WebGoat Plus: Uma Extensão da Ferramenta WebGoat para o Ensino de Vulnerabilidades de Segurança

Artur Ricardo Bizon[1], Gilvan Justino[1]

[1]Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

{abizon, gilvanj}@furb.br

Abstract. *Hands-on vulnerability exploitation activities can provide students a better understanding of security vulnerabilities. Thus, this paper proposes to provide an extension from the WebGoat application that allows the practice of OS Injection vulnerability. In this work tests were applied with nine students who used and evaluated the application. When evaluating the amount of completed assignments, it was noticed that 80% of students solved at least 60% of the assignments. As for user satisfaction, 88% of them would recommend the application to a third party. Thus, it is possible to conclude which the application may contributed in teaching OS Injection vulnerability.*

Resumo. *Atividades práticas de exploração de vulnerabilidades podem trazer um melhor entendimento ao aluno sobre falhas de segurança. Assim, o presente artigo se propôs a disponibilizar uma extensão à aplicação WebGoat que permita o aprendizado prático sobre a vulnerabilidade OS Injection. Neste trabalho foram realizados testes com nove alunos que utilizaram e avaliaram a aplicação. Ao avaliar a quantidade de atividades concluídas foi percebido que 80% dos alunos resolveram ao menos 60% das atividades. Quanto a satisfação dos usuários 88% deles recomendariam a aplicação para um terceiro. Sendo assim, é possível concluir que a aplicação contribuiu no ensino da vulnerabilidade OS Injection.*

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. Em seu relatório anual o [Federal Bureau of Investigation 2020] apresenta um aumento de prejuízos de aproximadamente 3 vezes entre os anos de 2016 (1,5 bilhões de dólares) até 2020 (4,2 bilhões de dólares). 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 e Ekstrom 2011].

Para a prevenção de possíveis prejuízos causados por essas falhas, podem ser utilizadas ferramentas que realizam testes automatizados de segurança ou 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 [Stefinko; Piskozub e Banakh 2016; Xie, Lipford e Chu 2011].

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 e 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, o objetivo geral deste trabalho é incorporar à ferramenta WebGoat uma extensão para que estudantes possam conhecer e aprender sobre a vulnerabilidade *OS Injection*. Tendo como objetivos específicos: (i) disponibilizar um tutorial teórico que introduza os conceitos da vulnerabilidade *OS Injection* através da ferramenta WebGoat; (ii) disponibilizar exercícios práticos para demonstrar a vulnerabilidade OS Injection; (iii) avaliar com uma turma do curso de Sistemas da Informação se este trabalho cumpriu seu papel de ensinar sobre a vulnerabilidade *OS Injection*. O projeto seguiu 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.

2. Fundamentação Teórica

Na presente seção são apresentados os conteúdos que fundamentam a pesquisa. Primeiramente na seção 2.1 são apresentados aspectos fundamentais da segurança da informação. Por sua vez na seção 2.2 é conceituada a vulnerabilidade de segurança *OS Injection*, bem como apresentada como ela ocorre e suas principais variações. Já na seção 2.3 é discorrido sobre o ensino prático de técnicas de *hacking* em universidades, bem como discutido os riscos e implicações éticas no ensino deste conteúdo. Por fim, na seção 2.4 é apresentada a última versão da aplicação WebGoat.

2.1. 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 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. 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 e Mattord 2017].

2.2. 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 falha tem origem na vulnerabilidade de *Command Injection*, na qual a fraqueza ocorre quando o sistema gera um comando a partir da união de um texto pré-definido com um dado de entrada informado pelo usuário. Se esse dado de entrada não for devidamente tratado o usuário pode inserir caracteres que alteram o funcionamento esperado do comando gerado pela instrução [Common Weakness Eumeration 2020b, 2021; Howard, Leblanc e Viega 2010].

Segundo o trabalho de [Stasinopoulos, Ntantogian e Xenakis 2015], a vulnerabilidade *OS Injection* pode ser separada em duas categorias distintas, sendo elas, *OS Injection* ativo e *blind OS Injection*. Na categoria de *OS Injection* ativo (termo adaptado do original *result-based command injection*) o resultado do comando executado pelo sistema operacional é ativamente retornado para o usuário, ou seja, a aplicação exibe para o usuário o resultado do comando executado pelo sistema operacional de forma transparente. Por fim, a *blind OS Injection*, é uma variação em que o resultado do código executado não é retornado para o usuário, podendo ser um resultado processado pela aplicação ou simplesmente o resultado não é repassado ao usuário.

2.3. Educação e Hacking

[Trabelsi e McCoey 2016] afirmam que o ensino de técnicas de segurança ofensiva ou *hacking* é uma peça fundamental para o melhor entendimento do aluno sobre o pensamento hacker e como as falhas de segurança acontecem e são exploradas. 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, assim complementando o entendimento do aluno sobre falhas de

segurança, permitindo que ele implemente soluções apropriadamente seguras. Uma das formas de ensino com laboratórios práticos é o uso de atividades no estilo *Capture The Flag* (CTF). Um exercício de CTF consiste normalmente em duas equipes sendo que o seu objetivo é capturar um determinado dado chamado de *flag* (bandeira em inglês), porém cada equipe possui a sua própria bandeira que deve ser protegida da equipe adversária. A primeira equipe que capturar a bandeira do seu adversário vence o exercício [Vigna. 2003].

Entretanto, o ensino de *hacking* pode causar questionamentos éticos sobre o conhecimento transmitido aos alunos, visto que as mesmas técnicas apresentadas em sala de aula são utilizadas por agentes maliciosos para cometer crimes no mundo real. [Vigna 2003] defende que ensinar técnicas de *hacking* para os alunos não é um problema, pois assim como os chaveiros possuem a capacidade de abrir qualquer porta isso não os torna criminosos. Contudo, [Logan e Clarkson 2005] demonstram preocupação em relação à segurança dos laboratórios nos quais são feitos os exercícios de *hacking*. Alinhado a esta preocupação, em seu trabalho, [Vigna 2003] relata ter ocorrido um acidente durante as aulas. Segundo ele "um experimento gerou um efeito colateral que causou uma interrupção no roteamento na rede de um departamento vizinho durante algumas horas" [Vigna, 2003, traduzido pelo autor]. O relato de [Vigna 2003] reforça a ideia de [Logan e Clarkson 2005] da necessidade de os instrutores tomarem cuidado ao construir exercícios sobre ataques para não deixar a rede inteira da universidade vulnerável.

O trabalho de [Trabelsi e McCoey 2016] aborda a ideia da necessidade de incluir uma matéria sobre ética nas disciplinas que utilizam técnicas de ataques cibernéticos, pois como apresentado em sua pesquisa, após uma aula prática sobre a técnica de *sniffing* de rede, 88% dos alunos da classe admitiram que tentaram utilizar o ataque aprendido em sala de aula na rede da universidade, sendo que, 72% desses alunos afirmaram que seu objetivo era apenas diversão. Assim os autores consideraram que as intenções dos ataques realizados pelos alunos não eram maliciosas.

[Pashel 2006] sugere que além de uma formação ética para os alunos, é necessário mostrar as consequências no âmbito legal sobre o que acontece quando o aluno utiliza de forma ilícita as técnicas aprendidas em sala de aula. [Hartley 2016] demonstra que apesar da possibilidade de alguns alunos fazerem mal uso do conhecimento adquirido, os benefícios alcançados pelos alunos que utilizam os conhecimentos para fins éticos podem compensar eventuais prejuízos. Como apresentado, o ponto principal que os autores concordam é a necessidade de ensinar a parte ética além de simplesmente atacar um sistema, bem como as leis e as consequências que atos ilícitos podem causar ao aluno [Hartley 2016; Logan e Clarkson 2005; Pashel 2006; Trabelsi; McCoey 2016].

2.4 Versão Anterior do WebGoat

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]. 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. O WebGoat é dividido 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 divididas em três etapas. A primeira possui uma explicação de como determinada vulnerabilidade ocorre. A segunda etapa explica como é explorada esta falha. Por último, é apresentado ao usuário como evitar a vulnerabilidade. Já o módulo de desafios é feito no estilo CTF, em que o usuário tem que utilizar os seus conhecimentos adquiridos nas lições para conseguir resolver o desafio.

3. Descrição da Aplicação

Nesta seção é apresentada a descrição da aplicação desenvolvida no presente trabalho, sendo que na seção 3.1 é discorrido sobre a especificação do sistema mostrando alguns diagramas. Já na seção 3.2 descreve sobre a implementação da extensão do sistema WebGoat.

3.1 Especificação

O ambiente desenvolvido neste trabalho é composto por três aplicações distintas, como demonstrado na Figura 1. A Aplicação 1 é uma extensão ao sistema WebGoat. Esta extensão contém a implementação da lição sobre o *OS Injection*, sendo que ela é executada em paralelo com um servidor Apache dentro de um container Docker, que por sua vez é executado de forma local pelo usuário. No momento em que o usuário utiliza a extensão do WebGoat, os dados de suas ações são enviados para a Aplicação 2. Este segundo sistema é um serviço web hospedado em um servidor de nuvem gratuito sendo responsável por armazenar os dados enviados pela extensão do WebGoat. Por fim, a Aplicação 3 é um Jupyter notebook responsável por gerar gráficos de desempenho individual e médio dos usuários a partir dos dados armazenados na Aplicação 2.

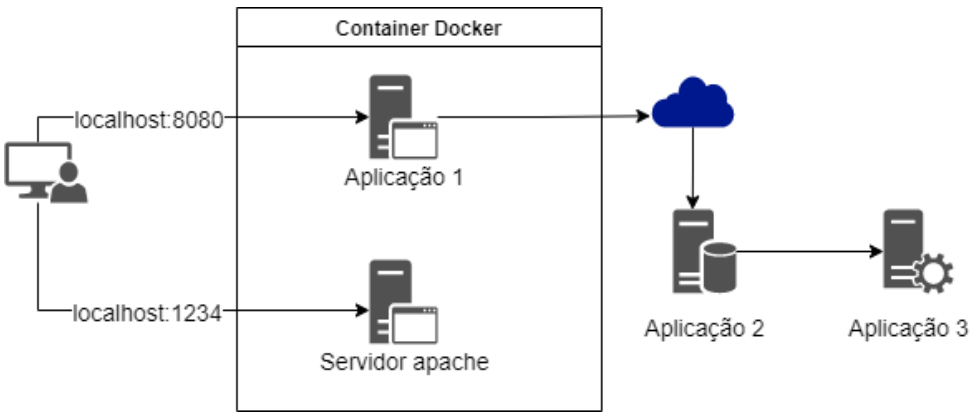

Figura 1. Visão geral do ambiente desenvolvido

Na Figura 2 é apresentado o diagrama de classes da extensão desenvolvida (Aplicação 1). Neste diagrama é possível observar que todas as classes herdam o comportamento da AssignmentEndPoint, esta classe tem por objetivo padronizar a parte de resposta ao usuário, fornecendo métodos padrões para mensagens de sucesso *succes*(), de resposta incorreta *failed*() e mensagens para informação do usuário *informationMessage*(). Outra função desta classe é sinalizar qual classe representa

determinada atividade. Assim as classes apresentadas neste diagrama são referentes a cada atividade.

Por sua vez estas classes seguem um padrão semelhante entre elas, já que elas possuem dois métodos principais além dos apresentados anteriormente. O primeiro é um método referente ao *end point* que deve ser acessado para resolver determinada atividade. Neste método é realizada a correção da resposta enviada pelo usuário e o envio do feedback ao usuário. O segundo método é responsável por coletar as informações que serão as respostas que os usuários devem enviar para a aplicação, caso a resposta seja um dado que pode variar, como por exemplo, o IP interno do container, é responsabilidade deste segundo método coletar esta informação para que seja possível a validação do dado fornecido pelo usuário.

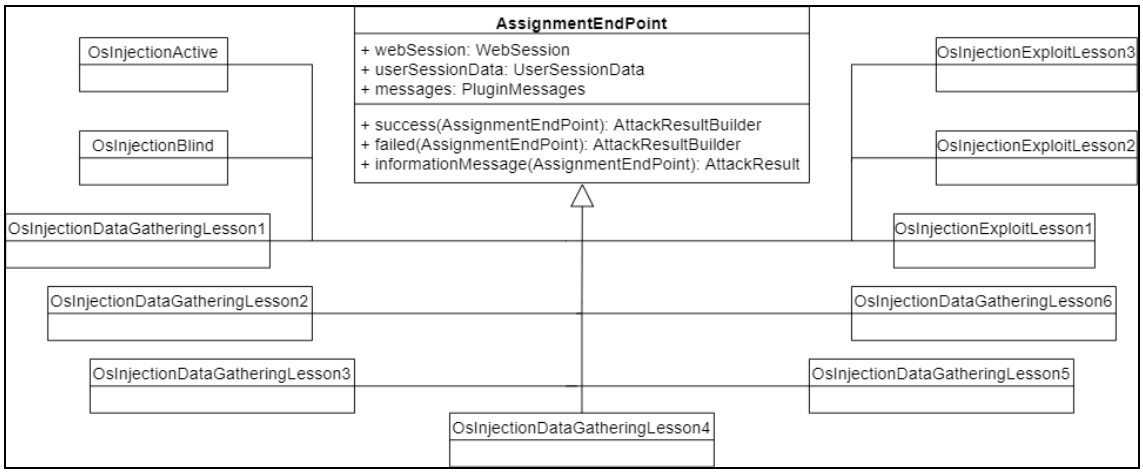

Figura 2. Diagrama de classes da extensão do WebGoat desenvolvida

3.2 Implementação da Extensão do WebGoat

A aplicação WebGoat possui uma estrutura bem definida permitindo que novas lições sejam desenvolvidas como módulos que são adicionados ao módulo responsável pelo gerenciamento das lições. Desta forma quando um novo módulo é adicionado à coleção de módulos do WebGoat, a aplicação automaticamente adiciona este módulo como uma lição no menu presente na interface do usuário. Cada lição pode conter conteúdos em texto que são fornecidos através da injeção de texto em um *template* HTML definido pelo desenvolvedor. As lições também podem conter exercícios, sendo que cada questão é referente a um *end point* REST definido por uma classe que herda o comportamento da classe AssignmentEndPoint, como explicado na seção anterior.

Ao todo, a lição desenvolvida para o ensino da vulnerabilidade *OS Injection* é organizada em 13 páginas, sendo a primeira e a última páginas de conteúdo teórico enquanto as demais contém atividades. Na página com conteúdo teórico são apresentados alguns comandos básicos para o sistema operacional Linux. Em seguida são abordados os conceitos do *OS Injection*, suas variações e exemplo de como é o código fonte de uma aplicação Java suscetível a esta vulnerabilidade. Neste momento da lição o usuário também é apresentado a alguns exercícios simples para compreender as variações da vulnerabilidade e entender como ocorre a injeção de comandos.

As próximas páginas são focadas apenas em atividades práticas, que por sua vez são divididas em duas categorias: coleta de dados e exploração. A categoria de coleta de

dados, tem o seu foco em orientar o aluno sobre as formas que ele pode utilizar a falha aprendida para coletar dados sensíveis sobre o servidor que hospeda a aplicação. Já a categoria de exploração demonstra algumas das formas que o usuário pode utilizar os dados coletados para prejudicar o sistema vulnerável. Por fim, a última página apresenta algumas técnicas recomentadas pela OWASP e CWE para mitigar a vulnerabilidade *OS Injection*. Também são apresentados através de exemplos como estas técnicas poderiam ser utilizadas para corrigir o código vulnerável apresentado nas primeiras páginas.

No Quadro 1 estão relacionadas as atividades da lição, com o seu objetivo a ser cumprido e o aspecto de segurança que será comprometido caso a atividade for concluída. No Quadro 1 é possível observar que a partir da página 4 até a página 10 o foco das atividades é coletar dados da aplicação, comprometendo a confidencialidade da aplicação que o usuário deve atacar. Já as páginas 11 e 12 tem o seu foco nos outros aspectos da tríade da segurança da informação, sendo que a atividade da página 11 demostra como o usuário pode comprometer a integridade da aplicação suscetível a *OS Injection*. Já a página 12 apresenta como o usuário pode utilizar esta vulnerabilidade para comprometer a disponibilidade.

Página	Nome da Atividade	Objetivo da atividade	Aspecto de Segurança Comprometido
2	OsInjecticonActive	Imprimir “Hello World” na tela	Não se aplica
3	OsInjectionBlind	Identificar o campo de entrada vulnerável	Não se aplica
4	OsInjectionDataGatheringLesson1	Coletar o nome do usuário corrente	Confidencialidade
5	OsInjectionDataGatheringLesson2	Contar quantidade de usuários root presentes no servidor	Confidencialidade
6	OsInjectionDataGatheringLesson3	Coletar o IP interno do servidor	Confidencialidade
7	OsInjectionDataGatheringLesson4	Coletar a versão do *kernel* Linux do servidor	Confidencialidade
8	OsInjectionDataGatheringLesson5	Coletar o UID do usuário corrente no servidor	Confidencialidade
9	OsInjectionDataGatheringLesson6	Coletar o PID dos processos que tem o nome “apache2”	Confidencialidade
10	OsInjectionExploitLesson1	Coletar uma *flag* escondida na página HTML do servidor Apache	Confidencialidade
11	OsInjectionExploitLesson2	Alterar o conteúdo da página HTML do servidor Apache	Integridade
12	OsInjectionExploitLesson3	Finalizar o servidor Apache	Disponibilidade

Quadro 1. Atividades desenvolvidas na aplicação, juntamente com o seu objetivo e aspecto de segurança relacionado.

Na Figura 3 é apresentada a interface gráfica do WebGoat juntamente com a lição desenvolvida por este trabalho. Esta lição pode ser acessada através do item *OS Injection* - indicado pelo item (b) da figura - presente no menu do lado esquerdo - item (a) da figura. No item (d) da Figura 3 é apresentada a estrutura adotada pelas atividades. Nesta estrutura a tela possui dois campos de entrada, sendo o primeiro simulando uma aplicação do mundo real e logo abaixo outro campo ao qual o usuário insere a resposta para a tarefa. No item (c) da Figura 3 também é possível observar uma das dicas fornecidas pelo sistema para auxiliar na solução da atividade. Cada exercício possui seu

próprio conjunto de dicas, seguindo um esquema de progressão de dicas superficiais até dicas específicas indicando links externos que demostram os comandos que o usuário deve executar.

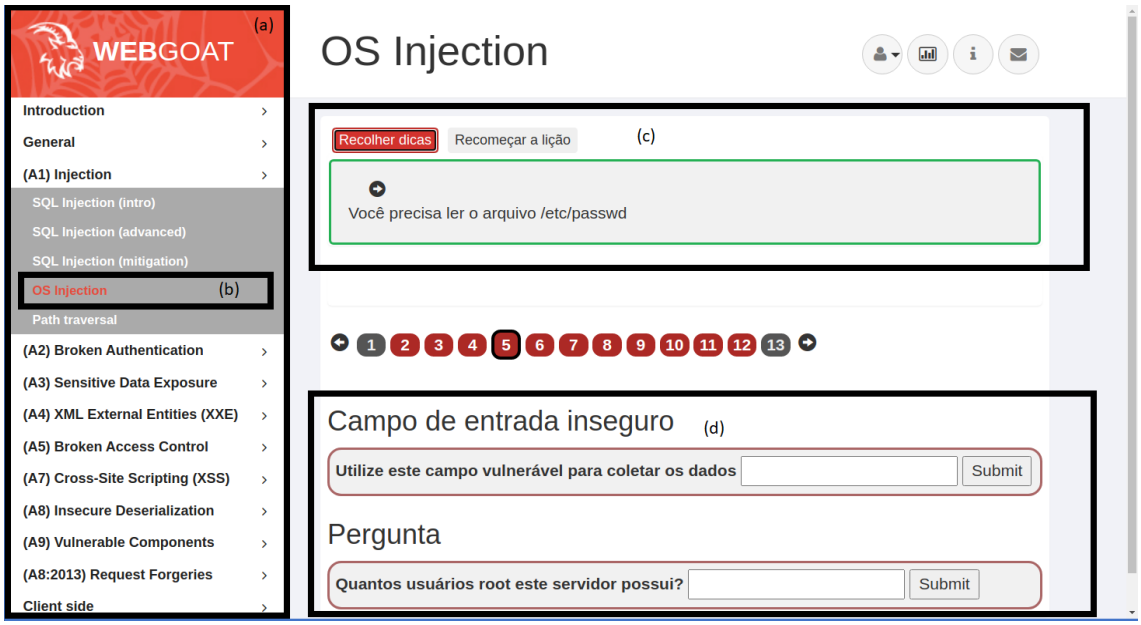

Figura 3. Interface da aplicação desenvolvida.

4. Resultados

Para testar a utilidade da aplicação desenvolvida no presente trabalho foram realizados testes com usuários. Os dados avaliados foram coletados através das informações de uso da aplicação de cada usuário. Ademais, ao final do teste, foi solicitado o preenchimento de um formulário sobre aprendizado no que tange a usabilidade do sistema. A metodologia de testes foi realizada da seguinte forma: inicialmente o sistema foi testado apenas com um aluno do curso de Ciência da Computação. Neste teste o autor esteve monitorando a tela do usuário durante todo o período de teste ao qual durou uma hora. Este teste individual foi realizado com o intuito de verificar a consistência do sistema desenvolvido. Após esta primeira fase de teste foi realizado um novo teste com duração de uma hora em uma turma do curso de Sistemas da Informação. Assim, ao todo, juntando as duas fases, o sistema foi testado por um total de 9 usuários. Vale ressaltar que em ambas as fases os alunos pertenciam ao sétimo semestre de seus respectivos cursos. Foram selecionados alunos desta fase pois eles já possuíam conhecimentos básicos de segurança da informação.

Na Figura 4 é apresentado um gráfico que demonstra a quantidade de usuários que conseguiram resolver determinada atividade. Nele pode ser observado que conforme o avanço das atividades a quantidade de usuários que conseguem concluí-las vai diminuindo. Uma hipótese para este efeito é a possível progressão de dificuldade das atividades. Como demostrado na Figura 4, apenas um usuário conseguiu resolver estas atividades. Entretanto mesmo com este indicador de dificuldade, 7 dos 9 usuários que utilizaram a aplicação responderam corretamente ao menos 60% das atividades. Isto pode indicar que a aplicação alcançou o seu objetivo de ensinar sobre a vulnerabilidade *OS Injection*.

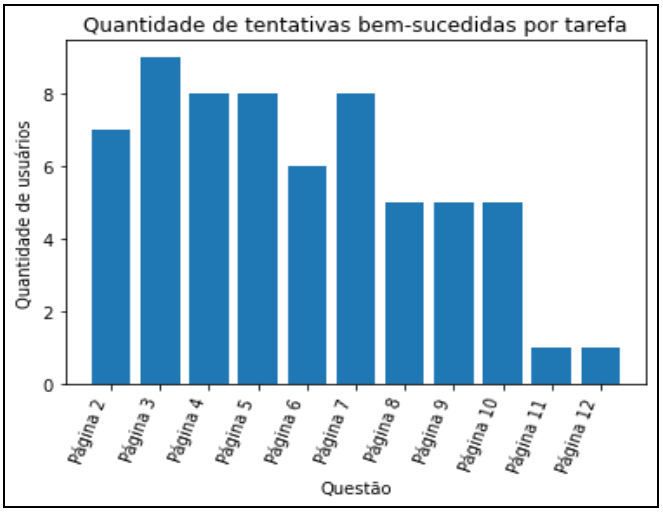

Figura 4. Quantidade de tentativas bem-sucedidas

Já a Figura 5 apresenta a quantidade média de injeções corretas e incorretas. Nesta figura é possível observar que a quantidade de injeções erradas vai diminuindo ao passo que os usuários vão avançando pelas páginas. Pode indicar uma curva de aprendizado dos alunos. Vale ressaltar que na Figura 5 não é considerado o conteúdo do comando que o usuário executou, apenas é validado se o comando foi corretamente injetado. Isso justifica a incongruência entre o número de tentativas corretas na atividade referente à página 11 com a quantidade de soluções apresentadas para a mesma atividade demostrado na Figura 4, de modo que na Figura 5 são executadas em média 6 injeções corretas enquanto na Figura 4 apenas um usuário conseguiu concluir esta atividade.

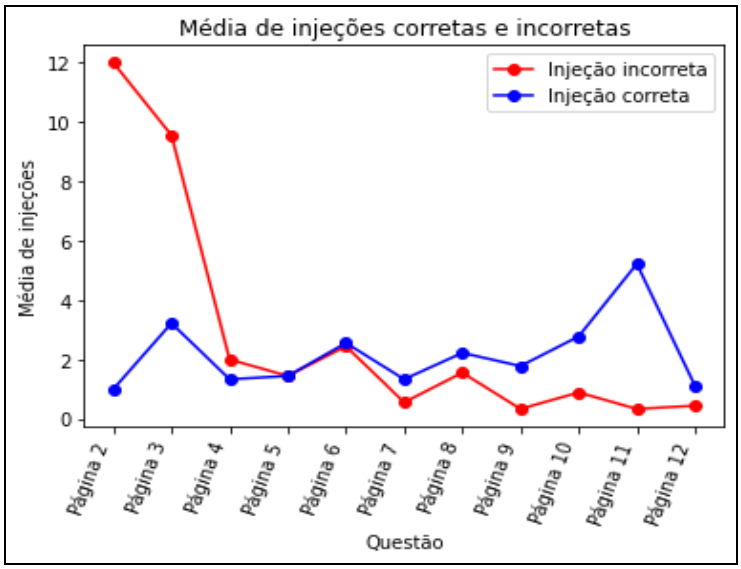

Figura 5. Quantidade média de injeções corretas e incorretas

Ao final do período de testes os usuários foram convidados a responder um questionário de usabilidade sobre os aspectos de aprendizagem e satisfação do usuário em relação a aplicação. As perguntas utilizadas no formulário foram baseadas nos trabalhos de [Lund 2001] e [Chin, Diehl e Norman 1988]. Na abordagem desses autores,

as perguntas recebem respostas na escala de 1 a 5, de tal forma que 1 é uma resposta muito negativa e 5 é muito positiva.

Na Tabela 1 são apresentadas as perguntas bem como as respostas fornecidas pelos usuários. Nesta tabela é possível observar que as perguntas relacionadas às mensagens, ou seja, da pergunta 1 até a número 4, foram, no geral avaliados positivamente pelos usuários. Entretanto como demostrado nas perguntas 5 e 6, os usuários relataram terem dificuldades em resolver os problemas fornecidos pela aplicação, sendo que apenas 44% dos alunos consideraram ser fácil de aprender a utilizá-la e somente 33% dos alunos consideraram fácil de resolver as atividades propostas a eles. Um ponto positivo foi a avaliação relacionada a satisfação dos usuários. Como demostra a Tabela 1 55% dos alunos se consideram satisfeitos ao utilizar a aplicação, enquanto 77% dos alunos afirmam que a aplicação é divertida de se utilizar, por fim ao questionar se eles recomentariam a aplicação a outra pessoa 88% dos alunos deram respostas positivas.

Pergunta	**Avaliação**				
	1	**2**	**3**	**4**	**5**
1 - As posições das mensagens na tela são consistentes?	0,00%	0,00%	11,11%	44,44%	44,44%
2 - Os campos de entrada são organizados?	22,22%	0,00%	0,00%	44,44%	33,33%
3 – A aplicação informa sobre o seu progresso?	0,00%	22,22%	22,22%	11,11%	44,44%
4 - As mensagens de erro são úteis?	11,11%	11,11%	11,11%	11,11%	55,56%
5 – Foi fácil aprender a utilizar a aplicação?	11,11%	33,33%	11,11%	22,22%	22,22%
6 - É fácil realizar as tarefas da aplicação?	0,00%	11,11%	55,56%	22,22%	11,11%
7 - As dicas apresentadas para cada questão são úteis?	11,11%	0,00%	22,22%	44,44%	22,22%
8 - Os materiais de apoio apresentados são úteis?	0,00%	0,00%	22,22%	66,67%	11,11%
9 - Estou satisfeito com a aplicação	0,00%	11,11%	33,33%	11,11%	44,44%
10 - A aplicação é divertida de se utilizar	0,00%	0,00%	22,22%	44,44%	33,33%
11 - Você recomentaria esta aplicação a um amigo	0,00%	0,00%	11,11%	66,67%	22,22%

Tabela 1. Respostas do questionário

Ao final do questionário foram deixados dois campos opcionais para os alunos comentarem sobre os aspectos positivos e negativos do sistema. Iniciando pelos pontos negativos, um item levantado é a dificuldade. Segundo um usuário o sistema possui uma "difícil curva de aprendizado" enquanto outro usuário acredita que por ele não possuir conhecimentos prévios sobre o assunto abordado ele ficou com uma "imagem negativa e difícil" da aplicação. Um terceiro usuário levanta uma questão que pode ser o principal motivo destas reclamações, segundo ele "É necessário ter um certo conhecimento com Linux, ou de certa forma pesquisar a respeito dos comandos que resolvem o problema".

Já o aspecto positivo citado pelos usuários é principalmente a possibilidade de aprender algo novo através da aplicação. Um usuário deixou o seguinte relato sobre a

sua experiência em utilizar a aplicação: “Não conhecia os conceitos apresentados, nunca utilizei comando Linux. Gostei bastante de ver que essas vulnerabilidades existem e de entender a importância de tratar elas”. Este comentário demonstra que mesmo uma pessoa não sabendo utilizar comandos Linux, ela tem condições de aprender e compreender o funcionamento da vulnerabilidade abordada através da aplicação desenvolvida por este trabalho. Outro usuário comenta que o período de teste definido foi insuficiente, entretanto segundo ele as atividades “são um desafio interessante”. Um terceiro usuário comenta sobre a facilidade de entendimento através dos conteúdos fornecidos, entretanto ele sugere que deveriam existir mais materiais sobre alguns comandos para ajudar nas atividades mais difíceis.

5. Conclusões

O presente estudo apresentou uma extensão para a aplicação WebGoat focada ao ensino da vulnerabilidade *OS Injection*. Esta extensão aborda as duas principais variações da falha, sendo elas a blind *OS Injection*, e o *OS Injection* ativo, apresentando os riscos que a falha pode trazer aos sistemas que contém esta vulnerabilidade. O sistema também apresenta formas que o usuário pode evitar que sua aplicação se torne suscetível a esta falha. As técnicas de mitigação apresentadas pela aplicação são feitas com base nas recomendações tanto da CWE quanto da OWASP. Também são fornecidos exemplos de códigos-fonte vulneráveis e seguros.

Para validar a aplicação foram realizados testes de uso com usuários, eles também avaliaram a aplicação através de um questionário sobre a usabilidade e satisfação. Estes testes indicam que a aplicação tem um potencial para auxiliar no ensino da falha tratada por este estudo. Assim o objetivo geral deste estudo que é disponibilizar uma extensão da aplicação WebGoat para que estudantes possam conhecer e aprender sobre a vulnerabilidade *OS Injection*, foi alcançado. Quanto aos objetivos específicos i e ii, estes foram cumpridos incorporando os conceitos do *OS Injection* na aplicação desenvolvida neste estudo, bem como na criação de exercícios que apresentassem o potencial de risco que a falha fornece ao sistema vulnerável.

Sobre o objetivo específico iii que é validar a utilidade do sistema desenvolvido, foi aplicado teste com usuários como descrito anteriormente. Nesse teste foram coletados dados de uso e respostas de um questionário sobre aprendizado e satisfação. Como demostrado nos resultados, a partir destes dados coletados foi possível criar gráficos que demostravam uma possível curva de aprendizado sobre a exploração do *OS Injection*. Outro ponto a ser levantado é a concordância que os dados de uso tiveram em relação as respostas dos questionários, desta forma fortalecendo os aspectos levantados por estes indicadores. Assim é possível afirmar que a aplicação pode contribuir positivamente no ensino da vulnerabilidade de *OS Injection*, desta forma o objetivo específico iii foi concluído.

Quanto as contribuições sociais, a aplicação desenvolvida neste estudo permite que os usuários possam explorar a vulnerabilidade *OS Injection* sem pôr o seu sistema operacional em risco, visto que a aplicação executa de modo virtualizado tornando a experimentação segura para os usuários. Outro aspecto relevante é a satisfação dos usuários, pois como apresentado na seção dos resultados, aproximadamente 88% dos usuários recomendariam a aplicação desenvolvida para um amigo, isso demostra que o sistema pode incentivar os alunos a estudar mais a fundo as falhas de segurança ou pode

despertar um maior interesse sobre a área. Vale ressaltar que a aplicação desenvolvida neste estudo está disponível no seguinte link https://github.com/ArturBizon/WebGoat/tree/os-injection.

Como sugestão para futuros trabalhos, é recomendado o estudo de formas para melhorar o engajamento dos alunos com a aplicação, podendo fazer uso de elementos de gamificação ou recursos visuais mais apelativos, bem como o desenvolvimento de abordagens que tornem o aprendizado de novas técnicas de segurança mais simples. Por fim, é sugerido também o desenvolvimento de um sistema que ensine sobre vulnerabilidades de segurança, podendo ainda abordar falhas que ocorrem em redes de computadores não somente em sistemas. Outra característica para este novo estudo é a desenvolvimento de funções que permitam ao professor acompanhar o desenvolvimento individual dos alunos através de seus dados de uso coletados.

6. Referências

Associação Brasileira de Normas Técnicas (2005) "NBR ISO/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.

Chin, John P.; Diehl, Virginia A.; Norman, Kent L (1998) "Development of an instrument measuring user satisfaction of the human-computer interface". In: SIGCHI Conference on Human Factors in Computing System, [S.I], 1988, Washington. Proceedings, Association for Computing Machinery. p. 213-218.

Common Weakness Enumeration (2020a) "2020 CWE Top 25 Most Dangerous Software Weaknesses". Disponível em: https://cwe.mitre.org/top25/archive/2020/2020cwetop25.html. Acesso em: 04 out. 2020.

Common Weakness Enumeration (2021) "CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')". Disponível em: https://cwe.mitre.org/data/definitions/77.html. Acesso em: 28 jun. 2021.

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

Federal Bureau of Investigation (2021) "2020 Internet Crime Report". Disponível em: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf. Acesso em: 03 jul. 2021.

Hartley, Regina D. (2015) "Ethical Hacking Pedagogy: an analysis and overview of teaching students to hack". Journal Of International Technology And Information Management, [S.I.], ano 4, v. 24, p. 95-104, Disponível em: https://scholarworks.lib.csusb.edu/jitim/vol24/iss4/6/. Acesso em: 15 jun. 2021.

Howard, Michael; Leblanc, David; Viega, John. (2010) "24 DEADLY SINS OF SOFTWARE SECURITY: programming flaws and how to fix them". New York: McGraw Hill.

Logan, Patricia Y.; Clarkson, Allen. (2005) "Teaching students to hack: curriculum issues in information security". In: 36th SIGCSE Techinical Symposium on

Computer Science Education. 36, Nova Iorque. Proceedings. Association for Computing Machinery, 2005. p. 157-161.

Lund, Arnold M. (2001) “Measuring usability with the USE questionnaire”. Usability interface, [S.I.] v. 8, n. 2, p. 3-6.

Open Web Application Security Project (OWASP) (2020) “OWASP WebGoat”. Disponível em: <https://owasp.org/www-project-webgoat/>. Acesso em: 04 out. 2020.

Pashel, B. A (2006) “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, Association for Computing Machinery. p. 197–200. https://doi.org/10.1145/1231047.1231088

Rowe, Dale C.; Lunt, Barry M.; Ekstrom, Joseph J. (2011) “The role of cyber-security in information technology education”. In: 2011 Conference on Information Technology Education. 3. 2011, Nova Yorque. Proceedings, Association for Computing Machinery. p. 113-122.

Stasinopoulos, Anastasios; Ntantogian, Christoforos ; Xenakis, Christos. (2015) “Commix: Detecting and exploiting command injection flaws”. In: Blackhat Europe 2015, Amsterdam. Proceedings, Blackhat, [9] p., Disponível em: https://www.blackhat.com/docs/eu-15/materials/eu-15-Stasinopoulos-Commix-Detecting-And-Exploiting-Command-Injection-Flaws-wp.pdf. Acesso em: 13 junho 2021.

Stefinko, Yaroslav; Piskozub, Andrian; Banakh, Roman. (2016) “Manual and automated penetration testing. Benefits and drawbacks. Modern tendency”. In: 13th International Conference on Modern Problems of Radio Engineering, Telecommunications And Computer Science (TCSET), 13, 2016, Liviv. Proceedings, IEEE. p. 488-491.

Trabelsi, Zouheir; McCoey, Margaret. (2016) “Ethical Hacking in Information Security Curricula”. International Journal Of Information And Communication Technology Education (IJICTE), [S.L.], ano 1, v. 12, p. 1-10.

Vigna G. (2003) “Teaching Network Security through Live Exercises” In: Security Education and Critical Infrastructures. WISE 2003. IFIP Advances in Information and Communication Technology, vol 125, p. 3-18.

Xie, Jing; Lipford, Heather Richter; Chu, Bill (2011) “Why do programmers make security errors?”. In: 2011 IEEE Symposium on Visual Languages And Human-centric Computing (VL/HCC). [S.I], 2011, Pittsburgh. Proceedings IEEE. p. 161-164. DOI 10.1109/VLHCC.2011.6070393.

Whitman, Michael E; Mattord, Herbert J (2017) “Principles of Information Security”. 6. ed. Boston: Cengage Learning. 656 p.