DOI: 10.5748/20CONTECSI/PSE/SEC/7222

eLocator: e207222

TEACHING INFORMATION SECURITY: A PRACTICAL STUDY ON SQL INJECTION

Rafael Fernando Diorio – https://orcid.org/0000-0002-5574-0941
Instituto Federal De Educação, Ciência E Tecnologia De São Paulo - Ifsp

TEACHING INFORMATION SECURITY:A PRACTICAL STUDY ON SQL INJECTION

ABSTRACT

Injection attacks are among the most dangerous attacks targeting Web applications. Among all types of injection attacks, SQL Injection (SQLi), which makes it possible to execute malicious SQL statements and may lead to full system compromise, is one of the most common attack vectors and poses a serious security risk to organizations. For this reason, it is crucial that Information Technology students and professionals, especially those focused on information security, are prepared to deal with these attacks. In this context, a practical study on SQL Injection attacks is discussed in this paper. To this end, based on a reference scenario, related attacks are reproduced for experimentation and discussion purposes. The experiments and discussion are based on a practical laboratory about Web application security, carried out with third-year students of an undergraduate computer science course to help them understand concepts related to the topic.

Keywords: Web Application Security, SQL Injection, Experimentation, Teaching and Learning, Information Security.

ENSINO DE SEGURANÇA DA INFORMAÇÃO: UM ESTUDO PRÁTICO SOBRE SQL INJECTION

RESUMO

Os ataques de injeção estão entre os ataques mais perigosos direcionados a aplicações Web. Entre todos os tipos de ataques de injeção, o SQL Injection (SQLi), que possibilita a execução de instruções SQL maliciosas e pode levar ao comprometimento total do sistema, é um dos vetores de ataque mais comuns e representa um sério risco de segurança para as organizações. Por esse motivo, é crucial que estudantes e profissionais da área de Tecnologia da Informação, em especial, focados em segurança da informação, estejam preparados para lidar com tais ataques. Nesse contexto, um estudo prático sobre ataques de SQL Injection é discutido neste trabalho. Para tal, com base em um cenário de referência, ataques relacionados são reproduzidos para fins de experimentação e discussão. Os experimentos e discussão baseiam-se em um laboratório prático sobre segurança de aplicações Web, realizado com estudantes do terceiro ano de um curso de graduação em informática para auxiliá-los na compreensão de conceitos relacionados ao tema.

Palavras-chave: Segurança de Aplicações Web, SQL Injection, Experimentação, Ensino e Aprendizagem, Segurança da Informação.

1. Introdução

Aplicações Web são alvo constante de ataques cibernéticos. Entre tais ataques, os ataques de injeção estão entre os principais e mais perigosos, listados entre os dez riscos de segurança mais críticos para aplicações Web pelo Open Web Application Security Project (OWASP) (OWASP Foundation, 2023). Entre todos os tipos de ataques de injeção, o SQL Injection (SQLi), que possibilita a execução de instruções SQL maliciosas,para o proveito do atacante,e pode levar ao comprometimento total do sistema, é um dos vetores de ataque mais comuns (Elmasri&Navathe, 2018, p. 1035; Halfond, 2006; Li et al., 2019). Em tais ataques, quando bem-sucedidos, um atacante pode contornar as medidas de segurança da aplicação e recuperardados confidenciais do banco de dados, bem como adicionar, modificar e excluir registros e, em alguns casos, executar comandos em nível de sistemae controlar o sistema que hospeda a aplicação(Elmasri&Navathe, 2018, p. 1035; Halfond, 2006; Li et al., 2019). Os efeitos relacionados podem ser extremamente críticos para organizações e indivíduos, com possíveis consequências incluindo a manipulação não autorizada do banco de dados, o vazamento de dados, o roubo/falsificação de identidade e a realização defraudes, entre outros(Elmasri&Navathe, 2018, p. 1035; Halfond, 2006; Li et al., 2019).

Diante desse cenário, a compreensão de conceitos, de técnicas e de abordagens sobre os ataques de SQL Injection, bem como a realização de experimentos práticos quanto aos mesmos, são cruciais no processo de formação de estudantes e profissionais da área de Tecnologia da Informação, com contribuições recentes abordando questões relacionadas, tais como nos trabalhos de Pinget al. (2020),Zivanic et al. (2022), Chen et al. (2019), Wang e Liu (2023) e Vyamajala et al. (2018), entre outros. Como exemplo, no trabalho de Pinget al. (2020), os autores discorrem sobre o uso da solução SQLi-labs para o ensino de SQL Injection. No trabalho de Zivanicet al. (2022), os autores apresentam uma aplicação utilizada para demonstrar ataques de SQL Injection. Nos trabalhos de Chen et al. (2019) e Wang e Liu (2023), os autores discorrem sobre o uso da solução DVWA para experimentações envolvendo a segurança de aplicações Web, incluindo ataques de SQL Injection. No trabalho de Vyamajalaet al. (2018), os autores discorrem sobre o uso das soluções JSQL e Acunetix em questões relacionadas aos ataques de SQL Injection.

Nesse contexto, com ênfase no processo de ensino e aprendizagem em segurança da informação e objetivando contribuir com os demais trabalhos relacionados ao tema, este trabalho discute um estudo prático sobre ataques de SQL Injection. Para tal, a partir de um cenário de referência, um ambiente computacional é utilizado para fins de experimentação e discussão. Os experimentos e discussão baseiam-se em um laboratório prático sobre segurança de aplicações Web, realizado com estudantes do terceiro ano de um curso de graduação em informática, em uma das aulas de segurança da informação, para auxiliá-los na compreensão de conceitos sobre SQL Injection. Essa discussão é importante, por exemplo, para conciliar teoria e prática em atividades de ensino relacionadas ao tema, bem como para que novas pesquisas e contribuições sejam realizadas no âmbito da segurança da informação, tais como para a prevenção e para a detecção dos ataques de SQL Injection.

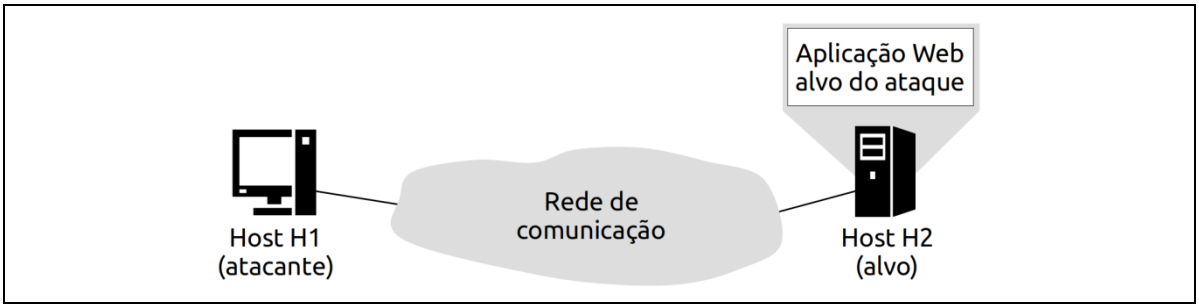
O restante deste trabalho está organizado da seguinte forma: o cenário de referência utilizado para fins de experimentação e discussão é descrito na Seção 2. Os materiais e métodos são descritos na Seção 3. Os resultados experimentais são descritos na Seção 4 e, por fim, a conclusão e os trabalhos futuros são descritos na Seção 5.

2. Cenário de Referência

O cenário de referência utilizado para fins de experimentação e discussão sobre ataques de SQL Injection é ilustrado na Figura 1.

Figura 1

Cenário de referência utilizado para fins de experimentação e discussão sobre ataques de SQL Injection.



Conforme ilustrado na Figura 1, dois hosts são utilizados no cenário de referência explorado neste trabalho: um host atacante (H1) e um host alvo (H2). Ambos os hosts estão interconectados por meio de uma rede de comunicação baseada em um ambiente computacional virtualizado, em que o host H1 atua como cliente e o host H2 atua como servidor para os serviços de hospedagem de páginas Web e de banco de dados, hospedando a aplicação Web alvo do ataque de SQL Injection.

Em tal cenário, a aplicação Web alvo do ataque de SQL Injection é acessível por meio do endereço http://www.exemplo.com.br/app e tem como referência um sistema acadêmico, com interfaces de acesso público, acessíveis por qualquer usuário, e interfaces de acesso restrito, acessíveis por usuários autenticados. O comportamento esperado da aplicação, sem vulnerabilidades, é que todas as informações disponibilizadas por meio de suas interfaces sejam limitadas aos propósitos das mesmas e que apenas usuários legítimos do sistema, previamente autenticados, tenham acesso aos conteúdos/recursos disponíveis por meio de suas interfaces de acesso restrito.

3. Materiais e Métodos

Para a implementação do cenário de referência ilustrado na Figura 1, a solução Oracle VM VirtualBox[1] foi utilizada, com ambos os hosts (H1 e H2) baseados em máquinas virtuais. Em tal cenário, o host H1 foi configurado utilizando o sistema operacional Kali Linux[2] 2023.2 e o host H2 foi configurado utilizando o sistema operacional Linux CentOSStream[3] 9. No host H2, os serviços de hospedagem de páginas Web e de banco de dados foram configurados, respectivamente, por meio das soluções Apache, com suporte ao PHP, e MySQL. A aplicação Web alvo do ataque, hospedada no host H2, foi desenvolvida utilizando as soluções PHP e MySQL.

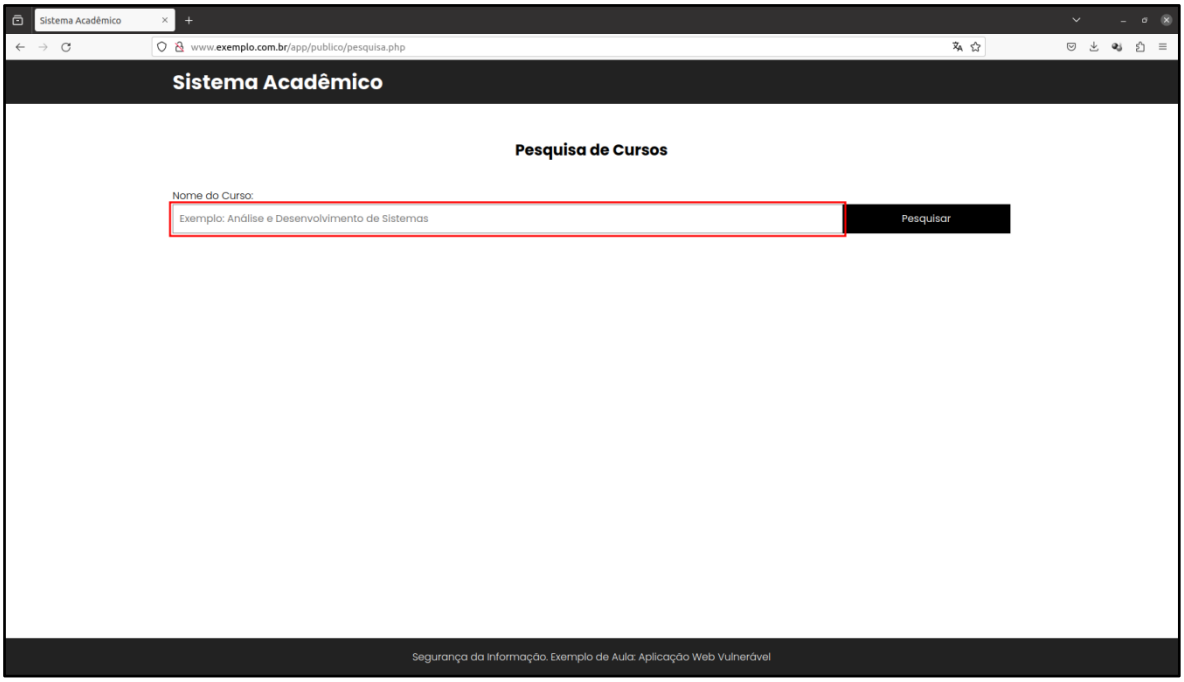
[1]Oracle VM VirtualBox: https://www.virtualbox.org/
[2]Kali Linux: https://www.kali.org/
[3]Linux CentOSStream: https://www.centos.org/centos-stream/

Nesse cenário, os ataques foram realizados do host H1 para o host H2, explorando o SQL Injection na aplicação Web hospedada junto ao mesmo de forma manual, sem a utilização de ferramentas automatizadas em tal processo. A exploraçãodo SQL Injectionfoi realizada de forma manual, em especial, para fins didáticos, objetivando apresentar possíveis técnicas e abordagens que podem ser utilizadas pelo atacante para a detecção da vulnerabilidade e para a obtenção de dados do banco de dados da aplicação.

Na aplicação Web alvo do ataque, as explorações tiveram como referência uma de suas interfaces de acesso público, utilizada para fins de pesquisa de cursos (Figura 2). Em tal interface, o SQL Injection foi explorado por meio do campo destinado ao nome do curso a ser pesquisado, ilustrado em destaque na Figura 2.

Figura 2

Interface da aplicação Web alvo do ataque de SQL Injection, com destaque para o campo utilizado para fins de experimentação.



Quanto ao código-fonte que trata da consulta disponibilizada por meio da interface ilustrada na Figura 2, a Figura 3 ilustra o trecho pertinente ao mesmo responsável por receber a entrada do usuário (linha 1), definir a consultaque será realizada ao banco de dados da aplicação (linha 2) e realizar a consultaem questão(linha 3).

Figura 3

Trecho do código-fonte da aplicação Web alvo do ataque que trata da consulta disponibilizada por meio da interface ilustrada na Figura 2.

```
1   $search = $_GET["search"];
2   $sql = "SELECT * FROM cursos WHERE nome LIKE '%$search%'";
3   $result = mysqli_query($conn, $sql);
```

Para fins de experimentação, considerou-se que o atacante desconhecia qualquer particularidade sobre a configuração da aplicação Web alvo do ataque, incluindo questões relacionadas ao seu banco de dados.Também considerou-se que o ataque de SQL Injection tinha como objetivo a obtenção de dados confidenciais do banco de dados da aplicação Web, mais especificamente, os *logins* e senhas de acesso para suas interfaces de acesso restrito.

4. Resultados e Discussão

Considerando o cenário de referência descrito na Seção 2 e os materiais e métodos descritos na Seção 3, uma possível abordagem a ser utilizada pelo atacante para detectar se a aplicação Web alvo do ataque é vulnerável ao SQL Injection se dá pelo uso de aspas simples enquanto entrada para a pesquisa de cursos. O uso de aspas simples, em certas circunstâncias, interrompe a consulta realizada pela aplicação ao banco de dados e possibilita identificar que a mesma é vulnerável. Nesse contexto, a Figura 4 ilustra a utilização de aspas simples enquanto possível abordagem do atacante para detectar se a aplicação Web alvo do ataque é vulnerável ao SQL Injection, com destaque para a entrada utilizada e para o erro gerado pela aplicação a partir da mesma. Por meio do erro gerado pela aplicação, é possível identificar que ela é vulnerável ao SQL Injection. Para efeitos de comparação, a Figura 5 ilustra a utilização de uma entrada sem o uso de aspas simples na aplicação Web alvo do ataque, com destaque para a entrada em questão e para o retorno gerado pela aplicação a partir da mesma.

Figura 4

Possível abordagem do atacante para detectar se a aplicação Web alvo do ataque é vulnerável ao SQL Injection, com destaque para a entrada utilizada, via "Resultado da pesquisapor:", e para o erro gerado pela aplicação a partir da mesma.

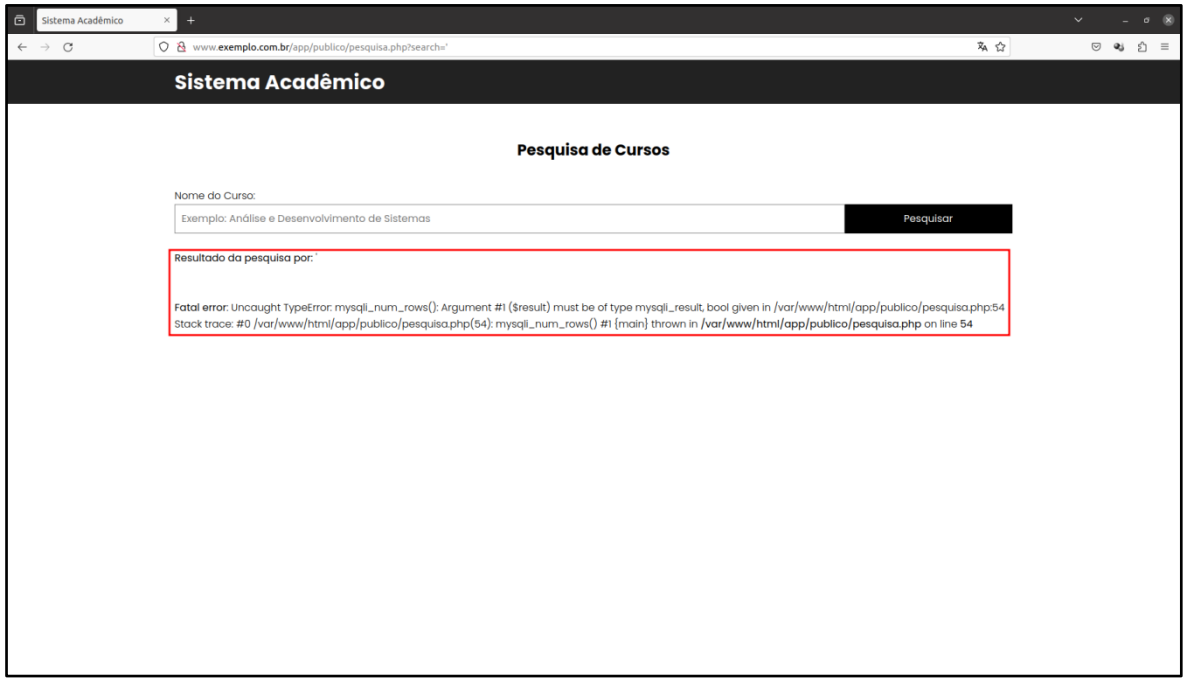
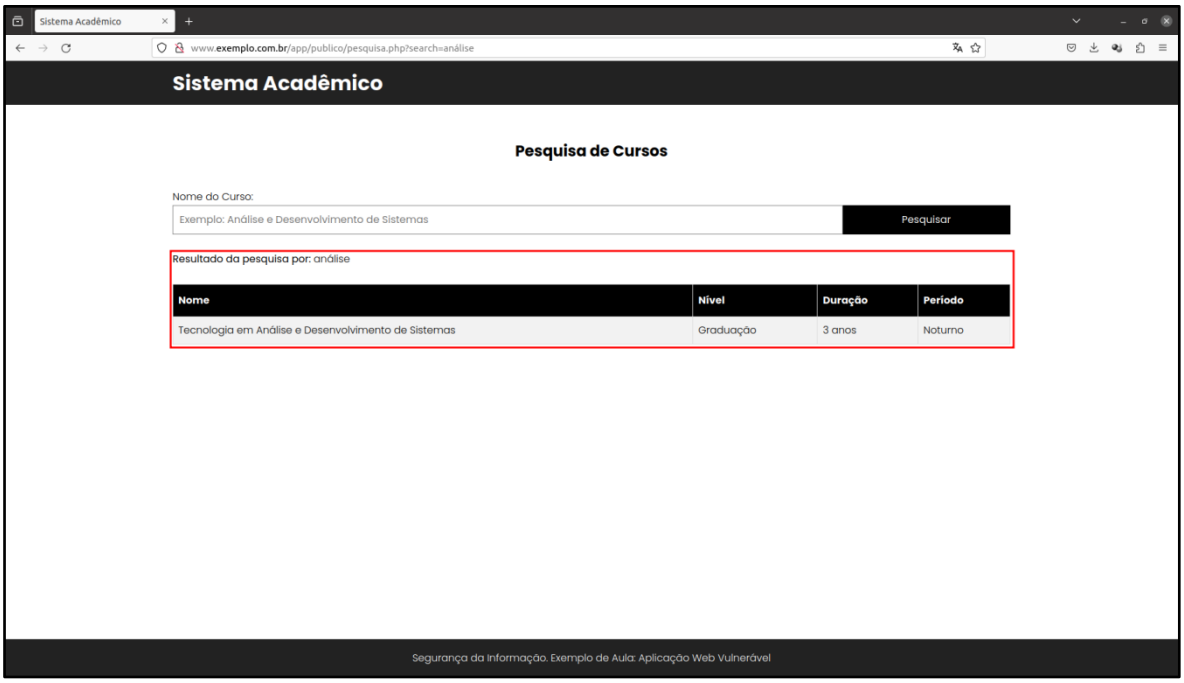


Figura 5

Utilização de uma entrada sem o uso de aspas simples para a pesquisa de cursos na aplicação Web alvo do ataque, com destaque para a entrada em questão, via "Resultado da pesquisapor:", e para o retorno gerado pela aplicação a partir da mesma.



Após detectar que a aplicação Web é vulnerável ao SQL Injection, o atacante pode prosseguir com o ataque. Para tal, uma possível abordagem a ser utilizada consiste na descoberta da quantidade de colunas da tabela atual. Essa descoberta é importante para, posteriormente, explorar as colunas identificadas para a obtenção de dados do banco de dados da aplicação. Nesse sentido, entre as formas possíveis, pode-se descobrir a quantidade de colunas da tabela atual por meio da ordenação de colunas. Na SQL, a cláusula ORDER BYpermite que o usuário ordene as tuplas no resultado de uma consulta pelos valores de um ou mais dos atributos que aparecem no resultado da consulta (Elmasri&Navathe, 2018, p. 179). Dessa forma, por meio de valores passados enquanto parâmetros para a ordenação e do comportamento da aplicação quanto aos mesmos, o atacante pode descobrir a quantidade de colunas da tabela atual. Nesse caso, se o número passado ao parâmetro for menor que o número total de colunas da tabela atual, o retorno da aplicação não deve mudar, uma vez que a consulta SQL é válida. Por outro lado, se o número passado for maior que o número total de colunas, um erro será gerado pela aplicação, nesse caso, pelo fato da coluna informada não existir na tabela atual.Por meio de tal abordagem, para a aplicação Web alvo do ataque, a partir dos retornos da aplicação ilustrados nas Figuras 6 e 7, pode-se observar que a tabela atual tem cinco colunas, uma vez que a ordenação pela coluna cinco manteve a consulta SQL válida (Figura 6) e a ordenação pela coluna seis implicou erro (Figura 7). Por meio de ambas as ilustrações, também pode-se observar que a pesquisa de cursos é realizada com base na entrada "-1". Essa abordagem é comumente utilizada para suprimir possíveis registros retornados pela aplicação, uma vez que, normalmente, não há registros negativos no banco de dados.

Figura 6

Possível abordagem do atacante para descobrir a quantidade de colunas da tabela atual, via ordenação pela coluna cinco, com destaque para a entrada utilizada,via "Resultado da pesquisapor:", e para o retorno gerado pela aplicação a partir da mesma.

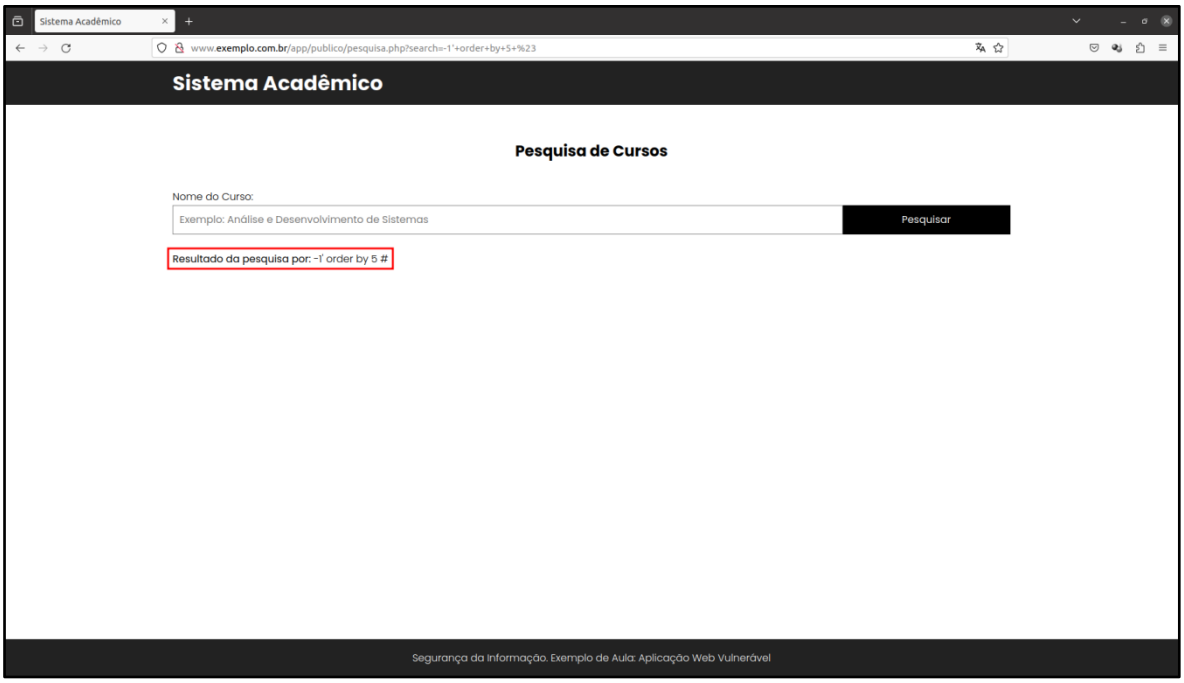
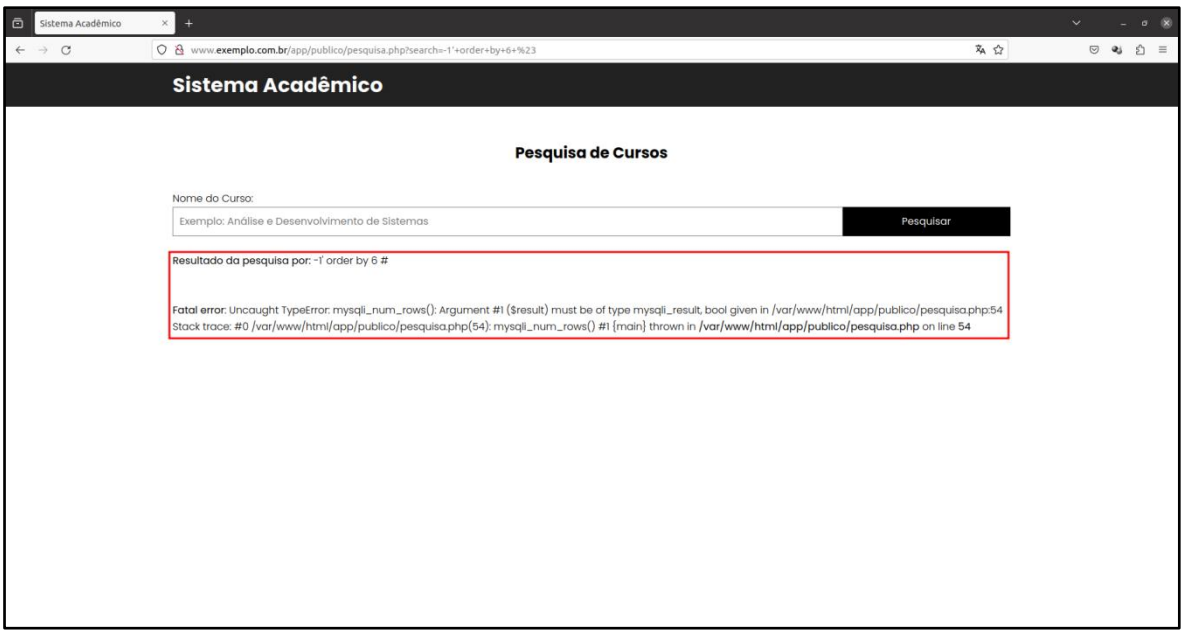


Figura 7

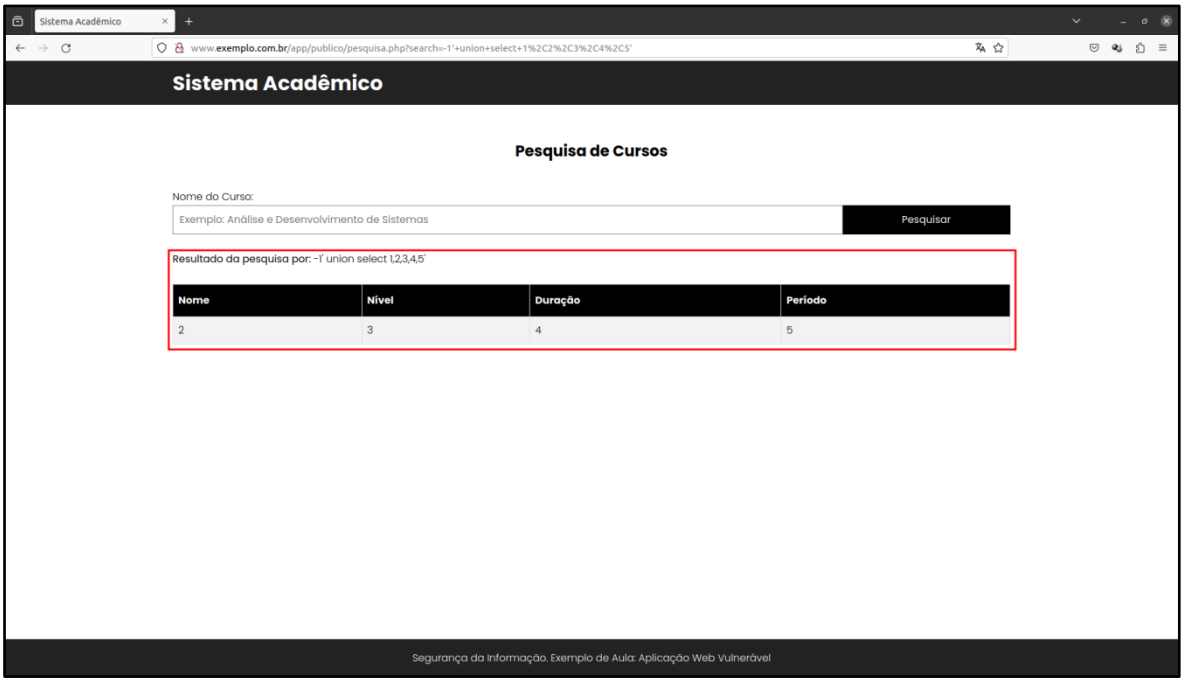
Possível abordagem do atacante para descobrir a quantidade de colunas da tabela atual, via ordenação pela coluna seis, com destaque para a entrada utilizada,via "Resultado da pesquisapor:", e para o erro gerado pela aplicação a partir da mesma.



Em seguida, após descobrir o número de colunas da tabela atual, uma possível abordagem a ser utilizada pelo atacante consiste na descoberta de quais colunas podem ser exploradas para a obtenção de dados do banco de dados da aplicação. Para tal, o atacante pode utilizar a operação UNION, que é uma operação de união de conjunto (Elmasri&Navathe, 2018, p. 176), para combinar o resultado da consulta padrão realizada pela aplicação com uma nova consulta definida por ele, por meio da instrução SELECT, que é uma instrução básica para recuperar informações de um banco de dados (Elmasri&Navathe, 2018, p. 169). Nesse caso, a instrução SELECT pode ter como base a utilização de identificadores para cada uma das colunas da tabela atual descobertas pelo atacante. A partir do resultado gerado pela aplicação, os identificadores exibidos permitem descobrir quais colunas da tabela atual podem ser exploradas para a obtenção de dados de seu banco de dados. Por meio de tal abordagem, para a aplicação Web alvo do ataque, as colunas dois, três, quatro e cinco da tabela atual podem ser exploradas para a obtenção de dados do banco de dados da aplicação, tal como ilustrado na Figura 8, com destaque para a entrada utilizada para tal descoberta e para o retorno gerado pela aplicação a partir da mesma.

Figura 8

Possívelabordagem do atacante para descobrir quais colunas da tabela atual podem ser exploradas para a obtenção de dados do banco de dados da aplicação, com destaque para a entrada utilizada, via "Resultado da pesquisapor:", e para o retorno gerado pela aplicação a partir da mesma.

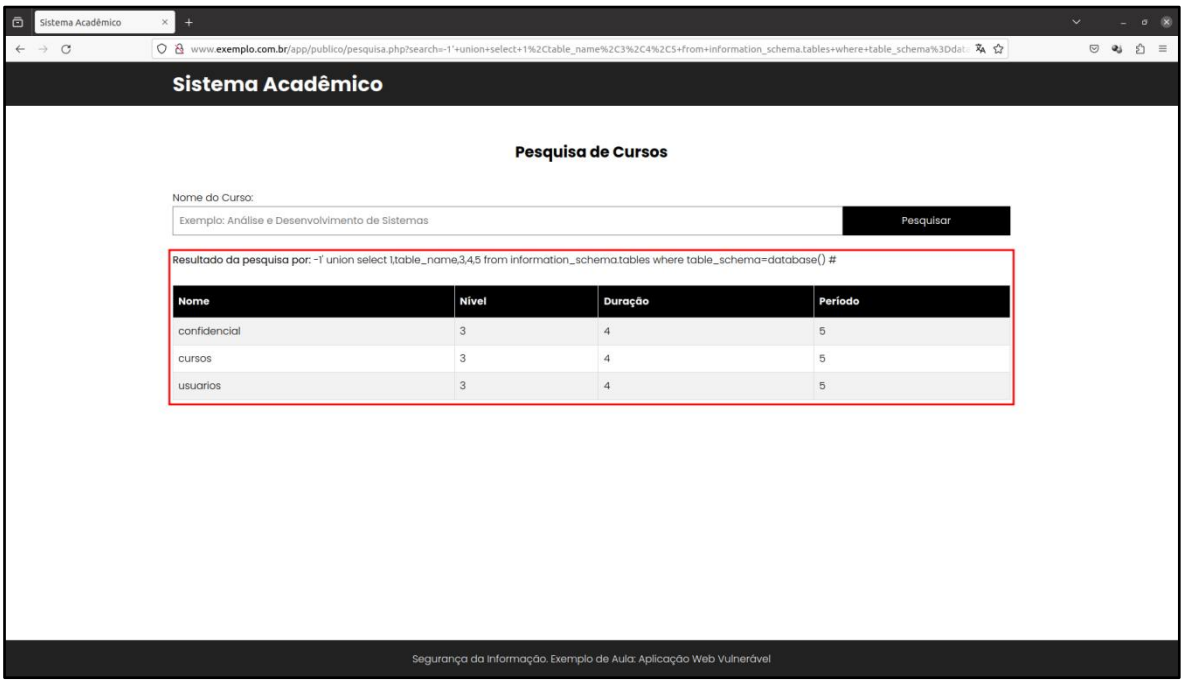


Na sequência, após descobrir quais colunas da tabela atual podem ser exploradas para a obtenção de dados do banco de dados da aplicação, uma possível abordagem a ser utilizada pelo atacante consiste na descoberta de quais são as tabelas de seu banco de dados. Para tal, o atacante pode realizar uma consulta ao INFORMATION_SCHEMA, que

provê informações sobre todos os esquemas no catálogo e todos os descritores de elemento nesses esquemas (Elmasri&Navathe, 2018, p. 162).Por meio de tal abordagem, para a aplicação Web alvo do ataque, pode-se observar que seu banco de dados tem as tabelas “confidencial”, “cursos” e “usuarios”, tal como ilustrado na Figura 9, com destaque para a entrada utilizada e para o retorno gerado pela aplicação a partir da mesma. Por meio de tal ilustração, também pode-se observar que as tabelas do banco de dados da aplicação são exibidas na coluna identificada previamente pelo número dois (Figura 8).

Figura 9

Possívelabordagem do atacante para descobrirquais são as tabelas do banco de dados da aplicação, com destaque para a entrada utilizada, via “Resultado da pesquisapor:”, e para o retorno gerado pela aplicação a partir da mesma.



Em seguida, após descobrir quais são as tabelas do banco de dados da aplicação, o atacante pode realizar consultas para cada uma das tabelas de interesse, inicialmente obtendo os campos da tabela por meio do INFORMATION_SCHEMA e, em seguida, consultando tais campos. Considerando o objetivo descrito na Seção 3, que é a obtenção de informações confidenciais do banco de dados da aplicação Web, mais especificamente, os *logins* e senhas de acesso para suas interfaces de acesso restrito, uma possível abordagem consiste na obtenção de informações da tabela “usuarios”.

Nesse contexto, a Figura 10 ilustra uma possível abordagem para a descoberta dos campos da tabela “usuarios” e a Figura 11 ilustra uma possível abordagem para a obtenção dos dados armazenados em tal tabela. Em ambas as ilustrações, as entradas utilizadas e o retorno gerado pela aplicação a partir das mesmas são apresentados em destaque.Para a aplicação Web alvo do ataque, pode-se observarque a tabela “usuarios” tem os campos “codigo”, “usuario”, “senha” e “tipo” (Figura 10), com dados armazenados pertinentes aos usuários “aluno” e “professor” (Figura 11).

Figura 10
Possível abordagem do atacante paradescobrir quais são os campos da tabela "usuarios" do banco de dados da aplicação, com destaque para a entrada utilizada, via "Resultado da pesquisapor:", e para o retorno gerado pela aplicação a partir da mesma.

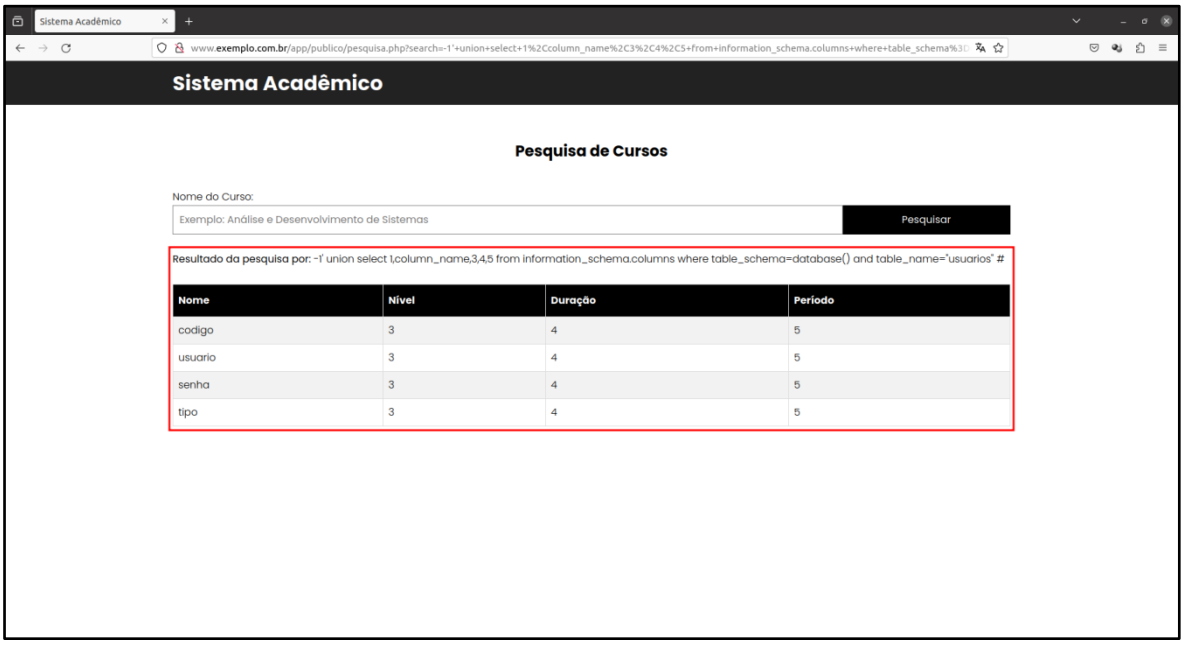


Figura 11
Possível abordagem do atacante para obter os dados armazenados na tabela "usuarios" do banco de dados da aplicação, com destaque para a entrada utilizada, via "Resultado da pesquisapor:", e para o retorno gerado pela aplicação a partir da mesma.

Por meio da Figura 11, pode-se observar que, entre os dados armazenados na tabela “usuarios” da aplicação Web alvo do ataque, há os hashes das senhas dos usuários “aluno” e “professor” que, inclusive, são iguais (“0f359740bd1cda994f8b55330c86d845”). Com base em talhash, de modo complementar ao SQL Injection, uma possível abordagem a ser utilizada pelo atacante consiste na tentativa de descoberta da senha referente ao mesmo. Para tal, duas possíveis soluções a serem utilizadas incluem hash-identifier[4], para identificação do hash, e John theRipper[5], para a descoberta da senha referente ao hash. Para o hash empregado para as senhas dos usuários “aluno” e “professor” (Figura 11), as Figuras 12 e 13 ilustram, respectivamente, sua identificação e a senha referente ao mesmo.

Figura 12

Possível abordagem do atacante, via solução hash-identifier, para identificar o hash empregado para a senha dos usuários “aluno” e “professor” (Figura 11), com destaque para o comando utilizado e para o retorno da solução quanto ao mesmo.

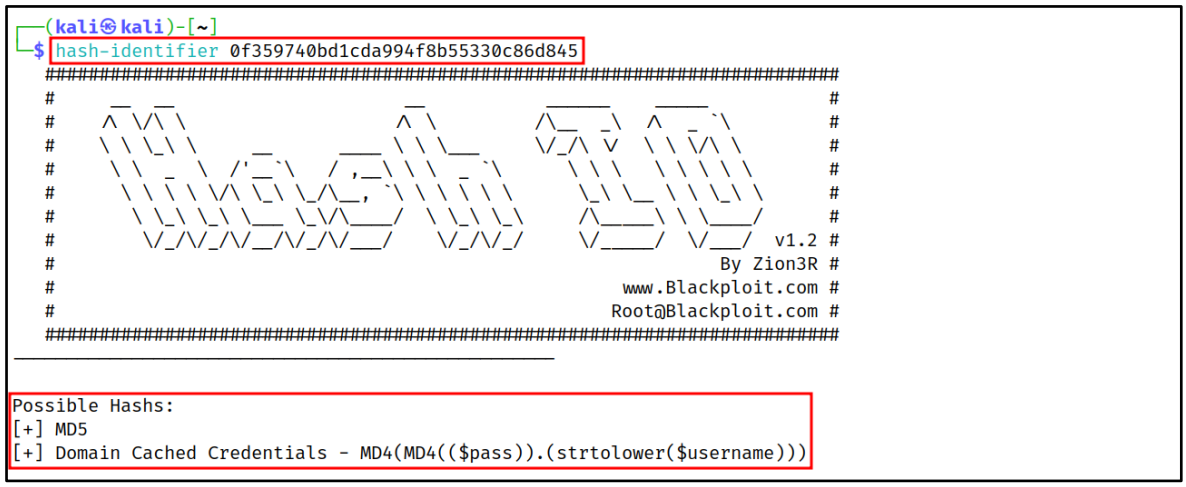
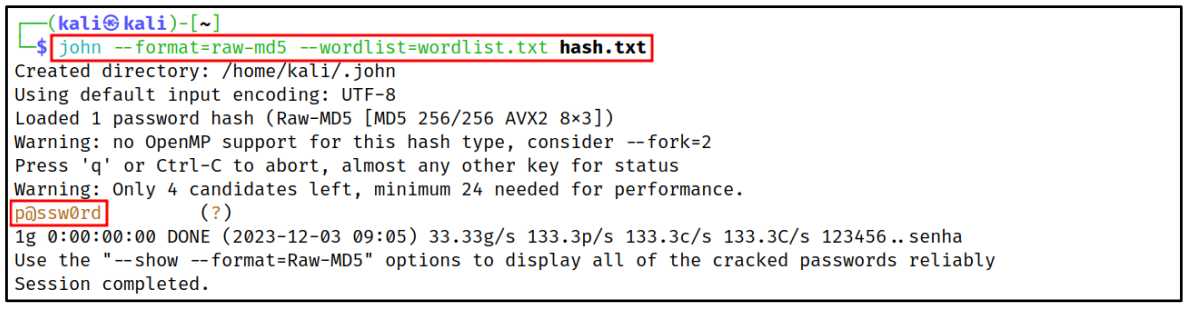


Figura 13

Possível abordagem do atacante,viasolução John theRipper, para descobrir a senha referente ao hash empregado para os usuários “aluno” e “professor” (Figura 11), com destaque para o comando utilizado e para o retorno da solução quanto ao mesmo.



[4]hash-identifier:https://www.kali.org/tools/hash-identifier/
[5]John theRipper:https://www.kali.org/tools/john/

Diante desse cenário, após descobrir os usuários e suas respectivas senhas de acesso ao sistema, o atacante poderia explorar a aplicação Web alvo do ataque utilizando a credencial de ambos, executando ações conforme as permissões e acessos concedidos para cada usuário. Adicionalmente, o atacante poderia utilizar as credenciais obtidas para realizar outros ataques, como ataques de força bruta(Diorioet al., 2019a) e/ou de phishing (Diorio et al., 2019b), por exemplo.

No âmbito do ataque de SQL Injection, após obter os dados de interesse do banco de dados da aplicação Web e com controle ao banco de dados em questão, o atacante poderia, por exemplo, adicionar, modificar e excluir registros, com efeitos relacionados podendo ser extremamente críticos, com possíveis consequências incluindo o vazamento de dados, o roubo/falsificação de identidade e a realização de fraudes, entre outros.

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo prático sobre ataques de SQL Injection. Para tal, com base em um cenário de referência, ataques relacionados foram reproduzidos para fins de experimentação e discussão.

Os experimentos e discussão foram baseados em um laboratório prático sobre segurança de aplicações Web, realizado com estudantes do terceiro ano de um curso de graduação em informática, em uma das aulas de segurança da informação, para auxiliá-los na compreensão de conceitos sobre SQL Injection. Essa discussão é importante, por exemplo, para conciliar teoria e prática em atividades de ensino relacionadas ao tema, bem como para que novas pesquisas e contribuições sejam realizadas no âmbito da segurança da informação, tais como para a prevenção e para a detecção dos ataques de SQL Injection.

Trabalhos futuros serão realizados para avaliar a percepção dos estudantes sobre o estudo prático descrito neste trabalho. Além disso, outros estudos práticos serão propostos, explorados e avaliados.

Referências

Chen, P., Zhao, M., Wang, J.,&Yu, H. (2019, August). Explorationandpracticeoftheexperimentteachingof web applicationsecuritycourse. In *2019 10th InternationalConferenceonInformation Technology in Medicine andEducation (ITME)* (pp. 381-384). IEEE.

Diorio, R. F., Serafim, E., Alves, K. R., & Meira, M. C. (2019a). Ataques de Força Bruta: Um Estudo Prático. In *2019 BrazilianTecnologySymposium (BTSym 2019)*.

Diorio, R. F., Serafim, E., Alves, K. R., & Meira, M. C. (2019b). A PracticalStudyonPhishingAttacks. In *16th CONTECSI - InternationalConferenceonInformation Systems and Technology Management*.

Elmasri, R., &Navathe, S. B. (2018). Sistemas de Banco de Dados (7ª ed.). Pearson Education do Brasil.

Halfond, W. G., Viegas, J.,&Orso, A. (2006, March). A classificationof SQL-injectionattacksandcountermeasures. In Proceedingsofthe IEEE internationalsymposiumonsecure software engineering (Vol. 1, pp. 13-15). IEEE.

Li, Q., Li, W., Wang, J., &Cheng, M. (2019). A SQL injectiondetectionmethodbasedonadaptivedeepforest. *IEEE Access, 7*, 145385-145394.

OWASP Foundation. (2023). OWASP Top Ten. https://owasp.org/www-project-top-ten/.

Ping, C., Jinshuang, W.,Lanjuan, Y., &Lin, P. (2020, September). SQL InjectionTeachingBasedonSQLi-labs. In *2020 IEEE 3rd InternationalConferenceonInformation Systems and Computer AidedEducation (ICISCAE)* (pp. 191-195). IEEE.

Vyamajala, S., Mohd, T. K.,&Javaid, A. (2018, May). A real-world implementationof SQL injectionattackusing open source tools for enhancedcybersecuritylearning. In *2018 IEEE InternationalConferenceonElectro/Information Technology (EIT)* (pp. 0198-0202). IEEE.

Wang, J.,& Liu, X. (2023, May). Researchon Software Security Basedon DVWA. In *2023 IEEE 3rd InternationalConferenceonElectronic Technology, Communication andInformation (ICETCI)* (pp. 38-42). IEEE.

Zivanic, S., Ruvceski, S., &Basicevic, I. (2022, May). Network securityeducation: SQL injectionattacks. In *2022 IEEE ZoomingInnovation in Consumer Technologies Conference (ZINC)* (pp. 77-80). IEEE.