



CRYPTOGRAPHIX

# Desenvolvendo Aplicações Seguras

... COM WEB-CRYPTO, NODE AND SOFTWARE LIVRE

# Objetivos do Workshop

- ▶ Introduzir os **pilares de segurança** de aplicações
- ▶ Apresentar as tecnologias de **criptografia para web**
- ▶ Ensinar como **estruturar uma aplicação segura**
- ▶ Demonstrar a implementação de **camadas de segurança** numa aplicação insegura
- ▶ Provocar **reflexões sobre a segurança** de aplicações web modernas



# Caso Exemplo: Secure-BOX

## ► A Empresa

**S-BOX** (Secure-Box) é uma *startup* que deseja oferecer um serviço de armazenamento seguro de documentos e arquivos na nuvem.

## ► O Problema

A equipe da **S-BOX** não possui nenhuma experiência em projetar ou construir aplicações seguras.

## ► Nosso Desafio

*Projetar e construir* um serviço de nuvem e uma web-aplicação, usando tecnologias de criptografia modernas e aplicando as melhoras práticas de segurança de aplicações.



# Agenda



Serviço nuvem inseguro	30
Pilares de segurança	30
Corrigindo as falhas	30
Serviço nuvem mais seguro	60
Avaliações e discussões	30

Apresentando ...

# APLICAÇÃO IN-BOX ( INSECURE-BOX ;)

# Aplicação IN-BOX

**Serviço na nuvem** - armazenamento ilimitado de documentos para usuários registrados.

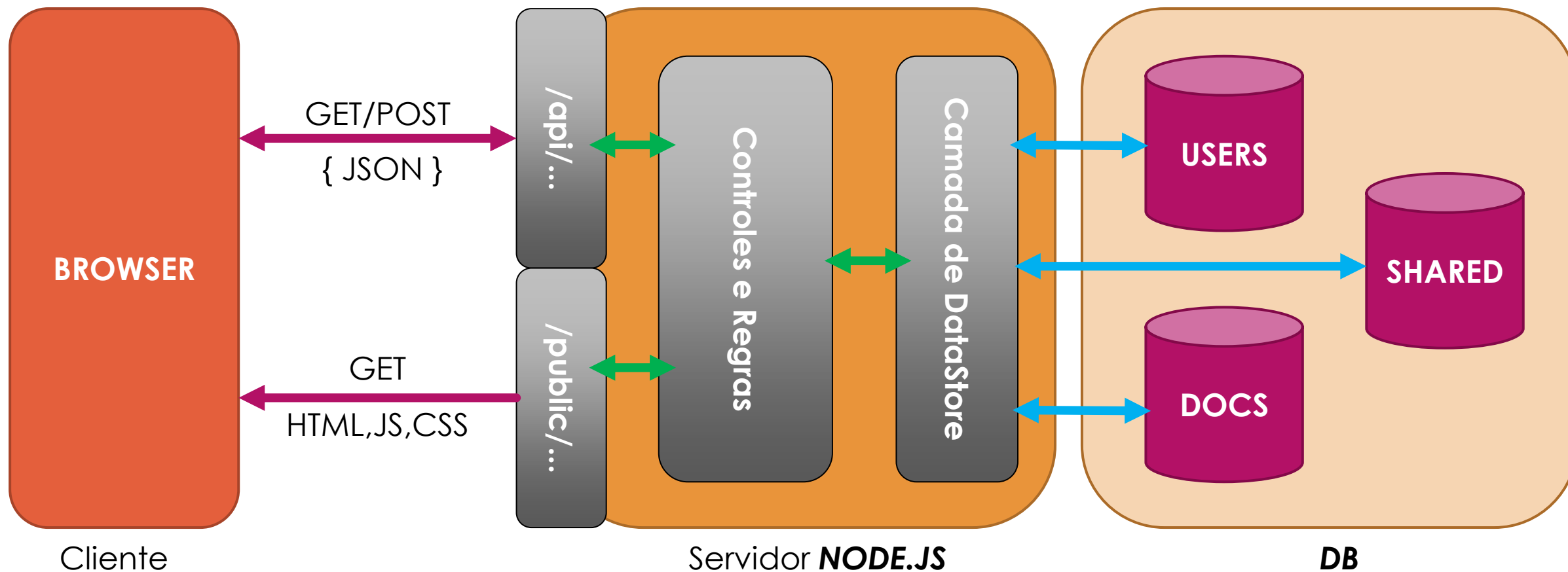
## Regras de Operação

- Cada usuário possui uma área privativa onde pode armazenar documentos.
- O usuário pode listar, criar, abrir, atualizar, deletar ou compartilhar documentos.
- Acesso a um documento é restrito ao seu dono, a menos que tenha sido compartilhado.

## WEB Interface (API)

/api/users	Registrar novos usuários;
/api/documents	Armazenar documentos (objetos) no formato JSON;
/api/shared	Compartilhar documentos com outros usuários;

# Arquitetura do IN-BOX



# Ambiente de Teste



<http://labs.cryptographix.org/workshops/2016-07-13/01-insecure-box>

No Chrome: tecle F12 (cmd-X-I no MAC)

No Firefox: Instale o plugin



Aba 'Rede/Network', para monitorar as comunicações com o servidor



<https://github.com/cryptographix/workshop-designing-a-secure-box-web-application>  
pasta: /01-insecure-box



# Registrar um Usuário

PUT /api/users/

```
{  
  username: 'fis117',  
  name:    'FISL Test User',  
  email:   'fis117@fisl.org.br',  
  password: 'P@rtoA!egre2016'  
}
```

RESPONSE OK (200)

```
{  
  username: 'fis117',  
  id: 'xyz1234438753abcdhg',  
  name:    'FISL Test User',  
  email:   'fis117@fisl.org.br',  
  password: 'P@rtoA!egre2016',  
}
```

id criado para  
este usuário

**OOPS!**  
Retornou a senha

# Login de Usuário

- ▶ Servidor não mantém sessão
- ▶ Cookie enviado ao cliente

The screenshot displays the network tab of a web browser's developer tools. On the left, a list of network requests is shown, with the 'login' request selected. The main panel on the right shows the details of this request. The 'Request Method' is POST, the 'Status Code' is 200 OK, and the 'Remote Address' is 54.207.1.69:80. The 'Response Headers' section is expanded, showing the following headers: Connection: keep-alive, Content-Length: 16, Content-Type: application/json; charset=utf-8, Date: Sun, 10 Jul 2016 16:39:56 GMT, ETag: W/"10-c2RoY+nt7m8FOksx1YjAhg", Server: nginx/1.8.1, Set-Cookie: user=fisl; Path=/, and X-Powered-By: Express. The 'Set-Cookie' header is circled in red. The 'Request Headers' section is also expanded, showing Accept: application/json, text/javascript, \*/\*; q=0.01, Accept-Encoding: gzip, deflate, and Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4.

Method	URL	Status	Size
logout	/workshops/2016-07-14/01-in...	200 OK	16 B
login	/workshops/2016-07-14/01-in...	200 OK	16 B
users	/workshops/2016-07-14/01-in...	200 OK	16 B

**Response Headers** [view source](#)

- Connection: keep-alive
- Content-Length: 16
- Content-Type: application/json; charset=utf-8
- Date: Sun, 10 Jul 2016 16:39:56 GMT
- ETag: W/"10-c2RoY+nt7m8FOksx1YjAhg"
- Server: nginx/1.8.1
- Set-Cookie: user=fisl; Path=/
- X-Powered-By: Express

**Request Headers** [view source](#)

- Accept: application/json, text/javascript, \*/\*; q=0.01
- Accept-Encoding: gzip, deflate
- Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4



## Pontos de Atenção

### 1. Como a senha sobe para o servidor?

- ▶ *Conexão WIFI*
- ▶ *Internet é uma rede pública*



HTTPS

### 2. Como a senha está gravada no banco de dados?

- ▶ *Acesso indevido*
- ▶ *Invasão do servidor*
- ▶ *Injeção de SQL*



PASSWORD  
HASHING



## Pontos de Atenção

### 3. Porque use-se cookie?

- ▶ *Servidor precisa saber quem está logado*



COOKIES DE  
SESSÃO

### 4. Como o servidor pode confiar no cookie?

- ▶ *Meio para comprovar user?*



JSON WEB  
TOKEN



## Pontos de Atenção

### 5. O documento está seguro no servidor?

- ▶ *Acesso indevido à base?*
- ▶ *Erro de implementação da api?*
- ▶ *Operador de datacenter mal-intencionado*



FULL-DISK  
ENCRYPTION



ENCRIPTAÇÃO  
CLIENTE

### 6. Como fazer compartilhamento seguro?

- ▶ *Falha no controle de permissões?*
- ▶ *Ataque de personificação?*



ENVELOPES  
SEGUROS

# Conclusão: ENCRENCA

*Está valendo o nome “IN[secure]-BOX”*

*Lembrete: Ataques 'clássicos'*

*SQL Injection, CSRF (Cross-site Request Forgery),  
XSS (Cross-Site Scripting), Falhas de código, ...*

<https://www.owasp.org>



E agora ...



# SEGURANÇA O QUE É?

# Missão: Proteger Dados

*Dados são como água, existem em diferentes Estados*

*Em repouso*

*Em trânsito*

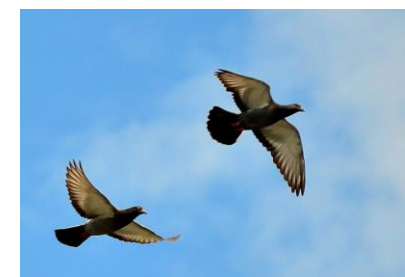
*Em uso*



*Proteção de Dados é Difícil*

*Uma cópia é igual o original*

*Uma vez solta, não volta*





# Modelos de Proteção

## Castelos:

- **Proteção: barreira física e/ou lógica**

Muralhas

Portões

Cofres

- **Proteger as entradas**
- **Guardar chaves, senhas, ...**
- **O QUE, ONDE, QUEM pode acessar?**

## Códigos:

- **Proteção: transformação da informação**

Confidencialidade

Integridade

Autenticidade de  
dados

Identificação

Segura

Autenticidade de  
pessoas e máquinas

- **Chave é “zipar a necessidade de proteção”**

# Segurança = Proteção

## ► Criptografia

*Confidencialidade*

*Integridade*

*Autenticidade*

## ► Chaves criptográficas

*Problema: guarda das chaves*



# Segurança = Proteção

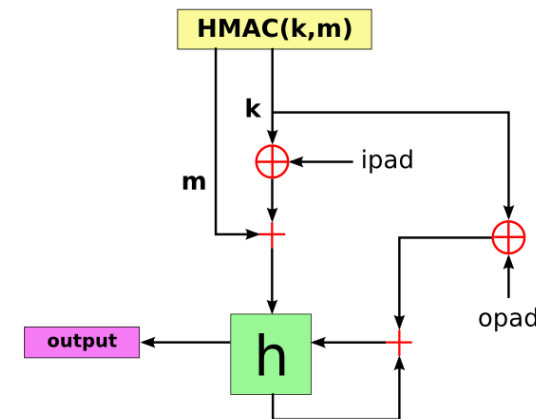
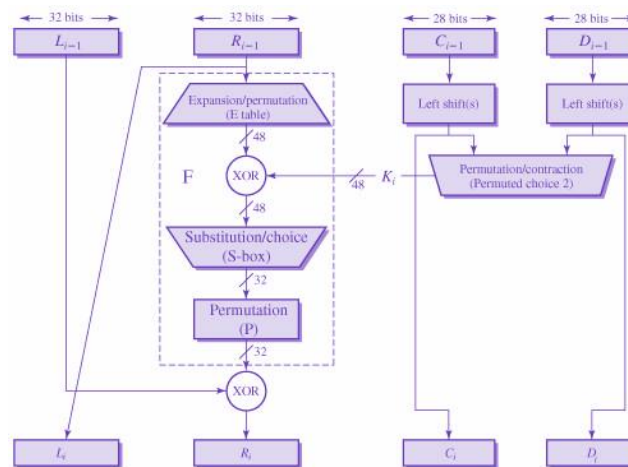
## ► Criptografia em Camadas

*Primitivas*

*Serviços Criptográficos*

*Serviços de Segurança*

*Cola e superbonder*



# Cenários e Técnicas



Login to Wikipedia

Username:

Password:

☒ Remember me (up to 30 days)

[Forgot your password?](#)

Login ▶

 Sign in using our secure server

**Autenticação  
Presencial**



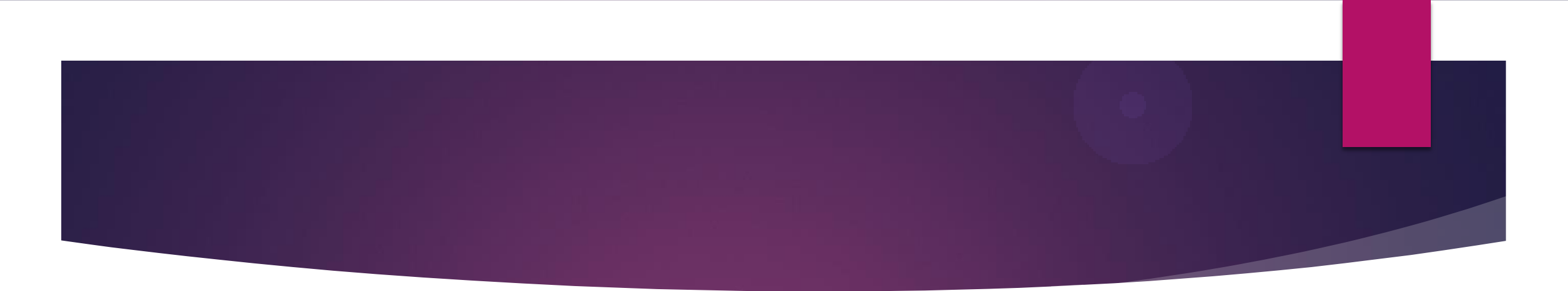
**Proteção Local**



**Autenticação  
Remota M2M**



**Proteção em Trânsito**  
**Datagrama**  
**Canal seguro**



# ADICIONANDO SEGURANÇA DO LADO SERVIDOR

# Ambiente de Teste



<http://labs.cryptographix.org/workshops/2016-07-13/02-secure-box-normal>



<https://github.com/cryptographix/workshop-designing-a-secure-box-web-application>

pasta: /02-secure-box-normal

# HTTPS - Encriptação da comunicação

## O QUE É:

*Canal seguro entre cliente e servidor*

## PROTEÇÃO:

*Captura da senha e demais dados em trânsito*

## CUIDADOS:

*MITM*

*=>*

*Certificate PINNING*

*Downgrading*

*=>*

*Configuração da TLS*

*Chaves*

*=>*

*Permissões, arquitetura*





# Password Hashing

## O QUE É:

*Guardar a senha 'hasheada' com funções especiais (KDF)*

## PROTEÇÃO:

*Captura da base de senhas*

## CUIDADOS

*Algoritmo*

*=> PBKDF2, Bcrypt, Scrypt, Argon2*

*Dificuldade*

*=> Aumentar trabalho do atacante*

*SALT*

*=> Valor diferente para cada usuário (+PEPPER)*

**NUNCA** usar  
HASH simples





# JWT - JSON Web Token

## O QUE É:

Token de **AUTENTICAÇÃO** gerado durante **LOGIN**.

## PROTEÇÃO:

**CSRF - Cross Site Request Forgery**

Substitui os  
**SESSION COOKIE**



## DETALHES:

*API moderna*

*Como usar?*

**=> Autenticado, paradigma REST**

**=> Cabeçalho HTTP (Authentication:)**

# Links Úteis

## MITM

<https://letsencrypt.org/>

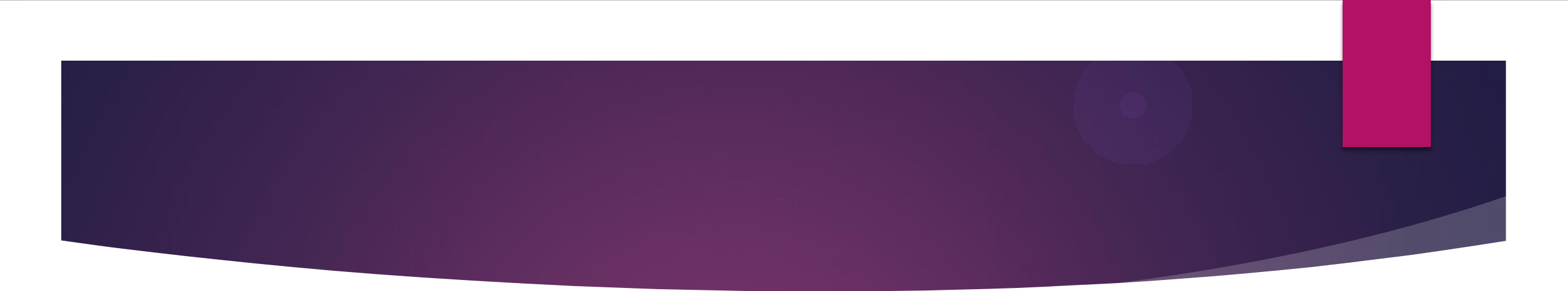
## JWT

[jwt.io](http://jwt.io)

[www.conect2id.com/products/nimbus-jose-jwt](http://www.conect2id.com/products/nimbus-jose-jwt)

<https://securedb.co/community/jwt-vs-jws-vs-jwe>





# ADICIONANDO SEGURANÇA COM WEB-CRYPTO

# O que é WEB-CRYPTO

▶ *API (low-level) disponível nos navegadores modernos*

▶ *Primitivas e serviços criptográficos*

*Encrypt, Decrypt, Sign, Verify*

*Key Generation, Key Derivation*

▶ *Proteção para chaves*

*Import, Export, Wrap, Unwrap, Storage via IndexedDB*

▶ *Fonte de números aleatórios*

`window.crypto.subtle`

Baseado em  
**PROMISES**

Segurança  
**S.O.P.**

# WEB-CRYPTO – Casos de Uso

*Protected Document Exchange*

*Secure Messaging*

*Cloud Storage*

*Multi-factor Authentication*

*Document Signing*

*Data Integrity Protection*



# Links Úteis

## WEB-CRYPTO

<https://www.w3.org/TR/WebCryptoAPI/>

<http://caniuse.com/cryptography>

<https://github.com/diafygi/webcrypto-examples>

<https://developer.mozilla.org/en-US/docs/Web/API/Crypto/subtle>



# Ambiente de Teste



<http://labs.cryptographix.org/workshops/2016-07-13/03-secure-box-plus>



<https://github.com/cryptographix/workshop-designing-a-secure-box-web-application>

pasta: /03-secure-box-plus

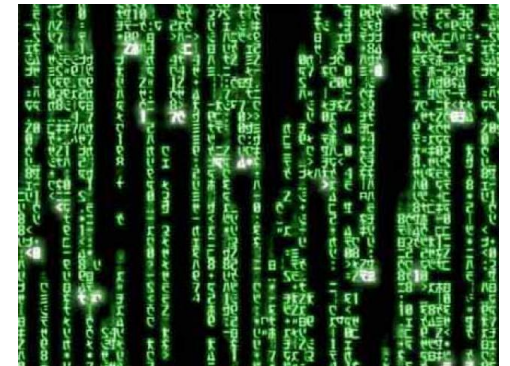
# Encriptação de Dados no Navegador

## O QUE É:

*Encriptação local de dados antes do envio para o servidor*

## PROTEÇÃO:

*Captura da base de dados na nuvem, ou durante a comunicação*



## PORQUE ...

*Não basta FDE no servidor?*

*=> Operadores, admin mal intencionados*

*Não basta HTTPS ?*

*=> MITM tipo web filter*

*Não encriptar na nuvem ?*

*=> Guarda da chave, posse, ...*



# Passos necessários

*Setup (uma vez)*

*Gerar* PAR de chaves, e *guardar* LOCALMENTE

*Salvar Documento*

*Encryptar* documento, antes de enviar para o servidor

*Abrir Documento*

*Decryptar* documento, após recuperação do servidor



# Problemas

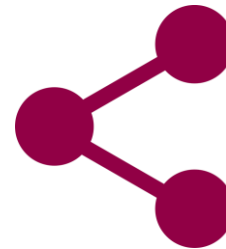
*Perda das chaves*



*Múltiplos usuários*



*Compartilhamento de documentos*



*Compatibilidade com navegadores antigos / diferentes*





# COMPARTILHAMENTO SEGURO

# Compartilhamento Seguro

Como resolver o problema de compartilhar algo que está criptado ?

## CLIENT

Clonar a chave de  
criptação do  
documento

2

Envelopar com a chave  
pública do destinatário

3

## SERVER

1  
Compartilhar a chave  
pública, vinculado ao  
usuário destinatário

1

4  
Compartilhar envelope  
com destinatário pelo  
/shared/

4

# Compartilhamento Seguro

## EXEMPLO DE COMPARTILHAMENTO

<https://webcryptoapiex.github.io/secretnote/>

## IMPLEMENTAÇÃO NO SECURE-BOX



# CONTATOS

<http://cryptographix.org> (beta)

[twitter.com/@Crypto\\_Graphix](https://twitter.com/@Crypto_Graphix)

[facebook.com/cryptographix](https://facebook.com/cryptographix)

**e-mail:**

[info@cryptographix.org](mailto:info@cryptographix.org)

[sean.wykes@nascent.com.br](mailto:sean.wykes@nascent.com.br)

[jairo@tondin.inf.br](mailto:jairo@tondin.inf.br)

Thanks