CRIANDO APIS SEGURAS

QUEM SOU EU?

VINÍCIUS CAMPITELLI

- Co-fundador do Curseduca
- Membro do PHPSP
- GitHub e Twitter: @vcampitelli
- Slides: viniciuscampitelli.com

AGENDA

AGENDA

- Autenticação e autorização

AGENDA

- Autenticação e autorização
- Melhores práticas com access tokens

AGENDA

- Autenticação e autorização
- Melhores práticas com access tokens
- Throttling e Rate Limiting

AGENDA

- Autenticação e autorização
- Melhores práticas com access tokens
- Throttling e Rate Limiting
- Camuflagem de IDs sequenciais

AGENDA

- Autenticação e autorização
- Melhores práticas com access tokens
- Throttling e Rate Limiting
- Camuflagem de IDs sequenciais
- Criptografando e assinando requisições e respostas

AUTENTICAÇÃO E AUTORIZAÇÃO

AUTENTICAÇÃO

> *É o ato de estabelecer ou confirmar algo (ou alguém) como autêntico (...)*

Fonte: Wikipedia

AUTENTICAÇÃO

> *É o ato de estabelecer ou confirmar algo (ou alguém) como autêntico (...)*

Fonte: Wikipedia

AUTENTICAÇÃO

Para autenticar algo ou alguém, precisamos de uma credencial que o identifique.
Ela precisa ser emitida por algo (ou alguém) que o autorizador confie.

“O QUE VOCÊ SABE”

Autenticação baseada no conhecimento

“O QUE VOCÊ SABE”

Autenticação baseada no conhecimento

Exemplos:

- Usuário e senha
- Certificado digital
- Client ID e Client Secret

“O QUE VOCÊ TEM”

Autenticação baseada na propriedade

“O QUE VOCÊ TEM”

Autenticação baseada na propriedade

Exemplos:

“O QUE VOCÊ TEM”

Autenticação baseada na propriedade

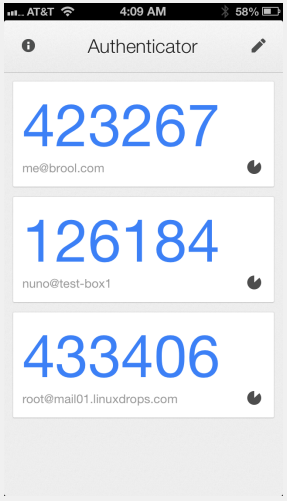
Exemplos:

Token ou cartão físico

“O QUE VOCÊ TEM”

Autenticação baseada na propriedade

Exemplos:



Token ou cartão físico

Código via aplicativo

“O QUE VOCÊ É”

Autenticação baseada na característica

“O QUE VOCÊ É”

Autenticação baseada na característica

Exemplos:

- Impressão digital
- Identificação de íris
- Reconhecimento facial
- Reconhecimento de voz

AUTORIZAÇÃO

> *É garantir que apenas usuários autorizados consumam os recursos protegidos de um sistema computacional*

Fonte: Wikipedia

AUTORIZAÇÃO

> *É garantir que apenas usuários autorizados consumam os recursos protegidos de um sistema computacional*

Fonte: Wikipedia

AUTORIZAÇÃO

Após saber quem é o cliente no processo de Autenticação, preciso agora entender o que ele pode fazer. Quais recursos ele pode consumir? Por quanto tempo?

COMO EFETUAR AUTENTICAÇÃO E AUTORIZAÇÃO?

COMO EFETUAR AUTENTICAÇÃO E AUTORIZAÇÃO?

Para autenticação, podemos utilizar o OpenID ou implementar nosso próprio sistema *(por exemplo, através do banco de dados)*.

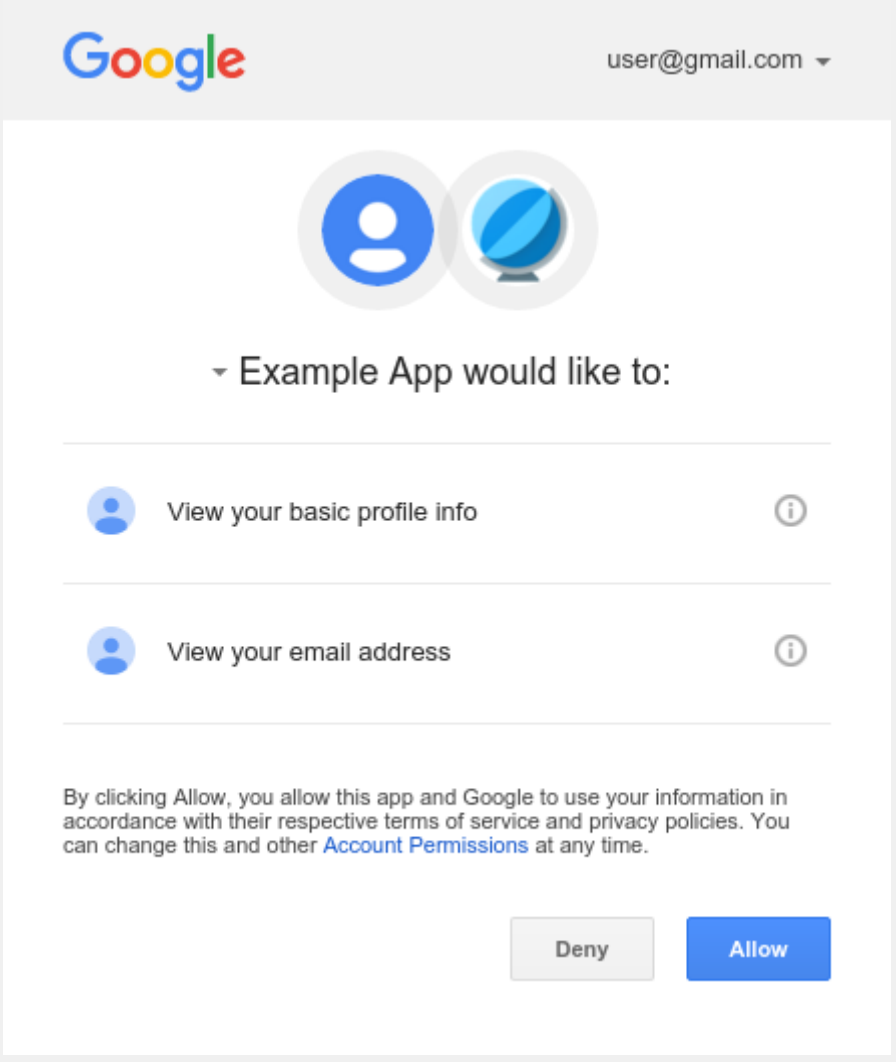
COMO EFETUAR AUTENTICAÇÃO E AUTORIZAÇÃO?

Para autenticação, podemos utilizar o OpenID ou implementar nosso próprio sistema *(por exemplo, através do banco de dados)*.

OAuth 2.0 é o protocolo mais conhecido de **autorização**. Ele, por si só, não contempla processos de autenticação.

FAMOSA TELA DE AUTORIZAÇÃO VIA OAUTH 2

Utilizando o *Grant* de *Authorization Code*
(saiba como escolher o Grant Type certo nesse link)



Referência: Shotgun Debugging

REFERÊNCIAS

- OpenID
- Auth0
- OAuth 2.0
- Para PHP
  - The PHP League: OAuth 2.0 Server
  - Meu slides sobre OAuth2

MELHORES PRÁTICAS COM *ACCESS TOKENS*

O QUE É UM *ACCESS TOKEN*?

O QUE É UM *ACCESS TOKEN*?

> *É um objeto que descreve o contexto de segurança de um processo (...)*

Fonte: MSDN

E NO MUNDO DE APIS?

O processo de autenticação e autorização pode ser caro computacionalmente *(consultas a banco de dados, servidores de autenticação externos, criptografia de senha etc)*

E NO MUNDO DE APIS?

Ao invés de sempre termos essa carga a cada requisição a nossos servidores, podemos ter um único *endpoint* de autenticação / autorização que irá realizar essa etapa uma vez e gerar um *access token* com as informações necessárias para autenticar e autorizar mais rapidamente o cliente nas requisições seguintes

E NO MUNDO DE APIS?

Esse *token* deve possuir um tempo de vida limitado, tanto para diminuir efeitos colaterais caso haja um vazamento quanto porque sua informação pode ficar desatualizada

E NO MUNDO DE APIS?

Esse segundo processo pode ser automatizado através de *refresh tokens*

E NO MUNDO DE APIS?

Esse segundo processo pode ser automatizado através de *refresh tokens*

Eles são utilizados para regerar um *access token* automaticamente, sem necessitar da intervenção manual do usuário

COMO DEIXAR OS *TOKENS* SEGUROS?

COMO DEIXAR OS *TOKENS* SEGUROS?

- Utilizando algoritmos de criptografia seguros e com assinatura;

COMO DEIXAR OS *TOKENS* SEGUROS?

- Utilizando algoritmos de criptografia seguros e com assinatura;
  - AES-256-GCM, ChaCha20+Poly1305, Ed25519 entre outros

COMO DEIXAR OS *TOKENS* SEGUROS?

- Utilizando algoritmos de criptografia seguros e com assinatura;
  - AES-256-GCM, ChaCha20+Poly1305, Ed25519 entre outros
- Tendo cuidado com implementações de bibliotecas com falhas de segurança;

COMO DEIXAR OS *TOKENS* SEGUROS?

- Utilizando algoritmos de criptografia seguros e com assinatura;
  - AES-256-GCM, ChaCha20+Poly1305, Ed25519 entre outros
- Tendo cuidado com implementações de bibliotecas com falhas de segurança;
  - Exemplo: JWT com alg=none

COMO DEIXAR OS *TOKENS* SEGUROS?

- Não faça *commit* de chaves e outros segredos, nem deixe-os *hardcoded* na aplicação

COMO DEIXAR OS *TOKENS* SEGUROS?

- Não faça *commit* de chaves e outros segredos, nem deixe-os *hardcoded* na aplicação
- Não guarde informações muito sensíveis *(como senhas e outras credenciais)*

COMO DEIXAR OS *TOKENS* SEGUROS?

- Não faça *commit* de chaves e outros segredos, nem deixe-os *hardcoded* na aplicação
- Não guarde informações muito sensíveis *(como senhas e outras credenciais)*
- Se precisar armazenar do lado do cliente, utilize cookies seguros *(flags* `httpOnly` *e* `secure`*)* ao invés do *LocalStorage*

COMO DEIXAR OS *TOKENS* SEGUROS?

- Apenas trafegue *tokens* no cabeçalho ou corpo da requisição, nunca na URL

COMO DEIXAR OS *TOKENS* SEGUROS?

- Apenas trafegue *tokens* no cabeçalho ou corpo da requisição, nunca na URL
- Não exiba informações do *token* em seus *logs*

EXEMPLO DE IMPLEMENTAÇÃO

```php
use Lcobucci\JWT\Builder;

$time = time();
$token = (new Builder())
    ->issuedBy('http://example.com') // Configures the issuer (iss claim)
    ->permittedFor('http://example.org') // Configures the audience (aud claim)
    ->identifiedBy('4f1g23a12aa', true) // Configures the id (jti claim), replicating as
    ->issuedAt($time) // Configures the time that the token was issue (iat claim)
    ->canOnlyBeUsedAfter($time + 60) // Configures the time that the token can be used (
    ->expiresAt($time + 3600) // Configures the expiration time of the token (exp claim)
    ->withClaim('uid', 1) // Configures a new claim, called "uid"
    ->getToken(); // Retrieves the generated token
```

lcobucci/jwt

BIBLIOTECAS

- JSON Web Tokens: implementação mais conhecida de tokens
  - lcobucci/jwt
  - firebase/php-jwt
- PASETO: implementação com design "mais seguro" por padrão
  - paragonie/paseto

REFERÊNCIAS

- Meu slides sobre a libsodium, biblioteca moderna de criptografia
- Symfony Guard

THROTTLING E RATE LIMITING

O QUE É THROTTLING?

O QUE É THROTTLING?

É a desaceleração intencional do processamento de uma requisição para prevenir sobrecarga do servidor

O QUE É THROTTLING?

Imagine um *endpoint* que consuma grande recursos computacionais *(como, por exemplo, o processo de autenticação e autorização descritos anteriormente)*

O QUE É THROTTLING?

Se um agente malicioso identificar esse recurso, ele pode se tornar muito visado para ataques, causando sobrecarga no nosso servidor e podendo gerar paralisação e até queda total do serviço

O QUE É THROTTLING?

Nesses casos, podemos configurar que o servidor irá aceitar somente `x` requisições em algum(ns) *endpoint(s)* por um certo período de tempo

O QUE É THROTTLING?

Após esse valor `x`, as requisições entrarão em uma fila para serem processadas alguns instantes depois, assim que as primeiras tiverem sido liberadas

MAS ENTÃO O QUE É RATE LIMITING?

MAS ENTÃO O QUE É RATE LIMITING?

Ao contrário do *Throttling*, em a requisição será processada com um atraso, *Rate Limiting* é fazer o servidor se recusar a responder após um certo número de requisições

MAS ENTÃO O QUE É RATE LIMITING?

Nesses casos, devemos emitir um status HTTP `503 Service Unavailable` ou `429 Too Many Requests`

COMO ESSAS DUAS PRÁTICAS SE RELACIONAM ENTRE SI?

Geralmente, primeiro aplicamos uma política de *Throttling* para desacelerar o processamento nas `x` primeiras requisições, e após um outro número `y`, iremos simplesmente parar de responder

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Você deve configurá-las em seu servidor Web *(por exemplo, Apache, IIS ou nginx)* ou no seu serviço de DNS *(por exemplo, o Cloudflare)*

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Tomando como exemplo o *nginx*, utilizaremos o módulo `ngx_http_limit_req_module` *(que implementa o mais conhecido algoritmo desse tipo de prática, o Leaky Bucket)*

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

```
limit_req_zone $binary_remote_addr zone=login_zone:10m rate=10r/s;

server {
    location /login {
        limit_req zone=login_zone burst=5;

        # outras configurações padrões do meu bloco location...
    }
}
```

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

O código anterior irá proteger nosso *endpoint* `/login`, permitindo apenas 10 requisições por segundo por IP *(ou seja, 1 a cada 100ms)*

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Mas também permitimos um *burst*, permitindo que 5 requisições extras sejam colocadas na fila de processamento antes de serem enviadas para o `upstream`

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Nesse caso, se recebermos de um mesmo IP 10 requisições em um período de 100ms, o que ocorrerá?

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Nesse caso, se recebermos de um mesmo IP 10 requisições em um período de 100ms, o que ocorrerá?

- A 1ª requisição será processada instantaneamente

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Nesse caso, se recebermos de um mesmo IP 10 requisições em um período de 100ms, o que ocorrerá?

- A 1ª requisição será processada instantaneamente
- As 2ª, 3ª, 4ª, 5ª e 6ª requisições serão colocadas em uma fila e serão processadas sequencialmente após o término da anterior

ONDE E COMO CONFIGURAR ESSAS POLÍTICAS?

Nesse caso, se recebermos de um mesmo IP 10 requisições em um período de 100ms, o que ocorrerá?

- A 1ª requisição será processada instantaneamente
- As 2ª, 3ª, 4ª, 5ª e 6ª requisições serão colocadas em uma fila e serão processadas sequencialmente após o término da anterior
- As 7ª, 8ª, 9ª e 10ª requisições serão negadas

REFERÊNCIAS

- Como implementar no CloudFlare
- Como implementar no nginx

CAMUFLAGEM DE IDS SEQUENCIAIS

O QUE É UM ID SEQUENCIAL E POR QUE ESCONDÊ-LO?

Ao criarmos *endpoints* no formato `/user/1`, `/user/2`, `/user/3` etc, estamos fornecendo uma informação muito valiosa sobre a quantidade de registros que possuímos daquela entidade

O QUE É UM ID SEQUENCIAL E POR QUE ESCONDÊ-LO?

Isso pode ser utilizado tanto por alguém mal intencionado quanto por algum competidor.

O QUE É UM ID SEQUENCIAL E POR QUE ESCONDÊ-LO?

Isso pode ser utilizado tanto por alguém mal intencionado quanto por algum competidor.

Cuidado com espionagem industrial! 

COMO RESOLVER?

VERSÃO STANDARD

COMO RESOLVER?

VERSÃO STANDARD

COMO RESOLVER?

- Crie um segundo campo na tabela com um valor único para aquele registro

VERSÃO STANDARD

COMO RESOLVER?

- Crie um segundo campo na tabela com um valor único para aquele registro
  - Esse valor pode ser um GUID, um hash do ID incremental ou um valor aleatório...

VERSÃO STANDARD

COMO RESOLVER?

- Crie um segundo campo na tabela com um valor único para aquele registro
  - Esse valor pode ser um GUID, um hash do ID incremental ou um valor aleatório...
- Então, transforme o recurso `/user/<IdIncremental>` em `/user/<Identificador>`

VERSÃO STANDARD

COMO RESOLVER?

REFERÊNCIAS

- Funções para gerar valores aleatórios:
  - random_bytes() para PHP 7
    - Para PHP 5, use o polyfill paragonie/random_compat
- Funções para gerar hash:
  - hash_hmac()
- Artigo no blog da Paragon

VERSÃO STANDARD

COMO **NÃO** RESOLVER?

Funções que **não devem** ser utilizadas:

- Para gerar valores aleatórios:
  - ~~rand()~~
  - ~~mt_rand()~~
  - ~~uniqid()~~
- Para gerar hash:
  - ~~sha1()~~
  - ~~md5()~~

COMO RESOLVER?

COMO RESOLVER?

VERSÃO HARDCORE

COMO RESOLVER?

Utilizando a biblioteca paragonie/ciphersweet, que deriva uma chave para cada coluna a partir de uma chave-mestra

REFERÊNCIA

- Artigo no blog da Paragon

CRIPTOGRAFANDO E ASSINANDO REQUISIÇÕES E RESPOSTAS

PRECAUÇÃO EXTRA

Para usuários avançados e sistemas críticos, podemos adicionar uma camada extra de proteção: criptografar os dados da requisição e da resposta

PRECAUÇÃO EXTRA

> *Mas, espera aí! Eu já uso HTTPS. Meu dados já estão sendo criptografados usando TLS.*

PRECAUÇÃO EXTRA

Após o estabelecimento do protocolo, realmente os dados transitados estão criptografados, mas ainda podemos sofrer um ataque de Man in the Middle

MAN IN THE MIDDLE

- Existem diversos *hops* entre o cliente e o servidor. Quem garante que todos os *gateways* do caminho são seguros?

MAN IN THE MIDDLE

- Existem diversos *hops* entre o cliente e o servidor. Quem garante que todos os *gateways* do caminho são seguros?
- Se você não for o administrador da sua rede local, alguém pode instalar uma Autoridade Certificadora em sua máquina e emitir certificados próprios

MAN IN THE MIDDLE

- Nem todos os sites utilizam HSTS

MAN IN THE MIDDLE

- Nem todos os sites utilizam HSTS
  - Ou seja, pode haver uma conexão HTTP antes do redirecionamento para HTTPS pelo servidor

MAN IN THE MIDDLE

- Nem todos os sites utilizam HSTS
  - Ou seja, pode haver uma conexão HTTP antes do redirecionamento para HTTPS pelo servidor
- Teoria da conspiração: grandes empresas possuem *backdoors* para bisbilhotar seu tráfego

PRECAUÇÃO EXTRA

Antes de seguir nessa etapa, faça um Threat Modeling e certifique-se que realmente valhe a pena

PRECAUÇÃO EXTRA

Antes de seguir nessa etapa, faça um Threat Modeling e certifique-se que realmente valhe a pena

Compensa instalar um sistema de segurança de última geração em um depósito que não possui algo de extremo valor dentro?

PRECAUÇÃO EXTRA

> *SIM! Meu sistema é crítico e preciso disso.*

PRECAUÇÃO EXTRA

> *SIM! Meu sistema é crítico e preciso disso. Ou eu não confio em grandes corporações.*

PRECAUÇÃO EXTRA

Você pode exigir que toda requisição tenha seu corpo criptografado pelo cliente, utilizando um segredo em que só vocês dois sabem

PRECAUÇÃO EXTRA

Por exemplo, você pode utilizar uma terceira informação fora o Client ID e o Client Secret e usá-la como chave assimétrica, ou fornecer um certificado digital para seu cliente

PRECAUÇÃO EXTRA

Do lado do servidor, você utiliza essa informação sigilosa para descriptografar o corpo da requisição - se falhar, você emite um erro e para o processamento

PRECAUÇÃO EXTRA

Entretanto, esse processo de descriptografia pode ser muito custoso operacionalmente, podendo ser um outro ponto de ataque DoS *(lembre-se do que falamos na seção de* Throttling e Rate Limiting*)*. Você pode, então, utilizar um processo de assinatura digital além da criptografia.

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura
  - Envia a requisição somente com os dados criptografados

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura
  - Envia a requisição somente com os dados criptografados
- Para cada resposta seguinte, ele deve:

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura
  - Envia a requisição somente com os dados criptografados
- Para cada resposta seguinte, ele deve:
  - Verificar a assinatura

FLUXO DE CRIPTOGRAFIA E ASSINATURA

O cliente...

- Autentica na sua API fornecendo as credenciais
- Para cada requisição, assina e criptografa os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura
  - Envia a requisição somente com os dados criptografados
- Para cada resposta seguinte, ele deve:
  - Verificar a assinatura
  - Descriptografar o corpo

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura
- Descriptografar o corpo

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura
- Descriptografar o corpo
- Ao responder à requisição, deve assinar e criptografar os dados *(combinação do corpo da requisição + URL)*;

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura
- Descriptografar o corpo
- Ao responder à requisição, deve assinar e criptografar os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura
- Descriptografar o corpo
- Ao responder à requisição, deve assinar e criptografar os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura

FLUXO DE CRIPTOGRAFIA E ASSINATURA

A cada requisição recebida, o servidor deve...

- Verificar a assinatura
- Descriptografar o corpo
- Ao responder à requisição, deve assinar e criptografar os dados *(combinação do corpo da requisição + URL)*;
  - Utiliza um algoritmo de hash para assinar os dados
  - Criptografa os dados mais a assinatura
  - Envia a requisição somente com os dados criptografados

CONCLUINDO...

CONCLUINDO...

- Segurança não é trivial

CONCLUINDO...

- Segurança não é trivial
  não é fácil

CONCLUINDO...

- Segurança não é trivial
  não é fácil
  não é para leigos

CONCLUINDO...

- Segurança não é trivial
  não é fácil
  não é para leigos
- Nenhum sistema operante é **invencível**

CONCLUINDO...

- Segurança não é trivial
  não é fácil
  não é para leigos
- Nenhum sistema operante é **invencível**,
  mas você **deve dificultar** o trabalho de atacantes

CONCLUINDO...

- Segurança não é trivial
  não é fácil
  não é para leigos
- Nenhum sistema operante é **invencível**,
  mas você **deve dificultar** o trabalho de atacantes
- Saiba equilibrar e avaliar a relação de custo-benefício entre segurança e usabilidade

OBRIGADO!

SLIDES

viniciuscampitelli.com

CONTATO

GitHub e Twitter
@vcampitelli