Uma metodologia para o incremento da segurança no desenvolvimento de aplicações web com CodeIgniter

Elias Spanhol[1], Roger Sá da Silva[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Veranópolis – RS – Brasil

elias.spanhol2@gmail.com

Resumo. *A tecnologia é parte do dia a dia do ser humano. A entrada de dados em plataformas digitais e redes sociais é muito comum, deixando-os cada vez mais vulneráveis ao ataque de cibercriminosos ou usuários que buscam explorar as vulnerabilidades destes sistemas. Sendo assim, este trabalho tem como objetivo a criação de um protótipo de uma aplicação web juntamente com uma metodologia para incremento de segurança, enfatizando a confiabilidade dos dados fornecidos por futuros usuários. Dessa forma, o trabalho mostra que um software desenvolvido com seus devidos padrões de segurança se torna muito mais seguro, reduzindo, assim, os riscos de falhas e o roubo de informações.*

Abstract. *Technology is part of human life. The entry of data into digital platforms and social networks is very common, leaving them increasingly exposed to attack by cybercriminals or users who seek to exploit the vulnerabilities of these systems. Therefore, this work aims to create a prototype of a reb application together with a methodology to increase security, emphasizing the reliability of the data provided for future users. In this way, the work shows that a software developed with its due security standards becomes much safer, friendly, thus, the risks of failures and the theft of information.*

1. Introdução

Após o surgimento da internet as páginas web, que antes eram somente estáticas, foram substituídas por sistemas mais complexos, nos quais os usuários poderiam interagir na produção de conteúdo, seja através da publicação de vídeos ou por meio da troca de informações em uma rede social.

Desde essa mudança na perspectiva de desenvolvimento de sistemas web, os usuários mal intencionados têm explorado cada vez mais as vulnerabilidades desses sistemas. O Tribunal Regional Eleitoral (2017) afirma que "danos causados por código malicioso, *hackers* e ataques de *denial of service* estão se tornando cada vez mais comuns, mais ambiciosos e incrivelmente mais sofisticados".

Conforme Ribeiro (2013), devido ao aumento dos ataques, a maior dificuldade na criação de sistemas web foi garantir a proteção do sistema desenvolvido da melhor forma possível, para que esses ataques não sejam eficazes.

Com o avanço da tecnologia e o crescimento da área de sistemas de informação, a gestão de vulnerabilidades de softwares não acompanhou tal desenvolvimento em tão

pouco tempo. Por isso, Khan e Zulkernine (2009) deixam claro que uma das fontes para os problemas de segurança é a ausência de planejamento e controle de segurança durante o processo de desenvolvimento de software.

Compreende-se por segurança na internet, de acordo com Ribeiro (2013), o cuidado com elementos como: infraestrutura, computadores e informações fornecidas, grandes alvos de cibercriminosos. A partir disso, Holanda e Fernandes (2009-2011) destacam que a segurança da informação em ambientes tecnológicos depende, em grande parte, de sua aplicação na aquisição de desenvolvimento de software.

Destaca-se que um software construído, desde o início, com padrões de segurança diminui muito os riscos no vazamento e roubo de informações. Segundo Sonmez e Kilic (p. 25858, 2021) "[...] é melhor produzir um sistema seguro no início do que abraçar defesas após a implantação". Diante desse contexto, o presente trabalho tem por objetivo propor uma metodologia para o incremento de segurança no desenvolvimento de aplicações web com o framework Codeigniter e banco de dados MySQL. Ademais, pretende-se desenvolver um protótipo de página web que irá demonstrar a implementação da metodologia da segurança de dados proposta.

Nesse sentido, o trabalho está organizado da seguinte forma, na seção 2 será feita a fundamentação teórica a respeito da segurança no desenvolvimento web. No tópico 3, serão descritos os métodos e como realizar um protótipo de aplicação web e a metodologia proposta e utilizada no protótipo. Por fim, na seção 4, está disposta a apresentação e discussão dos resultados, e na seção 5, as considerações finais deste trabalho.

2. Fundamentação Teórica

A segurança é vista como uma qualidade de serviço que garante o fornecimento do próprio serviço, mesmo diante de ações de indivíduos não autorizados no sistema, sem que ocorram violações de segurança. Com a popularização e a constante evolução da internet, onde a maioria dos sistemas passa a fazer parte e depender da rede, as boas práticas de segurança em sistemas web são cada vez mais imprescindíveis [FERRÃO; MACEDO; KREUTZ; 2018].

De acordo com Pressman (2006), "Software de computador é o produto que os profissionais de software constroem e, depois, mantém ao longo do tempo. Abrange programas que executam em computadores de qualquer tamanho e arquitetura, conteúdo que é apresentado ao programa a ser executado e documentos tanto em forma impressa quanto virtual que combinam todas as formas de mídia eletrônica".

Segundo Zavala e Alvarado (2018), "Atualmente o desenvolvimento web está crescendo e crescendo em um ritmo vertiginoso, mais e mais aplicativos vão trabalhar na internet, incluindo aplicativos, telefones celulares e outros dispositivos [...]". Quando são instalados em um servidor host tornam-se, imediatamente, acessíveis aos usuários a qualquer hora e em qualquer lugar via internet com determinados dispositivos conectados à rede.

As violações de segurança sempre existiram e ocorrem devido à exploração das vulnerabilidades existentes nos sistemas. Um erro de programação, de configuração ou mesmo de operação, podem permitir que usuários não autorizados entrem na aplicação ou que usuários autênticos executem ações não autorizadas podendo, assim, comprometer o funcionamento correto da aplicação [Bishop e Bailey, 1996].

Conforme Shuaibu e Ibrahim (2018), "A segurança é um atributo do software e dos aplicativos da web, isso sempre deve conter cuidados especiais. Um simples defeito em software ou na web pode deixar os usuários abertos a invasores que encontram tal defeito para exploração". Um exemplo desses defeitos ou falhas é o *injection*, que ocorre quando informações não confiáveis são enviadas ao servidor como parte de um comando ou solicitação. Essas informações incluem scripts SQL que induzem o servidor a executar comandos indesejados ou permite o acesso a dados não autorizados [OWASP, 2013].

Essa vulnerabilidade consiste, basicamente, em injetar comando SQL em campos de formulário ou até mesmo na URL da aplicação. Segundo Lin, Chen e Liu (2008) esta vulnerabilidade acontece por falta de senso de segurança do programador, os autores, ainda, alertam para a possibilidade do atacante inserir códigos hostis na aplicação, que pode ser utilizado, inclusive, para a futura execução de ataques mais elaborados, tais como o XSS (Cross-Site Scripting).

Ao se construir um software deve-se conhecer as vulnerabilidades que podem ameaçar o sistema, assim, sabe-se em qual fase do ciclo de vida deste, elas podem ser evitadas ou removidas. Fonseca (2013) relata que as vulnerabilidades são brechas em um sistema computacional, quando trata-se de aplicações, essas mesmas vulnerabilidades são chamadas de bugs.

Devido ao grande avanço no desenvolvimento de aplicações web, é cada vez mais importante desenvolver aplicações robustas de forma organizada no menor tempo possível. Isso força os desenvolvedores a parar de criar aplicativos do zero e usar cada vez mais ferramentas para ajudar a produzir softwares [ANDRADE, 2011].

Com todas essas possibilidades de invasões e ataques, cabe aos desenvolvedores tomarem certos cuidados e usarem ferramentas que ajudem a evitar tais problemas. Neste protótipo foi usado o banco de dados MySQL, cuja função é armazenar, pesquisar, classificar e recuperar informações com eficiência. O servidor do MySQL gerencia o acesso aos dados para garantir que vários usuários possam processar os dados ao mesmo tempo, fornece acesso rápido aos dados e garante que apenas users autorizados possam acessá-los. Trabalha com questões de segurança em um sistema de privilégios e senha muito flexível e segura, que permite a verificação baseada em host. Também possui segurança de senha por criptografia de todo o tráfego de palavras chaves quando conecta-se a um servidor [MYSQL, 2022].

O desenvolvimento de aplicações web utilizando frameworks se expandiu de tal forma que se tornou imprescindível o uso destes aparatos para construção de sistemas voltados para internet/intranet/web [SOUZA, 2010]. Um framework é uma base de onde

se pode desenvolver algo maior ou mais específico. É uma coleção de códigos-fontes, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de softwares [MINETTO, 2007].

CodeIgniter é um kit de ferramentas para aplicações PHP. Sua finalidade é possibilitar o desenvolvimento mais rápido dos projetos. Consegue-se concentrar no programa minimizando a quantidade de códigos necessários para uma tarefa específica. Esta ferramenta é lançada sob uma licença de código aberto no estilo Apache/BSD. Portanto, pode ser usado para qualquer finalidade.

O codeIgniter foi criado com os objetivos de instanciamento dinâmico, onde os componentes são carregados e rotinas executadas apenas quando requisitadas. Também tem o objetivo de acoplamento flexível, ou seja, quanto menos um componente depender do outro, mais reutilizável e flexível um sistema se torna. E por fim, a singularidade do elemento, onde cada classe e suas funções são autônomas a fim de serem as mais úteis possíveis.

No trabalho de Ribeiro (2013) é abordado duas aplicações web onde uma é feita sem a implementações de segurança e outra com as validações, assim, pode-se comparar uma e outra. Já em Andrade (2011) é feito um comparativo entre frameworks, apresentando suas qualidades e vulnerabilidades.

Diferentemente dos trabalhos relacionados, este apresenta uma metodologia de segurança no desenvolvimento web, colocando algumas técnicas de segurança em prática. Por fim, segundo Campos (2007) pode-se descrever que uma ameaça consiste em uma possível ação que, se concretizada, poderá produzir efeitos indesejados ao sistema, comprometendo a confidencialidade, a integridade, a disponibilidade e/ou a autenticidade.

3. Métodos

Este trabalho foi realizado em duas grandes etapas. Na primeira, desenvolveu-se a metodologia de incremento de segurança com base nos problemas preconizados pela OWASP. Já na segunda, implementou-se a metodologia junto ao desenvolvimento de um protótipo de aplicação web no CodeIgniter.

3.1. Metodologia de incremento de segurança

De primeiro modo, foi realizada uma pesquisa de possíveis riscos e falhas que uma aplicação web pode apresentar, bem como, formas de prevenir ou reduzir o risco dessas situações. Em Lyra (2008), relata-se que as violações de segurança ocorrem devido à presença de um agente em busca de um feedback sobre informações de empresas/pessoas, podendo se categorizar como um “ataque”.

A Fundação OWASP (Open Web Application Security Project) é uma organização sem fins lucrativos que trabalha para melhorar a segurança do software, ou seja, é a fonte para desenvolvedores e tecnólogos protegerem a web. A OWASP disponibiliza um top 10 de principais riscos de segurança de aplicativos web, esse ranking é um documento de conscientização padrão para desenvolvedores e segurança

de aplicativos da web. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicações da web. Algumas dessas vulnerabilidades são tratadas neste trabalho e descritas, detalhadamente, posteriormente [OWASP, 2013].

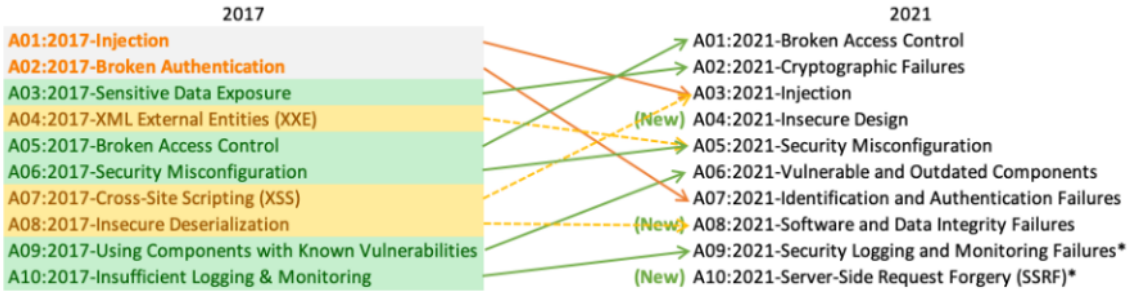

Figura 1. Os 10 principais riscos de segurança de aplicativos web segundo a OWASP [OWASP, 2013]

Com a proposta de metodologia deste trabalho, foram analisados os problemas que a OWASP apresenta sobre segurança em aplicações web, conforme figura 1, e com isso, as ações foram pensadas e realizadas no protótipo.

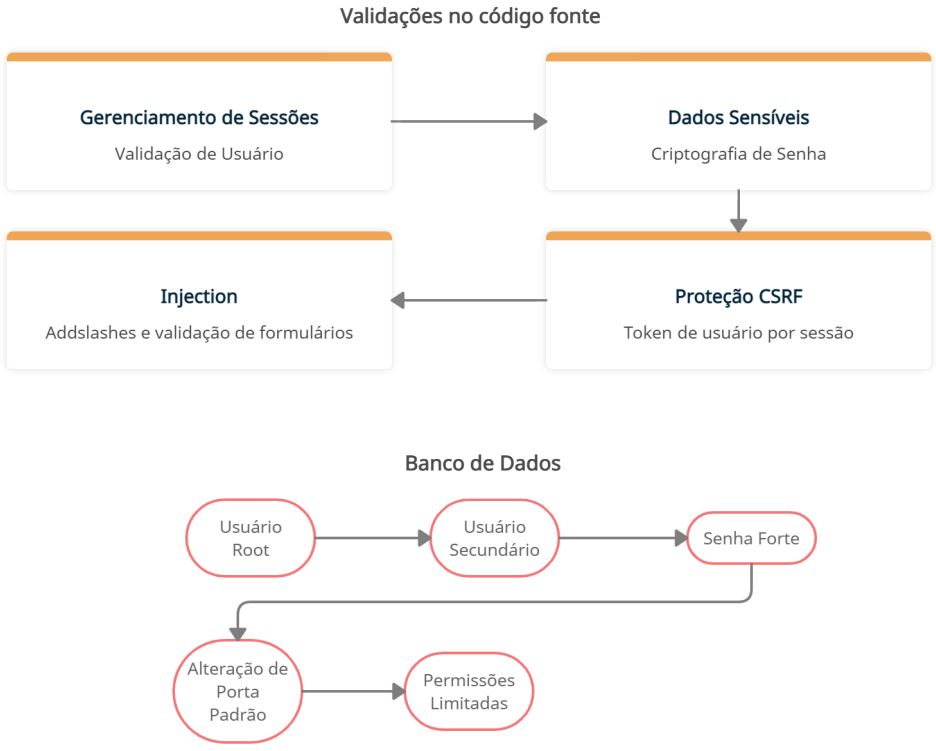

Figura 2. Metodologia de segurança proposta e ordem de implementação [autor]

No diagrama da figura 2, mostra-se a modelagem para a metodologia de incremento de segurança e a ordem de tratamentos a serem realizados em uma aplicação web, juntamente com as sugestões de quais recursos disponíveis no framework CodeIgniter podem ser utilizados para validar cada questão.

Com as validações descritas e realizadas posteriormente se cumpre algumas das possíveis maneiras de deixar um sistema mais seguro, conforme o top 10 da OWASP. Dessa forma, se aborda as melhores em relação a:

- injection;
- gerenciamento de sessões;
- dados sensíveis;
- solicitações CSRF (Cross-Site Request Forgery);
- criação de um usuário secundário para o banco de dados (não foi utilizado o usuário principal, ou seja, o root);
- permissões limitadas (banco de dados);
- alteração da porta padrão (banco de dados).

Sendo assim, com os tratamentos apresentados sugere-se esta metodologia de validações, a qual tem o intuito de dificultar o acesso a informações de uma aplicação web caso esta sofra ataques. Por meio dela, pretende-se, também, tratar de algumas das principais vulnerabilidades das aplicações, deixando-as, assim, mais seguras.

3.2. Desenvolvimento do protótipo e implementação da metodologia

Neste trabalho, desenvolveu-se no padrão MVC (*Model, View, Controller*) um protótipo para demonstração das técnicas de segurança. O mesmo foi projetado em formato de agenda, onde o usuário fornece dados pessoais, podendo executar as operações de listar, inserir, editar e excluir. O projeto, também, dispõe de uma tela de login e de cadastro de usuário.

Após a apresentação do protótipo e também do top 10 da OWASP, conforme a figura 1, a primeira validação adotada no código fonte do protótipo foi a validação de usuário com login ativo. Esse processo foi feito com variáveis de sessão comparando se o nome de usuário e senha são válidos para se conseguir acessar determinada página.

Segundo a documentação do CodeIgniter, quando uma página é carregada, a classe de sessão verifica se um cookie de sessão válido é enviado pelo navegador do usuário. Se um cookie de sessão não existir (ou se não corresponder a um armazenado no servidor ou tiver expirado), uma nova sessão será criada e salva. Se existir uma sessão válida, suas informações serão atualizadas. Com cada atualização, o identificador da sessão pode ser gerado novamente, se configurado para isso [CODEIGNITER, 2022].

No código fonte do protótipo essa validação de sessão é estabelecida através de um filtro criado, onde o mesmo verifica se o perfil do usuário tem ou não permissão para acessar aquela página. Essa sessão é iniciada automaticamente para todos os controllers desse CRUD.

Neste filtro, é iniciado o serviço de sessão e verificado se aquele perfil de usuário tem ou não permissão para acessar determinadas páginas, conforme figura 3.

```php
public function before(RequestInterface $request, $arguments = null){
    $session = service('session');
    if(! in_array($session->get('perfil'), $arguments)){
        die('Voce não tem permissão para acessar essa página');
    }
}
```

Figura 3. Trecho de código mostrando a validação de sessão [autor]

Juntamente com a sessão, também foi adotado o método de proteção CSRF. Este ocorre quando uma requisição HTTP é feita entre sites na tentativa de se passar por um usuário legítimo. Quem utiliza desse tipo de ataque normalmente espera que o usuário alvo esteja autenticado no site onde a requisição fraudulenta será realizada, a fim de obter mais privilégios e acessos a operações [TEDESCO, 2018].

A proteção CSRF é ativada nos filtros do CodeIgniter (App\Config\Filters.php), pois a mesma não vem habilitada por padrão. Posteriormente, foi usado o comando csrf_field() para criar um campo oculto com token único por usuário. Esse token fica salvo na sessão do usuário no servidor e, quando o formulário é postado, o token enviado pelo formulário é comparado com o que se tem na sessão, armazenado no servidor. Com o filtro CSRF global ativo, ele inserirá automaticamente um CSRF oculto nos formulários [TEDESCO, 2018].

Outra configuração feita nos filtros do CodeIgniter foi ativar a classe Honeypot. A classe Honeypot possibilita determinar quando um bot faz a solicitação usando o formulário da aplicação. Isso é feito anexando campos de formulário a qualquer outro formulário, e esse campo fica oculto de um humano, mas acessível a um bot. Quando os dados são inseridos no campo, assume-se que a solicitação vem de um bot e o CI4 lançará "*HoneypotException*" e travará a página da web para abordar a solicitação feita pelo bot e impedirá que a aplicação torne-se uma fonte de spam [CODEIGNITER, 2022].

Para dados sensíveis como a senha do usuário foi utilizada a criptografia. Esta, segundo Howard e LeBlanc (2005), é algo muito importante para garantir a privacidade e integridade dos dados e deixa muito mais forte a autenticação.

Para tanto, foi criado um método no qual trata-se a senha que o usuário digitou e esta é inserida no banco de dados de uma forma criptografada. Por meio desse processo, em caso de uma invasão, a senha não estará salva como foi digitada, e sim em um formato de "código criptografado", tornando-a mais difícil de ser decifrada.

```php
protected function hashSenha($data){
    $data['data']['senha'] = password_hash($data['data']['senha'], PASSWORD_DEFAULT);
    return $data;
}
```

Figura 4. Trecho de código mostrando a criptografia de senha [autor]

Nessa metodologia também foi utilizada a validação de formulários. Os mesmos foram configurados com seu tipo de conteúdo correto (por exemplo, um campo de senha estar com seu input *type* como *password*) e ajustados como campos obrigatórios para preenchimento, para que o usuário não consiga fazer uma inserção no banco com

informações em branco. Além disso, foram descritas suas mensagens de erro em casos de informações incorretas, e mensagens de sucesso ao concluir um processo.

Para as configurações no MySQL, foi criado um usuário secundário para o qual registrou-se uma senha forte (com uso de caracteres especiais, mais de 8 dígitos, letras maiúsculas e números) e permissões diferentes que a do usuário principal (*root*). As autorizações dadas a esse usuário foram para as operações de *create*, *select*, *insert*, *update* e *delete*. Também foi realizada a troca da porta padrão na qual o servidor de banco de dados MySQL executa: ao invés da porta 3306 utilizou-se a porta 2200.

Por fim, este trabalho foi desenvolvido na linguagem de programação PHP com auxílio do framework CodeIgniter 4 e o banco de dados MySQL, juntamente com a metodologia de segurança na qual são tratadas as vulnerabilidades de uma aplicação web. Ainda, algumas vulnerabilidades são testadas para a validação da implementação da metodologia, conforme evidenciado na seção dos resultados.

4. Apresentação e discussão dos resultados

Segundo Howard e LeBlanc (2005), a propriedade privada é protegida desde o início dos tempos. Mesmo nossos ancestrais mais antigos tinham leis que puniam as pessoas que atacavam, danificavam ou roubavam a propriedade privada. No mundo digital não é diferente, pois sempre que há a necessidade de proteger bens privados, cabe aos desenvolvedores a criação de soluções e aplicativos que protejam o patrimônio digital.

Por tal, é preciso garantir a segurança do software, porém, segundo Braz (2009), apesar das normas de segurança serem bem claras, as mesmas não disponibilizam de uma forma detalhada de implantação nos processos, tornando difícil para as empresas saber, exatamente, como atendê-las em sua plenitude.

Dito isso, propôs-se neste trabalho uma metodologia para o desenvolvimento de aplicações web, com algumas das principais vulnerabilidades do top 10 da OWASP tratadas e testadas. A partir do diagrama da figura 2, formalizou-se a metodologia conforme descrita em detalhes no quadro 1.

Quadro 1. Detalhes da metodologia de incremento de segurança [autor]

Ação a ser realizada	**Recurso(s) a ser(em) utilizado(s)**	**Vulnerabilidade tratada**	**Resultado esperado**
Validação de usuário com login ativo.	Criação e gerenciamento de sessões de usuário no CodeIgniter.	Acesso indevido a páginas e dados pelo não gerenciamento de sessões.	Impedir o acesso a páginas da aplicação diretamente pelo URL sem o prévio login, com verificação de usuário e senha.
Proteção de dados sensíveis de usuários.	Utilização de função para criptografia da informação desejada.	Acesso a uma informação confidencial do	Impedir que um usuário mal intencionado tenha

		usuário. Com a utilização da criptografia a informação acessada deve ser decodificada / descriptografada.	acesso a informação que o usuário informou.
Proteção CSRF	Criação de campo oculto com token único por usuário (campo oculto criado nos formulários).	Solicitações HTTP involuntárias que chegam na aplicação onde o usuário está autenticado.	Não permitir que um usuário mal intencionado induza um usuário legítimo a executar ações sem seu consentimento.
Proteção contra SQL Injection	Utilização de função PHP para não aceitar que caracteres especiais cheguem através de formulários, juntamente com a validação dos mesmos (campos obrigatórios e com conteúdo correto).	Entrada de caracteres especiais através de *scripts* sql em formulários.	Não aceitar *scripts* sql vindos de formulários.

Seguindo esta metodologia, consegue-se tratar algumas das principais ameaças e minimizar vulnerabilidades, como mostrado nos testes da próxima seção.

4.1. Teste de gerenciamento de sessões

A vulnerabilidade das sessões de usuários permite que invasores explorem pontos fracos ou brechas na autenticação e nas funções de gerenciamento de sessão, como o uso de senhas e IDs de sessão para representar outros usuários [OWASP, 2013].

A implementação dos recursos relacionados a autenticação e gerenciamento de sessão é relativamente difícil de executar corretamente do ponto de vista da segurança. Dessa forma, os invasores podem até explorar outros bugs de implementação do sistema que ele não pode acessar sem assumir a identidade de outro usuário [OWASP, 2013].

Para realizar esse teste foi usada a ferramenta W3AF para encontrar as URLs do sistema. Esse aparato é um framework de ataque e auditoria de aplicações web e nela já existem alguns perfis de testes prontos, porém há, também, a possibilidade de se criar novos perfis para verificações.

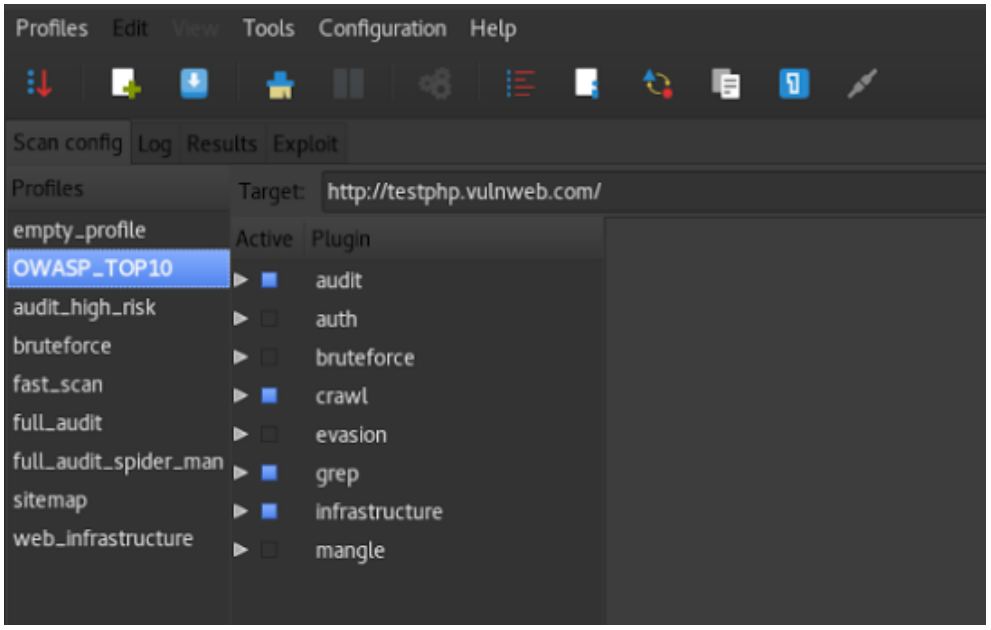

Figura 5. Perfis na interface da ferramenta W3AF [W3AF, 2013]

W3AF é uma estrutura de auditoria e ataque de aplicativos web. O objetivo dessa ferramenta é criar uma estrutura para ajudar o desenvolvedor a proteger seus projetos, encontrando e explorando todas as vulnerabilidades de aplicações web [W3AF, 2013].

Neste teste, foi realizada a busca de todas as URLs da aplicação com o auxílio do W3AF. Com o scanner realizado e encontrando todas URLs do projeto, foi realizado de forma manual o teste tentando acessar link a link as páginas e recursos da aplicação, sem efetuar o login nas mesmas. Com essa verificação, ao tentar acessar os endereços, houve um direcionamento automático para a página de login (conforme figura 6), resultado esperado após o incremento de segurança proposto na metodologia. Assim, não foi possível acessar a aplicação sem que o usuário estivesse autenticado.

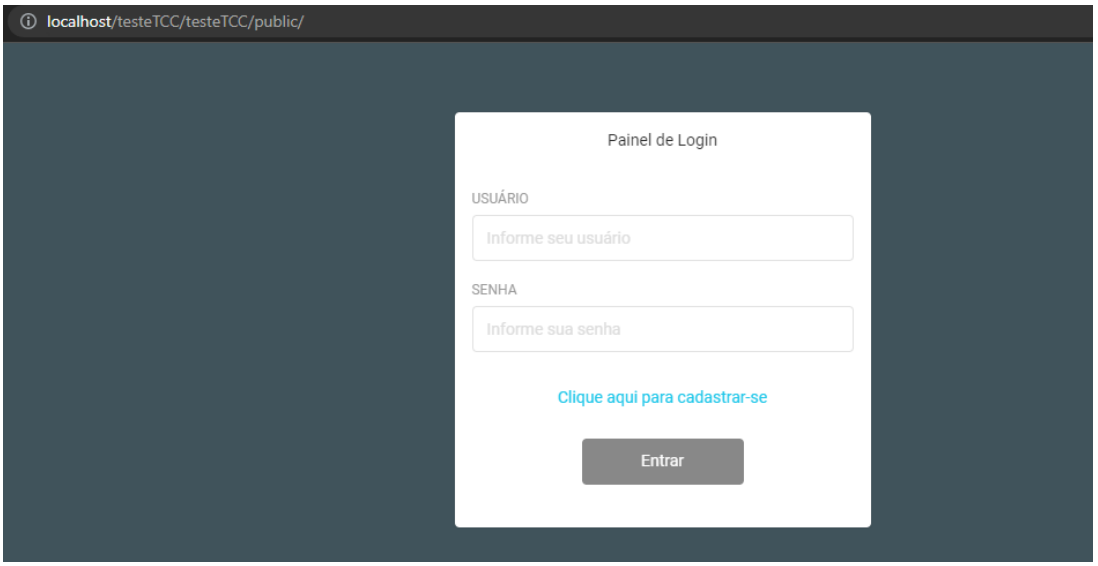

Figura 6. Resultado de tentativa de acesso à página da aplicação sem sessão de login criada [autor]

Dessa forma, caso um possível ataque tivesse acesso às URLs do protótipo e tentasse acessar a aplicação diretamente com o link, ele não iria conseguir acesso sem se autenticar nas páginas.

4.2. Teste de *SQL Injection*

Conforme Junior (2013), esse é um tipo de ataque que explora falhas de segurança nas interações do banco de dados. O desconhecimento dos desenvolvedores sobre as práticas de implementação deixa o sistema vulnerável, permitindo que hackers ataquem via *sql injection*.

As vulnerabilidades do *sql injection* acontecem quando é permitida a entrada de caracteres especiais que alteram o fluxo da consulta *sql* no banco de dados, ou seja, esse teste é feito nos formulários da aplicação através da inserção de um trecho de um comando *sql*.

Para este teste foi usado um trecho de *script sql* em um dos formulários do CRUD, conforme mostra a figura 7. Com essa injeção em uma aplicação vulnerável, seria possível ter acesso a toda a tabela de usuários desse projeto.

Figura 7. Representação do teste de *Injection* de SQL em formulário da aplicação [autor]

O comando **‘ or 1=1 or ‘’=’** faz com que o usuário e senha informados sejam sempre verdadeiros, ou seja, com isso seria possível ter um acesso indevido ao sistema. Dito isso, para se tratar essa questão no código fonte foi usado a função PHP *addslashes*. Essa função tem o intuito de substituir caracteres oriundos de formulários, com o objetivo de não aceitar informações indevidas (conforme figura 8). Este comando é aplicado nos métodos *post* de usuário e senha na aplicação.

Com essa função aplicada, ao inserir um *script* SQL em um formulário a aplicação entende essa informação como falsa, não deixando o possível “usuário” acessar. Dessa forma, elimina-se a possibilidade de *injection* através de formulários, ou seja, minimiza-se o risco de as tabelas e informações do banco de dados serem acessadas e fraudadas.

Figura 8. Representação de script SQL sendo recusado em submissão de formulário da aplicação [autor]

5. Considerações finais

Este trabalho mostrou que implementar técnicas de codificação segura envolve certos cuidados, pois cada aplicação requer certos direitos e privilégios de usuário. Pretendeu-se que a proposição de uma metodologia de incremento de questões de segurança de dados no desenvolvimento de aplicações web no *framework* CodeIgniter promova os benefícios de menos riscos às aplicações e minimização de certas vulnerabilidades. Isso se dá, também, pelas qualidades relacionadas à segurança de uso de um *framework* que auxilia em validações de informações, juntamente com um sistema de banco de dados corretamente configurado.

Pode-se perceber que utilizar frameworks para desenvolver aplicações PHP é um bom caminho para alcançar resultados confiáveis, pois eles são bastante maduros e permitem construir aplicações cada vez mais flexíveis, além de compor um processo de desenvolvimento mais rápido. Percebe-se, ainda, que a metodologia sobre segurança proposta no trabalho se mostrou adequada no tratamento e previsão de ataques, dificultando possíveis ataques.

Apesar das limitações como a não implementação de soluções de segurança para todas as vulnerabilidades citadas no top 10 da OWASP, além da não realização de testes para as todas validações realizadas neste projeto, visto que muitas validações são garantidas pela própria ferramenta CodeIgniter, entende-se que os resultados obtidos são satisfatórios.

Com essa metodologia e técnicas utilizadas na parte de segurança consegue-se deixar a aplicação web mais segura, diante de ameaças que podem ser tratadas no próprio código fonte e no banco de dados. Dessa forma, o trabalho mostra que um software desenvolvido com seus devidos padrões de segurança, se torna muito mais seguro, reduzindo, assim, os riscos de falhas e o roubo de informações.

Por fim, para trabalhos futuros, sugere-se a inclusão nesta metodologia de soluções para o restante das vulnerabilidades da OWASP, podendo-se também realizar testes com cada uma delas, ampliando-se também o uso para diferentes frameworks e bancos de dados.

Referências

ANDRADE, Fernando Francisco de. **Desenvolvimento de aplicações web com a utilização dos frameworks CodeIgniter e Doctrine**. 2011. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

BISHOP, M. e BAILEY, D. (1996). **A critical analysis of vulnerability taxonomies**. Technical Report CSE-96-11, Department of Computer Science at University of California, Davis.

BRAZ, F. **Instrumentação da análise e projeto de software seguro baseada em ameaças e padrões**. Tese (Doutorado em Engenharia Elétrica) - Faculdade de tecnologia, Brasília - Brasil. 2009.

CAMPOS, A. **Sistema de segurança da informação: controlando os riscos**. 2.ed. – Florianópolis : Visual Books, 2007.

CODEIGNITER (2022). CodeIgniter Foundation. Disponível em: <https://codeigniter4.github.io/CodeIgniter4/libraries/sessions.html?highlight=session>. Acesso em: 29 out. 2022.

DE HOLANDA, Maristela Terto; FERNANDES, Jorge Henrique Cabral. Segurança no desenvolvimento de aplicações. **Gestão da Segurança da Informação e Comunicações - CEGSIC 2009-2011**, 2009.

FONSECA, Peixinho Francisco Marmo da, F. M. M. L. Ivo de C. **Segurança de Redes e Sistemas: Princípios e práticas**. [S.l.]: RNP/ESR, 2013. p.251.

FERRAO, Isadora Garcia; DE MACEDO, Douglas DJ; KREUTZ, Diego. **Investigação do impacto de frameworks de desenvolvimento de software na segurança de sistemas web.** 16a Escola Regional de Redes de Computadores (ERRC). https://bit.ly/2WQAfZd, 2018.

HOWARD, MICHAEL; LEBLANC, DAVID. **Escrevendo código seguro: estratégias, técnicas práticas para codificação segura de aplicativos em um mundo em rede**. 2.ed. – Porto Alegre: Bookman, 2005.

JUNIOR, Antonio Carlos Gonçalves. **Um estudo de segurança da informação: injeção de sql**.

KHAN, M. U. A.; ZULKERNINE, M. **Activity and Artifact Views of a Secure Software Development Process**. International Conference on Computational Science and Engineering. Vancouver - Canadá. IEEE, 2009. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5283250&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5282954%2F5282955%2F05283250.pdf%3Farnumber%3D5283250>. Acesso em: 30 out. 2022.

LIN, Jin-Cherng; CHEN, Jan-Min; LIU, Cheng-Hsiung. **An Automatic Mechanism for Sanitizing Malicious Injection**. In: Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for. IEEE, 2008. p. 1470-1475.

LYRA, MAURÍCIO ROCHA. **Segurança e auditoria em Sistemas de Informação**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2008.

MINETTO, Elton. **O nascimento de um site com Codeigniter – Intro.** Disponível em: <http://www.codeigniter.com.br/tutoriais/74/o-nascimento-um-site-com-codeigniter-into>. Acesso em 30 out. 2022.

MYSQL (2022). Oracle. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>. Acesso em: 29/10/2022

OWASP (2013). The Open Web Application Security Project. Disponível em: <https://www.owasp.org/index.php/Main_Page>. Acesso em: 15 set. 2022.

PRESSMAN, Roger S. **Engenharia de Software**. Sexta Edição. São Paulo; MacGraw-Hill. 2006.

RIBEIRO, Wellington Z. **Aplicação de Boas Práticas de Segurança no desenvolvimento de Sistemas Web**. São João da Boa Vista, 2013.

SHUAIBU, Musa Bala; IBRAHIM, Ruqayyat Ahmad. **Web application development model with security concern in the entire life-cycle**. International Conference on Engineering Technologies and Applied Sciences, jan 2018.

SONMEZ, Ferda Ozdemir; KILIC, Banu Gunel. **Holistic web application security visusalization for multi-project and multi-phase dynamic application security test results**. IEEE Access, 2021.

SOUZA, Fernando. **Uso de frameworks para desenvolvimento *WEB* e mitos que já deveriam ter desaparecido.** Disponível em: <http://www.franciscosouza.com.br/2010/01/22/uso-de-frameworks-para-desenvolvimentoweb-e-mitos-que-ja-deveriam-ter-desaparecido/>. Acesso em 30 out. 2022.

TEDESCO, Kennedy. **Cross-Site Request (CSRF) e abordagens para mitigá-lo**, 2018. Disponível em: <https://www.treinaweb.com.br/blog/cross-site-request-forgery-csrf-e-abordagens-para-mitiga-lo#:~:text=CSRF%20Tokens,na%20sess%C3%A3o%2C%20l%C3%A1%20no%20servidor>. Acesso em: 02/11/2022.

TRIBUNAL REGIONAL ELEITORAL - TRE/BA (2017). Segurança da informação. Disponível em: <https://www.tre-ba.jus.br/transparencia-e-prestacao-de-contas/governanca-e-gestao/seguranca-da-informacao>. Acesso em: 22 dez. 22.

ZAVALA, Álvaro; Alvarado, Iván. **The importance of good pratices in the development of secure web applications and their repercussions for not implementing them**. Proceedings of the 2018 IEEE 38th Central America and Panama Convertion, Concapan, 2018.

W3AF (2013). w3af.org. Disponível em: <https://w3af.org/>. Acesso em: 01 dez. 2022.