



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

1	Exercício 1	3
2	Exercício 2	3
3	Exercício 3	3
4	Exercício 4	3
5	Exercício 5	3
6	Exercício 6	4
7	Exercício 7	4

Introdução

O trabalho realizado para esta 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.

- Modelos de Controlo de Acesso
- Vulnerabilidades em aplicações nativas, base de dados e web

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 papéis, 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 $r \leq r1$.

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 gruyere-repass.

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>](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 exhibe ao utilizador.

Stored XSS - A vulnerabilidade foi explorada criando um snippet com o seguinte conteúdo: `exploit!`, que regista um evento quando o utilizador passa o rato por cima do snippet, mostrando em janela de diálogo os cookies do utilizador.

6.3

Como já vimos, a app Gruyere é vulnerável a ataques do tipo XSS (stored, upload e reflected). Para receber as cookies de uma vítima na sua app, o atacante necessita de confeccionar um script que, sem o conhecimento da vítima, faça um pedido GET para a sua aplicação com as cookies na query string. Como as cookies só estão disponíveis em documentos da mesma origem (SOP), o atacante precisa de injectar esse script numa página do domínio da aplicação Gruyere, o que já vimos que é possível.

O ataque foi executado, tendo sido verificada a existência de um documento na BD na rota `siserie3-result` da app do atacante.

7 Exercício 7

7.1

Autenticou-se o utilizador Alice na aplicação Elgg. Efetuou-se um pedido GET para a rota `/profile/:user` com `user=alice`. Constatou-se que nesse pedido foram enviados os headers `referer` e `cookie` com uma cookie com o nome `Elgg`. De seguida, foi efectuado um pedido POST para a rota `/action/friends/add`. Além de ser enviado o header `cookie` com a mesma cookie do pedido GET, foram enviados na query string os seguintes parâmetros:

- `__elgg_token`

- __elgg_ts
- friend

O parâmetro friend é único por utilizador e não é alterado. Os parâmetros __elgg_token e __elgg_ts (timestamp) são gerados por cada pedido do formulário de edição do perfil.

7.2

Foi configurado o adaptador de rede para funcionar no modo bridge. A figura 1 apresenta o ping feito com sucesso à VM-elgg e à VM-attacker.

Configurou-se a aplicação do atacante em VM-attacker. Colocou-se o conteúdo do ficheiro anexo attacker.html em index.html. A partir deste momento, a aplicação que responde por 192.168.1.122 envia o ficheiro malicioso.

Copiou-se a aplicação Elgg para a pasta /var/www/home da VM-elgg.

Editou-se o documento hosts no host para que os endereços da app elgg e da app do atacante passem a ser resolvidos localmente. Adicionaram-se as seguintes duas linhas ao ficheiro:

- 192.168.1.122 www.csrfabbattacker.com
- 192.168.1.123 www.csrfabelgg.com

Acedeu-se a partir da máquina host à aplicação Elgg para realizar login como o utilizador Alice. Abriu-se uma outra janela do browser para se aceder à aplicação atacante. A figura 2 mostra este passo do ataque. Em seguida, acedeu-se ao relatório de actividade da aplicação Elgg, de onde consta o efeito do ataque (Figura 3).

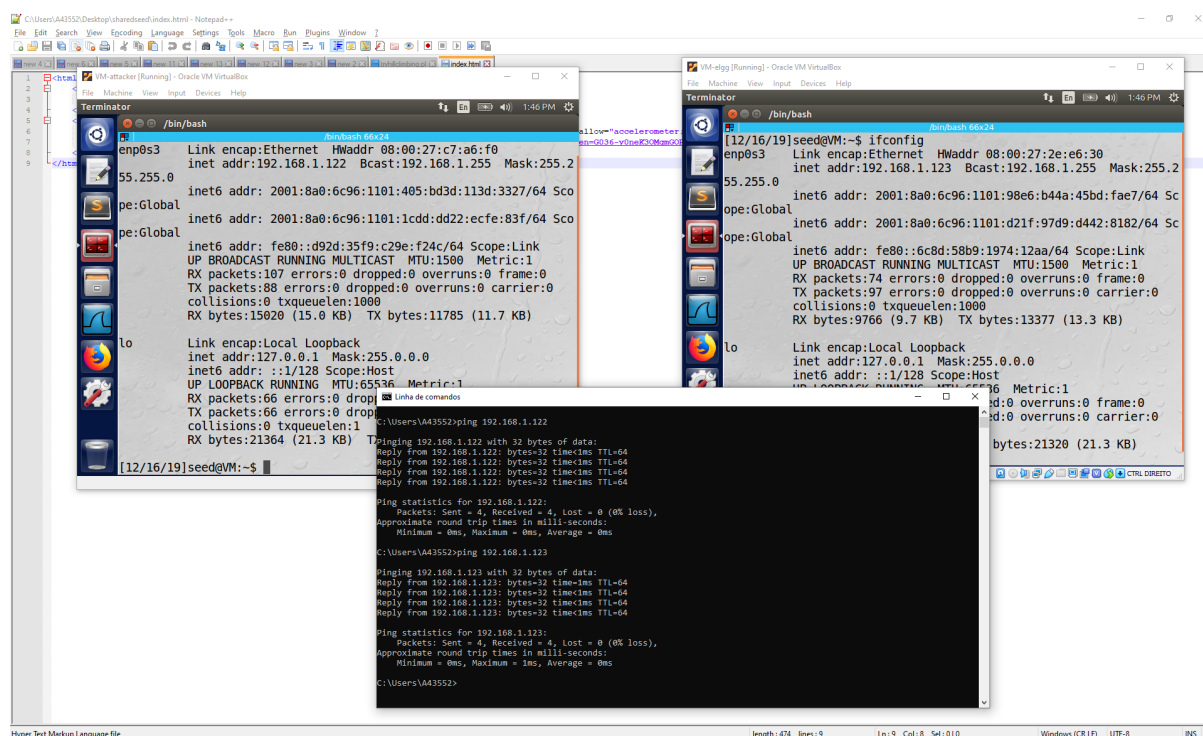


Figura 1: Endereços IP das VM's e estabelecimento do ping desde a máquina host

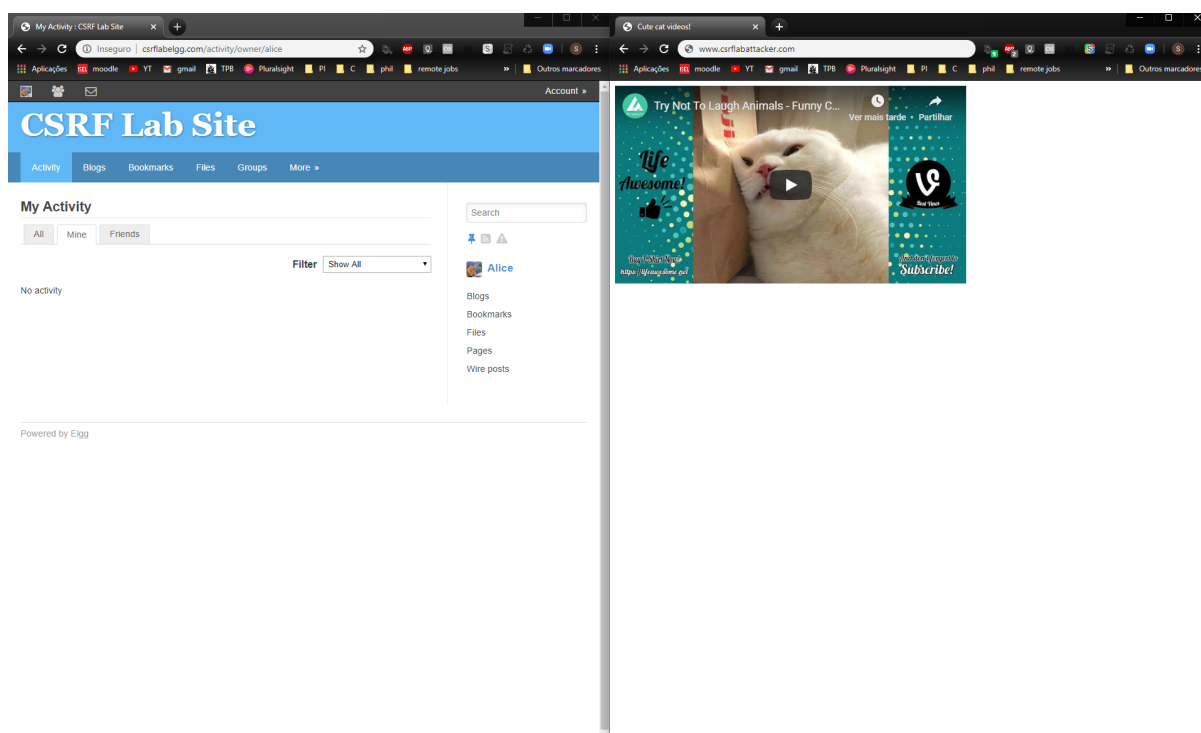


Figura 2: Vítima visita a aplicação atacante

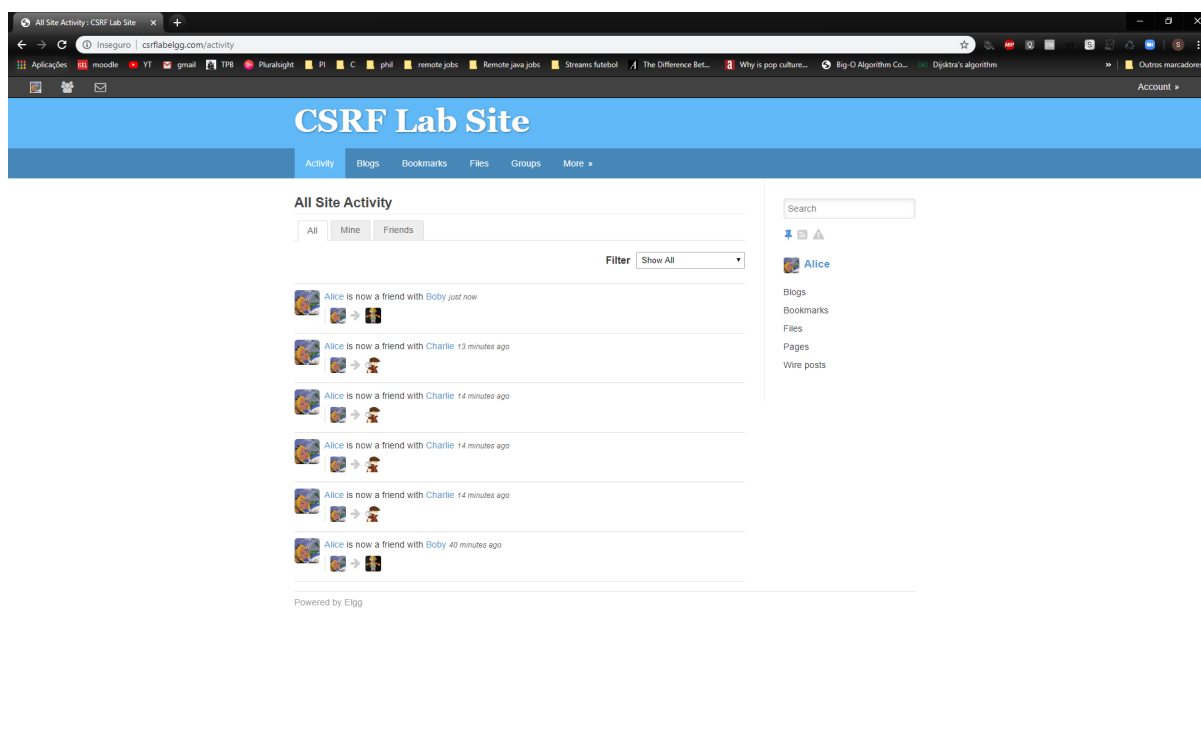


Figura 3: Como efeito do ataque, Alice é agora amiga de Bobby