Victor Seretni Guimarães - Desafio Profissional Engenharia de Software

**1. INTRODUÇÃO**

Esse trabalho é uma forma de aprender e aplicar métodos de segurança da informação em sistemas online.

**1.1. CONTEXTO E JUSTIFICATIVA DO TRABALHO PRÁTICO**

O mundo moderno está cada vez mais dependente de sistemas digitais, seja para trabalho, entretenimento, educação, entre outros. Junto com essas possibilidades, vêm as responsabilidades – ao mesmo tempo cada vez mais informações pessoais estão sendo registradas em nuvem, como dados bancários, nomes, endereços, números de telefone. Com isso, é essencial que haja contramedidas contra as tentativas de invasão e falhas não intencionais para conter o vazamento desses dados sensíveis. Nesse trabalho, está sendo configurado um ambiente virtual, para testar vulnerabilidades em um ambiente controlado. Ao invés de fazer diretamente na máquina, vamos hospedar uma segunda máquina onde serão realizados os testes. Para isso, será usado o WebGoat, uma ferramenta de  
treinamento para prática de testes de penetração em aplicações web, por meio do Oracle VM VirtualBox, que servirá de programa intermediário, dentro do Kali Linux, que é o aplicativo mais adequado a situação.

**1.2. OBJETIVOS**

Os objetivos desse trabalho são estudar e aplicar sobre brechas na segurança de sistemas online, quais ataques podem ser feito com elas e o que pode ser feito para evitar tais invasões.

**1.3. METODOLOGIA**

Serão analisados tanto o processo back end por trás das vulnerabilidades como os resultados front end, para conseguir uma melhor visão do que realmente acontece com sistemas online.

**2. AMBIENTE VIRTUAL**

**2.1. INSTALAÇÃO E CONFIGURAÇÃO DO VIRTUALBOX**

Pelo link <https://www.virtualbox.org/>, baixei a versão mais recente (7.0.8) para Windows.

**2.2. INSTALAÇÃO E CONFIGURAÇÃO DO LINUX NA MÁQUINA VIRTUAL**

Pelo link <https://www.kali.org/get-kali/#kali-platforms>, baixei a máquina virtual do Kali para VirtualBox. Tive problemas no download, resolvidos após desativar temporariamente o antivírus do Google Chrome e do Windows, visto que o aplicativo é vulnerável de propósito.

**2.3. INSTALAÇÃO E CONFIGURAÇÃO DO WEBGOAT**

Após instalar o Java Runtime Environment usando o comando no console sudo apt-get install default-jre, baixei o WebGoat com o comando wget <https://github.com/WebGoat/WebGoat/releases/download/v2023.4/webgoat-2023.4.jar> . Acessei o link gerado e criei uma conta.

**3. VISÃO GERAL DO WEBGOAT**  
**3.1. DESCRIÇÃO E FUNCIONALIDADES DO WEBGOAT**

WebGoat é uma ferramenta educacional que provê um sistema propositalmente vulnerável, guiando o usuário a como acessar tais falhas e o que pode ser feito com elas.  
**3.2. COMO ACESSAR E NAVEGAR NO WEBGOAT**

Após a instalação, é gerado um link, que no meu caso foi <http://127.0.0.1:8080/WebGoat>, diferente do documentado no guia do Trello. Então, acessei usando o Firefox como uma página na web.

**4. PRÁTICAS COMUNS DE SEGURANÇA EM APLICAÇÕES WEB**  
**4.1. CONCEITOS BÁSICOS DE SEGURANÇA EM APLICAÇÕES WEB**

Navegadores Web se comunicam com servidores Web usando o Hyper Text Transfer Protocol (HTTP, ou Protocolo de Transferência de Hipertextos). Qualquer interação em uma página web, como clicar em um link, enviar respostas de um formulário ou pesquisar uma palavra, o browser envia uma requisição HTTP para o servidor. Nela, estão inclusos:

Uma URL (Uniform Resource Locator, ou o endereço do site) que identifica o servidor e o recurso de destino por exemplo, um arquivo HTML, um ponto de dados no servidor ou uma ferramenta a ser usada).

Um método que define a ação necessária (por exemplo, para obter um arquivo ou para salvar ou atualizar alguns dados). Os diferentes métodos e suas ações associadas estão listados abaixo :

Informações adicionais podem ser codificadas com a solicitação(por exemplo, dados de formulário HTML). As informações podem ser codificadas como :

Paramêtros de URL : Solicitações GET codificam dados na URL enviada ao servidor, adicionando pares de nome/valor ao final dela— por exemplo, http://mysite.com?name=Fred&age=11. Você sempre tem um ponto de interrogação (?) separando o resto da URL dos paramêtros de URL, um sinal de igual (=) separando cada nome de seu valor associado, e um "E" comercial (&) separando cada par. Os paramêtros URL são inerentemente inseguros, pois podem ser alterados pelos usuários e reenviados. Como resultado, os parâmetros de URL/ solicitações GET não são usados para solicitações que atualizam dados no servidor.

Cookies do lado do cliente. Os cookies contêm dados de sessão sobre o cliente, podendo usá-los para determinar o estado de sua sessão e permissões aos recursos.

**4.2. IDENTIFICAÇÃO DE VULNERABILIDADES COMUNS EM**  
**APLICAÇÕES WEB**

Um tipo de ataque comum é o XSS, uma classe de ataques em que um invasor injeta scripts (roteiros ou instruções) da parte do cliente através do site nos navegadores de outros usuários. Com o código injetado chegando ao navegador a partir do site, o código é considerado como confiável pelo servidor, podendo fazer coisas como enviar o cookie de autorização do site do usuário ao invasor. Então ele pode fazer login em um site como se fosse o usuário, tendo acesso a dados sensíveis.  
**4.3. BOAS PRÁTICAS PARA MITIGAÇÃO DE VULNERABILIDADES EM**  
**APLICAÇÕES WEB**

A melhor defesa contra as vulnerabilidades do XSS é remover ou desativar qualquer marcação que possa conter instruções para executar o código. Para HTML, isso inclui elementos, como <script>, <object>, <embed> e <link> .

**4.4. SQL INJECTION**

As vulnerabilidades de injeção de SQL permitem que usuários mal-intencionados executem código SQL no banco de dados de um servidor, permitindo a interação mesmo que antes proibida do cliente com o servidor, independentemente das permissões do usuário. Um ataque de injeção bem-sucedido pode falsificar identidades, criar novas identidades com direitos de administração, acessar todos os dados no servidor ou destruir/modificar os dados para torná-los inutilizáveis.  
**5. CONCLUSÃO**

Uma aplicação web não pode confiar em nenhum dado do navegador. Todos os dados do usuário devem ser limpos antes de serem exibidos ou usados em consultas SQL e chamadas do sistema de arquivos.

**5.1. SÍNTESE DOS RESULTADOS E CONCLUSÕES DO TRABALHO**  
**PRÁTICO**

É de grande aprendizado ver na prática o que pode ser feito com um sistema web, que é utilizado todos os dias por inúmeras pessoas, e que possuem tantas informações sensíveis. É de suma importância aprender o que pode acontecer e o que pode ser feito para que nada não intencional aconteça.

**5.2. LIMITAÇÕES DO TRABALHO E SUGESTÕES PARA TRABALHOS**  
**FUTUROS**

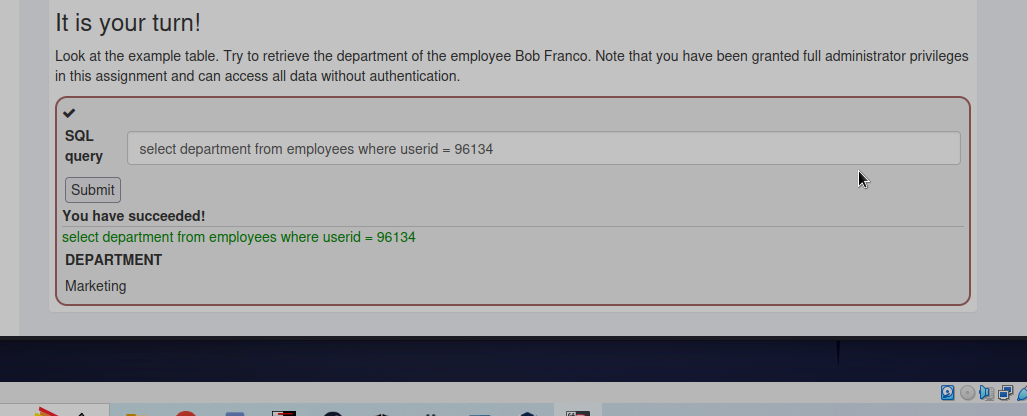
Sessões práticas em aula com a interação da turma seria melhor do que o trabalho individual e a longa distância.  
**6. REFERÊNCIAS BIBLIOGRÁFICAS**

<https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web>

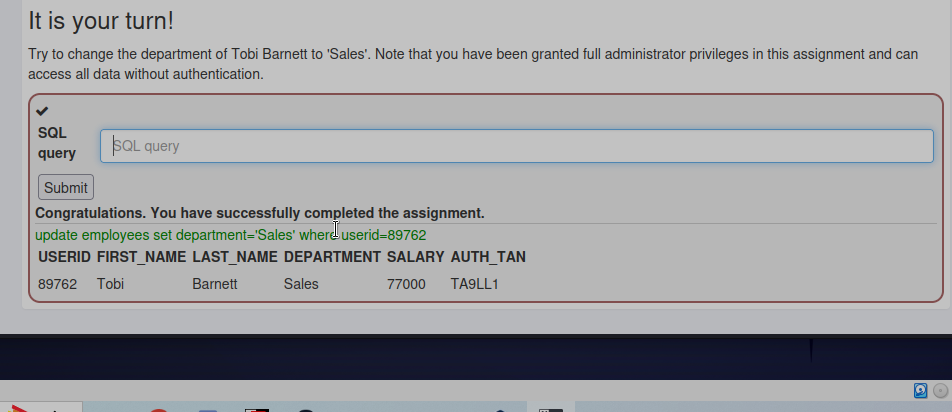
https://www.monitoratec.com.br/blog/metodologia-de-testes-de-software/

**7. RESULTADOS DA PRÁTICA NA AULA A3 – SQL INJECTION**

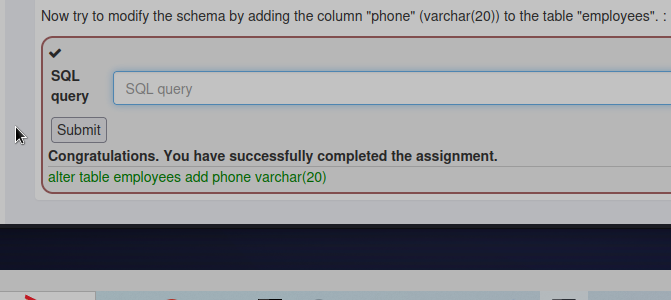
**7.1**



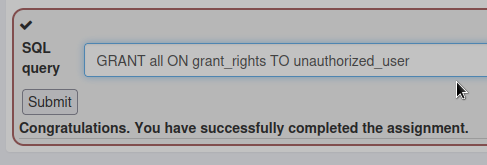
**7.2**



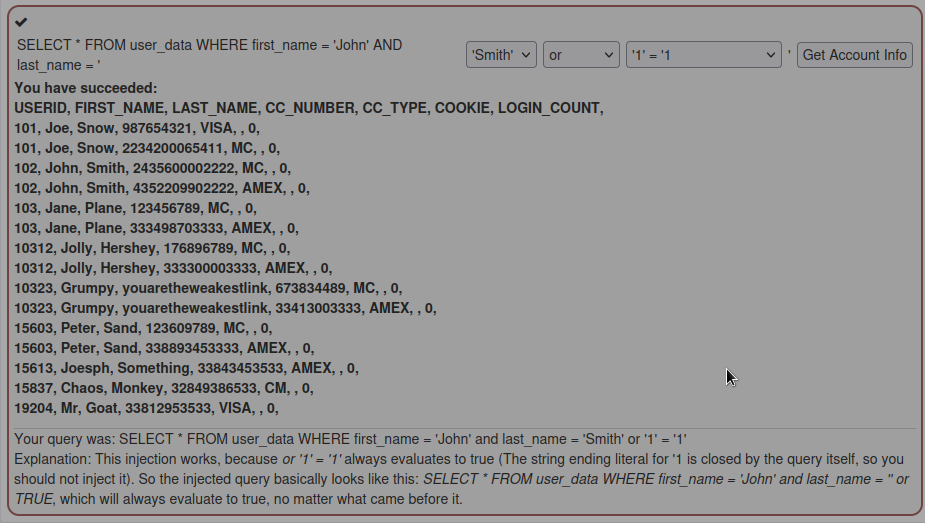
**7.3**



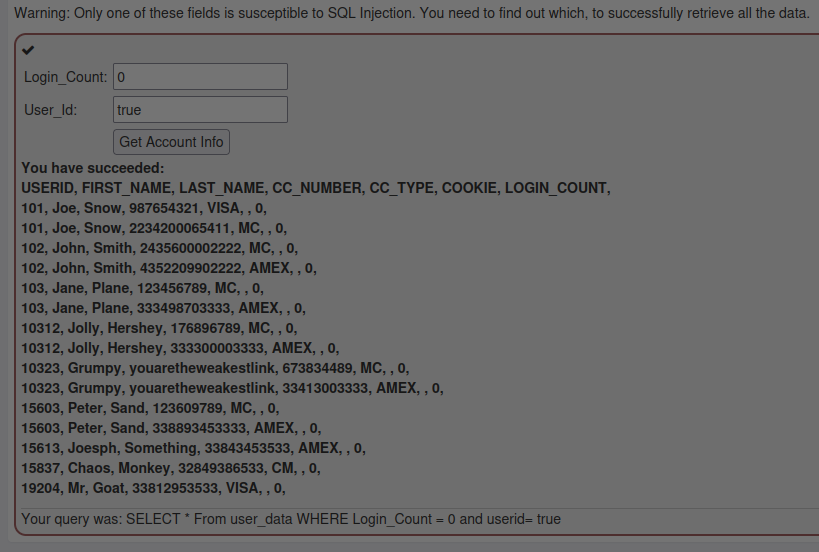
**7.4**



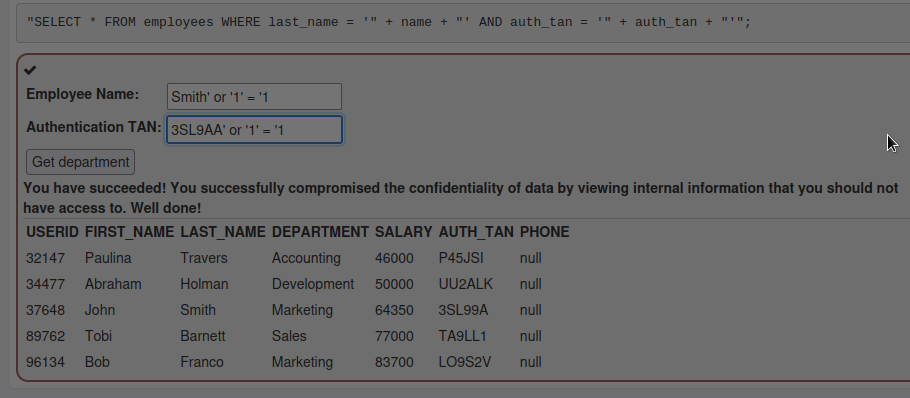
**7.5**



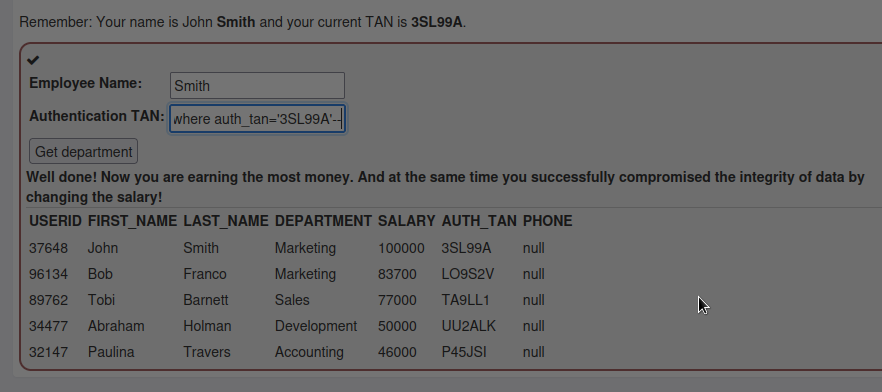
**7.6**



**7.7**



**7.8**



**7.9**

