Project 2
Application Security Verification Standard (ASVS)

Universidade de Aveiro

Segurança Informática e nas Organizações

Francisco Gonçalves nMec: 108538

Índice

Resumo

O projeto centra-se no padrão de Verificação de Segurança de Aplicações (ASVS) da OWASP, versão 1.0. O nosso objetivo principal foi aprimorar uma loja online de merchandise do DETI (Departamento de Eletrónica, Telecomunicações e Informática) da Universidade de Aveiro para que cumpra os requisitos de segurança do nível 1 do ASVS.

Inicialmente, realizámos uma auditoria completa de conformidade da aplicação web existente, identificando as áreas que não estão em conformidade com o ASVS nível 1.

Após a auditoria, implementámos as melhorias de segurança necessárias: a correção dos problemas identificados durante a auditoria, seguindo as diretrizes do ASVS sem remover funcionalidades da loja.

Adicionalmente, escolhemos e implementámos duas funcionalidades de segurança da lista fornecida: a avaliação da força das senhas e o armazenamento criptografado de dados críticos.

No ficheiro anexado, apresentamos duas versões da loja online: a versão original (app_org) e a versão melhorada (app_sec). Neste relatório explicamos as melhorias de segurança implementadas e o impacto das mesmas, obtidas através do concerto de 10 “key issues”.

LEVEL 1 ASVS

Password strength evaluation: requiring a minimum of strength for passwords according to V2.1, with breach verification using an external service;

Versão original

Na versão original, estava apenas implementado um limite mínimo de 12 caracteres para a senha. Adicionei requisitos para caracteres, o que não era necessário.

```python
def sign_up():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password1']
        password_confirm = request.form['password2']
        username = request.form['username']
        errors = {}

        # Validate email
        try:
            valid = validate_email(email)
            email = valid.email  # Update with normalized email
        except EmailNotValidError as e:
            errors["email"] = str(e)

        # Stronger password requirements
        if len(password) < 12:
            errors["length"] = "Password must be at least 12 characters long."
        if not re.search(r"[a-z]", password):
            errors["lowercase"] = "Password must contain at least one lowercase letter."
        if not re.search(r"[A-Z]", password):
            errors["uppercase"] = "Password must contain at least one uppercase letter."
        if not re.search(r"[0-9]", password):
            errors["number"] = "Password must contain at least one number."
        if not re.search(r"[@#$%^&+=_!]", password):
            errors["special"] = "Password must contain at least one special character (e.g., @, #, $, etc.)."
        if password != password_confirm:
            errors["confirm"] = "Passwords do not match."

        if not errors:
            if not verify_user(email):
                hashed_password = generate_password_hash(password)
                add_user(email, hashed_password, username)  # Store hashed password
                return redirect(url_for('auth.login'))
            else:
                flash('Email is already in use.', 'error')
                return render_template('sign_up.html', email_in_use=True, email=email, username=username)

        return render_template('sign_up.html', errors=errors, email=email, username=username)
    else:
        return render_template('sign_up.html')
```

Versão segura

Na versão aprimorada, foi implementada os ASVS:

- 2.1.2,
- 2.1.8,
- 2.1.9
- 2.1.12

Como se pode ver pelo código eu verifico o tamanho da password é menos que 128 (ASVS 2.1.2). Removi também todos os requerimentos anteriores (ASVS 2.1.9).

Adicionei também em javascript forma de ver a força da password de acordo com as seguintes condições presentes no código. Quantas mais condições forem preenchidas mais preenchida ficará a barra (ASVS 2.1.8)

```
let strength = 0;
if(password.value.length >= 12) strength++;
if(/[a-z]/.test(password.value)) strength++;
if(/[A-Z]/.test(password.value)) strength++;
if(/[0-9]/.test(password.value)) strength++;
if(/[@#$%^&+=_!]/.test(password.value)) strength++;

if(strength === 1 || strength === 2) {
    strengthBar.className = "strength-bar weak";
} else if (strength === 3) {
    strengthBar.className = "strength-bar fair";
} else if (strength === 4) {
    strengthBar.className = "strength-bar good";
} else if (strength === 5) {
    strengthBar.className = "strength-bar strong";
} else {
    strengthBar.className = "strength-bar";
}
```

V2.8.1 - Multi-factor Authentication (MFA)

A Autenticação Multi-Fator (MFA) é uma medida de segurança essencial, projetada para reforçar o processo de login tradicional, exigindo que os utilizadores forneçam mais do que uma forma de identificação. A MFA adiciona uma camada extra de defesa contra ameaças cibernéticas, geralmente envolvendo algo que o utilizador sabe (palavra-passe) e algo que possui (por exemplo, uma palavra-passe temporária baseada em tempo ou um fator biométrico).

Versão original

Na versão anterior, não havia a implementação de autenticação multifatorial (MFA). A ausência dessa camada de segurança adicional deixava a aplicação vulnerável a acessos não autorizados, mesmo que as credenciais de login fossem comprometidas.

Versão segura

Introduzi o processo de configuração de Autenticação Multi-Fator (MFA) usando TOTP (Time-based One-Time Password). O código gera uma chave secreta (totp_secret) para o usuário, a armazena junto com os dados do usuário, e cria um URI de provisionamento TOTP. Esse URI é convertido em um código QR, que é então gerado como uma imagem base64 para ser exibido na página de registro (sign_up.html). O utilizador pode digitalizar este código QR com uma aplicação como o Google Authenticator para configurar a MFA.

```
totp_secret = pyotp.random_base32()

add_user(email, generate_password_hash(password), username, totp_secret)

totp_uri = pyotp.totp.TOTP(totp_secret).provisioning_uri(name=email, issuer_name="eDeti shop")
qr = qrcode.make(totp_uri)
img_io = BytesIO()
qr.save(img_io, 'PNG')
img_io.seek(0)
qr_data = img_io.getvalue()
qr_b64 = base64.b64encode(qr_data).decode('ascii')
qr_image = f"{qr_b64}"

return render_template('sign_up.html', totp_uri=totp_uri, totp_qr_image=qr_image)
```

Este código verifica a autenticidade de um código TOTP (Time-based One-Time Password) durante o processo de login. Primeiro, ele recupera a chave secreta TOTP do usuário (user_totp_secret) com base no email. Em seguida, cria uma instância de TOTP e verifica se o código TOTP inserido pelo usuário é válido, permitindo uma margem de erro de três períodos (valid_window=3). Se o código for válido, a sessão do usuário é configurada como permanente, os dados da sessão são limpos, e o usuário é autenticado, armazenando seu email e nome de utilizador na sessão. Por fim, o contador de tentativas de login é reiniciado, e o usuário é redirecionado para a página inicial (views.home).

```
totp = pyotp.TOTP(user_totp_secret)

is_totp_valid = totp.verify(totp_code, valid_window=3)

if is_totp_valid:
    session.permanent = True
    session.clear()
    session['user'] = {'email': email, 'username': get_username(email)}
    clear_login_attempts(email)
```

V2.1.7 - Password Security

Esta issue do ASVS destaca a importância de verificar se as senhas usadas em registo, login ou mudança de senha já foram comprometidas. Isso pode ser feito localmente ou via API, sem expor a senha. Se não resolvido, utilizadores podem usar senhas já violadas, aumentando o risco de ataques e compromissos de segurança, levando a possíveis violações de dados e danos.

Versão original

Na versão original da aplicação, não havia nenhum mecanismo que verificasse se as passwords tinham sido violadas, o que deixava a segurança das contas dos utilizadores vulnerável. Embora a aplicação pudesse garantir que as senhas seguiam um certo padrão de complexidade, isso não impedia que fossem senhas já comprometidas em violações anteriores. Sem essa verificação, utilizadores poderiam utilizar senhas fracas ou já comprometidas, aumentando significativamente o risco de ataques como “credential stuffing”. Isso representava uma falha de segurança crítica, uma vez que senhas comprometidas são frequentemente exploradas por atacantes.

Versão segura

Na versão melhorada da aplicação, foi implementada uma verificação obrigatória das passwords contra uma base de dados de senhas comprometidas. A função check_breached_password utiliza a API do serviço "Have I Been Pwned?" para comparar a senha fornecida com uma lista de passwords conhecidas por terem sido comprometidas. Se a senha estiver comprometida, o utilizador é informado e solicitado a escolher uma senha diferente. Esta medida aumenta significativamente a segurança, prevenindo o uso de senhas vulneráveis e protegendo os utilizadores contra ataques de força bruta ou "credential stuffing".

```python
def check_breached_password(password):
    sha1_password = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    prefix, suffix = sha1_password[:5], sha1_password[5:]
    url = f"https://api.pwnedpasswords.com/range/{prefix}"

    response = requests.get(url)
    if response.status_code != 200:
        raise RuntimeError("Error fetching breached passwords")

    hash_suffixes = (line.split(':') for line in response.text.splitlines())
    for hsuffix, count in hash_suffixes:
        if suffix == hsuffix:
            return True
    return False
```

```python
    password = re.sub(r'\s+', ' ', password).strip()
    password_confirm = re.sub(r'\s+', ' ', password_confirm).strip()


    if check_breached_password(password):
        errors["breached"] = "This password has been compromised in a data breach. Please choose a different password."

    try:
        valid = validate_email(email)
        email = valid.email
    except EmailNotValidError as e:
        errors["email"] = str(e)

    if len(password) < 12 or len(password) > 128:
        errors["length"] = "Password must be at least 12 characters long and less than 128 characters."
    if password != password_confirm:
        errors["confirm"] = "Passwords do not match."
```

V5.1.4 - Input Validation Requirements

A validação e verificação dos dados enviados na nossa aplicação é crucial para garantir que correspondem ao esperado para cada campo. Este processo é essencial para mitigar vulnerabilidades associadas à "CWE-20: Improper Input Validation". Na nossa aplicação, onde os utilizadores podem deixar avaliações nos produtos, a validação adequada dos dados assegura que apenas entradas válidas são aceites, prevenindo possíveis ataques ou comportamentos indesejados.

Versão original

Na versão original, este espaço para reviews tinha apenas o escape de caracteres perigosos como medida preventiva, protegendo contra vários tipos de ataques, como o “Cross Site Scripting”. No entanto, a validação implementada não é suficientemente segura. Um utilizador, por exemplo, pode enviar uma review com um tamanho excessivo, sobrecarregando o servidor, já que não existe um limite de caracteres. Além disso, a aplicação permite o envio de qualquer caractere, incluindo muitos que não são relevantes numa review, como os caracteres unicode “┼“, “◀ ”, “▒”, etc. Permitir o uso desses caracteres pode levar a abusos desnecessários e não é, de todo, uma boa prática.

Versão segura

Para resolver isto foi implementada medidas preventivas dos problemas falados anteriormente como por exemplo foi posto um limite de 300 caracteres nas reviews e foi limitado o uso do tipo de caracteres que não sejam seja uma letra do alfabeto ou um número, o utilizador é alertado e a mensagem não é enviada.

```
    if not ratings or not ratings.isdigit():
        errors['ratings'] = 'Ratings is required.'
    elif not p_review or not p_review.strip():
        errors['p-review'] = 'Review is required and cannot be empty.'
    elif len(p_review) > 300:
        errors['p-reviewsize'] = 'Review must be at most 300 characters.'
    elif not ALLOWED_CHARACTERS.match(p_review):
        errors['p-reviewchars'] = 'Review contains invalid characters.'
    else:
        try:
            ratings = int(ratings)
            if ratings < 1 or ratings > 5:
                errors['ratings'] = 'Ratings must be between 1 and 5.'
            else:
                add_review(email, ratings, p_review.strip(), product_id)
        except ValueError:
            errors['ratings'] = 'Ratings must be a number.'
    reviews = get_reviews(product_id)
```

```
ALLOWED_CHARACTERS = re.compile(r'^[\w\s.,!?@#&*()-_=+;:\'"\u00A0-\uFFFF]+$')
```

V3.7.1 - Defenses Against Session Management Exploits

A CWE-778, conhecida como "Insufficient Session Expiration," ocorre quando uma aplicação web não valida adequadamente a sessão do utilizador antes de permitir transações sensíveis ou modificações de conta. Isso pode levar a ataques como o uso de sessões comprometidas, onde um atacante pode executar ações críticas sem que o utilizador precise reautenticar-se ou passar por uma verificação secundária.

Versão original

Na versão original, a aplicação permitia que um utilizador autenticado realizasse transações sensíveis e modificações de conta sem a necessidade de verificar novamente a validade da sessão ou solicitar reautenticação. Isso abria portas para ataques em que um atacante, com acesso a uma sessão válida, poderia realizar ações críticas indevidamente.

Versão segura

Na versão melhorada, foi implementada uma verificação obrigatória da validade da sessão para todas as transações sensíveis e modificações de conta. Além disso, foi adicionada uma reautenticação ou verificação secundária para garantir que apenas o utilizador legítimo possa executar ações críticas, aumentando assim a segurança da aplicação contra o uso indevido de sessões comprometidas.

```python
@views.route("/pay", methods=['POST'])
def pay():
    if 'user' not in session:
        return redirect(url_for('auth.login'))

    user = session['user']['username']
    email = session['user']['email']
    password = request.form.get('password')
    errors = {}


    if not password:
        errors['password'] = 'Password is required.'
        return render_template('checkout.html', user=user, errors=errors)


    stored_password = get_user_password(email)
    if not (stored_password and check_password_hash(stored_password, password)):
        errors['password'] = 'Invalid password. Please re-enter your password.'
        return render_template('checkout.html', user=user, errors=errors)


    try:
        pay_cart(user)
    except Exception as e:
        errors['payment'] = f"Payment failed: {str(e)}"
        return render_template('checkout.html', user=user, errors=errors)


    return redirect(url_for('views.home'))
```

V12.1.1 - File Upload

A CWE-400 é uma vulnerabilidade de segurança na web que ocorre quando o servidor não manipula adequadamente as entradas do utilizador. Esta falha pode resultar em comportamentos inesperados, injeção de código malicioso e exposição de dados sensíveis, devido à ausência de validação e tratamento apropriado das entradas.

Versão original

Na versão original, não havia uma verificação do tamanho dos ficheiros ao adicionar uma imagem para um produto à venda. Esta falta de validação permitia a injeção de malware e a potencial exposição de dados sensíveis.

Versão segura

Na versão atualizada, corrigiram-se as vulnerabilidades mencionadas, implementando um limite máximo de tamanho para os ficheiros. Criou-se uma variável global, MAX_FILE_SIZE, com o valor definido de 5 MB, para limitar o tamanho dos ficheiros carregados. Esta medida reduz significativamente o risco de injeção de malware e protege a integridade dos dados armazenados.

```python
    errors = {}
    MAX_FILE_SIZE = 5*1024*1024

    if 'avatar' in request.files:
        file = request.files['avatar']
        if file and allowed_file(file.filename):
            file.seek(0, os.SEEK_END)
            if file.tell() > MAX_FILE_SIZE:
                errors["avatar"] = "File size exceeds 5MB."
                return render_template('profile.html', email=session['use
```

V3.4.1 - Cookie-based Session Management

CWE-614 é uma vulnerabilidade de segurança na web que ocorre quando cookies de sessão são transmitidos de forma insegura. Se o atributo Secure não estiver configurado corretamente, os cookies de sessão podem ser transmitidos por conexões HTTP não seguras, expondo os usuários a ataques de intercetação de dados, como sequestro de sessão.

Versão original

Na versão original, o atributo SESSION_COOKIE_SECURE não estava configurado, permitindo que os cookies de sessão fossem transmitidos por meio de conexões HTTP não seguras. Isso expunha os usuários a riscos de ataques de intercetação de dados (Man-in-the-Middle), onde os tokens de sessão poderiam ser capturados e utilizados por atacantes mal-intencionados.

Versão segura

Na versão melhorada, os erros mencionados foram corrigidos configurando o atributo SESSION_COOKIE_SECURE como True. Esta modificação garante que os cookies de sessão só sejam transmitidos através de conexões HTTPS seguras, diminuindo significativamente o risco de intercetação de dados sensíveis e sequestro de sessão. Essa melhoria foi implementada ao adicionar a linha de configuração app.config['SESSION_COOKIE_SECURE'] = True no arquivo de configuração global do aplicativo.

V12.5.1 - File Download Requirements

Este tópico aborda uma prática de segurança que garante que o servidor web esteja configurado para servir apenas ficheiros com extensões específicas. Isto evita a divulgação acidental de informações ou códigos-fonte. Por exemplo, ficheiros de backup (.bak), temporários (.swp), comprimidos (.zip, .tar.gz, etc.), entre outras extensões comuns, devem ser bloqueados. Essa configuração minimiza a exposição de ficheiros sensíveis, que podem conter informações confidenciais ou detalhes de implementação que não devem estar acessíveis publicamente.

Versão original

Na versão original, não havia uma verificação adequada dos tipos de ficheiros servidos. Isso significava que, se um ficheiro tivesse a extensão .zip ou outras mencionadas, o servidor web poderia disponibilizá-lo diretamente aos utilizadores sem restrições. Essa falta de verificação comprometia a segurança, permitindo o acesso a ficheiros sensíveis ou potencialmente perigosos, como ficheiros de backup, temporários ou comprimidos, o que poderia resultar em fuga de informações ou exposição de códigos-fonte.

Versão segura

Na versão melhorada deste código, foi implementada uma verificação obrigatória do tipo de ficheiro antes de permitir o upload de avatares, assegurando que apenas imagens válidas são aceites. Além disso, salva os ficheiros apenas em diretórios designados, prevenindo o acesso ou a sobreposição de ficheiros sensíveis.

```python
@auth.route('/change_avatar', methods=['POST'])
def change_avatar():
    email = session['user']['email']
    if 'user' not in session:
        return redirect(url_for('auth.login'))
    errors = {}

    # Handle avatar upload
    if 'avatar' in request.files:
        file = request.files['avatar']
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(current_app.root_path, 'static/uploads', filename))
            print(current_app.root_path)

            # Update the avatar path with the relative path
            update_avatar(email, filename)

            # Update session data
            session['user']['avatar'] = filename
            errors["avatar"] = "Avatar updated successfully."
        else:
            errors["avatar"] = "Invalid file type. Please upload a valid image file."

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

V8.3.2 - Sensitive Private Data

Garantir que os utilizadores possam remover ou exportar os seus dados sob pedido é uma prática essencial de segurança e privacidade. Isto assegura que, ao solicitar a remoção, todos os dados pessoais sejam eliminados dos sistemas, reduzindo o risco de exposição de informações sensíveis. Além disso, a exportação permite que os utilizadores obtenham uma cópia dos seus dados num formato acessível, reforçando o seu controlo sobre as informações partilhadas. Estas práticas não só promovem a confiança dos utilizadores, mas também cumprem as exigências legais em vigor.

Versão original

Na versão original da aplicação, não havia qualquer funcionalidade que permitisse aos utilizadores remover ou exportar os seus dados sob pedido. Os dados dos utilizadores eram armazenados indefinidamente sem uma opção clara de exclusão, o que deixava os utilizadores sem controlo sobre as suas informações pessoais.

Versão segura

Na versão segura da aplicação, foi implementado um método para que os utilizadores possam remover os seus dados sob pedido. A rota /delete_account permite que os utilizadores autenticados solicitem a exclusão completa da sua conta. Quando o pedido é efetuado, o servidor verifica se o utilizador está autenticado e, em seguida, procede à eliminação de todos os dados associados ao endereço de e-mail do utilizador na base de dados.
A função delete_user_account executa esta ação de forma segura, removendo a entrada correspondente na tabela de utilizadores. Após a eliminação dos dados, a sessão do utilizador é limpa, assegurando que ele é desconectado, e é redirecionado para a página de login. Esta implementação garante que os utilizadores têm controlo sobre as suas informações, podendo removê-las completamente da aplicação sempre que desejarem.

```python
def delete_user_account(email):
    conn = sqlite3.connect(db_path)
    c = conn.cursor()
    c.execute("DELETE FROM Users WHERE email = ?", (email,))
    conn.commit()
    conn.close()

@auth.route('/delete_account', methods=['POST'])
def delete_account():
    if 'user' not in session:
        return redirect(url_for('auth.login'))

    email = session['user']['email']

    delete_user_account(email)

    session.clear()
    return redirect(url_for('auth.login'))
```

V3.2.2 - Session Binding

Os tokens de sessão não tiverem pelo menos 64 bits de entropia, eles se tornam vulneráveis a ataques como força bruta ou adivinhação, onde um atacante pode conseguir prever ou gerar um token válido. Isso pode resultar em sequestro de sessão, permitindo que o atacante assuma a identidade do usuário, acesse dados confidenciais, execute ações maliciosas em seu nome e comprometa toda a segurança da aplicação.

Versão original

Na versão original da aplicação, os tokens de sessão não possuíam o nível de entropia desejado, deixando-os vulneráveis a ataques. Esses tokens, que deveriam garantir a segurança das sessões dos utilizadores, eram gerados com baixa aleatoriedade, o que aumentava o risco de serem adivinhados por atacantes. Sem a entropia mínima de 64 bits recomendada, a segurança das sessões dos utilizadores estava comprometida, tornando possível que um atacante pudesse sequestrar sessões e obter acesso não autorizado a informações pessoais e ações do utilizador na aplicação.

```
SECRET_KEY = b'KXEHT'
```

Versão segura

Na versão melhorada deste código, foi implementado um mecanismo para garantir que os tokens de sessão possuam a entropia necessária para assegurar a segurança dos utilizadores. Agora, os tokens são gerados com pelo menos 64 bits de entropia, o que torna os tokens altamente imprevisíveis e resistentes a ataques de força bruta ou tentativa de adivinhação. Esta melhoria cumpre a recomendação da OWASP, assegurando que os tokens de sessão protejam efetivamente a identidade e os dados pessoais dos utilizadores, mitigando os riscos associados ao sequestro de sessão.

```
SECRET_KEYSEC = b'mV9c@2T#3jW^7b8L$x*QZp1n%A!5yK4X'
```

V8.3.3 - Sensitive Private Data

Este tópico aborda uma prática de segurança que garante que os utilizadores recebam informações claras sobre a coleta e o uso dos dados pessoais fornecidos. É essencial que os utilizadores compreendam como suas informações serão utilizadas e que tenham a oportunidade de fornecer consentimento explícito antes que seus dados sejam usados de qualquer forma. Essa abordagem minimiza o risco de uso indevido de informações pessoais e assegura a conformidade com regulamentos de privacidade e proteção de dados, protegendo tanto os direitos dos utilizadores quanto a reputação da organização.

Versão original

Na versão original da aplicação, os utilizadores não recebiam informações claras sobre o uso dos seus dados nem tinham a opção de fornecer consentimento explícito. Isso resultava no uso dos dados pessoais sem a devida autorização dos utilizadores, comprometendo a transparência e a conformidade com as normas de privacidade.

Versão segura

Na versão segura da aplicação, os utilizadores são claramente informados sobre a coleta e o uso dos seus dados pessoais. Antes de qualquer utilização, é obtido o consentimento explícito dos utilizadores, garantindo que eles tenham total controlo sobre as suas informações. Isso assegura a conformidade com as normas de privacidade e fortalece a confiança na aplicação.