A01:2021-Broken Access Control

Exceto para recursos públicos, negar por padrão.
O middleware Authentication, responsável pelo sistema de login, já faz a maioria do trabalho. Os recursos que não precisam de autenticação, estão listados na constante src/Controller/UsersController.php::ACTIONS_SEM_AUTENTICACAO

Implemente mecanismos de controle de acesso uma vez e reutilize-os em todo o aplicativo, incluindo a minimização do uso de Cross-Origin Resource Sharing (CORS).
Por característica do projeto, somente para o CRUD de Users é necessário um controle de autorização. Código que aplica o controle: src/Controller/UsersController.php->beforeFilter(). CORS não é utilizado.

Os controles de acesso ao modelo devem impor a propriedade do registro em vez de aceitar que o usuário possa criar, ler, atualizar ou excluir qualquer registro.
O KAW tem dois tipos de usuário, o comum e o root. A única diferença entre eles é que o root pode executar ações consideradas de configuração do sistema. O restante das permissões são iguais.

Os requisitos de limite de negócios de aplicativos exclusivos devem ser impostos por modelos de domínio.
As regras de negócio estão na cama de modelo (Tables e Entity)

Desative a lista de diretórios do servidor da web e certifique-se de que os metadados do arquivo (por exemplo, o .git) e os arquivos de backup não estejam presentes nas raízes da web (web roots).
No exemplo de configuração de produção, que está na nossa wiki, é ensinado a configurar o Apache2 para considerar a pasta kaw/webroot/ como a raiz da aplicação.
O arquivo kaw/.htaccess também faz o trabalho de redirecionar para a pasta webroot/

Registrar falhas de controle de acesso e alertar os administradores quando apropriado (por exemplo, falhas repetidas).
Criada a Atividades Suspeitas que com base nos logs registrados, mostra para o usuário atividades danosas na aplicação.
Arquivos:

- Sistema de logs: kaw/src/Log/
- A tela de Atividades Suspeitas está na home

Limite de taxa o acesso da API e do controlador para minimizar os danos do conjunto de ferramentas de ataque automatizado.
Foi criado o módulo de IPs Bloqueados. Quando um usuário erra consecutivamente as credenciais de acesso o IP é bloqueado.

Os identificadores de sessão com estado devem ser invalidados no servidor após o logout. Os tokens JWT sem estado devem ter vida curta, para que a janela de

oportunidade para um invasor seja minimizada. Para JWTs de longa duração, é altamente recomendável seguir os padrões OAuth para revogar o acesso.
Melhorado a segurança da Session e Cookies. Não é utilizado JWT ou OAuth
Ler o arquivo kaw/config/app.examplo.php > 'Session'. O tempo da sessão é determinado e controlado no server side

A02:2021-Cryptographic Failures

Classifique os dados processados, armazenados ou transmitidos por um aplicativo. Identifique quais dados são confidenciais de acordo com as leis de privacidade, requisitos regulamentares ou necessidades de negócios.
Todos os dados salvos no KAW são extremamente sensíveis. Sendo armazenados somente no banco de dados e não transmitidos para terceiros.

Não armazene dados confidenciais desnecessariamente. Descarte-o o mais rápido possível ou use tokenização compatível com PCI DSS ou mesmo truncamento. Os dados não retidos não podem ser roubados.
Somente os dados essenciais de cada módulo são salvos. Todos os dados são criptografados.

Certifique-se de criptografar todos os dados confidenciais armazenados.
Todas as informações críticas são criptografadas.

Certifique-se de que algoritmos, protocolos e senhas de padrão forte e atualizados estejam em vigor; use o gerenciamento de senhas adequado.

- Usando a Lib Sodium para criptografar e descriptografar
- Na documentação é incentivado a usar variáveis de ambiente dentro Apache2 para armazenar a chave de criptografia.

Criptografe todos os dados em trânsito com protocolos seguros, como TLS com cifras de sigilo de encaminhamento (FS), priorização de cifras pelo servidor e parâmetros seguros. Aplique a criptografia usando diretivas como HTTP Strict Transport Security (HSTS).

Forçado a conexão HTTPS e aplicando o HSTS: kaw/src/Application.php. O KAW só funciona se for utilizado HTTPS.

Desative o armazenamento em cache para respostas que contenham dados confidenciais.
Feito no commit f88e4e5979556c05b5b1b6d1e37eaecb8fad056a

Aplique os controles de segurança necessários de acordo com a classificação de dados.
Os dados mais sensíveis são criptografados.

Não use protocolos legados, como FTP e SMTP, para transportar dados confidenciais.
[não aplicável]

Armazene senhas usando fortes funções de hash adaptáveis e salgadas com um fator de trabalho (fator de atraso), como Argon2, scrypt, bcrypt ou PBKDF2.
Implementando a criptografia utilizando a lib Sodium.

Os vetores de inicialização devem ser escolhidos de acordo com o modo de operação. Para muitos modos, isso significa usar um CSPRNG (gerador de números pseudo-aleatórios criptograficamente seguro). Para modos que requerem um nonce, o vetor de inicialização (IV) não precisa de um CSPRNG. Em todos os casos, o IV nunca deve ser usado duas vezes para uma chave fixa.
Gerado um nonce único para cada informação criptografada, com o tamanho da constante SODIUM_CRYPTO_SECRETBOX_NONCEBYTES, através da função random_bytes()

Sempre use criptografia autenticada em vez de apenas criptografia.
Utilizando a função sodium_crypto_secretbox() já atende esse requisito.

As chaves devem ser geradas de forma criptograficamente aleatória e armazenadas na memória como um *array* de *bytes*. Se uma senha for usada, ela deve ser convertida em uma chave por meio de uma função de derivação de chave de base de senha apropriada.
No README, na seção GERANDO CHAVES DE SEGURANÇA, é recomendado usar a função sodium_bin2hex(sodium_crypto_secretbox_keygen()) para gerar a senha de criptografia. O KAW utiliza variáveis de ambiente para acessar a chaves de criptografia

Certifique-se de que a aleatoriedade criptográfica seja usada quando apropriado e que não tenha sido usada uma semente de uma forma previsível ou com baixa entropia. A maioria das APIs modernas não exige que o desenvolvedor propague o CSPRNG para obter segurança.
Para gerar o NONCE da função sodium_crypto_secretbox(), é utilizado o random_bytes(SODIUM_CRYPTO_SECRETBOX_NONCEBYTES); Conforme orientação da documentação https://www.php.net/manual/en/function.sodium-crypto-secretbox

Evite funções criptográficas e esquemas de preenchimento obsoletos, como MD5, SHA1, PKCS número 1 v1.5.
A função sodium_crypto_secretbox() implementa o algoritmo XSalsa20, um dos mais modernos. https://libsodium.gitbook.io/doc/advanced/stream_ciphers/xsalsa20

Verifique de forma independente a eficácia das configurações.
[falta implementar]

A03:2021-Injection

A opção preferida é usar uma API segura, que evita usar o interpretador inteiramente, fornece uma interface parametrizada ou migra para uma ferramenta de Mapeamento Relacional de Objeto (ORMs).
Sempre utilizando o ORM do framework.

Use validação de entrada positiva ou "*safelist*" do lado do servidor. Esta não é uma defesa completa, pois muitos aplicativos requerem caracteres especiais, como áreas

de texto ou APIs para aplicativos móveis.
[não aplicável]

Para quaisquer consultas dinâmicas residuais, escape os caracteres especiais usando a sintaxe de escape específica para esse interpretador. Nota: Estruturas SQL, como nomes de tabelas, nomes de colunas e assim por diante, não podem ter escape e, portanto, nomes de estruturas fornecidos pelo usuário são perigosos. Este é um problema comum em software de elaboração de relatórios.
Sempre utilizando o ORM do framework e seguindo a recomendação do CakePHP sobre usar key/left-hand side

Use LIMIT e outros SQL de controle em consultas para evitar a divulgação em massa de registros no caso de injeção de SQL.
Aplicado limit em todas as consultas possíveis do sistema. PRs:

- https://github.com/arthusantiago/KeyAnyWhere/pull/8
- https://github.com/arthusantiago/KeyAnyWhere/pull/9

A04:2021-Insecure Design

Estabeleça e use um ciclo de vida de desenvolvimento seguro com profissionais de AppSec para ajudar a avaliar e projetar controles relacionados à segurança e privacidade.
Durante o ciclo de desenvolvimento, existem etapas para se pensar a segurança da funcionalidade.

Use Modelagem de Ameaças para autenticação crítica, controle de acesso, lógica de negócios e fluxos de chaves.
[falta implementar]

Integre a linguagem e os controles de segurança às histórias de usuários.
[falta implementar]

Integre verificações de plausibilidade em cada camada da sua aplicação (do front-end ao back-end).
[falta implementar]

Escreva testes de unidade e integração para validar se todos os fluxos críticos são resistentes ao modelo de ameaça. Compile casos de uso de sucesso e casos de uso indevido para cada camada da sua aplicação.
[falta implementar]

Separe as camadas de nível no sistema e nas camadas de rede, dependendo das necessidades de exposição e proteção.
[falta implementar]

Separe os tenants de maneira robusta por design em todas as camadas.
[falta implementar]

Limite o consumo de recursos por usuário ou serviço.
[não aplicável]

A05:2021-Security Misconfiguration

Uma plataforma mínima sem recursos, componentes, documentação e outros desnecessários. Remova ou não instale recursos e estruturas não utilizadas.
O KAW tem uma lista pequena de dependências:
https://github.com/arthusantiago/KeyAnyWhere/network/dependencies

Um processo de proteção repetível torna mais rápido e fácil implantar outro ambiente que esteja devidamente bloqueado. Os ambientes de desenvolvimento, controle de qualidade e produção devem ser todos configurados de forma idêntica, com credenciais diferentes usadas em cada ambiente. Este processo deve ser automatizado para minimizar o esforço necessário para configurar um novo ambiente seguro.
[não aplicável]

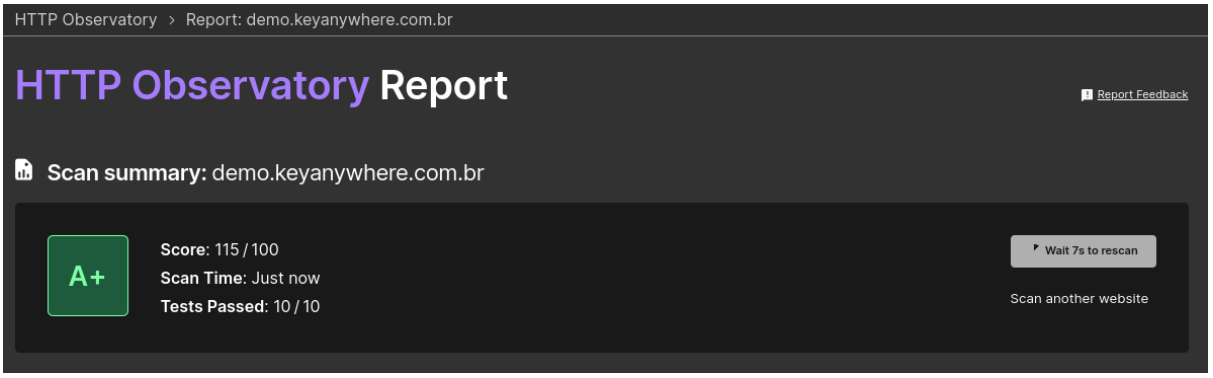
Uma tarefa para revisar e atualizar as configurações apropriadas para todas as notas de segurança, atualizações e patches como parte do processo de gerenciamento de patch (consulte A06: 2021-Componentes Vulneráveis e Desatualizados). Revise as permissões de armazenamento em nuvem (por exemplo, S3 bucket permissions).
[não aplicável]

Uma arquitetura de aplicação segmentada fornece separação eficaz e segura entre componentes ou tenants, com segmentação, conteinerização ou grupos de segurança em nuvem (ACLs).
[não aplicável]

Envio de diretivas de segurança para clientes, por exemplo, Security Headers.
Implementado nas PR abaixo:

- https://github.com/arthusantiago/KeyAnyWhere/pull/16
- https://github.com/arthusantiago/KeyAnyWhere/pull/17

Depois de implementar o CSP, o resultado do teste:



Um processo automatizado para verificar a eficácia das configurações em todos os ambientes.
[não aplicável]

A06:2021-Vulnerable and Outdated Components

Remova dependências não utilizadas, recursos, componentes, arquivos e documentação desnecessários.
O projeto tem poucas dependências instaladas.
https://github.com/arthusantiago/KeyAnyWhere/network/dependencies

Atualizar continuamente um inventário com as versões dos componentes do lado do cliente e do lado do servidor (por exemplo, estruturas, bibliotecas) e suas dependências usando ferramentas como versions, OWASP Dependency Check, retire.js, etc. Monitore continuamente fontes como Common Vulnerability and Exposures (CVE) e National Vulnerability Database (NVD) para vulnerabilidades nos componentes. Use ferramentas de análise de composição de software para automatizar o processo. Inscreva-se para receber alertas de e-mail sobre vulnerabilidades de segurança relacionadas aos componentes que você usa.
O Dependabot está ativo no github. Também está ativo a ferramenta que cria automaticamente um PR quando acha uma dependência com vulnerabilidade.

Obtenha componentes apenas de fontes oficiais por meio de links seguros. Prefira pacotes assinados para reduzir a chance de incluir um componente malicioso modificado (consulte A08: 2021-Software e Falhas de Integridade de Dados).
Utilizando o ecossistema do Composer e Packagist

Monitore bibliotecas e componentes sem manutenção ou que não criem patches de segurança para versões anteriores. Se o patch não for possível, considere implantar um patch virtual para monitorar, detectar ou proteger contra o problema descoberto.
[falta implementar]

A07:2021-Identification and Authentication Failures

Sempre que possível, implementar a autenticação multifator para evitar preenchimento automatizado de credenciais, força bruta e credenciais roubadas
Implementado o TFA.

Não permita ou implante nenhuma credencial padrão, especialmente para usuários administradores.
Não existe usuário/senha padrão. Esses dados são criados na configuração inicial da ferramenta.

Implementar verificações de senha fraca, como testar novas ou alteradas contra a lista das 10.000 piores senhas.

Não é possível salvar uma senha fraca, na criação de uma senha é exigido um nível mínimo de segurança e entropia.
Existe um banco de dados com quase 1 milhão de senhas inseguras que é utilizado para avisar o usuário antes de cadastrar.

Alinhe o comprimento da senha, a complexidade e as políticas de rotação com Instituto Nacional de Padrões e Tecnologia (NIST) as diretrizes do 800-63b na seção 5.1.1 para segredos memorizados ou outras políticas de senha modernas e baseadas em evidências.
Para senhas de usuário o comprimento mínimo é 12 caracteres. Ao cadastrar uma nova entrada o gerador de senha gera por padrão uma de 14 caracteres.

Use um gerenciador de sessão integrado, seguro do lado do servidor que gere um novo ID de sessão aleatória com alta entropia após o login. O identificador de sessão não deve estar na URL, deve ser armazenado com segurança e invalidado após o logout.
Utiliza o gerenciador de sessão do Cake e ela é salva no banco de dados. Utilizado um alto comprimento de sessão, e seu ID é gerado utilizando letras, números e caracteres especiais. O ID da sessão não é usado na URL.

Certifique-se de que o registro, a recuperação de credenciais e os caminhos da API sejam protegidos contra ataques de enumeração de contas usando a mesma mensagem para todos os resultados.
[não aplicável]
Não aplicável ao KAW, já que todos os recursos são privados e acessíveis somente depois de autenticação.

Limite ou atrase cada vez mais as tentativas de login com falha, mas tome cuidado para não criar um cenário de negação de serviço. Registrar todas as falhas e alertar os administradores quando o preenchimento de credenciais, força bruta ou outros ataques são detectados.
Implementado um sistema de logs que registra atividades suspeitas e possíveis tentativas de invasão. O usuário pode errar no máximo 3 vezes a senha, depois disso o IP dele é bloqueado.

A08:2021-Software and Data Integrity Failures

Use assinaturas digitais ou mecanismos similares para verificar se o software ou os dados são provenientes da fonte esperada e não foram alterados.
[não aplicável]

Certifique-se de que as bibliotecas e dependências, como npm ou Maven, estão consumindo repositórios confiáveis. Se você tiver um perfil de risco mais alto, considere hospedar um repositório interno conhecido como bom que foi examinado.
A gestão de dependências é feita utilizando o Composer. Poucas dependências são utilizadas, diminuindo a superfície de ataque.

Certifique-se de que uma ferramenta de segurança da cadeia de suprimentos de software, como OWASP Dependency Check ou OWASP CycloneDX, é usada para verificar se os componentes não contêm vulnerabilidades conhecidas.
O Dependabot está habilitado no github. O Dependabot faz uma verificação semanalmente por notificações de vulnerabilidade nos pacotes instalados, e cria uma pull request para atualizar a dependência.

Certifique-se de que haja um processo de revisão para mudanças de código e configuração para minimizar a chance de que código ou configuração maliciosos possam ser introduzidos no seu pipeline de software.
O código novo só pode ser mergeado na branch main via PR. Isso possibilita que a análise do código possa acontecer antes. Durante o processo de desenvolvimento existem etapas para analisar as implicações de segurança antes da implementação e depois da implementação.

Certifique-se de que seu pipeline de CI/CD tenha uma segregação adequada, configuração e controle de acesso para garantir a integridade do código que flui através dos processos de construção e implantação.
[não aplicável]

Certifique-se de que dados serializados não assinados ou não criptografados não sejam enviados a clientes não confiáveis sem algum tipo de verificação de integridade ou assinatura digital para detectar adulteração ou retransmissão dos dados serializados.
Nas solicitações é adicionado um token segurança no header, para proteger contra ataques do tipo CSRF. Também é feita verificação dos dados enviados do frontend para o backend via requisição assíncrona.

A09:2021-Security Logging and Monitoring Failures

Garantir que todas as falhas de login, controle de acesso e validação de entrada no lado do servidor possam ser registradas com contexto de usuário suficiente para identificar contas suspeitas ou maliciosas e mantidas por tempo suficiente para permitir análise forense atrasada.
O sistema de log implementado garante esse critério.

Garantir que os logs sejam gerados em um formato que as soluções de gerenciamento de logs possam facilmente consumir.
Informações salvas no Banco de dados da ferramenta.

Garantir que os dados de log sejam codificados corretamente para evitar injeções ou ataques nos sistemas de registro ou monitoramento.
O registro de log não é permitido pelo frontend.

Garantir que transações de alto valor tenham uma trilha de auditoria com controles de integridade para evitar adulteração ou exclusão, como tabelas de banco de dados somente para adição ou similares.
[não aplicável]

As equipes de DevSecOps devem estabelecer monitoramento e alerta efetivos para que atividades suspeitas sejam detectadas e respondidas rapidamente.
O Minha Segurança é uma ferramenta interna do KAW. Ela procura dar uma visão geral de como estão as atividades no sistema e o que pode ser melhorado na segurança do usuário.

Estabelecer ou adotar um plano de resposta e recuperação de incidentes, como o National Institute of Standards and Technology (NIST) 800-61r2 ou posterior.
[não aplicável]

A10:2021-Server-Side Request Forgery

Higienize e valide todos os dados de entrada fornecidos pelo cliente;
Todo dado que é enviado ao servidor é validado na model e em alguns casos, na controller.
PR de referencia: https://github.com/arthusantiago/KeyAnyWhere/pull/10/files

Aplique o esquema de URL, porta e destino com uma lista de permissões positiva;
[não aplicável]

Não envie a resposta crua ao cliente
[não aplicável]

Desabilite redirecionamentos de HTTP;
[não aplicável]

Tenha cuidado com a consistência URL contra ataques que mirem a resolução de nomes através do DNS e CWE-367.
[não aplicável]

Não implemente outros serviços de segurança relevantes em sistemas frontais (por exemplo, OpenID). Controle o tráfego local nesses sistemas (por exemplo, localhost)
[não aplicável]

Para frontends com grupos de usuários dedicados e gerenciáveis, use criptografia de rede (por exemplo, VPNs) em sistemas independentes para as necessidades de proteção muito altas.
[não aplicável]