CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
( ) PRÉ-PROJETO (X) PROJETO	ANO/SEMESTRE:2020/2

WEBGOAT PLUS: UMA EXTENSÃO DA FERRAMENTA WEBGOAT PARA O ENSINO DE VULNERABILIDADES DE SEGURANÇA

Artur Ricardo Bizon

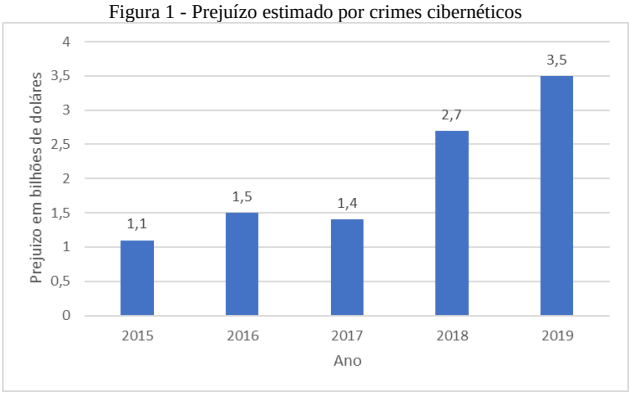
Prof. Gilvan Justino – Orientador

1 INTRODUÇÃO

A segurança da informação é um tema que está recebendo atenção nos últimos anos visto que o aumento de prejuízos causados por crimes cibernéticos vem aumentando consideravelmente, como é possível observar no gráfico da Figura 1. Conforme pode ser visto, os prejuízos mais que triplicaram entre os anos de 2015 e 2019. Um crime cibernético pode ocorrer, por exemplo, quando uma vulnerabilidade de segurança é explorada por um agente malicioso. Essas vulnerabilidades normalmente surgem a partir de bugs nos códigos escritos pelos desenvolvedores do sistema (ROWE; LUNT; EKSTROM, 2011).

Comentado [LPdAK1]: Especificar aonde

Figura 1 - Prejuízo estimado por crimes cibernéticos



Fonte: Federal Bureau of Investigation (2019)

Para a prevenção de possíveis prejuízos causados por essas falhas, podem ser utilizadas ferramentas que realizam testes automatizados de segurança ou pode-se contratar um profissional para realizar testes manuais, conhecido como hacker ético ou *pentester*. As duas abordagens são aplicadas com o intuito de descobrir vulnerabilidades antes que elas sejam exploradas de forma a causar prejuízos (XIE; LIPFORD; CHU, 2011; STEFINKO; PISKOZUB; BANAKH, 2016).

A utilização de ferramentas de testes automatizados de segurança, buscam analisar o código fonte do software para identificar possíveis falhas ~~no código fonte~~nele (HALDAR; CHANDRA; FRANZ, 2005), tal como a falta de validação de campos de entrada, cujo valor submetido pelo usuário pode interferir no comportamento da aplicação. Quando uma falha é encontrada, a ferramenta indica a região do código fonte que pode estar vulnerável e que deve ser ajustada.

Comentado [LPdAK2]: Padronizar termo: sofrware x aplicação x programa x etc

~~A o~~Outra alternativa é a contratação de um profissional que ~~realiza~~ realize testes de invasão. Este profissional utiliza das mesmas técnicas que um *hacker* utiliza para invasão de sistemas. Ao final do trabalho, o profissional fornece um relatório explicando como foi realizado o ataque e quais vulnerabilidades foram encontradas neste percurso. Se comparado com a utilização de ferramentas automatizadas, a atuação deste especialista tende a ser mais lenta e cara para as empresas. Entretanto existe um ganho na identificação de vulnerabilidades ao se contratar um *pentester* (STEFINKO; PISKOZUB; BANAKH, 2016). Como apresentado no trabalho de Amankwah, Chen e Kudjo (2020), as ferramentas automatizadas não são capazes de identificar a grande variedade de vulnerabilidades existentes.

Apesar destas estratégias auxiliarem na descoberta de falhas de segurança, elas não ajudam a diminuir a criação de novos bugs pelos desenvolvedores, ou seja, uma ferramenta ou um especialista descobrirá a falha e os desenvolvedores irão corrigi-la. Este processo não permite o refinamento técnico dos desenvolvedores para que eles parem ou diminuam a criação de bugs que podem vir a se tornar falhas de segurança (XIE; LIPFORD; CHU, 2011). Para que os desenvolvedores escrevam códigos mais seguros é preciso estudar as vulnerabilidades de software, conhecer como elas são exploradas por malfeitores e aprender a mitigá-las. Segundo o trabalho de

Xie, Lipford e Chu (2011), várias vulnerabilidades de segurança podem ser evitadas com conhecimentos e a utilização de práticas simples de desenvolvimento de software seguro.

Com a premissa de educar os desenvolvedores de software sobre vulnerabilidades de segurança, ferramentas como a WebGoat (OPEN WEB APPLICATION SECURITY PROJECT, 2020) foram desenvolvidas. Esta ferramenta apresenta ao usuário como determinada vulnerabilidade ocorre, como é explorada e como um desenvolvedor deve programar o sistema para que ele não fique vulnerável a esta falha. Entretanto, ainda existem vulnerabilidades de segurança que não são exploradas no WebGoat. Uma delas, é a vulnerabilidade conhecida como OS Injection, que está no *ranking* das 25 vulnerabilidades mais perigosas, segundo a Common Weakness Enumeration – CWE (COMMON WEAKNESS ENUMERATION, 2020).

Diante deste cenário, propõe-se incorporar à ferramenta WebGoat uma extensão para que estudantes possam se aprofundar no conhecimento da vulnerabilidade OS Injection. A proposta é seguir o modelo já definido pelo software, que conceitua a vulnerabilidade, explicando o impacto na segurança da informação e conduzindo a jornada do estudante na execução e validação de exercícios.

1.1 OBJETIVOS

O objetivo geral deste trabalho é disponibilizar uma extensão à ferramenta WebGoat para que seja possível ao estudante compreender a vulnerabilidade OS Injection.

São objetivos específicos deste trabalho:

a) disponibilizar um tutorial teórico que introduza os conceitos da vulnerabilidade OS Injection através da ferramenta WebGoat;

b) disponibilizar exercícios práticos para demonstrar a vulnerabilidade OS Injection;

c) avaliar o software desenvolvido com uma turma da disciplina de Desenvolvimento de Software Seguro;

d) ensinar conceitos teóricos e práticos sobre a vulnerabilidade OS Injection através da ferramenta WebGoat.

2 TRABALHOS CORRELATOS

Na presente sessão, são apresentados alguns trabalhos correlatos ao tema de estudo proposto. O primeiro é a aplicação web Hacksplaining (NETSPARKER, 2020), o segundo é a ferramenta Damn Vulnerable Web Application – DVWA (RANDOMSTORM, 2010), o terceiro é a aplicação web TryHackMe (TRYHACKME, 2020) e o quarto é a ferramenta CyExec (MAKI *et al*., 2020).

2.1 HACKSPLAINING

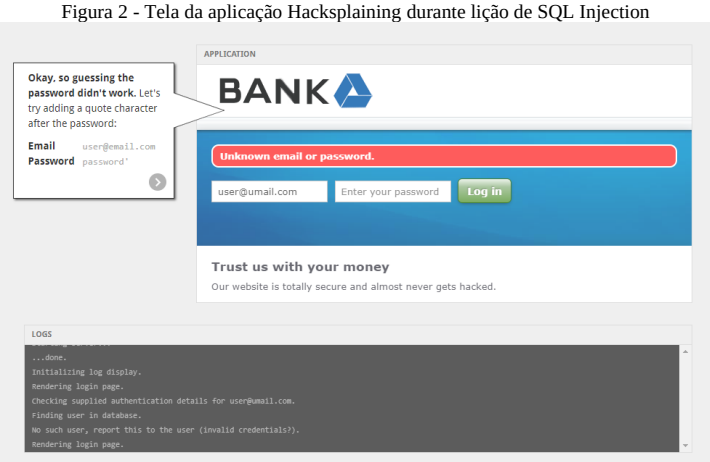
A aplicação Hacksplaining é uma aplicação web que tem por objetivo introduzir conceitos de exploração de vulnerabilidades para programadores (NETSPARKER, 2020). A aplicação possui diversas lições, que iniciam com explicações básicas de como a vulnerabilidade ocorre. Após essa introdução, o usuário é convidado a seguir alguns passos para explorar a vulnerabilidade em um sistema fictício. Tendo explorado a vulnerabilidade e a consciência de como ela ocorre, o usuário recebe dicas de como mitigar o problema, além de uma explicação dos riscos que aquela vulnerabilidade pode trazer para os sistemas do mundo real.

A aplicação tem o seu foco direcionado apenas para as vulnerabilidades mais comuns que podem acontecer em sistemas reais (NETSPARKER, 2020). Seu público-alvo são programadores que buscam o conhecimento básico sobre vulnerabilidades, sendo também utilizada por empresas que buscam capacitar seus programadores. Apesar de ser uma aplicação com fins comerciais, ela é disponibilizada de forma gratuita para qualquer pessoa utilizar a aplicação. Cobra-se uma assinatura anual apenas para empresas que desejam o acesso a mais de 5 contas distintas (NETSPARKER, 2020).

Comentado [LPdAK3]: Manter a referência. A referência é da frase e não do parágrafo!

Na Figura 2 é apresentada a tela da aplicação em uso. No centro da imagem pode ser observada a tela de uma aplicação fictícia ~~onde~~ em que o usuário realiza a lição. Neste caso, trata-se de uma aplicação que simula a tela de login de uma instituição financeira fictícia. No balão do lado esquerdo da imagem são apresentados os passos que o usuário deve fazer para concluir a lição. Por fim, na parte inferior da imagem são apresentados os logs da aplicação fictícia, para que o usuário tenha conhecimento de como o sistema está se comportando conforme as ações que ele realiza no sistema.

Figura 2 - Tela da aplicação Hacksplaining durante lição de SQL Injection



Fonte: Netsparker (2020)

2.2 DAMN VULNERABLE WEB APPLICATION – DVWA

A Damn Vulnerable Web Application (DVWA) é uma aplicação web de código aberto destinada ao ensino e estudo de vulnerabilidades web (RANDOMSTORM, 2010). Nesta aplicação, as lições são divididas em subprojetos, em que cada um possui uma vulnerabilidade distinta a ser explorada. Para cada projeto, são fornecidos links externos (para outros websites) que explicam o funcionamento de cada vulnerabilidade, além da possibilidade de selecionar a dificuldade da lição e visualizar o código fonte da lição que ~~está sendo~~é estudada.

Para que o usuário possa utilizar o DVWA é necessário que seja feito o seu download ~~da aplicação~~ e faça as ~~a execução de uma etapa de configuração~~ configurações com um banco de dados para o seu correto funcionamento (RANDOMSTORM, 2010). Com a aplicação instalada, pode-se utiliza-la localmente. ~~Após a instalação da aplicação, basta utilizá-la de forma local.~~ Vale ressaltar que apesar da aplicação ser executada de forma local ainda se faz ~~necessário~~ necessária a conexão com a internet para que o usuário possa acessar o conteúdo fornecido pelos links sugeridos pela aplicação.

Na Figura 3 é apresentada a tela da aplicação durante a lição de SQL Injection. Nesta figura é possível observar um campo de texto no qual o usuário deve submeter ao sistema dados que possam explorar a vulnerabilidade SQL Injection. No menu do lado esquerdo da figura podem ser observadas as lições disponíveis na aplicação, bem como alguns menus de configuração. Por exemplo, o último item do menu é destinado para a mudança da dificuldade da lição. ~~Por fim~~Ainda, logo abaixo do campo de entrada a aplicação apresenta links de documentos externos sugeridos para que o usuário adquira o conhecimento necessário para concluir a lição.

Comentado [LPdAK4]: Que eu saiba, deve haver borda nas imagens.

Figura 3 - Tela da aplicação DVWWA durante lição de SQL Injection

Fonte: Randomstorm (2010)

Cada lição no DVWA é apresentada como um sistema vulnerável. Desta forma, a cada lição o usuário pode escolher o nível de segurança da aplicação~~,~~ e, consequentemente, definir o grau de dificuldade para explorar a vulnerabilidade. Os níveis são divididos em três categorias: “baixo”, “médio” e “alto”. No nível baixo, a aplicação fica totalmente vulnerável. Neste nível a aplicação utiliza más práticas de segurança, sendo indicado para iniciantes, pois são ensinados os conceitos básicos de exploração da vulnerabilidade. No nível médio, a principal intenção é mostrar o código de uma aplicação insegura, para que os usuários melhorarem as suas habilidades em identificar e explorar vulnerabilidades de segurança. Por fim, o nível alto~~,~~ é o modo em que a aplicação serve como exemplo de boas práticas de segurança, pois é uma aplicação segura de qualquer vulnerabilidade. Logo, o seu propósito é o de apresentar boas práticas de codificação de sistemas seguros (RANDOMSTORM, 2010).

O DVWA também fornece uma funcionalidade que permite a comparação entre os códigos fontes de cada nível. Nesta função o usuário pode por exemplo, comparar o código fonte do nível baixo com o nível médio~~,~~ . ~~desta~~ Desta forma, ao comparar códigos de níveis distintos suas características são evidenciadas, facilitando a identificação de boas práticas que podem ser utilizadas em sistemas reais. Também podem se destacar os códigos que permitem que determinada vulnerabilidade ocorra, logo esse é um recurso que permite ao usuário identificar falhas de segurança em seu código fonte~~,~~ e corrigi-las através de boas práticas apresentadas por sistemas com um nível superior de segurança (RANDOMSTORM, 2010).

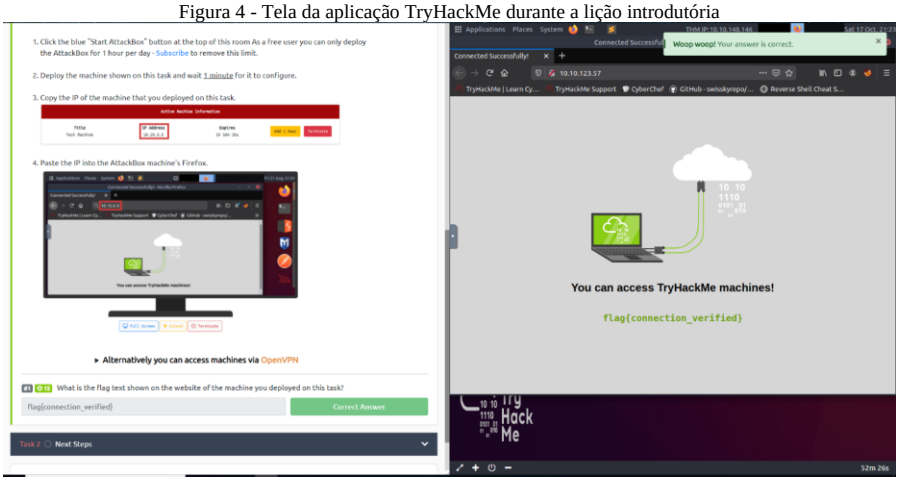
2.3 TRYHACKME

A aplicação TryHackMe é uma aplicação web comercial, entretanto ela possui planos gratuitos. Esta aplicação é destinada ao ensino de exploração de vulnerabilidades, logo seu público-alvo são pessoas que desejam aprender ou aprimorar suas técnicas de exploração de vulnerabilidades. A aplicação possui dois tipos distintos de módulos. O primeiro módulo é destinado ao ensino. Nele são apresentadas algumas vulnerabilidades e os passos que devem ser seguidos para que essas vulnerabilidades sejam exploradas. ~~No~~ O segundo módulo é destinado apenas a prática. Neste módulo são fornecidos desafios no estilo *Capture The Flag* (CTF). Um desafio CTF consiste em um invasor capturar um dado ou arquivo (*flag*) que sinalize que a sua invasão foi bem sucedida. Estes desafios estão separados em salas. Cada sala possui uma aplicação diferente da outra e o objetivo é invadir o sistema dessas salas e encontrar a *flag* (TRYHACKME, 2020).

Na Figura 4 pode ser observada a tela da aplicação TryHackMe. No lado esquerdo da figura é apresentada a lição, bem como as tarefas e os passos a serem seguidos para concluir a lição. No lado direito da figura é apresentada a tela de uma máquina virtual Kali Linux, na qual o usuário pode utilizá-la para explorar as vulnerabilidades a fim de concluir as lições ou desafios. Para utilização do TryHackMe existem dois planos, um gratuito e outro pago. O plano gratuito dá acesso à máquina virtual apenas durante uma hora por dia. A aplicação fornece conexão direta aos ~~seus~~ exercícios através de uma conexão por VPN, permitindo que os usuários possam

resolver as lições sem a necessidade de utilizar da máquina virtual destinada a ataques fornecida pela aplicação (TRYHACKME, 2020).

Figura 4 - Tela da aplicação TryHackMe durante a lição introdutória



Fonte: TryHackMe (2020)

Para utilizar essa aplicação não é necessária a instalação de nenhum software específico. ~~Porém~~ Contudo, caso o usuário queira utilizar seu próprio computador para realizar as lições é necessário que as ferramentas exigidas para concluir determinada lição sejam instaladas no computador do usuário. Quando o usuário seleciona uma lição para estudar, a aplicação inicializa um ambiente virtual que contém a lição. Após a inicialização completa, a aplicação fornece o IP da máquina virtual que contém a lição. Com isso, o usuário pode se conectar com este IP através da VPN ou pela máquina virtual de ataque. Caso o usuário queira utilizar a máquina virtual de ataques, a aplicação instancia uma máquina virtual com o sistema operacional Kali Linux. Nesta máquina já estão instaladas diversas ferramentas para *pentest* como por exemplo, o software Nmap, Metasploit entre outros. Com ~~O usuário tendo~~ o ambiente preparado para iniciar a lição, ~~o seu~~ objetivo do usuário passa a ser encontrar a *flag* da lição e fornecê-la no campo de resposta como é apresentado na Figura 4~~,~~ no lado esquerdo. Ali é possível visualizar o campo que deve ser fornecido a *flag* da lição~~.~~, que ~~Que n~~neste exemplo é o texto localizado na direita da tela que corresponde ao seguinte texto "flag{connection_verified}". Assim que o usuário informa o texto da *flag* no campo de entrada, é verificado se a resposta está correta e eventualmente ~~será~~ exibido uma mensagem de lição concluída.

2.4 CYEXEC

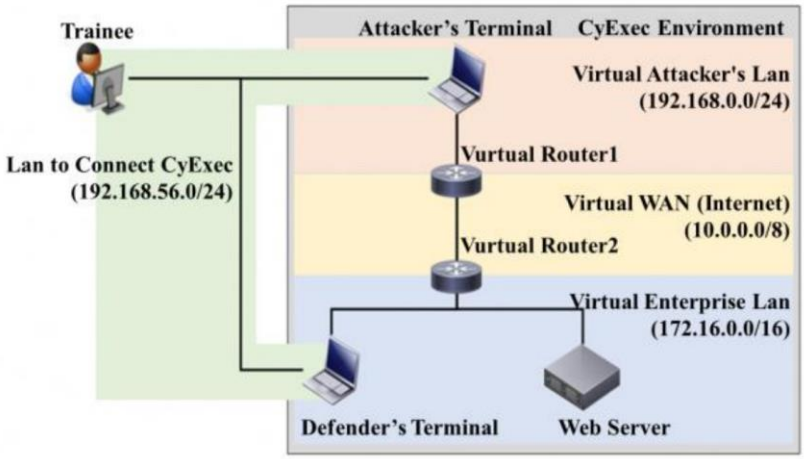
A ferramenta CyExec é uma ferramenta baseada na WebGoat, entretanto, o seu foco não é apenas no ensino de identificação e exploração de vulnerabilidades, mas também possui lições específicas de como construir uma defesa para um sistema real (MAKI *et al*., 2020). O fluxo desta ferramenta inicia com os exercícios já existentes no WebGoat. Essas lições são tratadas como introdutórias. Após esta etapa, os exercícios desenvolvidos no trabalho de Maki *et al*. (2020) são utilizados. Nestes exercícios, eles possuem tarefas específicas dependendo do "papel" que o aluno estiver atuando, que pode ser como atacante ou defensor.

Se o aluno utilizar o papel de atacante, terá que seguir lições como: executar ferramentas de *scan* para encontrar vulnerabilidades, explorar as vulnerabilidades até conseguir o controle da máquina alvo. Já o aluno com o papel de defensor deve aplicar técnicas de defesa para impedir o atacante. Dentre essas técnicas está a análise de *logs*, correção de vulnerabilidades que são exploradas e a implementação de um Web Application Firewall para conter o vazamento de informações da ferramenta de exercícios (MAKI *et al*., 2020).

A aplicação consiste em um ambiente virtual Docker que simula toda uma estrutura de rede, ~~onde~~ em que o aluno pode interagir através de um terminal de ataque ou terminal de defesa. Tanto o terminal do atacante quanto o terminal do defensor possuem sua própria rede local. Estas redes são conectadas a uma terceira que simula a rede global como mostrado na Figura 5. Na rede local do defensor, além do seu terminal, também está conectado um servidor web que contém a aplicação que deve ser defendida pelo aluno.

Comentado [LPdAK5]: Ferramenta ou aplicação?

Figura 5 - Estrutura da aplicação CyExec



Fonte: Maki *et al.* (2020)

3 SOFTWARE ATUAL

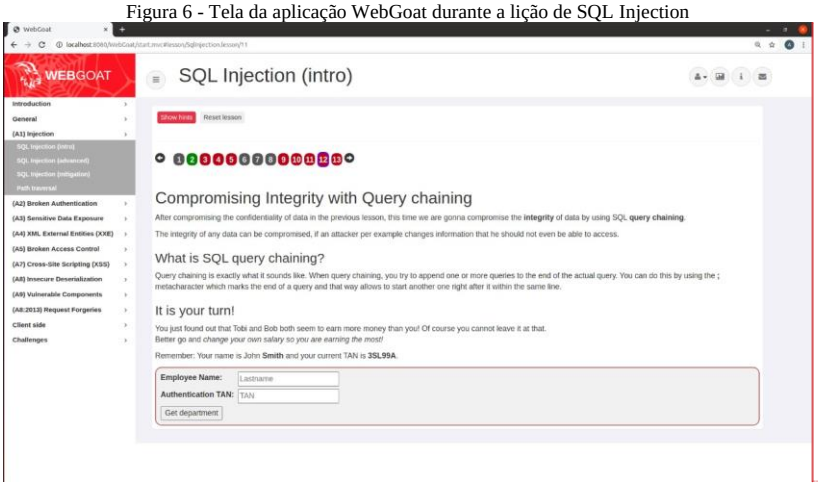
A aplicação WebGoat, que está na versão 8.1, é uma aplicação de código aberto destinada ao ensino de vulnerabilidades de segurança para programadores (OPEN WEB APPLICATION SECURITY PROJECT, 2020). ~~Sendo que todas~~Todas as vulnerabilidades ensinadas por esta ferramenta estão no *ranking* OWASP Top 10, que é uma lista que indica as vulnerabilidades de sistemas web mais comuns presentes na internet. ~~Esta~~ A WebGoat ~~aplicação~~ é dividida em dois módulos, sendo que o primeiro contém lições de vulnerabilidades e o segundo contém desafios.

No módulo de lições~~,~~ são apresentados ao usuário sistemas que propositalmente possuem falhas de segurança. Ao decorrer da lição, o usuário aprende a identificar, explorar e a mitigar a vulnerabilidade apresentada em determinada lição. Como citado anteriormente, as lições são dividias em três etapas. A primeira é uma explicação de como determinada vulnerabilidade ocorre. A segunda etapa explica como é explorada esta vulnerabilidade. Por último, é apresentado ao usuário como evitar a vulnerabilidade. Já no módulo de desafios, os desafios são feitos no estilo CTF, em que o usuário tem que utilizar os seus conhecimentos adquiridos nas lições para conseguir resolver o desafio.

Na Figura 6 é apresentada a tela do WebGoat durante a lição de SQL Injection. Nesta figura é possível observar o tutorial que explica ao usuário os conceitos teóricos, bem como aplica pequenas lições para que o usuário as execute para compreender de forma prática como é explorada a vulnerabilidade. Na parte esquerda há um menu ~~onde~~ em que é possível selecionar qual lição o usuário deseja aprender. O último item deste menu apresenta os desafios do segundo módulo da aplicação.

Comentado [LPdAK6]: Não ficou muito claro até pq é difícil de ler a figura. Sugiro escrever textualmente o que aparece na figura.

Figura 6 - Tela da aplicação WebGoat durante a lição de SQL Injection



Fonte: Open Web Application Security Project (2020)

Comentado [LPdAK7]: Tem uma borda somente no lado esquerdo.

4 PROPOSTA DA APLICAÇÃO

Na presente seção, é apresentada a justificativa do trabalho proposto, em seguida a sua metodologia de desenvolvimento e os seus requisitos.

4.1 JUSTIFICATIVA

No Quadro 1 são apresentadas as principais características dos trabalhos correlatos, cada característica pode assumir a condição de ~~atende~~ ATende (AT), ~~atende~~ Atende ~~parcialmente~~ Parcialmente (AP) e ~~não~~ Não ~~atende~~ Atende (NA). Neste quadro é possível perceber que todas as aplicações utilizam virtualização do ambiente em que são executados os exercícios, tornando as soluções mais portáveis e fáceis de replicar. Outro ponto a se atentar é a aplicação de desafios no estilo CTF.

Baseado na utilização das ferramentas correlatas pelo autor do presente projeto, foi observado na maioria das ferramentas um fluxo de ensino semelhante, ~~onde~~ de modo que as ferramentas seguem alguns passos definidos, sendo eles: ~~i~~Introduzir a parte conceitual da vulnerabilidade; aplicação prática dos conhecimentos adquiridos no passo anterior; e, ~~por fim é apresentado~~apresentações técnicas de mitigação para a vulnerabilidade ensinada na lição em questão. ~~Contudo a~~A ferramenta TryHackMe não contém a parte que explica como mitigar a vulnerabilidade, pois o foco da ferramenta é o ensino de exploração de vulnerabilidades ao contrário das demais ferramentas que tem seu foco ~~é~~de ~~o ensino~~ ensinar ~~d~~o desenvolvimento de softwares seguros.

Em relação a etapa de introduzir os conceitos de determinada vulnerabilidade, a aplicação DVWA apenas apresenta links externos com a documentação das vulnerabilidades. As demais aplicações guiam o usuário passo a passo para que seja atingido seu objetivo que é compreender e ter condições de explorar determinada vulnerabilidade de segurança. Entretanto, a aplicação DVWA se destaca na parte que ensina como mitigar determinada vulnerabilidade, pois das ferramentas analisadas neste estudo, é a única que permite o usuário visualizar e comparar o código fonte das aplicações fictícias fornecidas pela aplicação. Além de sistemas vulneráveis, o DVWA apresenta sistemas que seguem as boas práticas de segurança que por sua vez são sistemas que não possuem vulnerabilidades de segurança.

Comentado [LPdAK8]: Sugiro colocar isso daqui depois da tabela, pois não se relaciona a informações que constam na tabela.

Quadro 1 - Comparação das funcionalidades dos trabalhos correlatos~~. AT representa Atende, AP – Atende Parcialmente e NA Não Atende~~

Correlato / Funcionalidade	Hackplaining - (NETSPARKER, 2020)	DVWA - (RANDOMSTORM, 2010)	TryHackMe - (TRYHACKME, 2020)	CyExec – (MAKI *et al.*, 2020)
Descrição da vulnerabilidade	AT	AP	AT	AT
Técnicas de identificação	AT	AP	AT	AT
Técnicas de exploração	AT	AP	AT	AT
Técnicas de mitigação	AT	AP	NA	AT
Técnicas de ataque em gera	AT	AP	AT	AT
Técnicas de defesa em geral	NA	NA	NA	AT
Contém desafio estilo CTF	NA	AT	AT	AT
Execução em ambiente local	NA	AT	NA	AT
Utilização de ambiente virtual	AP	AT	AT	AT

Fonte: elaborado pelo autor

Comentado [LPdAK9]: Explicar esse quadro textualmente em parágrafo.

O Quadro 2 apresenta uma parcela das vulnerabilidades implementadas pelos trabalhos correlatos. Das vulnerabilidades apresentadas no **Erro! Fonte de referência não encontrada.**, nem todas foram implementadas por todas as ferramentas de ensino. Um exemplo é a vulnerabilidade OS Injection, que está listada na décima posição do *ranking* da CWE das 25 vulnerabilidades mais perigosas apresentado em Common Weakness Enumeration (2020). Também pode ser observado que o maior foco das ferramentas de ensino está na exemplificação das vulnerabilidades de Cross Site Scripting (XSS) e SQL Injection. Ambas as vulnerabilidades estão no ranking da Common Weakness Enumeration (2020), sendo o XSS considerado a vulnerabilidade mais perigosa do ano de 2020.

Quadro 2 - Vulnerabilidade ensinada. O "X" indica que a ferramenta contém

Correlato / Vulnerabilidade	Hackplaining - (NETSPARKER, 2020)	DVWA - (RANDOMSTORM, 2010)	TryHackMe - (TRYHACKME, 2020)	CyExec – (MAKI *et al.*, 2020)
Cross Site Scripting	X	X	X	X
Sql Injection	X	X	X	X
OS Injection	X	X	X	
Buffer Overflow	X		X	
Path Transversal	X		X	
Upload de arquivo não confiável	X	X	X	

Fonte: elaborado pelo autor

Considerando as comparações apresentadas até o momento, o presente projeto se propõe a estender a implementação da ferramenta WebGoat para acrescentar uma vulnerabilidade pouco explorada pelas ferramentas de ensino já existentes, como por exemplo, o OS Injection.

A criação deste tipo de ferramenta se torna importante para o auxílio no ensino de segurança da informação, pois o conhecimento gerado pelas técnicas de ataque pode evitar que muitas vulnerabilidades sejam exploradas de forma maliciosa. Sendo assim, o ensino de técnicas de *hacking* pode ser considerada uma peça crucial no ensino de segurança na computação (PARSHAL, 2006; TRABELSI; MCCOEY, 2016).

Trabelsi e McCoey (2016) afirmam que o ensino de técnicas de segurança ofensiva se torna uma peça fundamental para o melhor entendimento do aluno sobre o pensamento *hacker* e como as falhas de segurança acontecem. Os autores ainda frisam a importância da realização de atividades práticas para o ensino de

vulnerabilidades, pois nessas abordagens é permitido ao aluno testar técnicas utilizadas em ataques reais e em contra partida, o aluno também aprende como implementar soluções apropriadamente seguras.

Com base nos trabalhos de Trabelsi e McCoey (2016) e Parshal (2006), pode ser afirmado que o presente projeto tem relevância no sentido social, pois o objetivo do projeto é auxiliar na educação de novos alunos da disciplina de Desenvolvimento de Sistemas Seguros. Espera-se, com a conclusão deste trabalho, proporcionar maior entendimento aos alunos da disciplina de como as vulnerabilidades ocorrem e por sua vez como podem ser corrigidas.

4.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os Requisitos Funcionais (RF) e os Requisitos Não Funcionais da aplicação (RNF) são:

a) o software deve conceituar a vulnerabilidade *OS* Injection (RF);

b) o software deve conduzir o aluno na realização de exercícios (RF);

c) o software deve avaliar as respostas fornecidas pelo usuário (RF);

d) o software deve ser desenvolvido seguindo o modelo de extensão proposto pelo WebGoat (RNF);

e) o software deve ser desenvolvido em Java (RNF);

f) as lições devem estar disponíveis nos idiomas português e inglês (RNF).

4.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: pesquisar sobre a vulnerabilidade OS Injection bem como suas possíveis variações. Buscar também metodologias para o desenvolvimento de textos de caráter pedagógico;
b) análise da ferramenta atual: estudar o modelo de extensão da ferramenta Webgoat;
c) desenvolver parte teórica da lição: implementar o guia de como a vulnerabilidade pode ser explorada;
d) desenvolver parte prática da lição: desenvolver uma aplicação para testes, sendo propositalmente vulnerável a OS Injection;
e) testar a aplicação: realizar testes para validar a aplicação como um todo;
f) realizar testes com usuários: testar a aplicação com um grupo de usuários. O teste será validado a partir de questionários respondidos pelos usuários;
g) analisar as respostas dos questionários respondidos: pretende-se analisar os questionários respondidos para principalmente validar a eficiência da aplicação em ensinar conceitos sobre a vulnerabilidade OS Injection.

As etapas serão realizadas nos períodos relacionados no Quadro 3.

Quadro 3 - Cronograma

	ano									
	Fev.		mar.		abr.		mai.		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
Levantamento bibliográfico										
Análise ferramenta atual										
Desenvolver parte teórica da lição										
Desenvolver parte prática da lição										
Testar a aplicação										
Realizar testes com usuários										
Analisar questionários respondidos										

Fonte: elaborado pelo autor

5 REVISÃO BIBLIOGRÁFICA

Na presente seção, são apresentados alguns conceitos que fundamentam a produção deste trabalho, sendo eles: segurança da informação e a vulnerabilidade OS Injection.

5.1 SEGURANÇA DA INFORMAÇÃO

A segurança da informação é definida pela Associação Brasileira de Normas Técnicas. NBR ISO/IEC 27002:2005 (2005) como uma forma de garantir a integridade, confidencialidade e a disponibilidade da informação. Pode ainda contemplar outras propriedades como a autenticidade, responsabilidade, não repúdio e confiabilidade. Segundo Whitman e Mattord (2017) a confidencialidade, integridade e disponibilidade são considerados os pilares de um sistema seguro, ~~. a~~ A junção destas características também é conhecida como tríade Confidencialidade, Integridade e Disponibilidade (CIA) ou tríade da segurança da informação.

Whitman e Mattord (2017) conceituam que confidencialidade garante que apenas usuários que têm acesso ou privilégio o suficiente possam acessar alguma informação confidencial. A integridade se trata do quão íntegro é determinada informação, ou seja, deve garantir que a informação não sofra nenhum tipo de corrupção em seu conteúdo. Já a disponibilidade deve garantir que pessoas autorizadas acessem determinada informação quando precisarem sem que haja algum tipo de interrupção.

Quando um sistema sofre um ataque que prejudica ao menos uma das características da tríade da segurança da informação, pode ser afirmado que o ataque foi ~~bem sucedido~~bem-sucedido (FONTE, ANO). Um ataque na área da segurança da informação é uma ação contínua contra um sistema que pode causar a perda de alguma propriedade de segurança ao dono do sistema. Um ataque ocorre quando um agente malicioso se utiliza da exploração (*exploit*) de uma vulnerabilidade presente no sistema. Uma vulnerabilidade consiste em uma fraqueza de um sistema, como quando um campo em que seu conteúdo não é validado e seu valor interfere no comportamento da aplicação (WHITMAN; MATTORD, 2017).

Comentado [LPdAK10]: Referenciar!

5.2 OS INJECTION

OS Injection também conhecida como OS Command Injection é uma vulnerabilidade de segurança que permite que um agente malicioso execute comandos diretamente no sistema operacional. Esta vulnerabilidade tem origem na vulnerabilidade de Command Injection, ~~onde~~ em que este tipo de vulnerabilidade ocorre quando o sistema constrói um comando que é finalizado com um dado de entrada informado pelo usuário, porém caso esse dado de entrada não seja devidamente tratado, o usuário pode inserir caracteres que alteram o funcionamento esperado do comando gerado pela instrução (COMMON WEAKNESS ENUMERATION, 2020b, 2020c; HOWARD; LEBLANC; VIEGA, 2010).

No Quadro 4 é apresentado um código vulnerável à vulnerabilidade OS Injection. Neste exemplo, o programa solicita na linha 1 uma URL ao usuário para que seja verificad~~o~~a a latência da conexão com o endereço fornecido na linha 2. ~~Porém~~Contudo, como é possível perceber, não é realizada nenhuma validação sobre o dado que o usuário forneceu como entrada. Logo, o sistema permite que o usuário execute a seguinte entrada "| dir", que por sua vez internamente vai formar o seguinte comando "ping | dir". Neste exemplo, pode ser percebido que foi alterado o comportamento do comando que inicialmente era de consultar a latência, com a entrada do usuário o comando lista o diretório atual da aplicação. Isso ocorre, pois o caractere "|" funciona como um separador de comandos. Sendo assim, o sistema operacional executará dois comandos: o ping (que era a intenção do sistema) e o segundo comando com o comando que o usuário forneceu como entrada.

Quadro 4 - Exemplo de código vulnerável a OS Injection na linguagem Python

```
1 entrada = input("insira uma url")
2 os.system("ping " + entrada)
```

Fonte: Elaborado pelo autor

REFERÊNCIAS

AMANKWAH, Richard; CHEN, Jinfu; KUDJO, Patrick Kwaku; TOWEY, Dave. **An empirical comparison of commercial and open-source web vulnerability scanners**. Software: Practice and Experience, [S.L.], v. 50, n. 9, p. 1842-1857, 3 jul. 2020. Wiley. http://dx.doi.org/10.1002/spe.2870.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISSO/IEC 27002:2005: Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação. Rio de Janeiro, 2005. 120 p.

COMMON WEAKNESS ENUMERATION. **2020 CWE Top 25 Most Dangerous Software Weaknesses**. 2020a. Disponível em: https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html. Acesso em: 04 out. 2020.

COMMON WEAKNESS ENUMERATION. **CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection').** 2020b. Disponível em: https://cwe.mitre.org/data/definitions/78.html. Acesso em: 20 nov. 2020.

COMMON WEAKNESS ENUMERATION. **CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection').** 2020c. Disponível em: https://cwe.mitre.org/data/definitions/77.html. Acesso em: 20 nov. 2020.

FEDERAL BUREAU OF INVESTIGATION (FBI)**. 2019 Internet Crime Report**. 2019. Disponível em: https://pdf.ic3.gov/2019_IC3Report.pdf. Acesso em: 03 out. 2020.

HALDAR, Vivek; CHANDRA, Deepak; FRANZ, Michael. **Dynamic taint propagation for Java**. In: 21st Annual Computer Security Applications Conference (ACSAC'05). IEEE, 2005. p. 9 pp.-311.

HOWARD, Michael; LEBLANC, David; VIEGA, John. **24 DEADLY SINS OF SOFTWARE SECURITY: programming flaws and how to fix them**. New York: McGraw Hill, 2010.

MAKI, Nobuaki et al. **An Effective Cybersecurity Exercises Platform CyExec and its Training Contents**. International Journal of Information and Education Technology, v. 10, n. 3, p. 215-221, 2020.

NETSPARKER. **Hacksplaining**. Disponível em: https://www.hacksplaining.com/features. Acesso em: 04 out. 2020.

OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **OWASP WebGoat**. 2020. Disponível em: https://owasp.org/www-project-webgoat/. Acesso em: 04 out. 2020.

PARSHAL, B. A. **Teaching Students to Hack: Ethical Implications in Teaching Students to Hack at the University Level**. In: Annual Conference on Information Security Curriculum Development, 3. 2006, Kennesaw, Proceedings, Nova York: Association for Computing Machinery, 2006. p. 197–200. https://doi.org/10.1145/1231047.1231088

RANDOMSTORM. **Damn Vulnerable Web Application (DVWA)**. 2010. Disponível em: https://github.com/digininja/DVWA/blob/master/docs/DVWA_v1.3.pdf. Acesso em: 04 out. 2020.

ROWE, Dale C.; LUNT, Barry M.; EKSTROM, Joseph J. T**he role of cyber-security in information technology education**. In: Proceedings of the 2011 conference on Information technology education. 2011. p. 113-122.

STEFINKO, Yaroslav; PISKOZUB, Andrian; BANAKH, Roman. **Manual and automated penetration testing. Benefits and drawbacks. Modern tendency**. In: 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET). IEEE, 2016. p. 488-491.

TRABELSI, Zouheir; MCCOEY, Margaret. **Ethical Hacking in Information Security Curricula**. International Journal Of Information And Communication Technology Education (IJICTE), [S.L.], v. 12, n. 1, p. 1-10, jan. 2016. IGI Global. http://dx.doi.org/10.4018/ijicte.2016010101.

WHITMAN, Michael E; MATTORD, Herbert J. **Principles of Information Security**. 6. ed. Boston: Cengage Learning, 2017. 656 p.

XIE, Jing; LIPFORD, Heather Richter; CHU, Bill. **Why do programmers make security errors?**. In: 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, 2011. p. 161-164. DOI 10.1109/VLHCC.2011.6070393.

ASSINATURAS

(Atenção: todas as folhas devem estar rubricadas)

Assinatura do(a) Aluno(a): _____

Assinatura do(a) Orientador(a): _____

Assinatura do(a) Coorientador(a) (se houver): _____

Observações do orientador em relação a itens não atendidos do pré-projeto (se houver):

FORMULÁRIO DE AVALIAÇÃO – PROFESSOR AVALIADOR

Acadêmico(a): _____

Avaliador(a): Luciana Pereira de Araújo Kohler

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
	O problema está claramente formulado?	X		
	1. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	X		
	2. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?	X		
	3. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?		X	
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?	X		
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?	X		
	4. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?	X		
	5. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?	X		
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?	X		
	6. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?	X		
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?	X		
ASPECTOS METODOLÓGICOS	7. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?	X		
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		

PARECER – PROFESSOR AVALIADOR:
(PREENCHER APENAS NO PROJETO)

O projeto de TCC ser deverá ser revisado, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE.

PARECER: (X) APROVADO ( ) REPROVADO

Assinatura: Luciana P. de Araújo Kohler Data: 07/12/2020

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.