Segurança Informática

Série 3

Realizado por:

Diogo Leandro nº 44868

João Barata nº 44857

Questão 1:

1. No contexto dos sistemas de controlo de acessos, distinga entre modelos e políticas. Dê um exemplo no âmbito de um sistema operativo.

R: As políticas definem as regras do controlo de acessos e os modelos formalizam a forma de aplicação dessas políticas. Um bom exemplo disso seria as contas do Windows com acesso de administrador e as ações dentro do sistema operativo cuja sua execução apenas é permitida ao administrador, sendo a política a existência de ações apenas permitidas ao administrador e o modelo a rejeição de um determinado pedido feito por um utilizador não administrador a uma ação apenas permitida a administradores.

Questão 2:

1. Considere os modelos RBAC:
   1. No RBAC1, é possível existir uma sessão associada ao utilizador **U** e com o role **R** ativo, sem que **(u, r)** esteja na relação user assignment (UA)?

R: No modelo RBAC1 as roles estão estruturadas de forma hierárquica ou seja uma role mais acima na hierarquia herda todas as permissões das roles abaixo da mesma.

Logo,se for associada uma determinada role na relação user assignment a um user,o mesmo terá como roles activas todas as roles abaixo da hierarquia da sua role associada,o que significa que é possivel existir uma sessão associada a um user com um determinado role sem que a mesma esteja na relação UA.

* 1. Qual a relação entre o princípio de privilégios mínimos e o conceito de sessão?

R: O princípio de privilégios mínimos sugere que apenas os papéis necessários para ação corrente devem ser ativados enquanto que o conceito de sessão diz que assim que um utilizador inicia sessão, o processo iniciado deriva as permissões desse utilizador a partir dos papéis ou roles designados a esse utilizador.

Questão 3:

1. Explique sucintamente de que forma a vulnerabilidade CVE-2019-9766 pode levar a que um atacante consiga executar código arbitrário no computador da vítima.

R: Esta vulnerabilidade permite que fique inserido no stack um pointer para determinado código que o atacante inseriu em determinado sítio na aplicação, quando o stack pointer apontar para esse código este será executado.

Questão 4:

1. O que têm em comum as vulnerabilidades de buffer overflow e cross-site scripting (XSS)?

R: Ambos os ataques consistem na possibilidade de o atacante introduzir código que a aplicação vai correr posteriormente, código este que pode ser prejudicial para o funcionamento da aplicação

Questão 5:

1. Admita que a aplicação cliente que fez na segunda série tem uma vulnerabilidade de cross site request forgery no URL de callback da Google. Dê um exemplo de como essa vulnerabilidade pode ser explorada por um atacante.

R: Se essa vulnerabilidade se verificasse o atacante teria acesso a todas as informações acerca da conta do utilizador que iniciou sessão, o atacante poderia explorar esta vulnerabilidade acedendo aos cookies do utilizador e realizar o login,por exemplo,na respectiva conta bancária e realizar uma transferência para a conta do atacante sem o consentimento do utilizador.

6.1. )

<https://google-gruyere.appspot.com/609646299901679235229890844824895027074/>

6.2)

Upload file XSS-

<https://google-gruyere.appspot.com/609646299901679235229890844824895027074/grupo17SI/File%20Upload%20Xss.html>

Reflect XSS-<https://google-gruyere.appspot.com/609646299901679235229890844824895027074/%3Cscript%3Ealert(document.cookie)%3C/script%3E>

Store XSS-<https://google-gruyere.appspot.com/609646299901679235229890844824895027074/snippets.gtl>

(To work pass the mouse through the read this text)

6.3)

O atacante poderia colocar um script atráves de qualquer um dos métodos de Cross Site Scripting (Upload file XSS,Reflect Xss ou Store XSS) em que esse script iria realizar um pedido HTTP ao site do atacante e passaria como parâmetro da query string os cookies do utilizador através do campo document.cookie.

7.1) A alínea 7.1 consistia num  **cross-site request forgery** (**CSRF** ), em português falsificação de solicitação entre sites,que consite em persuadir a pessoa que esta a ser atacada a clicar num link ou abrir uma pagina maliciosa construída pelo atacante,que faz um pedido HTTP ao site que o utilizador atacado foi autenticado,neste caso esse site foi o Elgg, uma rede social.

Inicialmente,para forjar pedidos http ao Elgg,era necessário saber os headers necessários para fazer tanto um pedido Get como um pedido Post,para saber isso,foi utilizada uma extensão do Firefox denominada “HTTP Header Live”que mostra exatamente esses headers.

Posteriormente,depois de entender os headers necessários para fazer os respectivos pedidos,o objectivo seria fazer com que o utlizador atacado adicionasse o atacante aos amigos na rede social Elgg.

Para tal acontecer tínhamos que programar o site do atacante para que quando fosse aberto realizasse um pedido de amizade ao atacante sem ter de realizar qualquer clique,para isso,foi utilizado a tag img que faz um pedido HTTP Get ao url colocado no campo src.Logo no campo src colocamos o url que realiza o pedido de amizade.

7.2)O objectivo da alínea 7.2 era o mesmo da alínea 7.1,ou seja,realizar um pedido de **cross-site request forgery** (**CSRF** ),no entanto,em maquinas virtuais diferentes.

Para isso começamos por clonar a maquina virtual previamente criada para a alínea 7.1.

Posteriormente era necessário que essas duas maquinas conseguissem enviar informação entre elas.Para isso era necessário que estivessem ambas ligadas á mesma rede NAT.

Depois de criarmos a rede NAT constituída pelas 2 maquinas virtuais,fizemos uns testes a ver se era possível passar informação da maquina A para a maquina B e vice-versa através do comando ping.

Ao verificar que tudo estava a funcionar corretamente faltava apenas configurar o ficheiro de hosts do utilizador atacado para que quando fosse realizado o pedido Http para o Elgg através da sua maquina o mesmo apontasse para o IP do atacante.