Relatório

Segurança e Aplicações de Hardware Confiável

Matheus Bezerra

Ricardo

Rodrigo

Glossário

Introdução

Objetivo

Introdução

Este relatório apresenta o desenvolvimento de um projeto para a disciplina de Segurança e Aplicações de Hardware Confiável, cujo objetivo central é o desenho de um sistema de gerenciamento de chaves (Key Management Service – KMS) seguro e eficiente. O sistema proposto contempla múltiplos componentes interligados: usuários, servidores, HSMs e administradores, cada qual com papéis e níveis de confiança distintos. Enquanto usuários e HSMs são considerados implicitamente confiáveis, os administradores podem não ser totalmente confiáveis, o que impõe desafios adicionais ao desenho do sistema de confiança e à gestão das permissões.

Ao longo deste relatório, serão detalhadas as entidades envolvidas, as premissas de confiança adotadas e as estratégias propostas para garantir que o gerenciamento e o uso das chaves criptográficas ocorram de forma segura, mesmo em cenários onde nem todos os administradores podem ser plenamente confiáveis. A abordagem apresentada visa não apenas atender aos requisitos funcionais do sistema, mas também mitigar riscos e fortalecer a segurança das operações realizadas pelos usuários na nuvem.

Entidades Necessárias – Interesses, Funções e Características

Usuários: solicitará operações criptográficas de arquivos, isto é, cifrar, decifrar, assinar e verificar assinaturas. Eles são implicitamente confiáveis.

Administradores: decidirá quais usuários podem usar quais HSMs, gerindo as chaves do mesmo, devemos considerar administradores um entidade não confiável.

Servidores: receberá solicitações do lado do cliente e as encaminhará aos HSMs para processamento, não podendo armazenar dados referentes aos ficheiros enviados.

HSMs: armazenará e gerenciará as chaves do cliente. O comportamento destes é considerado confiável. É importante ressaltar que HSMs têm uma visão extremamente limitada de todo o sistema, mantendo apenas suas próprias chaves mestras para criptografar, descriptografar e assinar dados.

Objetivos

O objetivo deste projeto é desenvolver um sistema para gestão de chaves com entidades bem definidas. A partir disto, adicionamos outras novas features, a redundância e registro de ocorrências (logs) o sistema que serão explicadas abaixo:

Redundância

No cenário atual, a necessidade de sistemas redundantes é cada vez mais evidente diante do aumento das ameaças cibernéticas, da complexidade das infraestruturas digitais e da crescente dependência das organizações em relação à tecnologia. Com o avanço acelerado da digitalização e a migração massiva para ambientes em nuvem, empresas de todos os portes enfrentam riscos significativos de interrupção de serviços, perda de dados e ataques sofisticados, como ransomware e invasões à cadeia de suprimentos, ou até mesmo destruição física de componentes. A redundância, nesse contexto, torna-se uma estratégia essencial para garantir a continuidade dos negócios e a resiliência operacional. Sistemas redundantes, como a utilização de múltiplos HSMs para criptografia, permitem que, mesmo diante de falhas inesperadas em componentes críticos, o funcionamento dos serviços seja mantido sem interrupções perceptíveis para os usuários. Isso é especialmente importante em ambientes onde a disponibilidade e a integridade das informações são vitais, como setores financeiro, governamental e de saúde.

O sistema utilizará múltiplos HSMs para criar redundância na criptografia é uma arquitetura projetada para garantir a disponibilidade contínua e tolerância a falhas de suas operações este princípio permite:

- **Resiliência a falhas**: Elimina pontos únicos de falha (SPOF), crítico para sistemas que exigem alta disponibilidade.

- **Balanceamento de carga**: As operações criptográficas (criptografia, descriptografia, assinatura) são distribuídas entre os HSMs disponíveis, otimizando o desempenho e evitando gargalos.

- **Failover automático**: Se um HSM falhar ou ficar indisponível, outro dispositivo assume a operação das funções, garantindo continuidade sem interrupções.

Registro de Ocorrências

A presença de um sistema de logs em uma arquitetura baseada em múltiplos HSMs é fundamental para garantir a segurança, a rastreabilidade e a conformidade operacional do ambiente. Os logs registram todas as ocorrências relevantes, como operações de criptografia e descriptografia, tentativas de acesso, alterações de configuração e falhas de autenticação, criando um histórico detalhado de atividades dentro do sistema.

Protótipo – Características Básicas

Foi um projeto desenvolvido em Node.js em versão v20.11.0, além disso, utiliza base de dados em Postgres 17. Para realização de testes foi usado o Postman.

Protótipo - Entidades

Com isto, está esclarecido os objetivos que o sistema deve ter. A partir disto, foi desenvolvido um sistema que é capaz de cumprir todas estas demandas, a grande mudança foi criar 2 tipos distintos de servidores, que são:

1. Servidor Principal (MainServer): responsável por receber comandos dos utilizadores e administradores, tratar da autenticação do utilizador, enviar e-mails, armazenar e gerir a base de dados, e redistribuir os pedidos de criptografia aos Servidores HSM, que terão suas funcionalidades apresentadas a seguir.
2. Servidor HSM (HSMServer): é nestes servidores que são realizadas as operações criptográficas requeridas pelos utilizadores, eles só terão comunicação permitida com o Main Server, ou seja, qualquer operação realizada por este servidor foi filtrada e encaminhada pelo Main Server. Este tipo de servidor pode existir a quantidade que o utilizador quiser, contanto que seja no mínimo 2.
3. HSMs: Instalados no Servidor HSM, é nele que será realizada as operações criptográficas, recebidas a partir de pedidos do Servidor HSM.
4. Administradores: Tem o dever de gerenciar os utilizadores, podendo criá-los e deletá-los, além disso, conseguem saber informações sobre a base de dados, os logs de registro e o estado dos servidores.
5. Utilizadores: Terão a capacidade de definir sua password, a partir do momento que o administrador cria este utilizador. Além de conseguir realizar operações criptográficas de cifrar, decifrar, assinar e verificar assinatura.

Os detalhes específicos de cada entidade será explicada mais detalhadamente a frente.

Protótipo – Disposição

O Proof-of-Concept esboçado foi um sistema que possui 3 HSM Servers, que foram denominados em hsm1, hsm2 e hsm3:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Protótipo – Comunicação

Para comunicação externa, todos os acessos (de usuários e de administradores) ao Main Server ocorrem por canais HTTPS, protegendo os dados em trânsito contra interceptação e ataques man-in-the-middle. Já entre o Main Server e os HSM Servers, a comunicação é feita por HTTP devido a limitações de projeto, entretanto, é imposta o limite destes servidores apenas receberem requisições do Main Server.

Protótipo – Base de Dados

Para o Main Server, foi criado uma base de dados em Postgres com o objetivo de armazenar dados de autenticação e registro de logs, possuindo as seguintes tabelas:

**Users**

Esta tabela tem o objetivo de armazenar dados dos utilizadores para conseguir realizar ações de autenticação e reencaminhamento de funções para os HSM servers. Possuindo os seguintes campos:

1. **Id:** Número de identificação.
2. **Name:** String única do utilizador
3. **Email:** String com o endereço de e-mail do utilizador, que será utilizado para definição da password.
4. **Password:** String que fica armazenado o hash da password do utilizador.
5. **Salt:** Valor aleatório utilizado junto com a senha para aumentar a segurança do hash da senha.
6. **Role\_id:** Identificador da função do usuário no sistema, isto é, se for 1, é um utilizador e se for 2, é um administrador.
7. **Status:** Indica o status do usuário (ativo ou deletado).
8. **Created\_by:** Para fins de auditoria, fica registrado que administrador criou este utilizador.
9. **Server\_name\_1:** Nome do primeiro HSM server que este utilizador possui chaves criptográficas.
10. **Key\_Reference\_1:** Id de referência das chaves deste utilizador no server\_name\_1.
11. **Server\_name\_2:** Nome do segundo HSM server que este utilizador possui chaves criptográficas.
12. **Key\_Reference\_2:** Id de referência das chaves deste utilizador no server\_name\_2.

**Logs**

Esta tabela tem como objetivo armazenar todas as ações requisitadas ao Main Server e informações de suas respostas, essa foi a maneira encontrada para não precisar, por exemplo, armazenar ficheiros enviados e mesmo assim ter alguma forma de saber as ocorrências que aconteceram no servidor. Sendo assim, esta tabela tem os seguintes campos:

1. **Id:** Número de identificação.
2. **UserId:** Referência ao utilizador que fez esta requisição.
3. **Endpoint:** Tipo de requisição que o utilizador ou administrador fez.
4. **Ip\_address:** endereço de IP da requisição.
5. **Status:** Status da resposta.
6. **Created\_at:** Instante da ocorrência.

**Outras tabelas**

Também há a tabela **Roles** que armazena os valores de Utilizador e Administrador. Além disso, há a tabela chamada **Verification\_Codes**, que é utilizado para salvar os códigos de verificação para o utilizador definir a password.

Protótipo – Funções

Autenticação

Exceto para o **“User – Set Password”**, para conseguir realizar todas as ações presentes no servidor, é necessário autenticação, e para isto, é necessário fazer um pedido POST /auth/login com o seguinte formato de body:

Uma imagem contendo Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

No caso de uma autenticação bem-sucedida, será retornado um Bearer Token com dados de userId e a validade deste Token, que sempre será requisitado no campo Authorizathion nos pedidos HTTP/HTTPS. A validade deste token foi configurada para apenas 1 hora porque acredita-se que a grande maioria dos utilizadores não utilizará o servidor, durante uma sessão, mais do que algumas vezes.

Criar utilizadores

Esta função é exclusiva para uso de administradores, que enviará ao servidor os seguintes campos:

Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.

A partir deste pedido, o sistema terá o seguinte workflow:

Texto

O conteúdo gerado por IA pode estar incorreto.

Este código de verificação será usado pelo utilizador para definir sua password mais adiante, foi escolhido neste formato para evitar que o administrador tivesse alguma influência na escolha da password do utilizador, seguindo o princípio de que ele pode não ser confiável.

Deletar utilizadores

Esta função é exclusiva para uso de administradores, que enviará ao servidor o parâmetro **id**, alterando o status deste utilizador para ‘deleted’ mas sem apagá-lo da base de dados, possibilitando que se o administrador usar esta função de forma maliciosa, o único impacto que ele pode oferecer é o de impedir aquele utilizador de realizar operações com servidor, o que facilmente pode ser revertido e sem pedra de dados.

Outras Funções do Administrador

1. Get Users: Esta função é responsável por listar todos os usuários cadastrados no sistema.
2. Get Logs: Esta função permite consultar os logs do sistema com diversos filtros, aceitando parâmetros opcionais via query string, como userId, endpoint, status, startDate e endDate.
3. Clean Logs: Função responsável pela limpeza de logs antigos do sistema, requerendo uma data limite (beforeDate) no corpo da requisição, mas não permite limpar logs mais recentes que 2 meses, pois entende-se que 2 meses impediria o administrador apagar logs de ocorrências maliciosas.
4. Get Status: Função complexa que monitora o estado geral do sistema: coletando informações do servidor principal (MainServer), além de fazer ping em cada servidor HSM registrado e registrando se ele está online ou offline.

Set Password

Esta função é exclusiva para novos utilizadores que, após o administrador criar este utilizador, o mesmo receberá um e-mail com um código de verificação. O utilizador enviará um pedido ao Main Server com o seguinte endpoint “/user/set-password/:code” onde o :code ficará o código de verificação, e no body haverá o campo ‘newpwd’ onde o utilizador irá colocar sua password, após o envio de pedido ocorre o seguinte workflow:

Melhorias

1. Autenticação
2. HTTPS entre main server e HSM Server
3. Reverter deleção de utilizador