SeguraAí: confidencialidade de dados sensíveis com SGX

Felipe Antunes[1], **Filipe Garcia**[1] **e Diego Kreutz**[1]

[1]Ciência da Computação (CC), Engenharia de Software (ES)
e Laboratório de Estudos Avançados (LEA)
Universidade Federal do Pampa (UNIPAMPA)

{felipeantunesquirino,filipe.garcia1997}@gmail.com, kreutz@acm.org

Abstract. *Data leakage is still one of the major security issues. In 2018, large-scale leaks reached an impressive number of 1.1 billion records. To address this kind of security issue, Intel launched SGX, a low-cost technology for commodity hardware, which has been a research and development subject in both industry and academy. In this paper, we propose an architecture, named SeguraAí, to ensure the integrity and confidentiality of sensitive data using Intel® SGX. Some of the strengths of SeguraAí are interoperability, scalability, and compatibility with existing solutions. Our initial findings show that our solution can help to ensure the integrity and confidentiality of sensitive data of different applications.*

Resumo. *O vazamento de dados sensíveis ainda é um dos principais problemas de segurança. Apenas em 2018 houveram vazamentos de larga escala que chegam a 1.1 bilhões de registros. Recentemente, a Intel lançou SGX, uma tecnologia de baixo custo para hardware de prateleira, que tem sido objeto de pesquisa e desenvolvimento no meio corporativo e acadêmico. Este trabalho apresenta uma arquitetura, denominada SeguraAí, para garantir a integridade e confidencialidade de dados sensíveis utilizando Intel® SGX. Como pontos fortes da solução podem ser destacados a interoperabilidade, escalabilidade e compatibilidade com soluções existentes. As primeiras análises e resultados mostram que a solução é viável e pode contribuir na evolução e proteção de sistemas.*

1. Introdução

Vazamento de dados sensíveis, como nome de usuários, senhas, informações de cartões de créditos, entre outras informações, são um problema real, recorrente, cada vez mais crônico e preocupante. Dados de relatórios técnicos especializados e estatísticas online mostram que os vazamentos estão cada vez mais frequentes, impactando um número crescente de instituições e usuários. Em 2018, os 5 maiores incidentes de segurança causaram vazamentos entre 50 milhões a 1.1 bilhões de registros de usuários [Bisson 2018].

Um dos principais alvos dos ataques são dados de autenticação e autorização e dados pessoais dos usuários dos sistemas. Há diferentes trabalhos que investigam a aplicação de tecnologias de hardware (uma lista pode ser vista em [Antunes et al. 2018]) na protecção desse tipo de dados [Kreutz et al. 2014, Kreutz et al. 2016, Brekalo et al. 2016, Almalki 2017, Machida et al. 2018]. Por exemplo, recentemente, investigadores propuseram a utilização da tecnologia Intel® SGX para implementar serviços de gerenciamento de chaves [Chakrabarti et al. 2017], melhorar a segurança do processo de autenticação de usuários no Open Authorization 2.0 [Almalki 2017] e mitigar ataques de força bruta offline em senhas de

usuários [Brekalo et al. 2016]. Estes são alguns exemplos de projetos de soluções de segurança, específicas a um determinado contexto, assistidas por hardware.

Outro exemplo, um pouco mais genérico, é o apresentado por investigadores para resolver problemas de disponibilidade, integridade e confidencialidade de dados em infraestruturas de autenticação e autorização [Kreutz et al. 2014, Kreutz et al. 2016]. A solução é genérica, ou seja, pode ser teoricamente aplicada a qualquer protocolo de autenticação e/ou autorização. A arquitetura da solução é ancorada em técnicas de tolerância a falhas e intrusões, sem perder propriedades como disponibilidade e confidencialidade dos dados. No centro da solução reside um componente crucial, o componente confiável (TC), que pode ser implementado através de diferentes tecnologias. Os autores demonstraram o funcionamento e a operação da solução para os protocolos RADIUS e OpenID. Entretanto, apesar da especificação minuciosa e detalhada, os autores não apresentam uma implementação do TC em hardware específico. Além disso, a solução é bastante complexa em termos de interoperabilidade de sistemas, necessitando de várias modificações e adaptações para cada novo protocolo de autenticação e/ou autorização.

Dado o contexto apresentado, os principais objetivos do trabalho são: dar continuidade a trabalhos anteriores [Kreutz et al. 2014, Kreutz et al. 2016], propondo uma nova arquitetura capaz de atender serviços de autenticação e autorização e, também, outros sistemas que necessitem assegurar a integridade e confidencialidade de dados sensíveis, tomando como base as possibilidades e inovações permitidas pela tecnologia SGX; implementar um serviço de autenticação e autorização com a tecnologia SGX, utilizando o emulador OpenSGX [Jain et al. 2016]; e criar uma solução genérica (separação em duas camadas), desvinculada dos protocolos específicos de autenticação e autorização (e.g. RADIUS, TACACS, Kerberos, SAML, OpenID, OAuth), que permite interoperabilidade entre sistemas. Resumidamente, as principais contribuições deste trabalho são: uma visão geral do estado da arte da tecnologia SGX e do emulador OpenSGX; proposição e discussão de uma arquitetura genérica, escalável e segura de serviços de autenticação e autorização utilizando SGX e *web services* (WSs); e implementação e avaliação de um protótipo do serviço de autenticação utilizando OpenSGX.

O restante deste trabalho está organizado como segue. A solução proposta, *SeguraAí*, é apresentada e discutida na Seção 2. A implementação, os resultados, os trabalhos relacionados e as considerações finais são apresentados nas Seções 3, 4 e 5.

2. *SeguraAí*: a solução proposta

Na prática, não existe uma solução 100% segura contra vazamento de dados. O principal desafio reside em proteger os dados em caso de vazamento. *SeguraAí* é uma arquitetura de sistema que tem por objetivo oferecer serviços de segurança para proteger a integridade e confidencialidade de dados sensíveis em tempo de execução e nos sub-sistemas de armazenamento. Para isto, a tecnologia Intel® SGX é utilizada em combinação com outros componentes de software e hardware, como ilustrado na Figura 1. Como pode ser observado, há cinco componentes principais (cliente, WS/gateway, serviço confiável, dispositivo confiável e banco de dados) e quatro protocolos (CL, WS, DC e BD).

O **cliente** é um sistema (e.g. aplicação Web) ou protocolo (e.g. RADIUS, OpenID) que necessita manter a integridade e confidencialidade de informações sensíveis para autenticar ou autorizar um usuário ou ainda realizar operações comerciais (e.g. com-

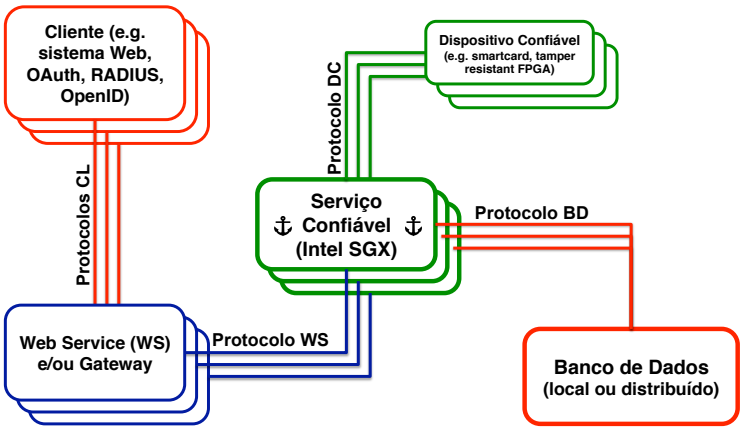

Figura 1. Visão geral da arquitetura *SeguraAí*

pra e venda utilizando dados de cartão de crédito). O **cliente** se comunica exclusivamente através do **WS/gateway** do *SeguraAí*. A comunicação pode ocorrer através de APIs REST do **WS** ou protocolos convencionais, suportados no **gateway**. O papel do **gateway** é manter a compatibilidade com sistemas legados. Por exemplo, as requisições RADIUS (convencionais) chegam até o **gateway** e são traduzidas para requisições ao **serviço confiável**, utilizando o **protocolo WS**. Neste exemplo, o **protocol CL** é o RADIUS.

O **WS/gateway** é um componente específico da solução *SeguraAí* responsável pela integração de clientes, interoperabilidade de sistemas e compatibilidade com sistemas legados. Todas as requisições dos **clientes**, utilizando diferentes *protocolos CL*, são recebidas e encaminhadas ao **serviço confiável** através de um *protocolo WS* específico, com uma interface bem definida. Um **WS/gateway** pode rodar como um serviço convencional do sistema operacional ou em um *enclave* SGX para aumentar o nível de segurança dos dados que por ele passam. Podem haver múltiplas instâncias deste componente para escalabilidade, ou seja, atender um número potencialmente grande de **clientes**.

O **serviço confiável** é o coração do *SeguraAí*, implementado utilizando a tecnologia Intel® SGX. Além disso, ele trabalha em conjunto com um **dispositivo confiável**, cuja principal finalidade é armazenar em hardware seguro chaves mestras e mecanismos essenciais de gerenciamento do **serviço confiável**, algo similar ao proposto em [Kreutz et al. 2016, Kreutz et al. 2014]. O **serviço confiável**, com auxílio do **dispositivo confiável** na inicialização dos *enclaves*, oferece serviços essenciais para os **clientes**, como controle de autenticação e autorização de usuários. As requisições são encaminhadas pelos **clientes** ao **WS/gateway** e repassadas ao **serviço confiável**, que irá processar efetivamente a requisição e enviar uma resposta. No caso de uma autenticação, pode ser algo simples como verificar o login e a senha do usuário e retornar verdadeiro ou falso como resultado da verificação. É importante ressaltar que os dados do usuário são utilizados, em tempo de execução, somente dentro dos *enclaves* do **serviço confiável**, garantindo a integridade e confidencialidade dos dados. Esses dados são armazenados num **banco de dados**, local ou distribuído (e.g. NoSQL/Redis), não confiável. Antes de armazenar os dados, o *enclave* realiza o selamento, ou seja, cifra os dados utilizando uma chave forte e um algoritmo de robusto. Portanto, mesmo na eventualidade de vazamentos de dados, o atacante terá acesso apenas aos dados selados (cifrados).

Tanto o **protocolo DC** quanto o **protocolo BD** são específicos à tecnologia sendo utilizada. No caso do **protocolo DC**, pode ser uma interface simples e robusta como proposto em [Kreutz et al. 2014]. Já no caso do **protocolo BD**, pode variar de acordo com o sistema de banco de dados sendo utilizado, que pode ir desde um arquivo de texto local até um serviço online, sob-demanda, como o Google BigQuery e o Microsoft Always Encrypted. Aliás, soluções como Always Encrypted permitem adicionar camadas extras de segurança fim-a-fim aos dados, o que vem a calhar com os objetivos do *SeguraAí*, ou seja, máxima proteção à integridade e confidencialidade dos dados sensíveis.

Na Seção 3 é apresentando um primeiro protótipo simplificado da arquitetura. O protótipo inclui uma versão simples de um **cliente**, **WS/gateway**, **protocolo WS** e **serviço confiável** de autenticação de usuários.

3. Implementação, Resultados e Discussão

Na primeira etapa, foi implementado um protótipo simplificado do *SeguraAí*, que consiste em um serviço de cadastro e autenticação de usuário, algo necessário a praticamente todos os sistemas. O protótipo utiliza o modelo cliente/servidor, onde o cliente realiza o papel do cliente e do WS da arquitetura *SeguraAí*. Já o servidor é um serviço confiável, implementado utilizando OpenSGX.

O protótipo utiliza o protocolo TCP/IP para troca de mensagens e especifica um protocolo de comunicação simples entre o WS e o serviço confiável. Essencialmente, o protocolo define o formato das mensagens e os mecanismos de segurança. O formato das mensagens é `operação:random:`E_k`(login:senha):HMAC` (**protocolo WS** simplificado – ver arquitetura na Seção 2), onde a operação pode ser `cadastro` ou `autenticação`. A mensagem é transmitida com uma assinatura HMAC, utilizando a primitiva HMAC-SHA256. Há duas chaves distintas para os HMACs, uma para a operação de `cadastro`, que pode ser realizada somente por administradores do sistema, e outra para `autenticação`, que pode ser solicitada por qualquer WS (registrado) do serviço. Vale ressaltar que a geração e distribuição das chaves está fora do escopo deste trabalho. Para este protótipo, é assumido que as chaves já foram previamente distribuídas de forma segura.

Além do HMAC, que é utilizado para verificar a autenticidade e integridade das requisições dos WSs, o algoritmo AES é utilizado para cifra simétrica. As chaves secretas são utilizadas tanto no HMAC quanto no AES. Os únicos dados que não são cifrados são a `operação` e o `random`. Este último serve para evitar ataques de *replay*.

O objetivo principal foi avaliar e discutir questões de desempenho do protótipo implementado. Para realizar os testes, foi utilizado um computador com um processador Intel core i7 5500u, 8 GB de RAM e a distribuição GNU/Linux Debian 9 X64 estável.

Os dados da Tabela 1 representam o tempo total de execução do protótipo para 2000 autenticações, o tempo médio de cada autenticação e o desvio padrão. Os tempos são em segundos. Nos experimentos, foram medidos os tempos do cliente, o que inclui a latência da rede, e da autenticação no servidor. Como pode ser observado na tabela, os testes foram realizados em três ambientes distintos de execução, nativamente no sistemas operacional hospedeiro da máquina, emulado com QEMU e emulado com OpenSGX. Vale ressaltar que, neste último caso, o OpenSGX utiliza o QEMU, ou seja, há dois níveis de emulação no ambiente.

Tabela 1. Tempo de execução

	tempo total	tempo médio	desvio padrão
Cliente nativo	0.248717	0.000124	0.000138
Cliente com QEMU	0.675755	0.000338	0.000144
Cliente com OpenSGX	210.875060	0.105595	0.014282
Autenticação nativa	0.087788	0.000043	0.000021
Autenticação com QEMU	0.467591	0.000234	0.000078
Autenticação com OpenSGX	209.038668	0.104623	0.014071

Como pode ser observado na Tabela 1, a execução nativa da autenticação possui o melhor desempenho em todos os casos, como era esperado. O desempenho da execução com QEMU vem em segundo lugar, como já era também esperado. No processo de autenticação, que é o mais importante da solução uma vez que será utilizado com grande frequência, a versão com QEMU é aproximadamente 5.3x mais lenta que a nativa. Já a execução com OpenSGX é aproximadamente 447x mais lenta que a QEMU. A principal fonte do overhead são as instruções e os mecanismos de segurança SGX (e.g. MEE – *Memory Encryption Engine*) que são emulados em software, tornando o processo de acesso à memória extremamente lento quando comparado com a implementação Intel® SGX nativa.

Apesar dessa diferença significativa entre as execuções QEMU e OpenSGX, pode-se observar que o tempo de uma autenticação é baixo, ficando em apenas 105.59ms, ou seja, algo imperceptível por um usuário normal. Na verdade, qualquer autenticação na ordem de até 200ms, por exemplo, é considerada aceitável sob a perspectiva do usuário [Kreutz et al. 2014], uma vez que é um tempo de resposta muito baixo para um usuário perceber a olho nu. Isto significa que, mesmo com todo o overhead do QEMU+SGX, trazido pelo OpenSGX, ainda temos um tempo de autenticação baixo e aceitável. Na prática, este tempo será ainda menor em um hardware com suporte a SGX, onde componentes essenciais como o MEE e os respectivos algoritmos de criptografia, são implementados e/ou acelerados em hardware.

4. Trabalhos Relacionados

Existem diferentes trabalhos que buscam resolver o problema da integridade e confidencialidade de chaves secretas, dados de autenticação e autorização de sistemas e outras informações sensíveis dos usuários [Kreutz et al. 2014, Kreutz et al. 2016, Almalki 2017, Brekalo et al. 2016, Machida et al. 2018, Alansari et al. 2017]. O maior desafio de qualquer solução é mitigar o vazamento, ou minimizar o efeito do vazamento, de dados sensíveis, como login, senha, dados complementares de autenticação (e.g. perguntas secretas) e outros dados privados dos usuários. As soluções variam em tecnologia, aspectos práticos, aplicabilidade em cenários reais, interoperabilidade, entre outras características.

A maioria das soluções existentes são específicas, ou seja, projetadas para determinados ambientes, aplicações ou contextos. Trabalhos recentes avançam um pouco o estado da arte ao propor arquiteturas mais gerais para infraestruturas de autenticação e autorização (AAIs) baseadas em protocolos como RADIUS e OpenID [Kreutz et al. 2014, Kreutz et al. 2016]. Entretanto, apesar de suportar múltiplos protocolos, são projetadas para contextos específicos, complexas e de baixa escalabildiade. Além disso, não pro-

movem a interoperabilidade de sistemas e são engessadas quanto a características como extensibilidade. Estes são alguns dos pontos endereçados pela arquitetura do *SeguraAí*, através de componentes como o serviço confiável, WS/gateway e banco de dados (ver Seção 2). Devido a limitação de espaço, uma descrição mais detalhada dos trabalhos relacionados, bem como uma tabela comparativa resumindo as nove características analisadas e discutidas, estão disponíveis em [Antunes et al. 2018].

5. Considerações Finais

SeguraAí é uma solução mais genérica, para garantir a integridade e confidencialidade de dados sensíveis, quando comparada a trabalhos relacionados. Em outra palavras, ela não foca em um caso único de aplicação, como é o caso da maioria das soluções similares.

Os resultados indicam que a solução proposta é viável com o suporte da tecnologia Intel® SGX. Os dados de desempenho, com OpenSGX, mostram que a solução pode ser aplicada para serviços essenciais e críticos, como os sistemas de autenticação e autorização de usuários. Apesar do *overhead* da emulação em software trazida pelo OpenSGX, o protótipo do *SeguraAí* atingiu um desempenho similar a sistemas nativos, como o FreeRADIUS (latência de autenticação na ordem de 100ms [Kreutz et al. 2014]).

Como trabalhos futuros podem ser mencionados: (1) analisar em detalhes os *trade-offs* entre segurança e desempenho; (2) investigar a sobrecarga da solução em máquinas que oferecem as últimas gerações da tecnologia Intel® SGX; (3) implementar um protótipo completo da arquitetura do *SeguraAí*; e (4) realizar um estudo de viabilidade técnica e comercial de inclusão da solução proposta.

Referências

Alansari, S., Paci, F., and Sassone, V. (2017). A distributed access control system for cloud federations. In *IEEE ICDCS*, pages 2131–2136. IEEE.

Almalki, M. S. (2017). *Enhancing the Security for User Authentication in Open Authorization 2.0 by Using OpenSGX Tool*. PhD thesis, Tennessee State University.

Antunes, F., Garcia, F., and Kreutz, D. (2018). Seguraaí: confidencialidade de dados sensíveis com sgx (versão estendida). `https://goo.gl/vPcN8C`.

Bisson, D. (2018). The 10 biggest data breaches. `https://goo.gl/sGY8GF`.

Brekalo, H., Strackx, R., and Piessens, F. (2016). Mitigating password database breaches with intel sgx. In *SysTEX*, pages 1:1–1:6. ACM.

Chakrabarti, S., Baker, B., and Vij, M. (2017). Intel SGX Enabled Key Manager Service with OpenStack Barbican. *ArXiv e-prints*.

Jain, P., Desai, S. J., Shih, M.-W., Kim, T., Kim, S. M., Lee, J.-H., Choi, C., Shin, Y., Kang, B. B., and Han, D. (2016). OpenSGX: An open platform for sgx research. In *USENIX NDSS*.

Kreutz, D., Bessani, A., Feitosa, E., and Cunha, H. (2014). Towards secure and dependable authentication and authorization infrastructures. In *IEEE PRDC*, pages 43–52.

Kreutz, D., Malichevskyy, O., Feitosa, E., Cunha, H., da Rosa Righi, R., and de Macedo, D. D. (2016). A cyber-resilient architecture for critical security services. *Journal of Network and Computer Applications*, 63:173–189.

Machida, T., Yamamoto, D., Morikawa, I., Kokubo, H., and Kojima, H. (2018). A secure framework for user-key provisioning to sgx enclaves. In *International Conference on Network-Based Information Systems*, pages 725–732. Springer.