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:

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 experiencia 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 Writting 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 confecção 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 confecção 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", 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:

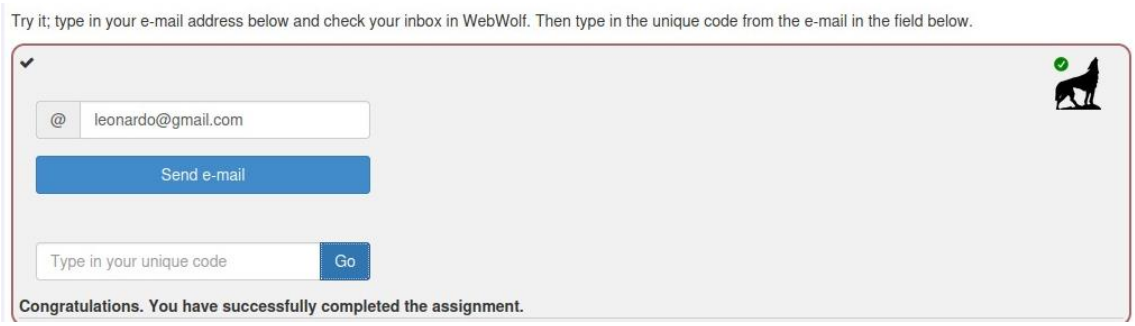
A SQL Injection é uma vulnerabilidade comum em aplicações web que ocorre quando dados não confiáveis são inseridos incorretamente em consultas SQL. Essa falha de segurança permite que um invasor execute comandos maliciosos no banco de dados, podendo manipular, modificar ou excluir dados, obter informações 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:

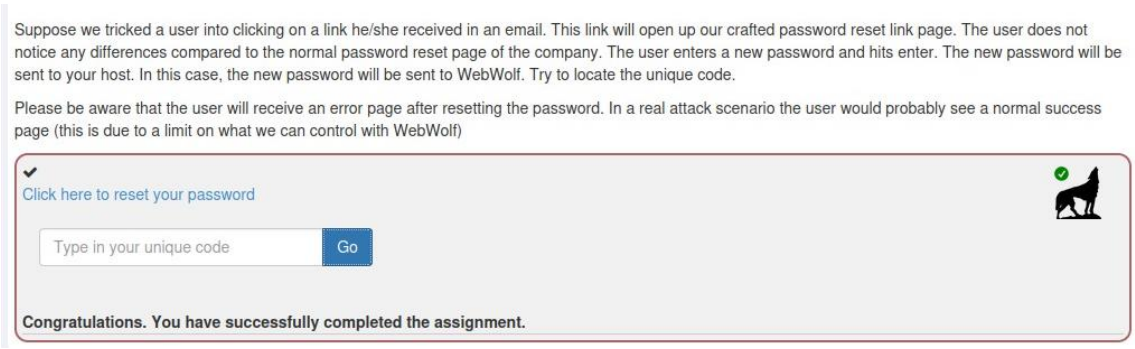
Introduction:

Atividade 1:



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:

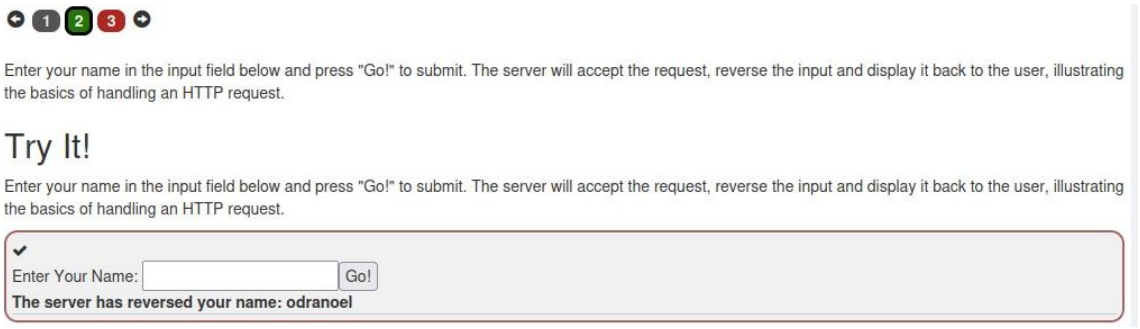


A atividade tinha como foco, demonstrar como links falsos de mudança de senha funcionam, o hyperlink presente na atividade redirecionava a uma página de alteração

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

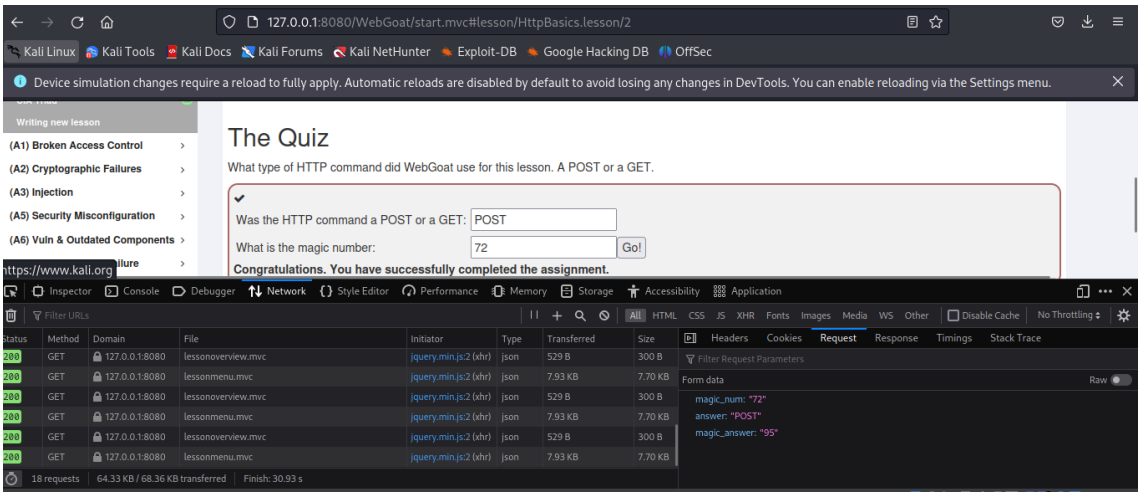
Básico HTTP:

Atividade 1:



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.

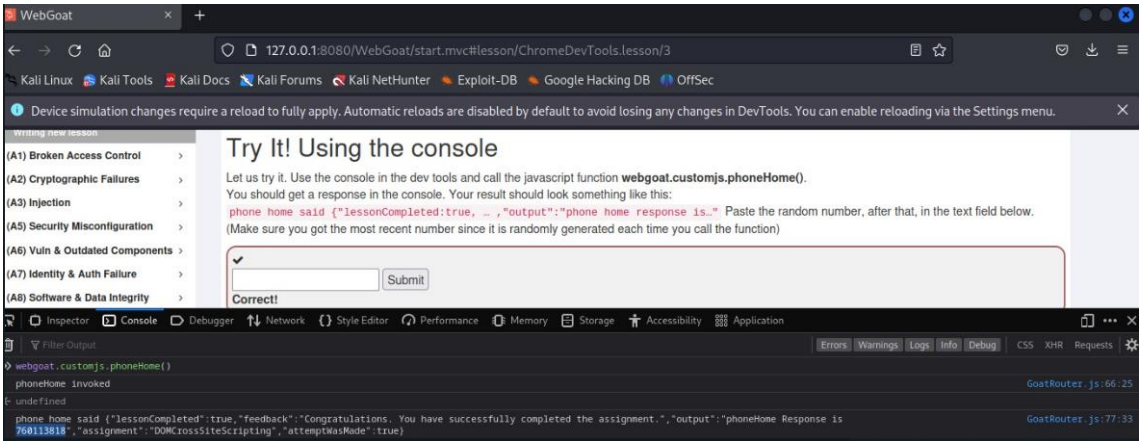
Atividade 2:



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 magico”.

Ferramentas de Desenvolvedor:

Atividade 1:



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:

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.
- [x] 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?

- [x] 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.

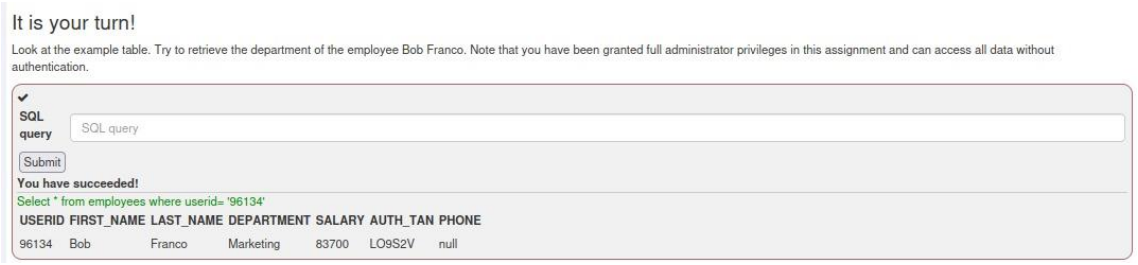
3. How could an intruder harm the security goal of availability?

- [ ] 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.
- [x] 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.
- [x] 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. Theres only a problem when confidentiality is harmed.

Atividade 2:



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

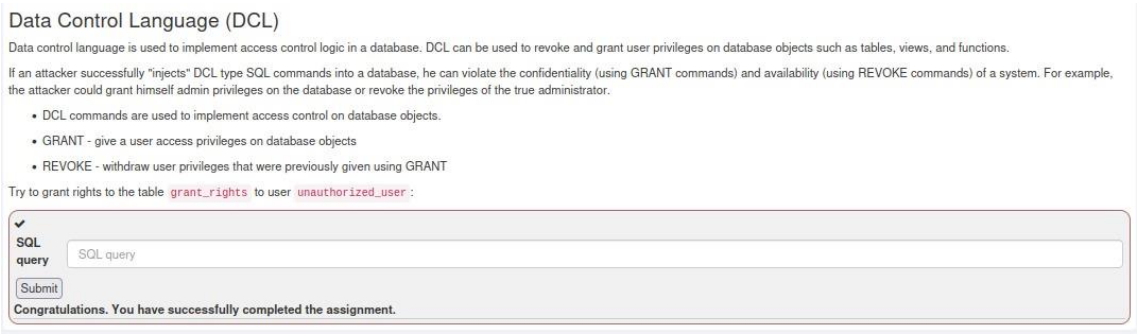
Atividade 3:

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

Atividade 4:

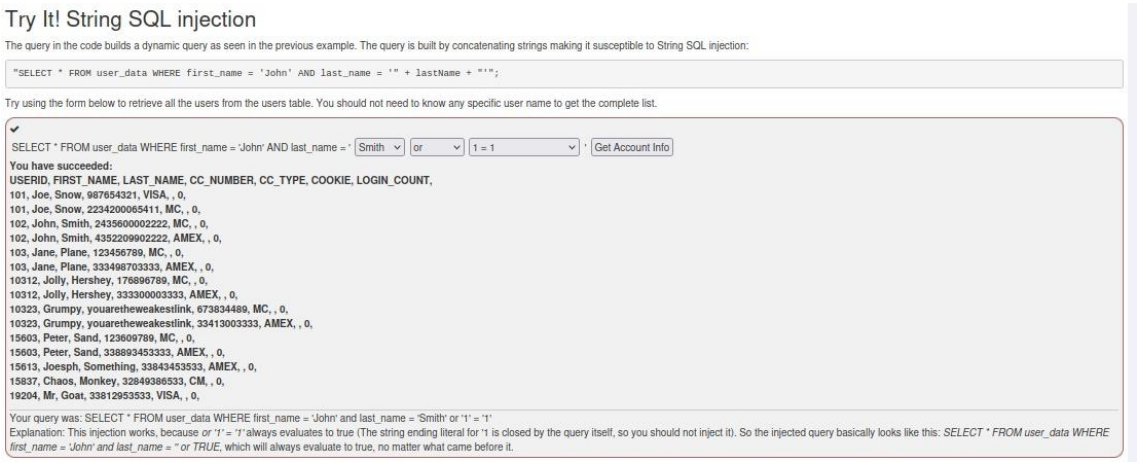
O objetivo da atividade era adicionar uma tabela chamado PHONE através do SQL e o comando alter table foi utilizado junto com as especificações exigidas da atividade.

Atividade 5:



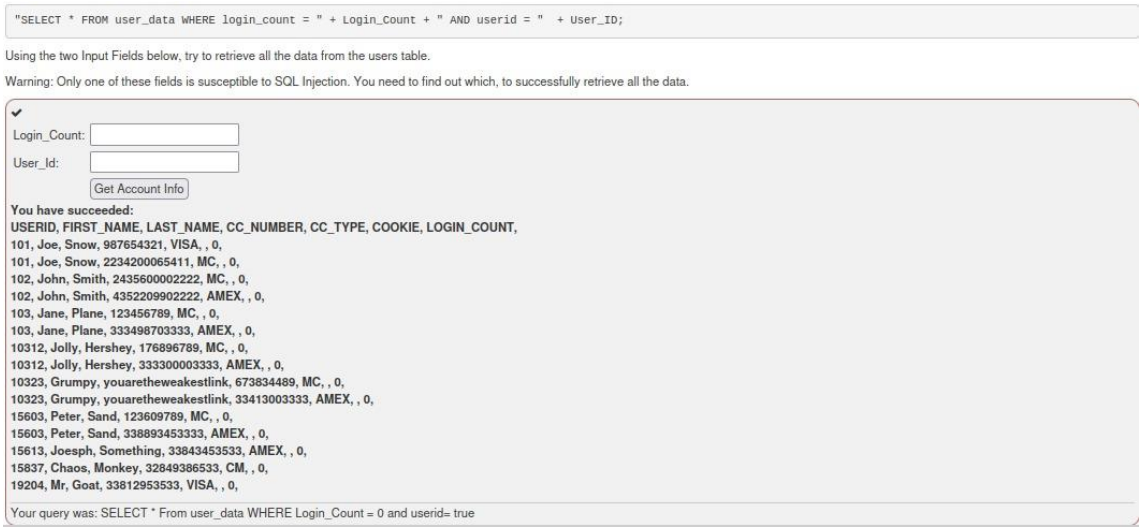
O objetivo da atividade foi garantir direitos para o usuário não autorizado, o comando utilizado foi “Grant All ON employees to unauthorized_user”.

Atividade 9:



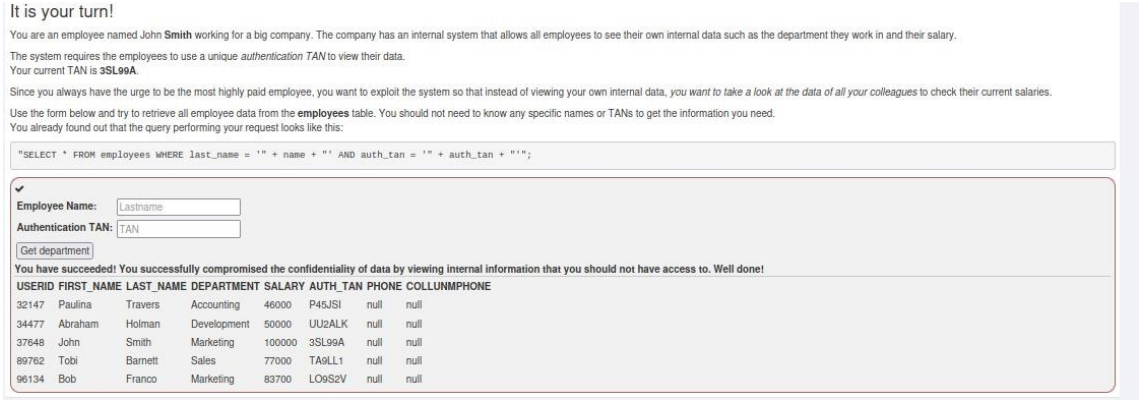
O objetivo da atividade foi descobrir a sintaxe correta da requisição, que por sua vez tem por intuito extrair informações determinadas do banco de dados.

Atividade 10:



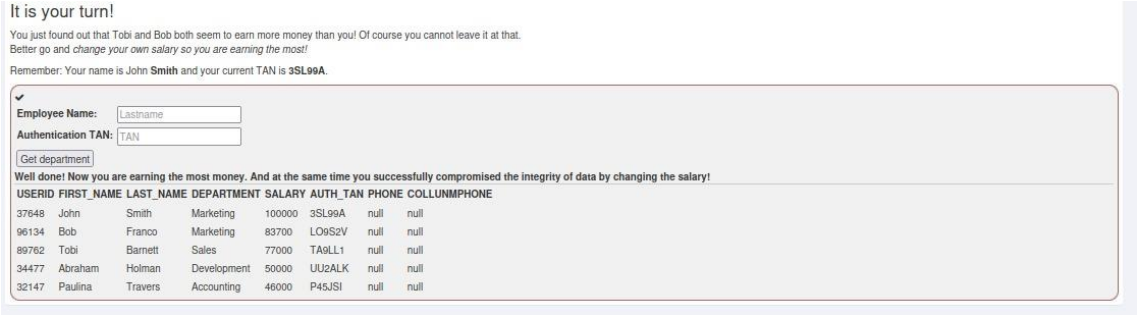
A atividade proposta teve como objetivo extrair os dados de uma tabela através do comando que foi especificado, os valores substituídos de login Count e User_Id foram, 0 e true respectivamente.

Atividade 11:



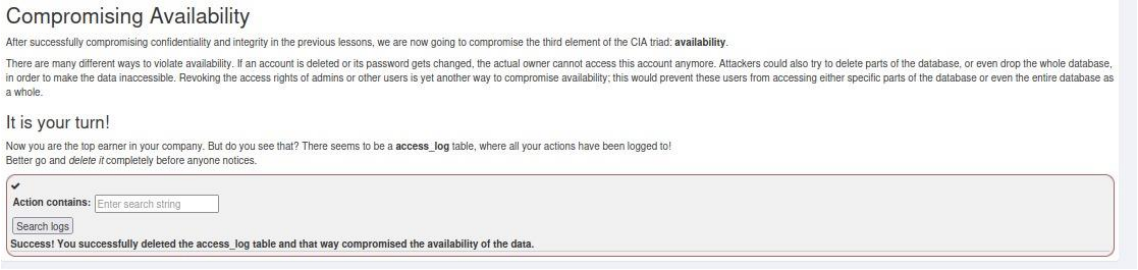
O objetivo desta atividade também era extrair informações da tabela de empregados, os valores a serem substituídos foram os nomes de empregados e o autenticador, Smith ' OR '1' ='1 respectivamente.

Atividade 12:



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:



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.