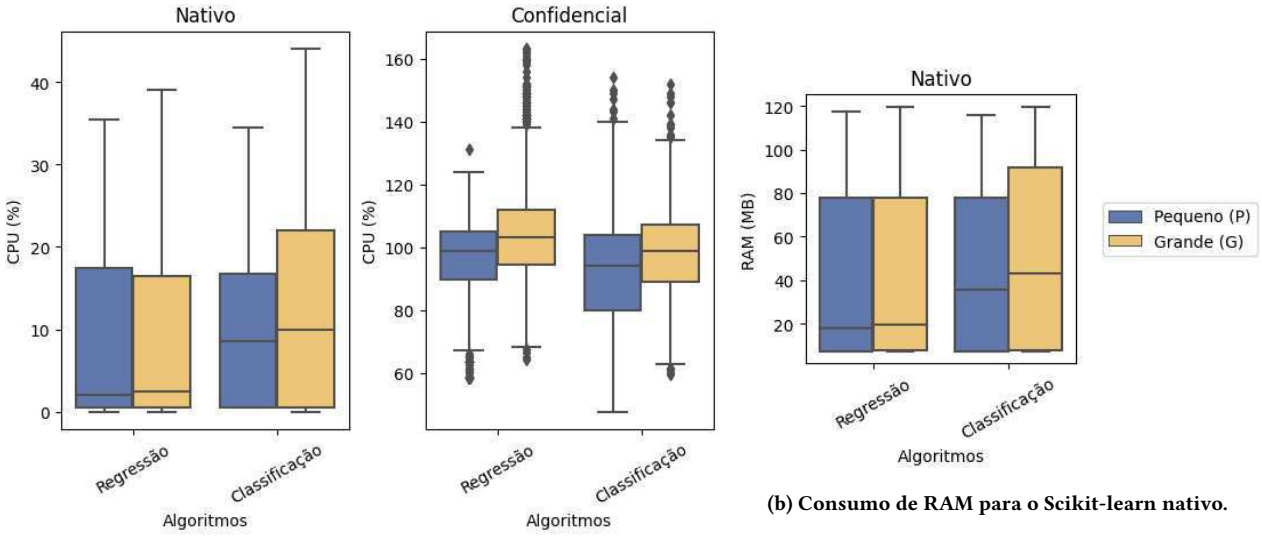
Tabela 2: Cenários para impacto de tempo na execução.

7.3 Resultados e Discussões

Os resultados relacionados ao experimento de consumo de recursos (Questão 1) são apresentados na Figura 4. A Figura 4a apresenta através de gráficos box plot a disposição do consumo de CPU do Scikit-learn, em execuções nativas e confidenciais, para combinações de algoritmos (regressão e classificação) e tamanho do *dataset* (P e G). Para que fosse possível coletar os dados de consumo nas execuções nativas, foi preciso adicionar um atraso de 2 s para que o medidor pudesse coletas os dados, visto que, a execução em si do código nativo (sem levar em consideração tempo de escalonamento, etc) levou em média 0.12 s. Ao observarmos os dados, podemos perceber que o consumo em geral de CPU para a biblioteca nas execuções nativas são baixos (75% das réplicas usaram menos de 25% de CPU) mesmo quando o *dataset* é mais volumoso. Quando executado em enclaves SGX, vemos uma demanda substancial de processamento, pois 75% das réplicas utilizaram por volta de 110% da CPU, demonstrando a necessidade de maior recurso dedicado quando atrelamos os mecanismos de segurança do TEE.

Em termos de memória principal (Figura 4b), a execução nativa do Scikit-learn obteve 75% das repetições abaixo de 80 MB, exceto o cenário de classificação com o *dataset* grande que chegou próximo dos 90 MB. Ao executarmos com Intel SGX, colocou-se 2 GB de memória *heap* que se demonstrou suficiente para executar todos os cenários sem problemas, contudo, não foi possível mensurar a quantidade utilizada de memória, pois o SCONE faz a alocação de toda a quantidade de *heap* mesmo que não seja utilizada pela aplicação.

Os resultados relacionados ao experimento de impacto de tempo sentido pelo usuário na execução do fluxo SEML (Questão 2) é demonstrado na Figura 5. Ele detalha através de um gráfico de colunas empilhadas a média de tempo gasto nas replicações para as fases do workflow do SEML (Figura 3), em execuções nativas e



(a) Consumo de CPU para o Scikit-learn nativo e confidencial.

(b) Consumo de RAM para o Scikit-learn nativo.

Figura 4: Consumo de recursos para Scikit-learn nativo e confidencial.

confidenciais, para combinações de algoritmos (regressão e classificação) e tamanho do *dataset* (P e G). No fluxo de execução nativo, observa-se uma média de 41 *s* para execução de todos os cenários, com destaque para classificação e regressão utilizando o *dataset* maior que tiveram os maiores tempos (44 *s* e 43 *s* respectivamente). Quando executado com os componentes SCONE, temos um padrão semelhante, porém com um salto no tempo, com uma média de 237 *s* para execução total dos cenários.

Diferente do que se podia imaginar, o que mais contribuiu com o tempo na execução confidencial não é a construção do modelo com Scikit-learn que obteve média de 14 *s* tanto para execução nativa como confidencial (respectivamente 14, 20840 *s* e 14, 20862 *s*), mas sim a fase de upload dos *datasets* para a nuvem através do Rclone SGX que respondeu sozinho por uma média de 107 *s* (em comparação aos 2 *s* na execução nativa) do tempo de execução dos cenários. Isso pode ser explicado pelo sistema operacional base utilizado para criar os contêineres dos componentes, enquanto o Rclone nativo utiliza a distribuição Alpine que é consideravelmente mais leve e rápida, o Rclone SGX usa Debian devido a restrições do processo de porte para Intel SGX. Situação essa que é minimizada quando olhamos a construção do modelo no SEML ML Provider e a execução do modelo com o SEML Model Runner que utilizam Alpine assim como suas versões nativas.

Em resumo, nossos experimentos demonstraram que ao protegermos a execução do processo de aprendizado de máquina acarretou-se em maior consumo de CPU somado ao aumento não proibitivo de RAM e tempo decorrido para realizar as operações. Dessa forma, torna-se uma solução viável para garantir a segurança dos dados.

8 CONCLUSÕES

Este trabalho apresentou o SEML, um processo para execução segura de aprendizagem de máquina em nuvem. Para fornecer segurança ao processamento dos dados, utilizamos o TEE da Intel aliado ao framework SCONE, aproveitando sua facilidade para portar aplicações nativas para enclaves SGX. Em nossa arquitetura, usamos componentes SGX para que as execuções ocorram dentro da memória protegida e que os dados sejam entregues apenas a partes confiáveis atestadas de maneira criptografada.

As etapas futuras para esta pesquisa incluem realizar testes exaustivos para avaliar o desempenho com *datasets* grandes e investigações a fim de otimizar o consumo de recursos dos componentes do SEML.

AGRADECIMENTOS

Agradeço primeiramente a Deus por tornar esse trabalho possível.

À minha família pelos esforços emprendidos para que eu pudesse realizar o sonho da graduação em Campina Grande.

Aos amigos que reuni ao longo desses três anos de atividade profissional no LSD (em especial Eduardo, Benardi, Matteus, Roberto, Amândio, João, Clenimar e Whasley) e de membros do time da Scontain (mantenedora do SCONE) pelas sugestões e apoio durante o desenvolvimento deste trabalho, que é uma variação de estudo de caso anterior.

A todos os meus professores, em especial: Andrey, Joseana, Raquel, Fubica, Lívia, Massoni, Campelo e Eanes.

Por fim, mas não menos importante, agradeço ao meu orientador, Andrey Brito, pelo conhecimento, paciência e, sobretudo, pelas preocupações e felicidades compartilhadas ao longo do desenvolvimento deste trabalho.

REFERÊNCIAS

[1] Felipe Albuquerque, Eduardo Alchieri, and Marcos Caetano e Priscila Solis. 2018. Integração de Replicação Máquina de Estados no Kubernetes. In *Anais do XIX Workshop de Testes e Tolerância a Falhas*. SBC, Porto Alegre, RS, Brasil. https://sol.sbc.org.br/index.php/wtf/article/view/2384

[2] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. Citeseer, 35.

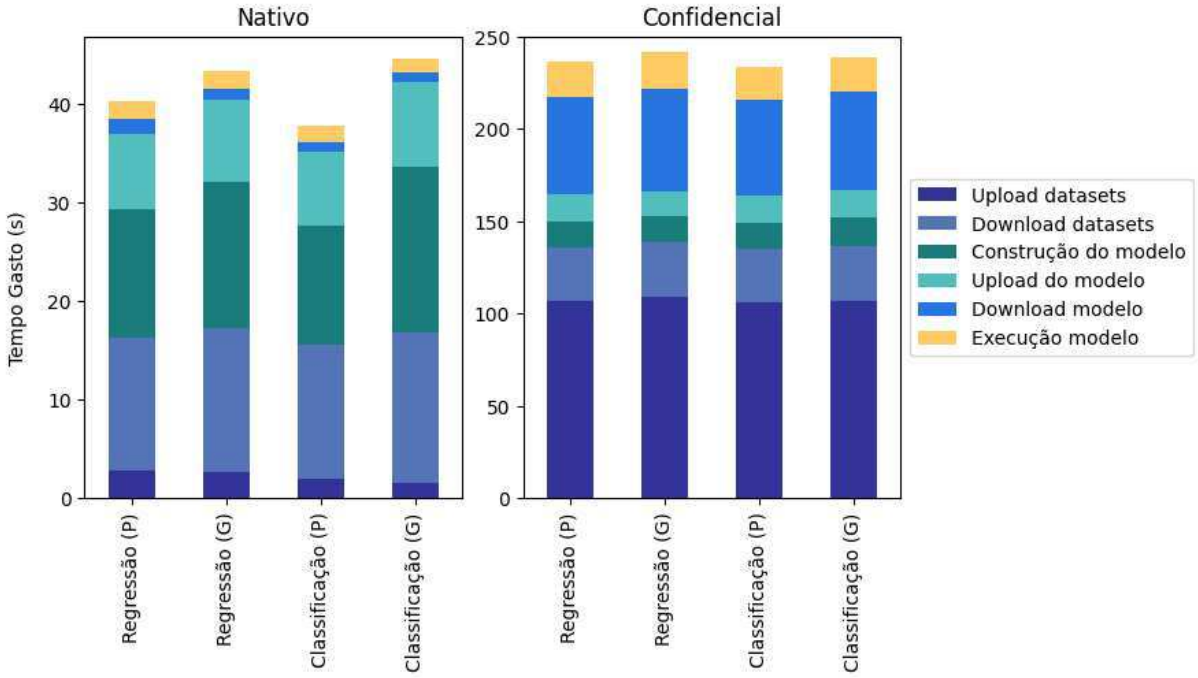


Figura 5: Tempo de execução entre o fluxo nativo e confidencial (SEML).

[3] Andrey Brito, Clenimar Souza, Fábio Silva, Lucas Cavalcante, and Matteus Silva. 2020. Processamento confidencial de dados de sensores na nuvem. *Sociedade Brasileira de Computação* (2020).

[4] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *ACM Queue* 14 (2016), 70–93. http://queue.acm.org/detail.cfm?id=2898444

[5] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.

[6] Laurence Goasduff. 2021. Gartner says cloud will be the centerpiece of new digital experiences. https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences

[7] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. 2018. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961* (2018).

[8] Michał Kucab, Piotr Boryło, and Piotr Chołda. 2021. Remote attestation and integrity measurements with Intel SGX for virtual machines. *Computers & Security* 106 (2021), 102300.

[9] Roland Kunkel, Do Le Quoc, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer. 2019. TensorSCONE: A Secure TensorFlow Framework using Intel SGX. https://doi.org/10.48550/ARXIV.1902.04413

[10] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.

[11] Maria Carolina Monard and José Augusto Baranauskas. 2003. Conceitos Sobre Aprendizado de Máquina. In *Sistemas Inteligentes Fundamentos e Aplicações* (1 ed.). Manole Ltda, Barueri-SP, 89–114.

[12] Andreas C Müller and Sarah Guido. 2016. *Introduction to machine learning with Python: a guide for data scientists*. "O'Reilly Media, Inc.".

[13] Hylson Netto, Lau Cheuk Lung, Miguel Correia, and Aldelir Fernando Luiz. 2016. Replicação de máquinas de estado em containers no kubernetes: uma proposta de integração. *Anais do XXXIV Simpósio Brasileiro de Redes de Computadores-SBRC* (2016).

[14] Sagar Sharma and Keke Chen. 2020. Confidential Machine Learning on Untrusted Platforms: A Survey. *arXiv preprint arXiv:2012.08156* (2020).

[15] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. 2018. VAULT: Reducing paging overheads in SGX with efficient integrity verification structures. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 665–678.

[16] Anji Reddy Vaka, Badal Soni, and Sudheer Reddy. 2020. Breast cancer detection by leveraging Machine Learning. *ICT Express* 6, 4 (2020), 320–324.

[17] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2512–2520.

[18] Peterson Yuhala, Pascal Felber, Valerio Schiavoni, and Alain Tchana. 2021. Plinius: Secure and persistent machine learning model training. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 52–62.