INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

SEGURANÇA INFORMÁTICA

LICENCIATURA ENGENHARIA INFORMÁTICA E COMPUTADORES

SÉRIE DE EXERCÍCIOS
Fase 3

Autores:
43552 - Samuel Costa
43320 - André Mendes

Docente:
José Simão

05 de Dezembro de 2019

Conteúdo

Introdução

O trabalho realizado para este fase pretende que os temas desenvolvidos durante as aulas sejam postos em prática. Para esta fase os exercícios focaram essencialmente a segunda parte da matéria.

- Autenticação baseadas em passwords
- Gestão de Identidade em Aplicações Web
- Modelos de Controlo de Acesso

Neste trabalho prático pretendemos responder aos vários exercícios propostos e implementar uma demonstração dos pontos referidos.

1 Exercício 1

Ambos são elementos do sistema de controlo de acessos (autorização). O modelo define as regras gerais seguidas no âmbito da autorização, enquanto as políticas são a concretização do modelo. No contexto do Sistema Operativo, identificamos a matriz de acesso como um modelo de controlo de acessos, tendo como sua concretização (política) a lista de controlo de acessos.

2 Exercício 2

2.1

Uma vez que o modelo RBAC1 comporta a hierarquia de papeis, é possível a existência de uma sessão com um user u com o role activo r, sem que (u,r) esteja na relação UA, se o utilizador tiver o role r1 e r <= r1.

2.2

O princípio de privilégios mínimos indica que por omissão se deve restringir os privilégios de acesso. Quando aplicado aos utilizadores, quer dizer que todos os utilizadores devem sempre utilizar o sistema com o mínimo de privilégios possível. Como princípio geral, o desenho do sistema deve conceder as permissões mínimas aos roles para executarem as ações necessárias, proibindo por omissão, em vez de proibir como excepção.

3 Exercício 3

Em CVE-2019-9766 trata-se de uma vulnerabilidade buffer-overflow baseada no stack. Esta vulnerabilidade é causada pela coincidência no stack de dados do programa (variáveis locais à função) e dados de controlo de fluxo (endereço de retorno para função chamadora e endereço do início da frame anterior). Se a escrita numa variável local exceder os limites dos dados do programa e esmagar os dados de controlo de fluxo, quando a função retornar pode ser executado código arbitrário. Neste caso, é a operação de conversão de um ficheiro que pode ser explorada, tirando partido da dimensão da zona do stack reservada a dados locais da função.

4 Exercício 4

Tanto a vulnerabilidade buffer overflow como cross-site scripting dizem respeito à execução de código arbitrário.

5 Exercício 5

Se o atacante conseguisse explorar a vulnerabilidade CSRF da rota /googlecallback, fazendo a vítima efectuar um pedido GET para essa rota com um state válido e o seu code, a aplicação tentaria trocar o code pelo access token do atacante, no que seria bem sucedida. Se a vítima prosseguisse o fluxo da aplicação, estaria a guardar os seus issues em tasks da conta do atacante.

6 Exercício 6

6.1

Foi iniciada uma instância com o id 472900268124152302695985496522790539409. A aplicação está disponível em https://google-gruyere.appspot.com/472900268124152302695985496522790539409/. Foi criada uma conta na app Gruyere com o utilizador LI51N-SI1920iG01 e a password gruyerepass.

6.2

Upload XSS - Foi elaborado o documento html upload_xss2.html que contem um elemento script que executa a função alert com as cookies do utilizador. Depois de carregar o ficheiro para o servidor, acedeu-se ao url fornecido. Em resultado foi realizado pelo browser um pedido GET para uma rota dentro do mesmo domínio, o que implicou que o browser envia as cookies que possui para esse domínio, fazendo com que a informação presente nas cookies do utilizador fosse acessível ao documento carregado.

Reflected XSS - Foi iniciada uma instância do browser com a flag –disable-xss-auditor. Foi elaborado o seguinte url:
https://google-gruyere.appspot.com/472900268124152302695985496522790539409/<script>
alert(document.cookie);</script>
, que injecta no documento um elemento script que executa código arbitrário, tirando partido do facto de a aplicação encaminhar directamente os dados de entrada na mensagem que exibe ao utilizador.

Stored XSS - A vulnerabilidade foi explorada criando um snippet com o seguinte conteúdo: <a onmouseover="alert(document.cookie)"href="#»exploit!, que regista um evento