INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DE SÃO PAULO
CÂMPUS VOTUPORANGA

MARIANA DE ALMEIDA RODRIGUES

BOAS PRÁTICAS DE PROGRAMAÇÃO NO DESENVOLVIMENTO DE APLICAÇÕES WEB

VOTUPORANGA
2021

Mariana de Almeida Rodrigues

BOAS PRÁTICAS DE PROGRAMAÇÃO NO DESENVOLVIMENTO DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do curso Técnico em Informática Integrado ao Ensino Médio no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Votuporanga.

Professor Orientador: Ubiratan Zakaib do Nascimento.

Votuporanga

2021

Mariana de Almeida Rodrigues

BOAS PRÁTICAS DE PROGRAMAÇÃO NO DESENVOLVIMENTO DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do curso Técnico em Informática Integrado ao Ensino Médio no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Votuporanga.

Professor Orientador: Ubiratan Zakaib do Nascimento.

BANCA EXAMINADORA:

_____

Prof. Me. André Luis Gobbi Primo

_____

Prof. Ma. Cristiane Paschoali de Oliveira Vidovix

_____

Prof. Me. Ubiratan Zakaib do Nascimento

Dedico este trabalho aos meus professores,
amigos e familiares, que me ajudaram e
apoiaram no desenvolvimento
deste projeto.

AGRADECIMENTOS

Agradeço a Deus por ter me dado força e sabedoria para concluir este projeto.

Sou grata a minha família pelo apoio nas horas mais difíceis, aos meus professores e servidores do IFSP, Câmpus Votuporanga, que sanaram todas as minhas dúvidas, durante o seu desenvolvimento. Não apenas eles, mas sou gratificada por ter estudado em uma instituição que me deu oportunidade e qualidade de ensino.

E principalmente, agradeço ao meu orientador, Bira, por ter caminhado junto comigo e me dado apoio e também, por ser meu amigo nas horas vagas.

Aos meus amigos, que me acompanharam durante todos esses anos, que fizeram os meus dias mais alegres, eu só posso dizer, obrigado!

“Nós só podemos ver um pouco do futuro,
mas o suficiente para perceber
que há muito a fazer”
Alan Turing

RESUMO

A segurança em aplicações *web* vem se tornando um assunto relevante da área de tecnologia. Contudo, ainda não é a parte principal de um projeto, pois, no começo da tecnologia, os ataques não eram tão engenhosos e elaborados quanto agora. Além disso, atualmente, os *hackers* maliciosos possuem diversas ferramentas e meios para atacar. Logo, a forma de projetar um sistema deve ser repensada. Existe a necessidade de se conhecer os métodos de ataque, para assim aplicar técnicas de correção, para tal, foram feitas pesquisas experimentais e bibliográficas visando se familiarizar mais com o tema proposto, assim como comprovar a necessidade de debater esse tema com mais afinco. Desta forma, o trabalho aborda características com o intuito de implementar mecanismos de segurança, de forma simples, seja utilizando *frameworks*, funções de linguagens de programação ou marcação. Além disso, o código escrito é comparado com uma aplicação vulnerável, em razão comprovar os métodos usados. De acordo com os resultados é possível comprovar que, com pequenas ações no *frontend*, evita-se que um sistema passe por uma catástrofe.

Palavras-chave: Segurança da Informação; Aplicações Web; Aplicação Vulnerável.

ABSTRACT

Security in web applications has become a relevant issue in the technology area. However, it is not yet the main part of a project, because, in the dawn of the technology, cyber attacks were not as well engineered and elaborate as now. Besides, nowadays malicious hackers can make use of a wider variety of tools and means to attack. Therefore, the way to design a system must be rethought. It is necessary to know the methods of attack, in order to apply correction techniques. To meet that purpose, experimental and bibliographic researches were carried out, both in order to become more familiar with the proposed theme, as well as to prove the need to debate this theme in more detail. Hence, this work addresses specific characteristics in order to implement security mechanisms, in a simple way, whether using framewoks, programming language functions or markup. In addition to this, the written code is compared to a vulnerable application, as it proves the methods used. According to the results, it is possible to prove that, with small actions on the frontend, it is possible to prevent a system from going through a catastrophe.

Keywords: Information Security; Web Aplications; Vulnerable Application.

Lista de figuras

Lista de Quadros

LISTA DE SIGLAS

CSS - Cascading Style Sheets (Folhas de Estilo em Cascata)
DDoS - Distributed Denial of Service (Negação Distribuída de Serviço)
DVWA - Damn Vulnerable Web App (Aplicativo da Web Vulnerável Maldito)
HTML - HyperText Markup Language (Linguagem de Marcação de Hipertexto)
IDE - Integrated Development Enviroment (Ambiente de Desenvolvimento Integrado)
PHP - PHP: Hypertext Preprocessor (PHP: Pré-Processador de Hipertexto)
SQL - Structured Query Language (Linguagem Estruturada para Consultas)
URL - Uniform Resource Locator (Localizador Uniforme de Recursos)
USB - Universal Serial Bus (Porta Serial Universal)

SUMÁRIO

1 INTRODUÇÃO

O trabalho tem a finalidade de mostrar boas práticas de programação, que pretendem deixar o desenvolvimento do *frontend*, neste caso destinado à *web*, mais seguro, mitigando ao máximo as possíveis vulnerabilidades. Para o norteamento dos principais problemas de segurança, foi usado o OWASP (Projeto Aberto de Segurança em Aplicações Web), que visa demonstrar um *ranking* das maiores falhas do ano e listá-las em tópicos, entre outras coisas como documentos, artigos, ferramentas do ramo (OWASP, 2020).

Segundo o autor Carvalho et al.(2013), talvez a indiferença em relação ao tema, seja pela falta de conhecimento sobre essas vulnerabilidades. Então, para trazer conhecimento, o trabalho irá mostrar uma ferramenta que contém diversos erros, para representar um site feito fora das normas de segurança, e desenvolver uma aplicação segura. Esta será um exemplo correto a ser seguido.

1.2 OBJETIVO GERAL

O objetivo geral é ressaltar a importância de uma programação pensada na segurança do *frontend*, que visa gerar mais segurança tanto para o desenvolvedor, quanto para o usuário. Além disso, apresentar também que a segurança é um assunto de extrema importância, que não deve ser feita a partir de lógicas ou retalhos de códigos, que não seguem algum padrão ou norma específica, para que seja discutida como a principal parte de um projeto em execução.

1.3 OBJETIVOS ESPECÍFICOS

- Demonstrar uma aplicação vulnerável, que não segue os devidos cuidados;
- Estudar a linguagem de programação PHP orientada a objeto;
- Explorar as diversas formas de proteger o sistema de ataques;
- Mostrar os resultados dos ataques em uma aplicação não segura;
- Mostrar as falhas de segurança que pode acontecer se a entrada e saída de dados não forem tratadas devidamente;

- Desenvolver uma aplicação simples com algumas técnicas de proteção no *frontend.*

1.4 JUSTIFICATIVA

A cada dia, mais pessoas estão conectadas no mundo virtual, oferecendo seus dados, muitas vezes de forma não intencional. Assim como cresce o número de usuários comuns, cresce também o número de usuários maliciosos, que estão tentando, a todo momento, acessar esses dados. Quando expostos de forma ilícita, como vazamento de fotos, contas de banco, informações pessoais e empresariais, podem se tornar uma fraude em uma eleição devido a controle de dados pessoais.

Segundo o autor Machado et al.(2019), as pesquisas mostram um aumento considerável tanto dos dados vazados, quanto a frequência com o que isso ocorre. Com isso, o mercado de trabalho vem procurando mais profissionais especializados em segurança, já que a maioria das empresas trabalha diretamente com dados pessoais, como os bancos, e falhas são inadmissíveis. Portanto, este trabalho ressalta o valor de se investir na área, pois existem diversas vulnerabilidades já conhecidas e muitas, a serem descobertas.

1.5 METODOLOGIA

Antes do desenvolvimento prático, foi realizado uma pesquisa exploratória e explicativa, com o intuito de levantar informações sobre o problema e, assim, poder formular hipóteses mais precisas. Além disso, foi feito pesquisas experimentais e bibliográficas para se familiarizar mais com o tema proposto.

Em seguida foi realizado um estudo de caso e teste prático, de modo a concretizar as análises feitas. Elas têm embasamento bibliográfico, além de artigos e trabalhos acadêmicos de outros profissionais. Além disso, é válido citar que o espaço em que esse estudo ocorreu foi no meio virtual, pois o foco do trabalho são aplicações *web*.

1.6 ESTADO DA ARTE

De acordo com os autores, L*ü*ders e Bianchini (2017), a gestão pública vem adotando medidas que melhoram a segurança das Cidades Inteligentes. Elas

formam uma rede de informações que em sua maioria, usa a linguagem de programação PHP em 82,9% e *softwares* livres. A segurança deve estar em todas as fases do desenvolvimento, pois, a cada dia, a sofisticação dos ataques vem crescendo. As principais vulnerabilidades testadas são: *SQL Injection,* consiste em adicionar caracteres ou comandos SQL nos campos de formulário através de parâmetros passados por URLs; *Cross-Site Scripting*, ela permite inserir códigos maliciosos JavaScript, por exemplo, para roubar informações contidas nos *cookies*; e *Any File Inclusion*, que é quando um atacante insere um arquivo em um URL, que pode executar instruções maliciosas.

O autor Souza (2012), diz que a segurança de aplicações *Web* não recebe a notoriedade necessária pelas empresas de desenvolvimento. Tendo isso em mente, utilizou-se a metodologia proposta pela OWASP, para realizar uma pesquisa qualitativa, com a finalidade de testar se os métodos propostos são possíveis de serem aplicados valorizando os princípios de segurança de confidencialidade, integridade, autenticidade e disponibilidade. O resultado foi que os desenvolvedores não utilizam normas de segurança durante o ciclo de vida dos projetos de *software*. Logo, a solução encontrada é adotar técnicas e recomendações de segurança no desenvolvimento de aplicações *Web* através de metodologias para tratar riscos e ameaças.

Os autores Ceron et al. ([s.d.]) também notaram que mecanismos de busca estão sendo usados como uma ferramenta para localizar *sites* vulneráveis. Algumas aplicações como *webmail*, aplicativos de gerenciamento remoto, fórum de discussões entre outros, contém as mesmas vulnerabilidades, sendo elas, *Cross Site Scripting, SQL Injection, Directory Transversal,* entre outras*.* Atualmente, para analisar as técnicas de ataque, estão sendo projetadas ferramentas como o *PHP Honeypot Project (PHP.HoP)*, que emulam aplicações *web* vulneráveis e coletam informações sobre os acessos.

Com os dados coletados, foi possível constatar que em um período de 53 dias, foi totalizado 4902 acessos aos serviços *web,* isso sinaliza que há uma alta procura por aplicações *web* vulneráveis. Mecanismos de busca como o Google e Yahoo são ferramentas de sondagem para portas e aplicações, é uma técnica eficiente, pois mecanismos tradicionais de segurança não a detecta.

O autor Monteverde (2014) defende que, o aumento do uso de aplicações *Web*, facilitou não só a se consolidar como um dos principais meios de comunicação mas também a expansão dos ataques a segurança, por outro lado, conhecer as principais vulnerabilidades, permite estabelecer medidas preventivas para garantir a segurança das aplicações, executando, por exemplo, os pilares centrais da segurança. Mas afinal o que são vulnerabilidades? São condições que podem virar falhas de segurança quando exploradas por alguém mal-intencionado. Elas são construídas ao longo do desenvolvimento a partir de uma série de fatores, por exemplo, prazos curtos para entrega do trabalho, falta de qualificação técnica dos desenvolvedores ou até mesmo uma ausência de revisão contínua e atualizações. Entender essas falhas e resolvê-las é o principal passo a ser tomado para evitar custos futuros com manutenção.

De acordo com o autor Taha (2017), grande parte das aplicações armazenam dados sensíveis, com isso em mente, foi desenvolvido um guia de testes de segurança usando a metodologia OWASP como base, para ajudar os desenvolvedores em diversos estágios de aprendizado. O guia conta com técnicas de teste, algumas delas são: a revisão e inspeção de manuais, que são documentos que contém informações sobre a arquitetura da aplicação; modelagem de ameaças, permite que os desenvolvedores identifiquem as ameaças antes de o *software* ser desenvolvido; revisão do código-fonte e muitas outras. Ele realiza testes de penetração em ambiente controlado, demonstra experimentos explicando as ferramentas usadas e apresenta as vulnerabilidades encontradas através dos procedimentos realizados.

Os autores Montanheiro e Carvalho (2018) assumem que, as equipes de desenvolvimento de sistemas apresentam uma eminente falta de conhecimento e atenção a respeito de segurança da informação, e é claro que isso ocasiona falhas. Uma justificativa para esse erro, é que cada vez mais os *softwares* são desenvolvidos em um prazo curto de tempo e são mais complexos. Então, os gestores optam por poupar custo e tempo, deixando a segurança como um acessório e não prioridade. Contudo, toda a equipe de desenvolvimento deveria receber um treinamento sobre o assunto, com a finalidade de mitigar os riscos nas aplicações e entender como as falhas podem gerar vulnerabilidades.

Isso é comprovado pelos autores Costa et al. (2018), que em uma entrevista com desenvolvedores *web*, visando mensurar o nível de conhecimento dos programadores em identificar, analisar e corrigir ameaças a segurança do código, recolheu dados alarmantes. Dos sessenta resultados obtidos pelo *survey, 73*% são respostas de profissionais com mais de 3 anos de experiência e 57% contém curso superior. Mesmo assim, a pesquisa revela que os motivos para não implementação de prevenções contra os ataques são: falta de conhecimento em 92%; prazos muito apertados em 70% e baixa prioridade em 47%.

Além disso, 90% dos entrevistados disseram que a formação acadêmica não oferece uma preparação adequada para lidar com segurança, outros 50% considera o ensino insuficiente, enquanto 37,5% afirmam que o assunto não foi abordado em sua formação. Dos que conhecem o assunto, 47% aprenderam através de livros, cursos ou documentações oficiais, enquanto 17%, aprenderam com a prática da profissão. E apenas 30% afirmaram ter aprendido durante a formação acadêmica. São números extremamente alarmantes quando esses profissionais desenvolvem diversos aplicativos, desde redes sociais até aplicativos de bancos, logo, isso pode afetar a vida de qualquer um e fazer diversos estragos.

2 CONCEITOS BÁSICOS SOBRE O TEMA EM ESTUDO

Para compreender melhor o tema abordado e a sua importância, abaixo se encontram descritos tópicos da segurança da informação.

2.1 SEGURANÇA DA INFORMAÇÃO

Nos dias atuais, disponibilizam-se as informações em diversos sites ou aplicativos. Logo, quando se concede o acesso, essas informações não pertencem mais ao dono, mas sim à *internet,* onde todos, inclusive os sites e aplicativos onde foram postadas, podem usá-las para seu próprio bem.

Tendo em vista o uso dos dados pessoais dos usuários pelas empresas, pode-se dizer que cada acesso vale ouro. É por meio dos cliques que os anúncios certos são direcionados as pessoas certas. Afinal, não é viável oferecer um produto a quem não quer comprar.

Em resumo, ao levantar essas questões, pode-se chegar a um final absoluto, quem está cuidado dessas informações valiosas? As organizações pelas quais se

navega e acessa todos os dias. São elas as responsáveis pelos dados coletados em suas plataformas.

Enfim, o que seria Segurança da Informação? Segundo Fontes (2017), é um conjunto de regras e ações que tem por razão proteger a informação. Sua existência coopera para que o sentido do que foi oferecido seja preservado para as empresas e organizações ou indivíduos.

2.2 PILARES DA SEGURANÇA

Para que a Segurança da Informação seja garantida, são necessários alguns pilares. Sendo eles: Disponibilidade, Integridade, Confidencialidade e Autenticidade.

A Disponibilidade garante que o acesso ao serviço desejado esteja funcionando. Um ataque que afeta diretamente este pilar, é o DDoS, que significa em português ataque distribuído de negação de serviço, ele usa diversas máquinas de modo a sobrecarregar um servidor. Um servidor atende um determinado número de usuários ao mesmo tempo, quando excedido, ele não consegue atender a nenhum pedido, podendo travar ou até mesmo desligar sozinho.

Já a Integridade cuida para que as informações não sejam alteradas ou apagadas sem que o dono dessas informações faça isso. De acordo com Dantas (2011), a Integridade é afetada quando ocorre inserções, substituições ou exclusões no assunto da informação, ou quando há alterações nas permissões de acesso de um arquivo restrito.

Assim como a Confidencialidade trata para que um arquivo seja acessado apenas pelo seu dono, ou pessoa autorizada. Qualquer usuário que acessar sem a permissão do gestor, está atingindo este pilar, e ao acontecer isso, o segredo da informação é perdido.

Por último, a Autenticidade específica que as entidades envolvidas no processo de comunicação sejam elas mesmas, e que a informação passada não seja modificada após seu envio.

Além dos quatro pilares citados, existem alguns tópicos ligados a Segurança da Informação, como, Não Repudio, quando o individuo nega que a algo foi enviada por ele; Auditoria, registrar as ações que ocorreram na rede, para que no futuro sejam verificadas as irregularidades; e Legalidade, a informação armazenada deve estar de acordo com as leis atuais do seu tempo.

2.3 ESPECIFIDADE DO TEMA

O foco principal do trabalho é apresentar vulnerabilidades comuns em diversos sistemas *web*, além disso, mostrar que é possível mitiga-las com técnicas simples no *frontend*, do próprio HTML.

Ademais, apresentar a segurança da informação de modo a facilitar seu intendimento, para que sua implementação também se torne fácil de ser realizada. Para isso, foi feito um sistema *web* com a entrada de dados, a parte que se comunica diretamente com o usuário, devidamente tratada, na qual foi implementado algumas técnicas de segurança, como a validação e filtração de campos, utilizando PHP e JavaScript.

Com isso, é possível comprovar que a segurança deve ser pensada desde a inserção dos dados até a sua manipulação.

3 DESENVOLVIMENTO

Dando início a parte prática da aplicação, serão necessárias algumas ferramentas e tecnologias, neste capítulo será descrito quais são estes requisitos técnicos e suas respectivas funções.

3.1 MATERIAIS E MÉTODOS

No desenvolvimento do trabalho foi usado a linguagem de programação, PHP e a linguagem de *scripting*, JavaScript, o Apache como servidor *web*, o MySQL para o gerenciamento do banco de dados, o VirtualBox para a virtualização do ambiente de ataque, o DVWA para exemplificar a aplicação vulnerável, o Metasploitable para, neste caso, carregar o DVWA, o NetBeans como ambiente de edição de código e o Bootstrap para o *design* do sistema.

Além dos requisitos citados, foi necessário também infraestrutura de rede para o desenvolvimento e pesquisa do trabalho e um computador para a execução da aplicação.

3.2 DESCRIÇÃO DAS PRINCIPAIS FERRAMENTAS

Abaixo, será descrito as ferramentas utilizadas e suas funcionalidades.

3.2.1 APACHE

O servidor *web* Apache mantém quase 50% das páginas na *internet* ativas. Ele tem seu código-fonte aberto e também é multiplataformas. Foi criado em 1995, por Rob McCool.

O nome Apache vem em das tribos de nativos americanos, que lutaram e restiram aos ataques sofridos, em semelhança a comunidade de *software* livre.

Segundo o autor Marcelo (2005), as principais vantagens desse servidor são, suporte a autenticação baseada em HTTP, *Logs* customizáveis, configuração rápida e simples, entre outros.

3.2.2 BOOTSTRAP

Bootstrap é um *framework* de desenvolvimento *web* de código aberto, baseado em HTML, CSS e JavaScript que facilita o *design* do site e também contribui para que o usuário se sinta mais à vontade, facilidade e segurança usando a aplicação, segundo o autor do site, Contributors, [s.d.].

3.2.3 DVWA

DVWA (Damn Vulnerable Web App), é uma aplicação propositalmente vulnerável, baseada em PHP e MySQL. Segundo o autor do site da aplicação DVWA, [s.d.], seus principais objetivos são ajudar os profissionais da área de segurança a testar suas habilidades sem prejudicar os usuários ou outros programadores, também ajudar os profissionais a entender como proteger uma aplicação.

3.2.4 JAVASCRIPT

JavaScript é uma linguagem de *scripting*, como o próprio nome diz, ela trabalha em conjunto com o HTML e o CSS, sua sintaxe se parece muito com a

linguagem C, permitindo a interação das páginas com o usuário. Pode ser usada do lado do cliente, mas também do lado do servidor.

Segundo o autor Prescotti (2016), seu nome não indica uma extensão do Java, que é linguagem de programação muito mais complexa. Logo, seu nome apenas vem de um momento em que a linguagem Java estava em seu auge, então resolveram renomeá-la, já que antes se chamava LiveScript.

3.2.5 METASPLOITABLE

Metasploitable é uma maquina virtual Linux que busca facilitar os testes de segurança, nele, consta diversos *softwares*, com diferentes vulnerabilidades, para que possam ser estudadas.

Segundo o autor Cordeiro et al.(2020), ele é usado para treinamentos de segurança, testar ferramentas de segurança e praticar técnicas, sem que isso interfira em um ambiente real.

3.2.6 MYSQL

MySQL é um servidor de banco de dados, ele é o mais utilizado no mundo, em razão de ser multiplataformas, rápido e ter o código-fonte aberto, isso acaba chamando a atenção de muitos programadores.

A linguagem SQL, como defende o autor Gonzaga (2000), é uma linguagem padronizada que facilita o armazenamento e o acesso às informações.

Ele foi criado em 23 de maio de 1995, pela empresa MySQL AB, antigamente chamada de TeX, e em 2009 foi comprada pela Oracle. O AB do nome da companhia é acrônimo para a palavra “*aktiebolang*”, de origem sueca, significa “sociedade anônima”.

De acordo com o autor Gonçalves (2007), o MySQL foi desenvolvido quando se precisava de um sistema de banco de dados rápido e flexível. Ele acabou sendo desenvolvido com base em um sistema chamado Msql.

3.2.7 NETBEANS IDE

De início, é necessário entender o que IDE. O autor Severo (2005) explica que IDE é um ambiente integrado de desenvolvimento, ou seja, é um conjunto de funcionalidades e atalhos que ajuda o programador a desenvolver seu trabalho.

Por consequência desse ambiente ser um meio facilitador para os profissionais da área, existem diversas IDEs disponíveis, porém, o NetBeans é considerada uma das melhores na categoria de código aberto do mercado, trazendo ferramentas que ajuda, por exemplo, em como mostrar falhas na digitação, variáveis não declaradas, métodos inexistentes, entre outras.

3.2.8 PHP

O nome PHP vem de um acrônimo recursivo para "*PHP: Hypertext Preprocessor*", originalmente "*Personal Home Page Tools*", criada em 1994, por Rasmus Lerdorf, engenheiro de *software* e membro da equipe Apache.

Segundo o autor Milani (2010), a ideia inicial de Lerdorf era apenas escrever alguns *scripts* em Perl, para ter estatísticas sobre o acesso ao seu currículo, disponível *on-line*. Ao passar do tempo, o criador foi aprimorando seu código, até que resolveu escrever algo em linguagem C, que pudesse criar outras aplicações *web*, o projeto foi batizado de *PHP/FI - Personal Home Page/Forms Interpreter*. Após isso, o projeto se tornou um sucesso e, em 1997 foi lançada a versão 2.0.

O PHP cria *scripts* embutidos no HTML do servidor, sendo um módulo oficial do http Apache, ele é multiplataformas, ou seja, se manterá no seu formato original em diversos sistemas.

3.2.9 VIRTUALBOX

Antes de saber sobre o *software* Virtual Box, é necessário entender o que é virtualização de ambiente. Virtualizar um ambiente significa dar características reais a algo abstrato, simular uma realidade sem danificar ou afetar o ambiente original e real.

Logo, o VirtualBox desempenha essa tarefa, ele simula uma máquina com um sistema operacional, disco rígido, memória, *internet* e etc, é possível executar programas dentro deste ambiente.

O autor Siqueira (2013), explica que o *software* em questão é multiplataformas e de código aberto, fornece diversas ferramentas, como, multiprocessamento (capacidade de criar máquinas virtuais com diversos processadores), é possível usar USBs conectados na máquina real na virtual, várias resoluções de tela independente da máquina real e outros.

3.3 DESENVOLVIMENTO PRÁTICO

Dando início ao desenvolvimento prático do trabalho, a primeira parte, foi a configuração da maquina virtual Metasploitable no VirtualBox.

Sua utilidade, para este projeto, consiste em abrigar a aplicação vulnerável DVWA, citado nos materiais e métodos. Para acessá-lo, é necessário iniciá-lo no VirtualBox, entrar com o *login* e senha informados por ele, em sequência, o IP da máquina deve ser colocado na URL do navegador, fazendo isso, irá aparecer algumas pastas, dentre elas, o DVWA, agora basta entrar na aplicação, que também apresenta informações de acesso (usuário e senha) na tela.

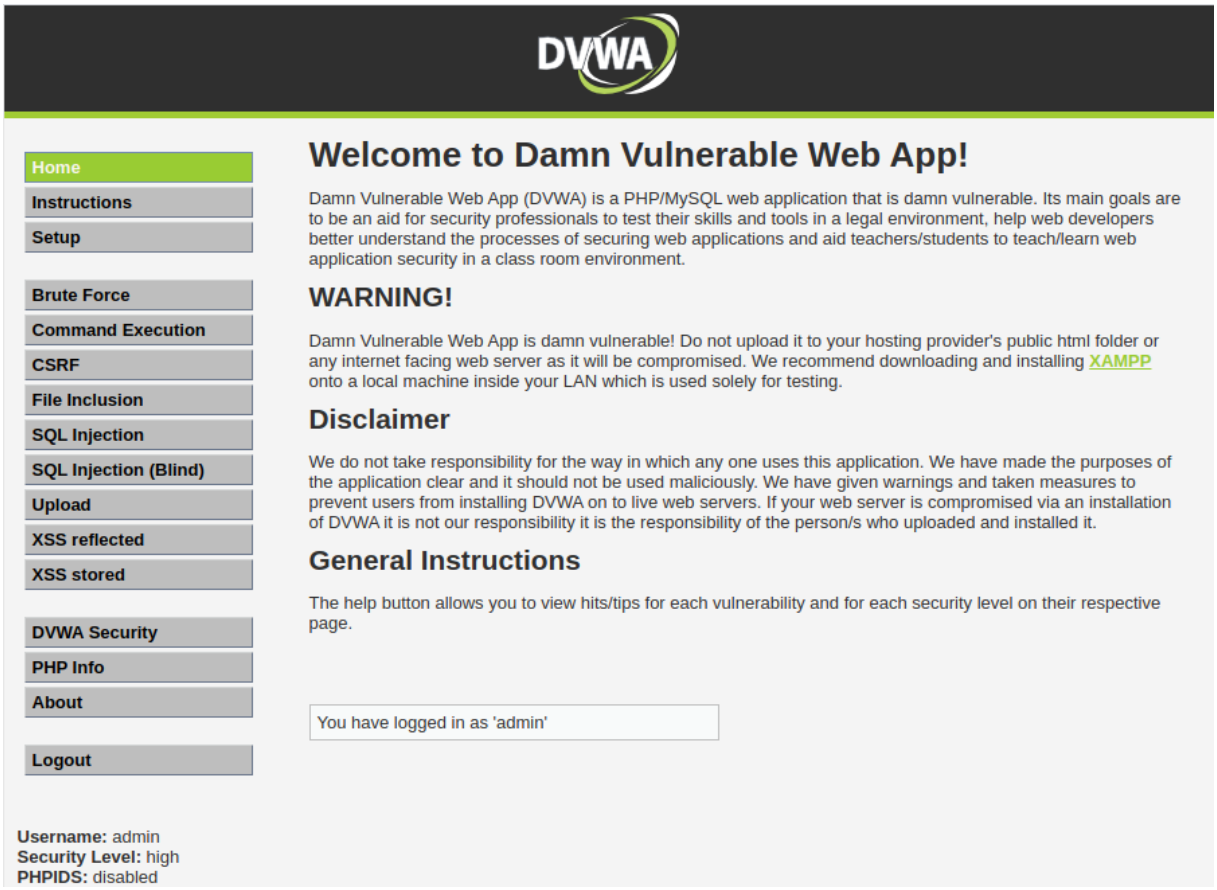
Acessando o DVWA, tem-se acesso ao *menu*, com várias vulnerabilidades, de acordo com a Figura 1, algumas delas serão testadas e tratadas no sistema protegido.

Em sequência, na segunda parte do projeto, foi desenvolvido uma aplicação própria, um site simples com cadastros, com foco em tratar os campos do *frontend*, como limitar a quantidade de caracteres inseridos.

O DVWA então, vai ser usado para mostrar o que acontece quando um campo não é tratado, em um sistema vulnerável.

Com isso em mente, no tópico abaixo será explicado as vulnerabilidades tratadas pelas boas práticas.

Figura 1: Menu DVWA



Fonte: Figura própria(2020).

3.3.1 VULNERABILIDADES TRATADAS

Para entender melhor o que foi feito, abaixo será dado uma breve explicação sobre as vulnerabilidades e a mitigação feitas no frontend.

3.3.1.1 BRUTE FORCE

O ataque de *Brute Force*, que traduzido significa Força Bruta, é usado para forçar a entrada de alguém não autorizado em um sistema. Isso acontece através de uma seleção de senhas comuns e fracas, usadas pela maioria dos usuários e testadas uma por uma, em sequência, até que se encontre uma que é validada pelo sistema.

Esse ataque pode ser mitigado apenas definindo limites na criação da senha, por meio de algumas *tags* do HTML, como exemplificado no Quadro 1. Logo, antes

que usuário possa criar uma senha fraca, o sistema o forçará a criar uma senha segura.

Quadro 1: Controle de Senha

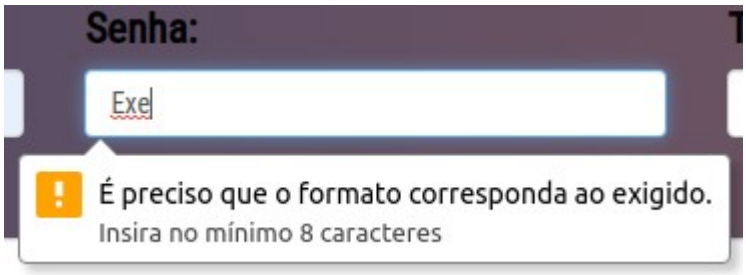
```
<input type="text" name="senha" class="form-control"
  placeholder="Letra maiúscula, minúscula e números"
  pattern="^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])[a-zA-Z0-9]{8,}$"
  title="Insira no mínimo 8 caracteres"
  minlength="8" required />
```

Fonte: Quadro próprio (2020).

A *tag pattern*, ou padrão, padroniza o tipo de inserção no campo referente. Neste caso, o padrão escolhido para a senha, precisa ter letras maiúsculas ([A-Z]), letras minúsculas ([a-z]), números ([0-9]) e ter no mínimo 8 caracteres ({8,}).

A *tag required*, ou requerido, torna o campo obrigatório, ou seja, não será possível que o usuário grave no sistema se o formulário não estiver preenchido ou dentro das especificações, pode-se ver na Figura 2.

Figura 2: Controle de Senha



Fonte: Figura própria(2020).

3.3.1.2 COMMAND EXECUTION

O ataque Command Execution, tem como objetivo, executar comandos no sistema operacional através de uma aplicação vulnerável. Normalmente, esses comandos são executados por meio dos privilégios concedidos pelo sistema.

Uma maneira de evitar esse ataque, é impedir a inserção de caracteres especiais em qualquer campo, e utilizando um filtro do PHP chamado *filter_input,*

que recebe a variável externa e a filtra, com base nos parâmetros escolhidos. Logo, para este caso foi escolhido o filtro *FILTER_SANITIZE_STRING*, que filtra caracteres especiais do campo, como se pode notar no Quadro 2.

Quadro 2: Filtragem de Campo

```
if (isset($_POST['nome'])) {
  $nome = filter_input(INPUT_POST, 'nome', FILTER_SANITIZE_STRING);
}
```

Fonte: Quadro próprio (2020),

3.3.1.3 CSRF

O ataque CSRF *(Cross-site request forgery)*, ou traduzido, Falsificação de Solicitação Entre Sites, foca em capturar uma sessão de um usuário autorizado, enquanto ela estiver ativa. Ou seja, o atacante precisa que o usuário informe seus dados antes que a sua sessão seja apagada ou desligada, pois ele se passará por esse usuário.

O ataque ocorre por links externos, ou até mesmo pelo link ‘Lembrar-me’ que é muito comum nas aplicações, pois é um conforto amais para o usuário. Além deles, é possível utilizar de uma autenticação por *cookie*, caso ele não seja expirado.

Cookie é uma ferramenta dos navegadores para guardar informações sobre os sites que os usuários acessam, como acessam e o que informam neles. Dito isso, algumas maneiras de impedir esse ataque, é definindo tempo de sessão para os usuários logados, evitando autenticações por links externos, seja em e-mails ou uma outra aba do site, pois pode ser falsificada.

Evitar passar informações via GET (método HTML), como mostra o Quadro 2, impedir que o navegador salve informação de acesso de maneira automática e definir um tempo limite de sessão entre os usuários, são algumas maneiras de evitar o ataque.

Quadro 3: Método POST.

```
if (isset($_POST['nome'])) {
```

Fonte: Quadro próprio (2020).

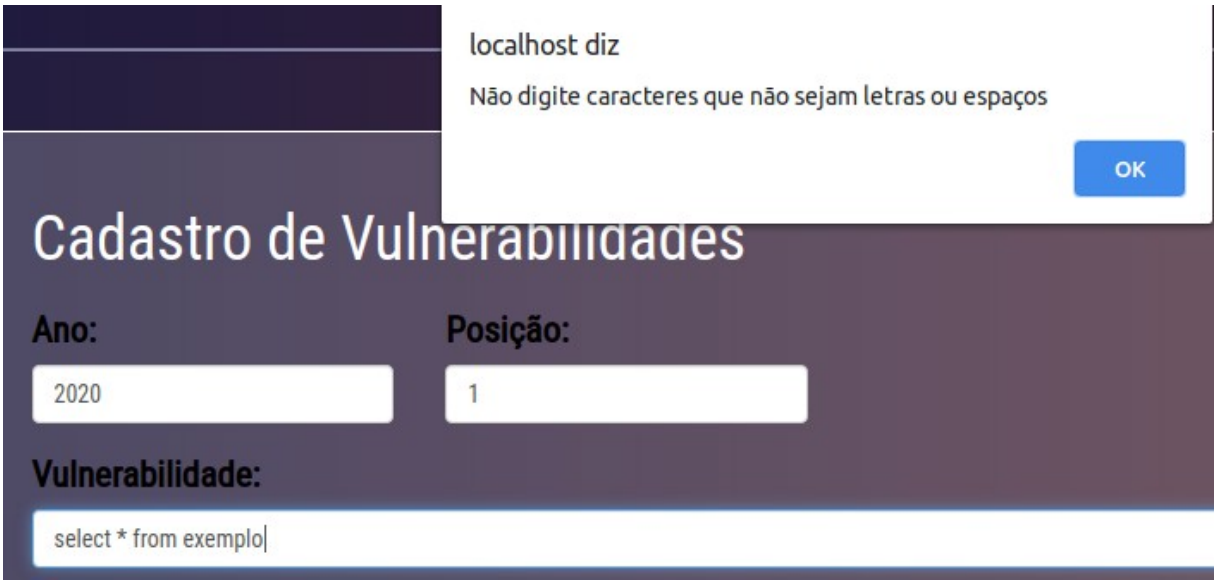
3.3.1.4 SQL INJECTION

O ataque de SQL Injection, ou injeção de SQL, consiste em inserir códigos SQL nos campos. Se o usuário malicioso colocar ‘1=1’, por exemplo, pode trazer todos os registros da tabela, pois é uma informação verdadeira, e também apagá-los.

Uma forma bem simples e comum de evitar o ataque é realizar a validação dos campos, como mostra Figura 3. Quando se bloqueia caracteres especiais, casos como ‘1=1’, não poderão acontecer, pois, o ‘=’, será bloqueado.

Além disso, utilizar já comandos preparados, pois eles limitam onde essa informação vai ser interpretada e a gravam como um dado qualquer, como mostra o Quadro 3. Não só isso, mas também evitar informar mais do que o necessário para o usuário, em razão de evitar o ataque *SQL Injection (Blind).*

Figura 3: Controle de Campo



Fonte: Figura própria (2020).

Quadro 4: Controle de Campo.

```
$sql = "INSERT INTO Cadastro_usuario (nome, usuario, senha, tipo)
  values(?,?,?,?);";

   $statement = $conexao->prepare($sql);

   $statement->bindParam(1, $nome);

   $statement->bindParam(2, $usuario);

   $statement->bindParam(3, $senha);

   $statement->bindParam(4, $tipo);

   $statement->execute();
```

Fonte: Quadro próprio (2020).

3.3.1.5 XSS REFLECTED

O ataque XSS Reflected (Cross-Site Scripting Reflected), visa inserir códigos maliciosos nas aplicações. Esses códigos são lidos normalmente pelo navegador, pois ele confia na aplicação, logo entende ser algo comum.

O ataque pode acontecer da seguinte forma, usuário malicioso insere um código de JavaScript que simula um *login,* em uma conversa de um fórum, o usuário atacado irá preencher esses campos, pois, não tem conhecimento do ataque. Isso é o suficiente para que o *hacker* tenha informações importantes.

Pode parecer algo simples e nem tão perigoso, mas essa prática pode até mesmo sequestrar a sessão da vítima, se tornando um ataque CSRF, que já foi discutido.

Um meio de se proteger é filtrar toda e qualquer entrada do usuário na aplicação, utilizar medidas que desconsidere códigos (caracteres especiais), como foi demonstrado anteriormente nas Figuras 3 e 2 e nos Quadros 1 e 2

3.4 APRESENTAR OS RESULTADOS

Depois de implementar as boas práticas no sistema, foi possível notar que um sistema que delimita a liberdade do usuário, passa mais segurança de uso quanto para quem programa, quanto para quem utiliza.

Tratar apenas os campos, pode parecer simples, no início. Contudo, com o decorrer do desenvolvimento, conclui-se que, este é um passo necessário contra os ataques. As boas práticas, quando implementadas, trabalham em conjunto com as mitigações do *backend*, gerando um sistema de segurança robusto.

Além de que, apenas tratando os campos, diversas vulnerabilidades já foram parcialmente tratadas, como *SQL Injection, Command Execution* e todas as outras já citadas.

4 TRABALHOS FUTUROS

Algumas das ideias para os trabalhos futuros são:

- Tratar as vulnerabilidades mais complexas, que vão além do *frontend*;
- Construir uma aplicação mais complexa, com diversidade de campos para testar os tratamentos;
- Utilizar outras ferramentas além das apresentadas, para que a aplicação seja segura de diferentes formas.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi abordado temas como, segurança em aplicações web, como filtrar a entrada do usuário, de forma simples e prática, e conclui-se que, é possível tratar o *frontend* de uma aplicação com poucas ferramentas, o suficiente para barrar diversos ataques.

Este trabalho foi muito importante para o meu aprofundamento pessoal no tema, uma vez que abordou assuntos antes desconhecidos por mim, o que me forçou a pesquisar sobre o assunto.

Além de me dar um conhecimento a mais sobre a área tive a oportunidade de entender a complexidade e a seriedade do profissional que trabalha com segurança da informação. Em resumo, ele me deu a oportunidade de crescer e gostar mais ainda do tema.

Também tive a oportunidade de ler diversos artigos, livros e teses, que me engajaram e foram referência neste trabalho. Em suma, a maioria dos objetivos, foram concluídos.

REFERÊNCIAS

CARVALHO, Fernanda Ramos et al.. Vulnerabilidades em aplicações web. Revista Eletrônica Científica de Ciência da Computação , [s.i.], v. 8, n. 1, p.1-1, jul. 2013. Disponível em: http://revistas.unifenas.br/index.php/RE3C/article/view/60. Acesso em: 20 mar. 2020.

CERON, J. M. et al. Vulnerabilidades em Aplicações Web: uma Análise Baseada nos Dados Coletados em Honeypots. p. 2, [s.d.].

CONTRIBUTORS, M. O., Jacob Thornton, and Bootstrap. Bootstrap. Disponível em: https://getbootstrap.com/. Acesso em: 12 ago. 2020.

CORDEIRO, Fabrício Alteff et al. Aplicação de Técnicas de Ethical Hacking-Demonstração do Uso de Ferramentas e Ambiente de Estudo para Acadêmicos ou Iniciantes em Segurança Web. 2020.

COSTA, P. V. et al. Nível de conhecimento de desenvolvedores sobre segurança em aplicações web: Pesquisa e análise. Anais da Escola Regional de Sistemas de Informação do Rio de Janeiro (ERSI-RJ), p. 92–99, 16 out. 2018.

DANTAS, Marcus Leal. Segurança da informação: uma abordagem focada em gestão de riscos. Olinda: Livro Rápido, p. 5-13, 2011.

DVWA - aplicativo da Web vulnerável maldito. Disponível em: http://www.dvwa.co.uk/. Acesso em: 12 ago. 2020.

FONTES, Edison Luiz Gonçalves. Segurança da informação. Saraiva Educação SA, 2017.

GONÇALVES, Edson. Desenvolvendo Aplicações Web com JSP Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax. Rio de Janeiro: Editora Ciência Moderna, 2007.

GONZAGA, Flávio S.; BIRCKAN, Guilherme. Curso de PHP e MySQL. Florianópolis, outubro, 2000.

LÜDERS, Edson; BIANCHINI, David Cidades Inteligentes: Segurança em aplicações WEB em Prefeituras. Aplicações PHP e Cobit 5. p. 4, 2017.

MACHADO, Rodrigo; KREUTZ, Diego; PAZ, Giulliano; RODRIGUES, Gustavo. Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de

Proteção de Dados. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES (ERRC), 17. , 2019 Anais da XVII Escola Regional de Redes de Computadores. Porto Alegre: Sociedade Brasileira de Computação, jan. 2020 . p. 154-159.

MARCELO, Antonio. APACHE: Configurando o servidor WEB para Linux. Brasport, 2005.

MILANI, André. Construindo Aplicações Web com PHP e MySQL. São Paulo: Novatec Editora, 2010.

MONTANHEIRO, L. S.; CARVALHO, A. M. M. Primeiros passos para o Desenvolvimento Seguro de Aplicações Web. . In: ANAIS ESTENDIDOS DO XVIII SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS. SBC, 25 out. 2018. Disponível em: https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/4162. Acesso em: 2 abr. 2020.

MONTEVERDE, W. A. Estudo e Análise de Vulnerabilidades Web. p. 82, 2014.

OWASP. Quem é a fundação OWASP? Disponível em: https://owasp.org/. Acesso em: 20 mar. 2020.

PRESCOTT, Preston. Programação em JavaScript. Babelcube Inc., 2016.

SEVERO, Carlos Emilio Padilla. NetBeans IDE 4.1: para desenvolvedores que utilizam a tecnologia Java. Rio de Janeiro: Brasport, 2005.

SIQUEIRA, Luciano Antonio. Máquinas virtuais com VirtualBox. Linux New Media do Brasil E, 2013.

SOUZA, L. L. D. Desenvolvimento Seguro de Aplicações Web Seguindo a Metodologia OWASP. p. 71, 2012.

TAHA, A. M. da C. Guia de testes de segurança para aplicações web. 2017.