

Proposta de um Catálogo de Padrões de Segurança da Informação

Luís Otávio Villela Antunes - 131021231

Orientador: Prof. Dr. Kleber Rocha de Oliveira

Problema

- Organizações necessitam **proteger** seus **ativos**.
- O **vazamento**, **destruição** ou **bloqueio** de acesso aos **dados** e/ou **funcionamento** de seus **sistemas informáticos** causam **grandes perdas financeiras e morais** para as organizações.
- Multas, condenações judiciais e desconfiança do mercado são potencialmente **letais**.
- **Yoder & Barcalow (1998)** argumentam que sistemas de informação são desenvolvidos **desconsiderando os critérios de segurança** da informação durante seu ciclo de implementação ou até mesmos na **política de controle dos dados e uso da informação** gerados por estes sistemas.

Segurança da Informação

- ISO 27000:2018 define Segurança da Informação "preservação da **confidencialidade, integridade e disponibilidade** da informação". "Envolvendo a aplicação e gerenciamento de **controles apropriados**, considerando-se uma vasta gama de ameaças, com o objetivo de garantir o sucesso e a continuidade do negócio" e "**minimizar** as consequências de incidentes".

Segurança da Informação

• Confidencialidade

- "Propriedade da informação não ser disponibilizada ou revelada à indivíduos, entidades ou processos **não autorizados**". ISO 27000:2018
- Em adição, Stallings e Brown (2014) abrangem o conceito da privacidade, argumentando que o indivíduo deve estar no controle de quais informações podem ser coletadas e armazenadas e para quem e por quem tais informações podem ser reveladas.

Segurança da Informação

- **Integridade**
- "Propriedade de acurácia e exatidão" ISO 27000:2018
- "Garante que informações e programas sejam alterados somente de maneira especificada e autorizada". Stallings e Brown (2014)

Segurança da Informação

- **Disponibilidade**
- "Propriedade de estar **acessível** e **utilizável**, **sob demanda**, por uma entidade **autorizada**"
ISO 27000:2018
- "Garante que os sistemas funcionem prontamente e que não haja negação de serviço a usuários autorizados" **Stallings e Brown (2014)**

Padrões

- DOUGHERTY, Chad *et al.* (2009) definem **Padrão (*Pattern*)** como uma solução geral reutilizável para um **problema recorrente** em um projeto. É uma **descrição ou modelo de como se resolver um problema**, sendo possível utilizá-lo em diversas situações.
- No contexto da **Segurança da Informação**, os Padrões visam **eliminar** a inserção acidental de **vulnerabilidades** no código e **reduzir** as **consequências** de tais vulnerabilidades. DOUGHERTY, Chad *et al.* (2009).

Metodologia de Pesquisa e Desenvolvimento

- Foram feitas **pesquisas bibliográficas e exploratórias** sobre **segurança da informação**, um estudo consistente sobre o **conceito de padrões** e buscas de artigos acadêmicos e profissionais relacionadas à **padrões de segurança**.
- Desta forma, os padrões de segurança sugeridos pelo autor baseiam-se em **modelos e estrutura de padrões consolidados** que visam garantir a **disponibilidade, a confidencialidade e integridade** da informação e, adicionalmente, **gestão de identidade e irretratabilidade**.

Padrão - "Senha Segura"

4.1 Senha Segura

Baseando-se no pattern Password Design and Use de Roser (2012), criou-se o padrão abaixo:

Nome: Senha Segura.

Intenção: Elucidar uma forma de criar e gerenciar senhas.

Contexto: Este *pattern* pode ser utilizado por quaisquer *softwares* que necessitam de autenticação por senha.

Problema: Senhas fracas possibilitam que pessoas não autorizadas possam se aproveitar da vulnerabilidade dessa condição. Adicionalmente, senhas devem ser fáceis de serem lembradas, porém devem ser difíceis de serem adivinhadas.

Forças: Restrição na composição da senha; repetição de senhas; facilidade de lembrança; validade das senhas;

Solução: Deve-se levar em consideração algumas condições para a criação e o gerenciamento da senha.

- Seu comprimento mínimo
- Uso de números, caracteres especiais (ex: @, #, %), letras maiúsculas e minúsculas
- Tempo de uso da senha
- Adição de salt à senha (conjunto de caracteres aleatórios)
- Método de segurança de armazenamento da senha

Padrão - "Senha Segura"

Exemplo: senha para acessar uma conta bancária na qual não se pode repetir números; senha de acesso a um *e-mail* onde não pode conter o login em seu corpo.

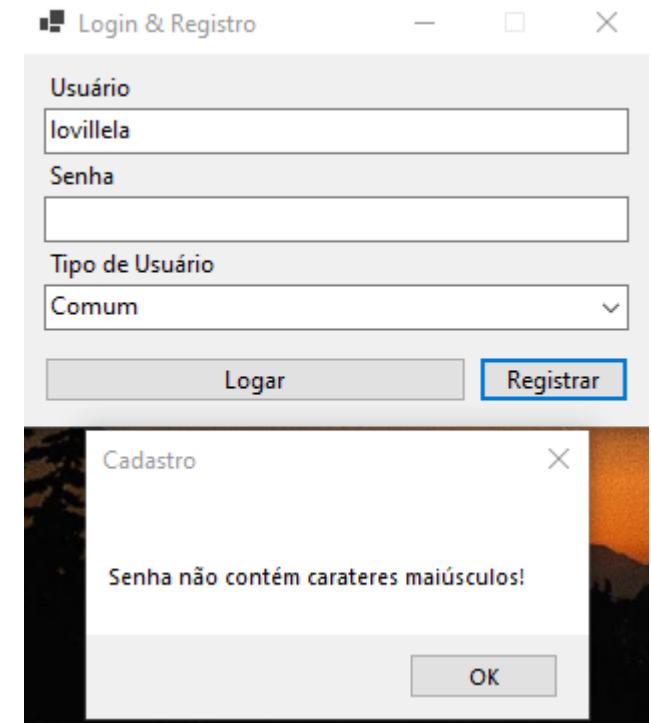
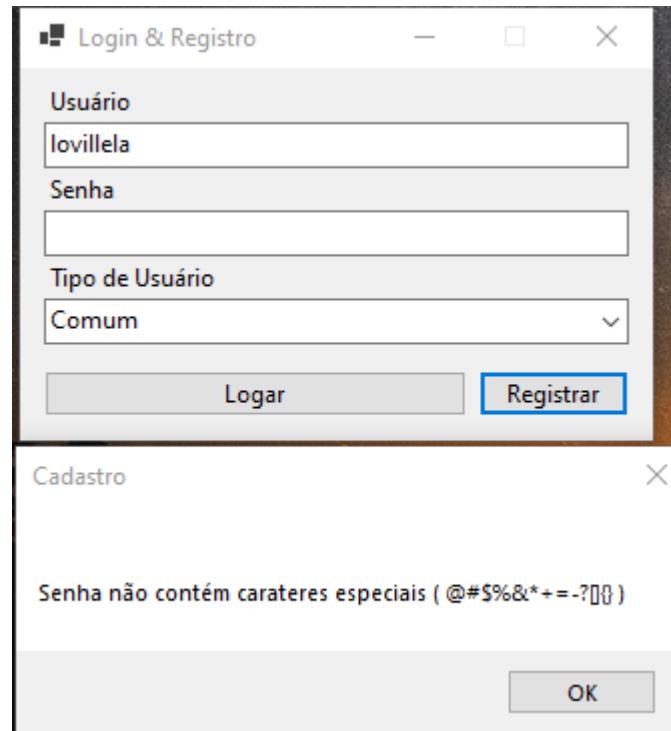
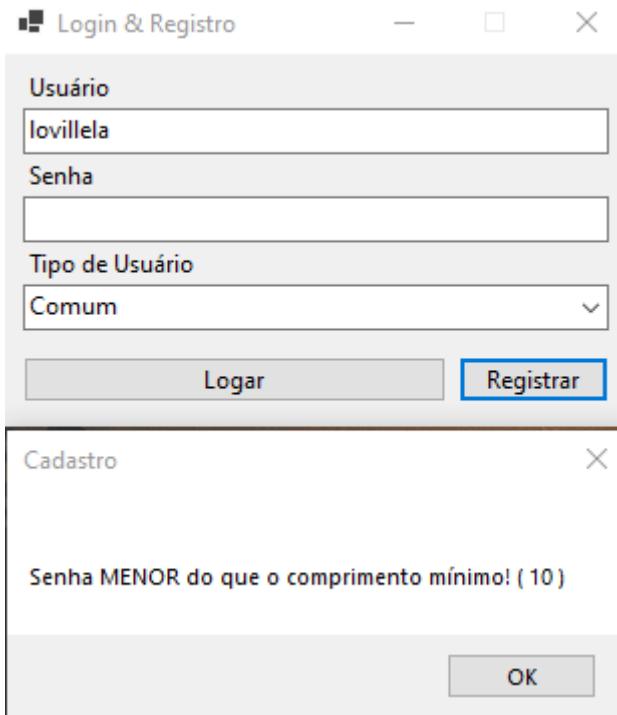
Um exemplo prático para o uso das condições de senha dá-se:

- Comprimento mínimo de 10 caracteres
- Utilização obrigatória de pelo menos 01 caractere especial e 01 caractere maiúsculo e 01 caractere minúsculo
- Senha válida indefinidamente
- Deve-se adicionar o salta antes da senha ser criptografada
- A senha deve ser armazenada depois de criptografada por *Hash SHA512*

Consequências:

- Utilização de um algoritmo de *Hash* para armazenar a senha impede que o agente malicioso tenha acesso a conta comprometida.
- A utilização do salt na senha impede que duas senhas iguais gerem o mesmo hash.

Padrão - "Senha Segura"



Usuário: teste (senha digitada: %Wonh45=ll60cv@#)

Hash:JrrQ7vDMBcRYjbwwPw+Q+V8I+HzA/T0BYJ/RqDbTPuvGNehT9FS3nJzEy2Z4mnW9WVTIA9
ZGbyy2xfkvprONSw==

Salt: sd)HxJ | gKkp%&C7;çwRh | .3doh.3Krq0I3f%CkYKH;G9PgN%Y\$x-*wetTrovu1F??OOWp-P@-
bGI#zV}_K9Epkec?IvF)5rEWv)MhBEue6:B9_0NMGtwu/Sc061W/r#)GXu8WzPç&W6h

Padrão - "Senha Segura"

Senha: 123456

ujJTh2rta8ItSm/1PYQGxq2GQZXtFEq1yHYhtsIztUi66uaVbfNG7IwX9eoQ817jy8UUeX7X3dMUVGTioLq0Ew==

Senha: 123456

ujJTh2rta8ItSm/1PYQGxq2GQZXtFEq1yHYhtsIztUi66uaVbfNG7IwX9eoQ817jy8UUeX7X3dMUVGTioLq0Ew==

Senha: senha123

+jwc3uhm6LV7ZE5VqoWtHwAeoURx2p1BzdMZXlYT9Li2//kF5/Gvs5VKPhguksUkl+Qd7PVxi1Ggm/rfUud/IA==

Senha: senha123

+jwc3uhm6LV7ZE5VqoWtHwAeoURx2p1BzdMZXlYT9Li2//kF5/Gvs5VKPhguksUkl+Qd7PVxi1Ggm/rfUud/IA==

Senha: password

sQnzu7wkTrgkQZF+0G1hi5AI3Qmzvv0bXgc5THBqi7mAsdd4Xll27ASbRt9fEyavWi6m0QP9B8lThf+rDKy8hg==

Senha: password

sQnzu7wkTrgkQZF+0G1hi5AI3Qmzvv0bXgc5THBqi7mAsdd4Xll27ASbRt9fEyavWi6m0QP9B8lThf+rDKy8hg==

Senha: password

sQnzu7wkTrgkQZF+0G1hi5AI3Qmzvv0bXgc5THBqi7mAsdd4Xll27ASbRt9fEyavWi6m0QP9B8lThf+rDKy8hg==

Senha: %Wonh45=1160cv@#

V2a8H7YbbuK9+DDY3fCMA0GiK8WXCtr3qXvXFJSTI9+En28EampLIljgr5rPslniDNohKb5AmrXLkUKC70PNw==

Senha: %Wonh45=1160cv@#

V2a8H7YbbuK9+DDY3fCMA0GiK8WXCtr3qXvXFJSTI9+En28EampLIljgr5rPslniDNohKb5AmrXLkUKC70PNw==

Senha: %Wonh45=1160cv@#

V2a8H7YbbuK9+DDY3fCMA0GiK8WXCtr3qXvXFJSTI9+En28EampLIljgr5rPslniDNohKb5AmrXLkUKC70PNw==

Senha: %Wonh45=1160cv@#

V2a8H7YbbuK9+DDY3fCMA0GiK8WXCtr3qXvXFJSTI9+En28EampLIljgr5rPslniDNohKb5AmrXLkUKC70PNw==

Padrão - "Senha Segura"

Senha: 123456

kIhoxt4ECvK5vta3tiAV0oahQrM+II EaHq4tKrGsZmh12e00c0Dw2xvk0j2Mm53niHLoF9p+D0bXTw9u4aWoRg==

Senha: 123456

6GicJmeRM9bsb0CyWj/WseeQZeBP80aeJkKdH38Hsjt1EGMVXSrNJlgukomokiJA+MIwOrPZoP6tpanFt9pGnQ==

Senha: 123456

UFSOA0Y+sUQ0a4lj6hp4aF2iW1v9P8M4t8EQ/KmMYq/z27Zam6gHi9JULh8qQy8s1AorCZN5Gx5/Ie/SgNB4zA==

Senha: password

L2ODUKh+MQzFM74YUxLvkOiakJL72Bw8enn+DwQX/S0xw5H0BYzmJjsiFEtsvg0LPvUCmptyKwLMs8a9BPBvUQ==

Senha: password

1mZucqrk/S5KFFq/8AlumnJrKHmz20iC+oZ10o+M9QNCnBSi5MET5uKyExcXVzY3Q7FRpnVN7NniLeASpYuKSA==

Senha: password

t5b+QJenq0ggfAAGS6INGDSpbkTrKZRCAYfc8NOJ+gnBShc4Ibn8u4dRYv3LHPgsCH0DAWH9i0Qvhx6ptuL4kQ==

Senha: %Wonh45=1160cv@#

Fqr9o7zAQFF3t/7cfkInFkAqcbg7VItArXWgc2qNUdrK5UT9LZnKU++A65LOP1OnR3RI/cHHMGm67w14X60C3w==

Senha: %Wonh45=1160cv@#

NHQ89aUNC59mObdCf4pH/5N0aTnf2DZYk/VIMq7No1CyDJumjYlGI2iFc8314hIRBpzAb8MePIiHqbz3g1076Q==

Senha: %Wonh45=1160cv@#

YwRhR0u76u40rZGXvS1JnNmB9ALlsp4uaFhCW3ayALdvKLyKqYxgy/vzC9RNz5+7YemRZuA6j9Pd3arrzpRjFg==

Senha: %Wonh45=1160cv@#

OQM7lFFiPvtfsb34g61TqJC/A2eV9Nzlm+T1M6gi90CTbwD49iMj7dpMIrN0wiXgTZwsdsxrmy+HaYf3zsEOXA==

Senha: %Wonh45=1160cv@#

cGJk+d0dQ+QH1SAhWNbSMm7zrY0bcuchs1PJbLYUK0wmqQcp8/afnY070aHMCKBgF21ZbTR0kGyX1KC5IhSrtg==

Senha: %Wonh45=1160cv@#

ZxR3Nzz6d7cTP1kFzmocXDxR5d+1FKg2LUORbmhqG72lo0ltium2v1xjI1yhITWY0vC/3cpoBK+Jrrsd03NefQ==

Padrão - "Senha Segura"

```
internal static (int, string) CriarSenha(ref string senhaInformada)
{
    bool possuiMaiusculo = false;
    bool possuiEspecial = false;

    //-
    if (senhaInformada.Length < ComprimentoMinimo)
    {
        senhaInformada = string.Empty;
        return (1, senhaPequena); //vê o tamanho da senha
    }
    //-
    foreach (var caractere in senhaInformada)
    {
        foreach (var caractereEspecial in CaracteresEspeciais)
        {
            if (caractere == caractereEspecial) //conta os caracteres especiais
            {
                possuiEspecial = true; //se encontrar 1, dá break!
                break;
            }
        }

        if (possuiEspecial)
        {
            break;
        }
    }

    if (!possuiEspecial)
    {
        senhaInformada = string.Empty;
        return (2, senhaSemEspeciais);
    }

    //-
    foreach (var caractere in senhaInformada)
    {
        if (char.IsLetter(caractere) && char.IsUpper(caractere)) //verifica se é letra e se é maiúscula
        {
            possuiMaiusculo = true; //se for, encerra o laço
            break;
        }
    }

    if (!possuiMaiusculo)
    {
        senhaInformada = string.Empty;
        return (3, senhaSemMaiusculo);
    }

    //-
    return (0, sucesso);
}
```

Padrão - "Senha Segura"

```
public static class GeradorSalt
{
    private static readonly char[] listaChars = Properties.Resources.listaDeCaracteres.ToCharArray();
    //ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789ç!@#$%&*()-_=+{}[]?/:.|  
3 references
    public static string GeraSalt()
    {
        int tamanho = Convert.ToInt32(Properties.Resources.tamanhoSalt); //140
        var saltLocal = new StringBuilder(tamanho); //string de caracteres
        byte[] dados = new byte[4 * tamanho];

        using (var cripto = new RNGCryptoServiceProvider()) //para garantir randomização
        {
            cripto.GetBytes(dados); //enche o vetor com números aleatórios
        }

        for (int i = 0; i < tamanho; i++)
        {
            var intAleatorio = BitConverter.ToInt32(dados, i*4); //pega um valor(gerado acima) da posicao do vetor
            var indice = intAleatorio % listaChars.Length;

            saltLocal.Append(listaChars[indice]);
        }

        return saltLocal.ToString(); //converte para string
    }
}
```

Padrão - "Senha Segura"

```
public static string CifrarSenha512(string senhaInformada, string salt)
{
    //Cifra a senha com SHA256 (implementação padrão do framework)
    //"salt" -> termo em inglês para um texto que é adicionado à senha
    string senhaCifradaLocal, //armazena o hash da senha, convertida para string
          senhaComSalt; //senhaInformada + salt
    byte[] dados, hash;

    senhaComSalt = senhaInformada + salt; //adiciona o salt à senha
    senhaInformada = string.Empty;
    ...

    dados = Encoding.UTF8.GetBytes(senhaComSalt); //converte os caracteres para bytes

    hash = SHA512.Create().ComputeHash(dados);

    senhaCifradaLocal = Convert.ToString(hash);

    return (senhaCifradaLocal);
}
```

Padrão - "Proteção de Dados"

- **4.6 PROTEÇÃO DE DADOS**
- Baseando-se no pattern *Encrypted Storage* de Kienzle et al. (2002) e o pattern *Encryption with user-managed keys* do catálogo *Privacy Patterns* do campus de Berkley da Universidade da Califórnia criou-se o padrão abaixo:
- **Nome:** Proteção de Dados
- **Intenção:** Este padrão propõe um método de proteção de dados sensíveis armazenados no sistema informático no evento do roubo dos mesmos.
- **Contexto:** Este padrão pode ser utilizado por qualquer sistema informático que necessite do armazenamento seguro de dados sigilosos.
- **Problema:** Apesar de todos os métodos utilizados para a proteção de um sistema informático (ex: senhas, firewalls) serem eficazes em suas funções, estes não são à prova de falhas (podem existir vulnerabilidades desconhecidas).
- Desta forma, o roubo de dados sigilosos como, por exemplo, laudos médicos e números de cartões de crédito podem causar transtornos a seus proprietários e danos extensivos às organizações que os armazenam.
- **Forças:** acesso não autorizado; armazenamento de dados sigilosos; gestão de chaves criptográficas.
- **Solução:** Para este pattern, há duas soluções possíveis:
 - 1) Encriptação efetuada pelo servidor

Padrão - "Proteção de Dados"

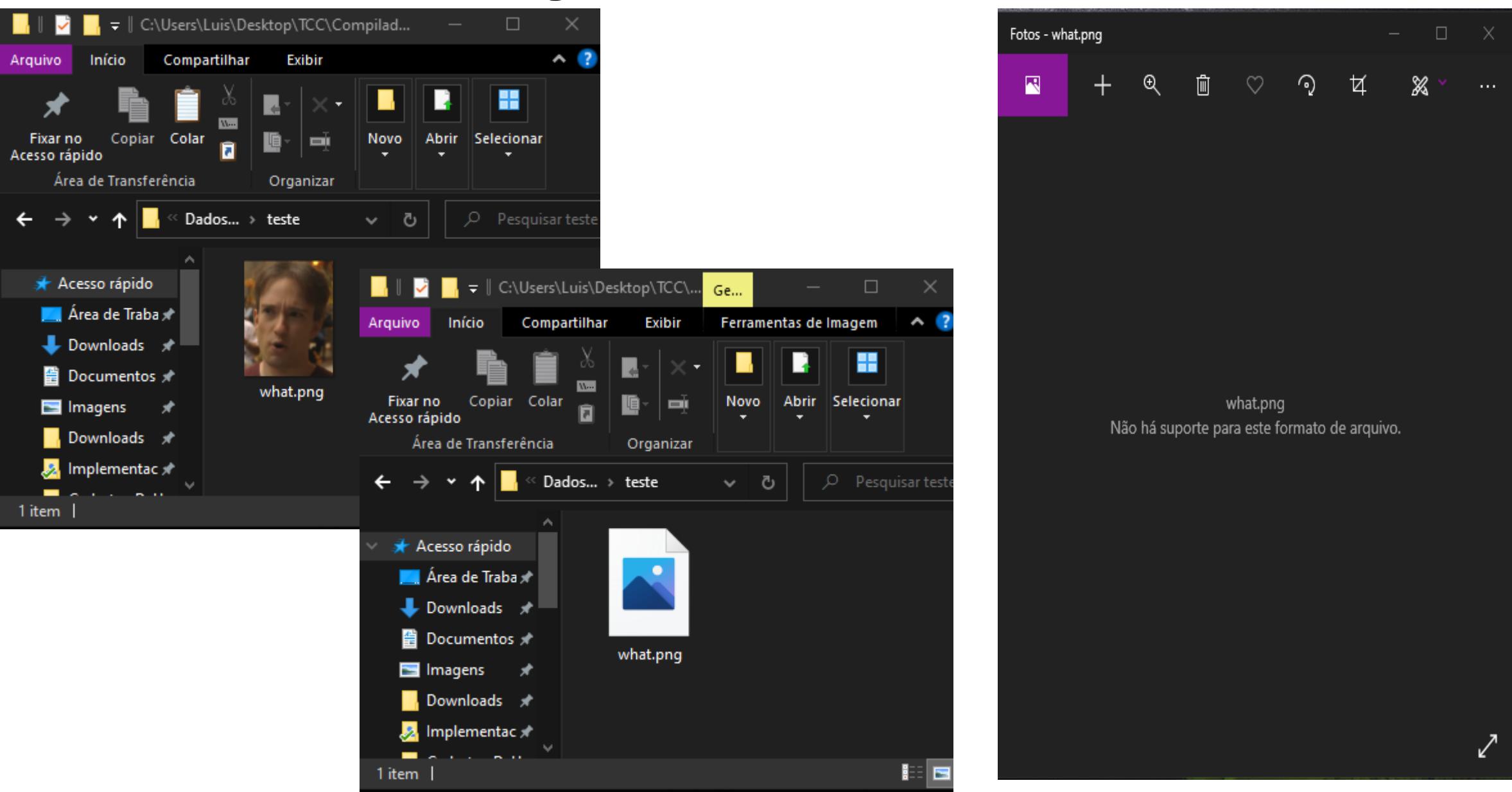
- Nesta solução o processo de decriptação e encriptação são efetuados pelo servidor, e naturalmente as chaves de criptografia são armazenadas no servidor. Desta forma, esta solução deve ser utilizada quando há a necessidade de terceiros acessarem os dados.
- Utilizando seu login e senha próprios, através de uma conexão segura (ex: HTTPS) o usuário acessa o sistema.
- O servidor carrega a chave criptográfica no módulo de encriptação/decriptação
- No caso de envio de dados ao sistema
- Os dados são enviados ao módulo de encriptação/decriptação
 - Os dados são encriptados
 - Os dados são armazenados
- No caso de acesso de dados do sistema
 - O servidor busca os dados
 - Os dados são decriptados
 - Os dados são enviados ao cliente

Padrão - "Proteção de Dados"

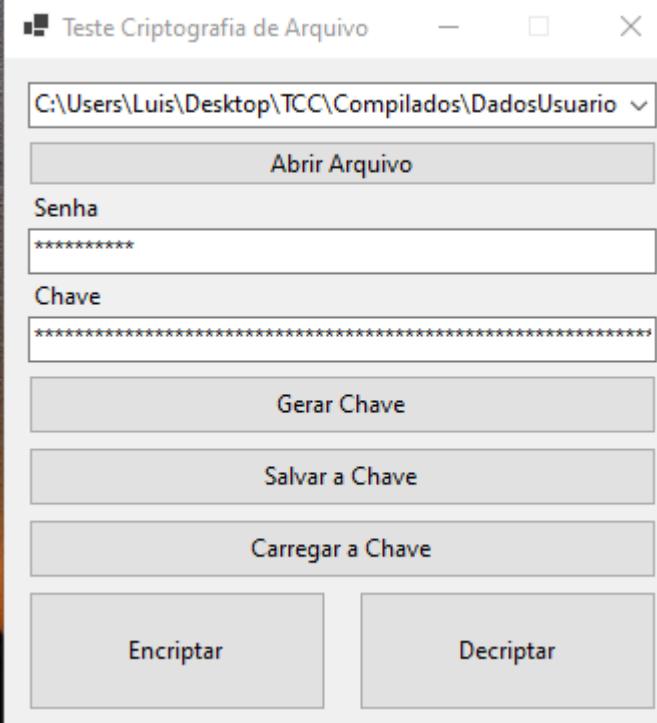
2) Encriptação efetuada pelo cliente

- Nesta solução o processo de decriptação e encriptação são efetuados pelo cliente, e naturalmente as chaves de criptografia estão em posse do mesmo. Desta forma, esta solução deve ser utilizada quando há a necessidade restringir terceiros de acessarem os dados.
- **Exemplo:** Dados armazenados pelos os usuários devem ser acessados apenas pelos mesmos, no caso dos dados já serem armazenados encriptados.
- **Consequência:**
 - Apenas usuários ou servidores em posse das chaves criptográficas podem acessar os dados
 - Impacto no desempenho do sistema, devido ao custo computacional elevado para encriptar e decriptografar os dados.
 - Garantia da confidencialidade da informação
 - Impossibilidade de acesso aos dados na eventualidade da perda das chaves criptográficas
- **Implementação:**

Padrão - "Proteção de Dados"



Padrão - "Proteção de Dados"



- Seleciona-se um arquivo
- Digita-se uma senha, esta não é armazenada
- Gera-se uma chave, esta deve ser salva

Senha + Chave -> Hash SHA256 -> Chave AES

Chave -> Hash SHA256 -> IV AES

Padrão - "Proteção de Dados"

```
internal static void EncriptaArquivo(string pathArquivo, ref byte[] chaveSegura, ref byte[] IV)
{
    using (var aes = Aes.Create())
    {
        aes.Key = chaveSegura;
        aes.IV = IV;
        aes.Padding = PaddingMode.PKCS7;
        aes.Mode = CipherMode.CBC;

        var encriptador = aes.CreateEncryptor(aes.Key, aes.IV);

        using(var streamMemoria = new MemoryStream(File.ReadAllBytes(pathArquivo)))
        {
            using (var streamArquivo = File.Create(pathArquivo))
            {
                using (var streamCripto = new CryptoStream(streamArquivo,
                                                encriptador, CryptoStreamMode.Write))
                {
                    streamCripto.Write(streamMemoria.ToArray(), 0, streamMemoria.ToArray().Length);
                    streamCripto.Dispose();
                }
            }
        }

        encriptador.Dispose();
        aes.Clear();
        aes.Dispose();
    }

    Array.Clear(chaveSegura, 0, chaveSegura.Length);
    Array.Clear(IV, 0, IV.Length);
}
```

Padrão - "Proteção de Dados"

```
internal static void DecriptarArquivo(ref string pathArquivo, ref byte[] chaveSegura, ref byte[] IV)
{
    using (var aes = Aes.Create())
    {
        aes.Key = chaveSegura;
        aes.IV = IV;
        aes.Padding = PaddingMode.PKCS7;
        aes.Mode = CipherMode.CBC;

        var decriptador = aes.CreateDecryptor(aes.Key, aes.IV);

        using (var streamMemoria = new MemoryStream(File.ReadAllBytes(pathArquivo)))
        {
            using (var streamArquivo = File.Create(pathArquivo))
            {
                using (var streamCripto = new CryptoStream(streamArquivo,
                    decriptador, CryptoStreamMode.Write))
                {
                    streamCripto.Write(streamMemoria.ToArray(), 0, streamMemoria.ToArray().Length);
                    streamCripto.Dispose();
                }
            }
        }

        decriptador.Dispose();
        aes.Clear();
        aes.Dispose();
    }

    Array.Clear(chaveSegura, 0, chaveSegura.Length);
    Array.Clear(IV, 0, IV.Length);
}
```

Conclusão

- A criação e o uso de Padrões para segurança, auxilia desenvolvedores em suas novas soluções.
- Reduz-se a quantidade de erros.
- Reduz-se o tempo de planejamento e desenvolvimento
- Reduz-se os custos com desenvolvimento e correções.

Referências

- ALEXANDER, Christopher et al. **A Pattern Language**. Oxford University Press, 1979.
- BARCALOW, Jeffrey; YODER, Joseph. **Architectural Patterns for Enabling Application Security**. 1998
- DOUGHERTY, Chad *et al.* **Secure Design Patterns**. 2009.
- ISO 27000:2018
- LAWRIE, Brown. STALLINGS, William. **Segurança de Computadores - Princípios e Práticas** - 2^a Ed. GEN LTC, 2014
- KIENZLE, Darrell M. *et al.* **Security Patterns Repository**. 2002.
- ROSER, Florian. **Security Design Patterns in Software Engineering**. 2012.