Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Licenciatura em Engenharia Informática, Redes e Telecomunicações
Segurança Informática
Terceira série de exercícios, Semestre de Inverno de 18/19
Data limite de entrega: 27 de dezembro de 2018

---

1 Enunciado

1. Considere o modelo RBAC_1.

   1.1. É possível existir uma sessão associada ao utilizador u e com o *role* r activo, sem que (u, r) esteja na relação *user assignment* (UA)?

   1.2. Qual a relação entre o princípio de privilégio mínimo e o conceito de sessão?

2. Ataques do tipo *cross-site request forgery* (CSRF) têm como alvo utilizadores autenticados numa aplicação web vulnerável (cujo estado de sessão é mantido através de *cookies*). O uso de *cookies* cifrados contribuí para dificultar este tipo ataque?

3. Em 2014 foi identificada uma vulnerabilidade na aplicação *bash* de vários sistemas Linux, Unix e OS X, a qual ficou conhecida como *Shellshock*. Uma das formas de explorar esta vulnerabilidade consistia no atacante conseguir executar comandos ou programas na máquina vulnerável. Para as alíneas seguintes tenha em conta as informações de base sobre a vulnerabilidade e sua exploração presentes na Secção 2, e use a versão Ubuntu 16.04 dos laboratório SEED.

   3.1. Realize as tarefas preparatórias 2.1 a 2.3 do guião SEED sobre a vulnerabilidade ShellShock [2]. Apresente um resumo das ações realizadas em cada tarefa e explique de que forma na 2.3 um atacante pode passar dados arbitrários para o contexto de execução de um *bash shell* através da *Common Gateway Interface* (CGI) do servidor Apache.

   3.2. Lance duas máquinas virtuais (atacante e vítima) e realize o ataque pedido na tarefa 2.4, com o objectivo de obter da máquina vítima o ficheiro `/var/www/SQLInjection/safe_home.php`. Identifique no ficheiro o utilizador e palavra-chave usados pela aplicação *web* em causa para aceder à base de dados. Note que com a ferramenta `curl` é possível adicionar cabeçalhos aos pedidos HTTP (opções `-H` e `--header`).

4. Considere o laboratório do project SEED sobre *SQL injection* [3]:

   4.1. Realize a configuração do sistema (ponto 2 do guião).

   4.2. Realize as tarefas 2.1 (do ponto 3.2) e 3.1 (do ponto 3.3). Apresente um sumário das ações realizadas.

   4.3. *Opcional* – Realize a tarefa 3.3 (do ponto 3.3) e apresente um sumário das ações realizadas.

5. Analise o efeito da política *same-origin* através do laboratório `Collaborative` [7] da versão Ubuntu 12.04 dos laboratório SEED:

   5.1. Realize a tarefa 2 da alínea 4 e descreva em que situações a política *same-origin* não permite o acesso ao DOM e *cookies*. No ponto 2 desta tarefa use um domínio diferente do *www.google.com* sugerido no guião.

   5.2. Realize a tarefa 3 da alínea 4 e descreva o resultado de pedidos `XMLHttpRequest` ao domínio *www.soplab.com* e a outro (ex: *www.example.com*).

6. Considere o laboratório do project SEED sobre *cross-site scripting* (XSS) e *cross-site request forgery* (CSRF) [4]:

   6.1. Realize as tarefas preparatórias de 3.1 a 3.4.

   6.2. Realize a tarefa 3.5 e responda às questões propostas:

      i. Qual o propósito das linhas identificadas com (1) e (2)?

      ii. Se não fosse possível usar o editor directo de HTML seria mesmo assim possível fazer o ataque?

2 Shellshock

Este anexo é uma adaptação do Capítulo 3 do livro *Computer Security - A Hands-on Approach* [1].

Uma das linhas de comandos mais utilizada em sistemas da família Unix/Linux é o *bash shell*, localizado em `/bin/bash`. Em *bash* podem ser definidas funções, como mostra o exemplo seguinte:

```
$ foo() { echo "Inside␣function"; } # define a função 'foo'
declare -f foo # apresenta o código da função 'foo'
foo() {[...]}
$ foo           # executa a função 'foo'
Inside function
$ unset -f foo
$ declare -f foo
$
```

As funções definidas no processo pai podem ser exportas e assim utilizadas no processo filho, como mostra o exemplo:

```
$ foo() { echo "Inside␣function"; }
$ declare -f foo
$ export -f foo
$ bash  # executa /bin/bash num processo filho
(child)$ declare -f foo
(child)$foo
Inside function
```

A vulnerabilidade Shellshock está relacionada com a possibilidade de passar funções de um processo pai para um processo *bash* filho. Para além do método apresentado anteriormente (e que não é uma vulnerabilidade) a outra forma de passar funções é através de variáveis de ambiente. Quando o processo pai exporta uma variável de ambiente cujo valor é uma *string* com a definição de uma função, o valor da variável é interpretado e transformado numa função no processo filho, como mostra o exemplo:

```
$ foo='() { echo "Inside␣function"; }'
$ echo $foo
() { echo "Inside␣function"; }
$ declare -f foo
$ export foo
$ bash
(child)$ echo $foo
(child)$                  # no processo filho não existe variável 'foo' ...
(child)$ declare -f foo   # ... mas sim uma função de nome 'foo'
(child)$ foo ()
{
  echo "Inside␣function"
}
$ foo
Inside function
```

Quando no processo filho é executado `echo $foo` não é apresentado nada porque, durante a criação deste processo, se for encontrada uma variável de ambiente que começa por `()`, será analisado o seu valor para a transformar numa função com o mesmo nome. Durante esta análise, devido a um erro de programação, são executados os comandos depois da segunda chaveta, como mostra o exemplo seguinte:

```
$ foo='() { echo "Inside␣function"; }; echo "Hi!";'
$ echo $foo
$ () { echo "Inside␣function"; }; echo "extra";
$ export foo
$ bash
Hi!          # o comando extra é executado
$ echo $foo
$
```

```
$ delcare -f foo
foo ()
{
  echo "Inside␣function"
}
```

Note que no segundo método de passar funções, o processo pai não precisa de ser o *bash*. Qualquer processo que queira chamar o *bash shell* e passar-lhe uma função, apenas terá de a definir previamente numa variável de ambiente.

Referências

[1] Wenliang Du, Computer Security: A Hands-on Approach, 1ª Edição, CreateSpace Independent Publishing Platform, 2017

[2] `http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Software/Shellshock/`, visitado a 27 de novembro de 2018

[3] `http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Web/Web_SQL_Injection/Web_SQL_Injection.pdf`, visitado a 27 de novembro de 2018.

[4] `http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Web/Web_XSS_Elgg/`, visitado a 27 de novembro de 2018.

[5] `http://www.cis.syr.edu/~wedu/seed/lab_env.html`

[6] `http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Documents/SEEDVM_VirtualBoxManual.pdf`, visitado a 27 de novembro de 2018.

[7] `http://www.cis.syr.edu/~wedu/seed/Labs/Web/Web_SOP_Collabtive/Web_SOP_Collabtive.pdf`, visitado a 22 de novembro de 2018.

[8] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options`, visitado a 22 de novembro de 2018.