CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(x) PRÉ-PROJETO ( ) 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

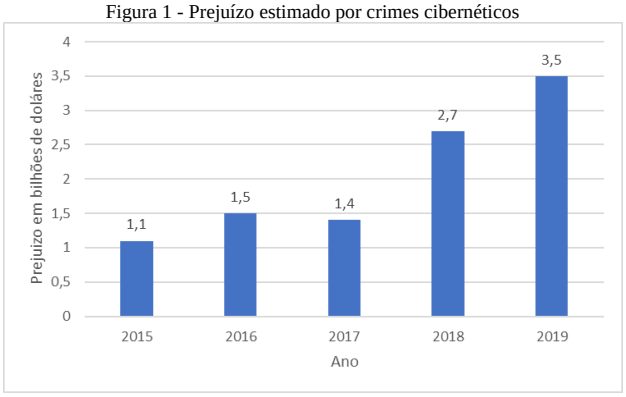
A segurança da informação é um tema que está recebendo atenção nos últimos anos visto que o aumento de prejuízos causados por crimes cibernéticos vem aumentando consideravelmente, como é possível observar no gráfico da Figura 1.

Comentado [AS1]: Não se faz parágrafo com uma única frase.

Fonte desta afirmação? O que é possível observar? Precisa descrever a imagem.

Comentado [AS2]: Coloque o recurso de referência cruzada para a figura.

Figura 1 - Prejuízo estimado por crimes cibernéticos



Fonte: Federal Bureau of Investigation (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). Para a prevenção de possíveis prejuízos causados por essas falhas, podem ser utilizadas ferramentas que realizam testes automatizados ou contratar um profissional para realizar testes manuais, conhecido como hacker ético ou *pentester*. Ambas as 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 para analisar o código fonte da aplicação tem por objetivo identificar possíveis falhas no código fonte (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. Caso alguma falha seja encontrada, a ferramenta indica a região do código fonte que pode estar vulnerável e deveria ser ajustada.

O profissional que realiza testes de invasão utiliza das mesmas técnicas que um *hacker* utiliza para invasão de sistemas (STEFINKO; PISKOZUB; BANAKH, 2016). Ao final do trabalho, o profissional fornece um relatório explicando como foi realizado o ataque e quais vulnerabilidades foram encontradas neste percurso (STEFINKO; PISKOZUB; BANAKHT, 2016). 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). Pois, como apresentado no trabalho de Amankwah, Chen e Kudjo (2020), as ferramentas automatizadas, não são capazes de identificar a grande variedade de vulnerabilidades existentes.

Comentado [AS3]: Coloca somente a referência no início da frase ou no final do parágrafo. Não precisa neste caso colocar em cada frase (referência repetida).

Apesar destas estratégias auxiliarem na descoberta de falhas de segurança, elas não ajudam a diminuir a criação de novos bugs pelos desenvolvedores (XIE; LIPFORD; CHU, 2011). Ou seja, uma ferramenta ou um especialista descobrirá a falha e os desenvolvedores irão corrigi-las. 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~~softwares, 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

Comentado [AS4]: Veja o comentário anterior.

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. Por exemplo, 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 muitas vulnerabilidades de segurança que não são exploradas no WebGoat. Uma delas, por exemplo, 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) ~~D~~disponibilizar a parte teórica que conceitua a vulnerabilidade *OS Injection*;

b) ~~D~~disponibilizar a parte prática que demonstra a vulnerabilidade *OS Injection*;

c) ~~A~~avaliar com uma turma da disciplina de Desenvolvimento de ~~software~~ Software ~~seguro~~ Seguro se este trabalho cumpriu seu papel de ensinar sobre as vulnerabilidades *OS Injection*.

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) e o terceiro é a aplicação web TryHackMe (TRYHACKME, 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 Hacksplaining 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. Isso pode ser um ponto limitante, pois a aplicação cobre apenas algumas vulnerabilidades, deixando várias outras de fora do seu catálogo.

Apesar de ser uma aplicação com fins comerciais, ela é disponibilizada de forma gratuita para qualquer pessoa que queira utilizar a aplicação. Cobra-se uma assinatura anual apenas para empresas que desejam o acesso a mais de 5 contas distintas (NETSPARKER, 2020).

Comentado [AS5]: Parágrafo muito curto.

O principal aspecto desta aplicação é a possibilidade de se explorar as vulnerabilidades em sistemas fictícios que se aproximam de aplicações reais. Isso acaba trazendo uma boa noção de como as vulnerabilidades ocorrem e são exploradas na prática.

Comentado [AS6]: Qual aspecto? Positivo? Negativo?

Outro ponto relevante é a forma com que as lições são estruturadas: cada passo é bem descrito para que qualquer programador possa entender como as vulnerabilidades ocorrem e como podem ser corrigidas caso existam em sistemas do mundo real. Uma característica desta ferramenta é que todas as aplicações fornecidas como exemplo de má implementação estão hospedadas nos servidores da Hacksplainig. Com isso, todos os recursos estão disponíveis na web sem a necessidade de efetuar o download de algo para estudar os problemas de segurança da informação.

2.2 DAMN VULNERABLE WEB APPLICATION – DVWA

A aplicação DVWA é uma aplicação web de código aberto destinada para o ensino e estudo de vulnerabilidades web (RANDOMSTORM, 2010). Nesta aplicação, as vulnerabilidades são divididas em subprojetos, onde 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.

A forma com que a vulnerabilidade é apresentada pode ser considerada um ponto fraco pois usuários menos experientes não têm qualquer outro auxílio, a partir da ferramenta, o que pode dificultar o seu aprendizado ou até mesmo frustrá-lo.

Outro ponto negativo desta aplicação é que se faz necessário o download da aplicação e a execução de uma etapa de configuração com um banco de dados para o seu correto funcionamento. Este processo de instalação e configuração acaba tornando a sua utilização menos prática, em comparação com outras ferramentas.

Um ponto forte que a aplicação DVWA possui é a possibilidade ~~do~~de o usuário escolher um nível de segurança. 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 nele é ensinado o básico da exploração da vulnerabilidade (RANDOMSTORM, 2010).

Já no nível médio, a principal intenção é mostrar o código de uma aplicação que ainda possui várias práticas insuficientes de segurança. Este nível é indicado para usuários que desejam melhorar as suas habilidades em identificar e explorar vulnerabilidades de segurança (RANDOMSTORM, 2010).

No nível alto, a aplicação serve como exemplo de boas práticas de segurança para códigos fonte pois é uma aplicação segura de qualquer vulnerabilidade. Logo, o seu propósito apenas é o de apresentar boas práticas de segurança ao usuário (RANDOMSTORM, 2010).

Por fim, o último ponto forte dessa aplicação, também está relacionado à funcionalidade de níveis, pois a aplicação também permite ao usuário comparar o código fonte aplicado em cada nível. Essa funcionalidade auxilia o usuário a identificar com mais facilidade o que é uma má prática de segurança em relação a uma boa prática, além de recomendar como pode adotar boas práticas.

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. 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. Além de um guia passo a passo a aplicação disponibiliza vídeos como exemplo da exploração da vulnerabilidade (TRYHACKME, 2020).

O segundo módulo é destinado apenas a prática, neste módulo são fornecidos desafios no estilo capture the flag (CTF). ~~u~~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 bandeira (TRYHACKME, 2020).

Ao contrário dos correlatos já apresentados, o TryHackMe é voltado para o refinamento das habilidades dos *pentesters*, logo os desafios tendem a ser mais complexos e existem mais vulnerabilidades que podem ser exploradas.

Um ponto positivo é que a utilização desta aplicação é através da web, sem ter a necessidade de fazer o download das aplicações para utilizá-las. Quando um usuário entra em uma sala, automaticamente um ambiente virtual é instanciado nos servidores da aplicação para que apenas um usuário tenha acesso para cada laboratório, sendo assim, se duas pessoas estiverem na mesma sala tentando explorar as mesmas vulnerabilidades uma não irá atrapalhar a outra.

Outro ponto forte do TryHackMe é que os próprios usuários podem criar suas próprias salas de desafios e desafiar os outros usuários da plataforma (TRYHACKME, 2020). Essa característica permite uma maior cobertura das vulnerabilidades existentes.

2.4 CYEXEC

A ferramenta CyExec é uma ferramenta baseada na ferramenta 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).

Comentado [AS7]: Não se faz parágrafo com uma única frase ou parágrafos com poucas linhas. Rever isso em toda a seção.

Comentado [AS8]: 1ª letra maiúscula

Comentado [AS9]: Não se faz parágrafo com uma única frase. Esta frase deve estar na justificativa e não nesta seção. Esta seção é somente de apresentação do correlato.

Comentado [AS10]: Frase extensa. Dividir em 2.

Comentado [AS11]: Nesta seção também tem parágrafos curtos. Rever.

Comentado [AS12]: Não se faz parágrafo com uma única frase.

Comentado [AS13]: *et al.* (em itálico e com "."). Rever todos no texto.

O fluxo desta ferramenta inicia com os exercícios já existentes no WebGoat. Essas lições são tratadas como introdutórias. Após esta etapa introdutória, 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 (MAKI et al, 2020). Já a 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 estão sendo exploradas e a implementação de um *Web Application Firewall* para conter o vazamento de informações da ferramenta de exercícios (MAKI et al, 2020).

3 SOFTWARE ATUAL

Como citado anteriormente a aplicação WebGoat é uma aplicação de código aberto destinada ao ensino de vulnerabilidades de segurança para programadores. Atualmente se encontra na versão 8.1 (OPEN WEB APPLICATION SECURITY PROJECT, 2020). A 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, onde ao decorrer da lição o usuário vai aprender 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 que explica 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, onde o usuário tem que utilizar os seus conhecimentos adquiridos nas lições para conseguir resolver o desafio.

Comentado [AS14]: Redundante

Como o foco da ferramenta é no ensino de conhecimentos básicos sobre vulnerabilidades para programadores, foram selecionadas aquelas que estão presentes no OWASP Top 10, um *ranking* que apresenta as vulnerabilidades com maior risco para aplicações web. Porém, nem todas as vulnerabilidades presentes neste *ranking* constam na aplicação do WebGoat.

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. Neste quadro é possível perceber que todas as aplicações utilizam virtualização do ambiente em que se são executados os exercícios, tornando as soluções mais portáveis e fácies de replicar.

Comentado [AS15]: Coloque o recurso de referência cruzada para o quadro.

Colocar o quadro 1 aqui

Um fator relevante é que, exceto a ferramenta TryHackMe, os outros correlatos possuem um funcionamento parecido, dividindo em etapas para apresentar a vulnerabilidade, explorá-la e explicar como mitigá-la. Porém a ferramenta TryHackMe não apresenta ao usuário como uma vulnerabilidade deve ser mitigada. O foco desta ferramenta é refinar as habilidades de ataque de um *pentester*, ao contrário das demais ferramentas que tem o objetivo de ensinar boas práticas de desenvolvimento de software seguro.

O Quadro 2 apresenta uma parcela das vulnerabilidades implementadas pelos trabalhos correlatos. Das vulnerabilidades apresentadas no Quadro 2, 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 (COMMON WEAKNESS ENUMERATION, 2020).

Pode ser observado também 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.

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 nas faculdades. 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 considerado uma peça crucial no ensino de segurança na computação (PARSHEL, 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 Parshel (2006), pode ser afirmado que o presente projeto tem relevância no sentido social, onde 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.

Quadro 1 Comparação das funcionalidades dos trabalhos correlatos. ~~Onde~~ 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

Quadro 2 Vulnerabilidade ensinada. Onde tem o "X" significa 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	
Buffer Overflow	X		X	
Path Transversal	X		X	
Upload de arquivo não confiável	X	X	X	

Fonte: elaborado pelo autor

4.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais (RF) e os requisitos não funcionais da aplicação (RNF) são:

RF1 – O ~~s~~Software deve conceituar a vulnerabilidade *OS Injection*;

RF2 – O ~~s~~Software deve conduzir o aluno na realização de exercícios;

RF3 – O ~~s~~Software deve avaliar as respostas fornecidas pelo usuário;

RFN1 – O software deve ser construído seguindo o modelo de extensão proposto pelo WebGoat;

RNF2 – O software deve ser escrito em Java;

RNF3 – As lições devem estar disponíveis nos idiomas português e inglês.

Comentado [AS16]: Faça uma lista a)... b)...

4.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) ~~Levantamento~~ levantamento bibliográfico: ~~Pesquisar~~ pesquisar sobre a vulnerabilidade *OS injection* bem como suas possíveis variações. Buscar também metodologias para o desenvolvimento de textos de caráter pedagógico;
b) ~~Análise~~ análise da ferramenta atual: ~~Estudar~~ estudar o modelo de extensão da ferramenta Webgoat;
c) ~~Desenvolver~~ desenvolvimento da parte teórica da lição: ~~Implementar~~ implementar o guia de como a vulnerabilidade pode ser explorada;
d) ~~Desenvolver~~ desenvolvimento da parte prática da lição: ~~Desenvolver~~ desenvolver uma aplicação para testes, sendo propositalmente vulnerável a *OS Injection*;
e) ~~Testar~~ testes da aplicação: ~~Realizar~~ realizar testes para validar a aplicação como um todo;
f) ~~Realizar~~ testes com usuários: ~~Testar~~ testar a aplicação com um grupo de usuários. O teste será validado a partir de questionários respondidos pelos usuários;
g) ~~Analisar~~ analise das respostas dos questionários respondidos.

As etapas serão realizadas nos períodos relacionados no Quadro 33.

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

Comentado [AS17]: início na segunda quinzena de fevereiro.

Comentado [AS18]: Arrumar de acordo com a correção do texto anterior

5 REVISÃO BIBLIOGRÁFICA

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

A segurança da informação é definida pela ISO/IEC 27002 (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 junção destas características também é conhecida como tríade *C.I.A (sigla em inglês para confidencialidade, integridade e disponibilidade)* ou tríade da segurança da informação.

Comentado [AS19]: Não é em itálico.

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 integro 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.

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 onde seu conteúdo não é validado e seu valor interfere no comportamento da aplicação (WHITMAN; MATTORD, 2017).

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**. 2020. Disponível em: https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html. Acesso em: 04 out. 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.

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.

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.

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

Comentado [AS20]: Referência não encontrada no texto.

Comentado [AS21]: Referência não encontrada no texto.

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 TCC I

Acadêmico(a): Artur Ricardo Bizon _____

Avaliador(a): Andreza Sartori_____

	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		
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	x		
	3. JUSTIFICATIVA 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. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?	x		
	Os métodos, recursos e o cronograma estão devidamente apresentados?		x	
	5. 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		
ASPECTOS METODOLÓGICOS	6. 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	
	7. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?		x	
	8. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?	x		
	9. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?		x	
	As citações obedecem às normas da ABNT?		x	
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?		x	

PARECER – PROFESSOR DE TCC I OU COORDENADOR DE TCC
(PREENCHER APENAS NO PROJETO):

O projeto de TCC será reprovado se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADO ( ) REPROVADO

Assinatura: Andreza Sartori_____ Data: 19/10/2020 _____

[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.