PROCESSO SELETIVO - Grupo de Resposta a Incidentes de Segurança

Nome: Fernanda Veiga Gomes da Fonseca
TAG - Segurança Web I - Entrega: 18/02/2020

1-

Com base no modelo TCP/IP, o HTTP é um protocolo da camada de aplicação e utiliza o TCP como protocolo de transporte.

Seu funcionamento é baseado em uma lógica cliente/servidor: primeiro, o cliente inicia conexão TCP com o servidor; em seguida, o servidor aceita conexão TCP do cliente; então, mensagens* HTTP são trocadas entre o navegador (cliente HTTP) e o servidor Web (servidor HTTP); por fim, a conexão TCP é encerrada.

* A troca de mensagens HTTP ocorre da seguinte forma: o cliente HTTP envia uma mensagem HTTP de pedido através da conexão TCP, pois deseja receber um objeto www.site.com.br/index.html, por exemplo; depois de o servidor HTTP receber a mensagem de pedido, formula a mensagem de resposta contendo objeto solicitado.

2-

Um “Response Code” é um código de estado emitido por um servidor em resposta à solicitação de um cliente feita a ele. No caso de uma solicitação HTTP, os códigos são separados em cinco grupos, nos quais o primeiro dígito define a classe de resposta.

Exemplo: o código 429 significa que o usuário enviou muitas solicitações em um determinado intervalo de tempo. Ele pode indicar que servidor está sob ataque e pode ser usado para desconectar conexões ou empregar uma forma de defesa, pois o servidor não precisa informá-lo, uma vez que o retorno a cada solicitação consome recursos valiosos.

3-

São campos existentes no cabeçalho das mensagens de requisição e resposta no HTTP. Eles definem os parâmetros de operação de uma

transação HTTP e permitem que o cliente e o servidor passem informações adicionais.

Um uso inseguro do cabeçalho pode residir no campo “Referer”. Ele contém o endereço da página da Web anterior à página solicitada no momento, o qual pode ser seguido, gerando roubo e vazamento de informações confidenciais, por exemplo.

4-

Os métodos no protocolo HTTP indicam o que o servidor deve fazer com o URL fornecido na mensagem de requisição do cliente.

O método GET é usado para recuperar informações do servidor usando parâmetros contidos na URL. O método POST é usado para performar ações (por exemplo, envio de informações ao servidor). Diferentemente do método GET, os parâmetros podem ser enviados pelo corpo da requisição e não apenas pela URL.

Dessa forma, o método POST é mais seguro que o método GET, porque não carrega informações possivelmente sensíveis anexadas à URL, evitando que fiquem salvas no histórico.

5-

O cache HTTP é o armazenamento temporário de páginas da Web. Reduz a latência e o tráfego de rede, diminuindo, pois, o tempo necesário para exibir uma representação do recurso. Quando um “web cache” possui um recurso requerido por um cliente em seu armazenamento, ele intercepta a solicitação e retorna sua cópia, evitando uma nova realização de download do servidor original. Além disso, sua atuação pode objetivar um conjunto de clientes (cacheamento público) ou apenas um cliente (cacheamento privado).

O atraso de resposta do servidor a requisições futuras apenas é reduzido caso determinadas condições especificadas nos cabeçalhos forem atendidas. Os campos do cabeçalho HTTP responsáveis pelo controle do cache são: “Age”, “Cache-Control” (é um cabeçalho genérico, porque é um conjunto de campos), “Expires”, “Pragma” e “Warning”.

6-

Um cookie HTTP é um fragmento de dados que um servidor envia para o navegador do usuário, o qual pode armazenar e enviar esses dados de volta na próxima requisição ao mesmo servidor. É um parâmetro que identifica a sessão, podendo ser utilizado para identificar se duas requisições vieram do mesmo navegador, permitindo que um usuário permaneça logado, por exemplo.

Há diversos ataques relacionados aos cookies, dentre eles Pharming, Man-in-the-middle e Cross-site Scripting. Considerando que o último encontra-se no OWASP-Top-Ten, ele pode ser visto como o principal ataque. Nessa modalidade, se um site permitir que um usuário insira conteúdos no mesmo e não bloqueie a injeção de um JavaScript malicioso, poderão ser roubados os cookies de outro usuário e personificá-lo (“cookie hijacking”). Para que o site não esteja vulnerável ao XSS, os cookies devem ser definidos como HTTPOnly, para que eles não possam ser lidos ou manipulados por JavaScript.

7-

É um documento de conscientização sobre os dez riscos de segurança mais críticos em aplicações Web. Recomenda-se que as empresas adotem este documento para garantir que seus aplicativos Web minimizem esses riscos e sugere-se que talvez seja o primeiro passo mais eficaz para mudar a cultura de desenvolvimento de software dentro de uma organização, levando à produção de códigos mais seguros.

8-

“Recon” trata-se do reconhecimento de um alvo e pode ser usado para planejar um ataque contra ele. É uma coleta de informações que pode fornecer a topologia da rede, bem como o hardware e o software utilizados. Por fornecer tal conhecimento, pode indicar o melhor plano de ação com base na “superfície de ataque”, pois

um invasor busca formular um plano que exija o mínimo de esforço, estreitando o foco inicial na parte mais fraca da segurança do alvo.

9-

a) É um ataque cujo objetivo é a execução de comandos arbitrários no sistema operacional host por meio de uma aplicação vulnerável. Eles são possíveis quando a mesma passa de forma insegura dados fornecidos pelo usuário para um shell do sistema. Os comandos do sistema operacional passados pelo invasor geralmente são executados com os privilégios da aplicação vulnerável, estendendo sua funcionalidade padrão, sem a necessidade de adição de código.

10-

a) É um ataque que realiza uma consulta SQL através dos dados de entrada do cliente na aplicação vulnerável. O atacante pode inserir ou manipular consultas criadas pela aplicação, que são enviadas diretamente para o banco de dados. Assim, é possível ler dados confidenciais, modificar dados (utilizando INSERT e DROP, por exemplo) e executar operações de administração (como o desligamento do sistema de gerenciamento do banco).

b) Quando uma aplicação é vulnerável à injeção de SQL e os resultados da consulta ao banco são retornados nas suas mensagens de respostas, a palavra-chave UNION pode ser usada para recuperar dados de outras tabelas no banco de dados, resultando em um ataque “Union-Based Injection”. A palavra-chave UNION permite retornar um único conjunto de resultados com mais de uma coluna através de diversas consultas SELECT individuais. Entretanto, isso somente é possível se elas retornarem o mesmo número de colunas e se os tipos de dados em cada coluna forem compatíveis.

c) O ataque “Blind-SQL-Injection” pode ser feito quando uma aplicação é vulnerável à injeção de SQL, mas suas respostas HTTP não contêm os resultados da consulta SQL, podendo ser

“Partial-Blind” ou “Full-Blind”. Uma vez que não é possível visualizar os resultados da consulta injetada nas respostas da aplicação, ataques utilizando a palavra-chave UNION não são eficazes. Entretanto, ainda é possível acessar dados não autorizados através da realização de uma série de perguntas do tipo “True” ou “False”.

11-

a) Nos ataques XSS (Cross-Site Scripting), scripts maliciosos são injetados em sites. Um invasor usa uma aplicação Web para enviar um código malicioso, geralmente por um script, pelo navegador, para um usuário final diferente. O navegador do mesmo o executará, pois não sabe que o script não é confiável. Dessa forma, o script mal-intencionado poderá acessar todos os cookies, tokens de sessão e outras informações confidenciais armazenadas no navegador e usadas no site. O invasor pode, então, executar quaisquer ações do usuário e acessar seus dados.

b) Existem três tipos principais: “Reflected XSS”, “Stored XSS” e “DOM-Based XSS”.

No primeiro, o script malicioso vem da solicitação HTTP atual, ou seja, a aplicação recebe dados em uma mensagem de solicitação HTTP e inclui esses dados na resposta imediatamente seguinte. Se o usuário visitar a URL criada pelo invasor, o script do invasor será executado em seu navegador, na sessão desse usuário com a aplicação. Assim, o script pode executar qualquer ação e recuperar quaisquer dados aos quais o usuário tem acesso.

No segundo, o script malicioso vem do banco de dados do site, porque ocorre quando um aplicativo recebe dados de uma fonte não confiável e inclui esses dados em suas respostas HTTP posteriores.

No terceiro, a vulnerabilidade existe no código do lado do cliente e não no código do lado do servidor, pois ocorre modificação prévia do DOM no navegador no lado do cliente. A resposta HTTP não é alterada, mas o código do lado do cliente contido na página é

executado de maneira diferente devido às modificações maliciosas que ocorreram no DOM. Diferentemente dos outros ataques XSS, o ataque não é realizado com base na página de resposta, explorando uma falha do servidor.

12-

a) Local File Inclusion (LFI) é o processo de execução de arquivos já estão presentes localmente no servidor após a obtenção dos mesmos. Isso é possível quando a aplicação utiliza o caminho para um arquivo como entrada.

b) O Remote File Inclusion (RFI) funciona de maneira semelhante ao LFI, porém são utilizados arquivos remotos.

c) O Path Traversal é um ataque que induz uma aplicação Web a divulgar o conteúdo de arquivos fora do diretório da aplicação. As entradas das aplicações podem ser manipuladas usando seqüências como ponto-ponto-barra (../) ou variações semelhantes para acessar as pastas do sistema de arquivos do servidor mais altas na hierarquia.

d) O Path Traversal é tipo de LFI, porque o LFI também pode executar arquivos após recuperá-lo. Dessa maneira, a realização do LFI permite que o Path Traversal também seja realizado.

13-

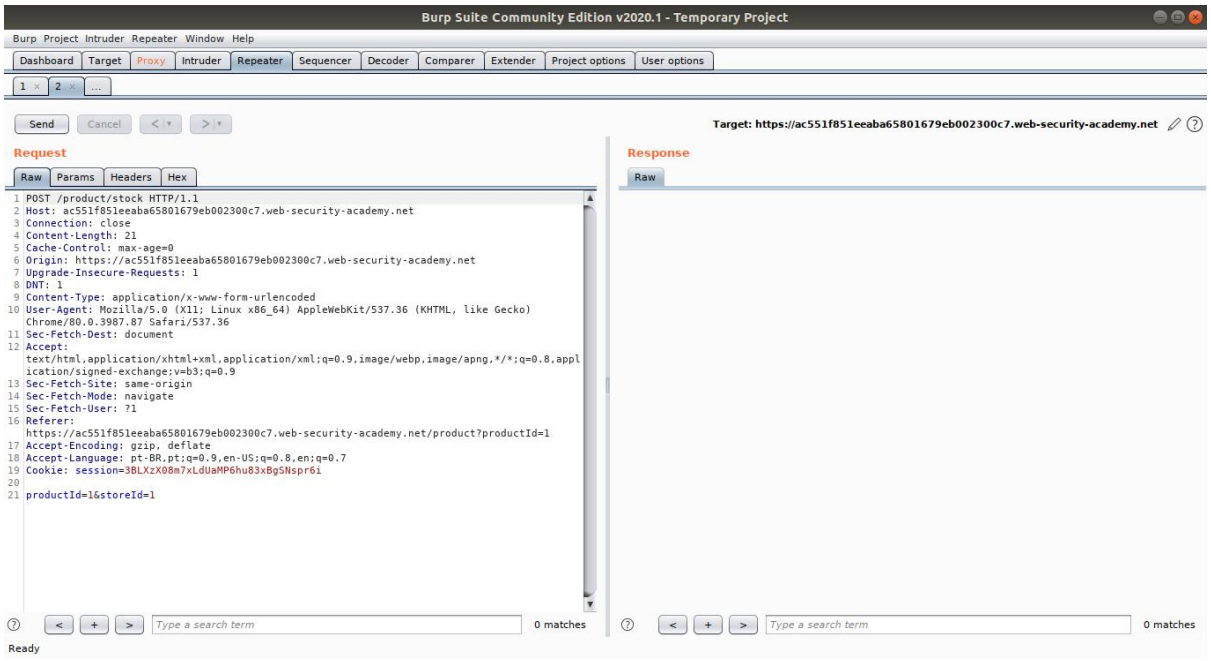
a) O CSRF é uma vulnerabilidade de segurança que permite que um atacante execute ações não desejadas pelo usuário, aproveitando-se da confiança do site na identidade do mesmo. Portanto, o invasor age como se fosse o usuário, podendo obter controle total dos dados e das funcionalidades, caso o usuário tenha uma função privilegiada na aplicação.

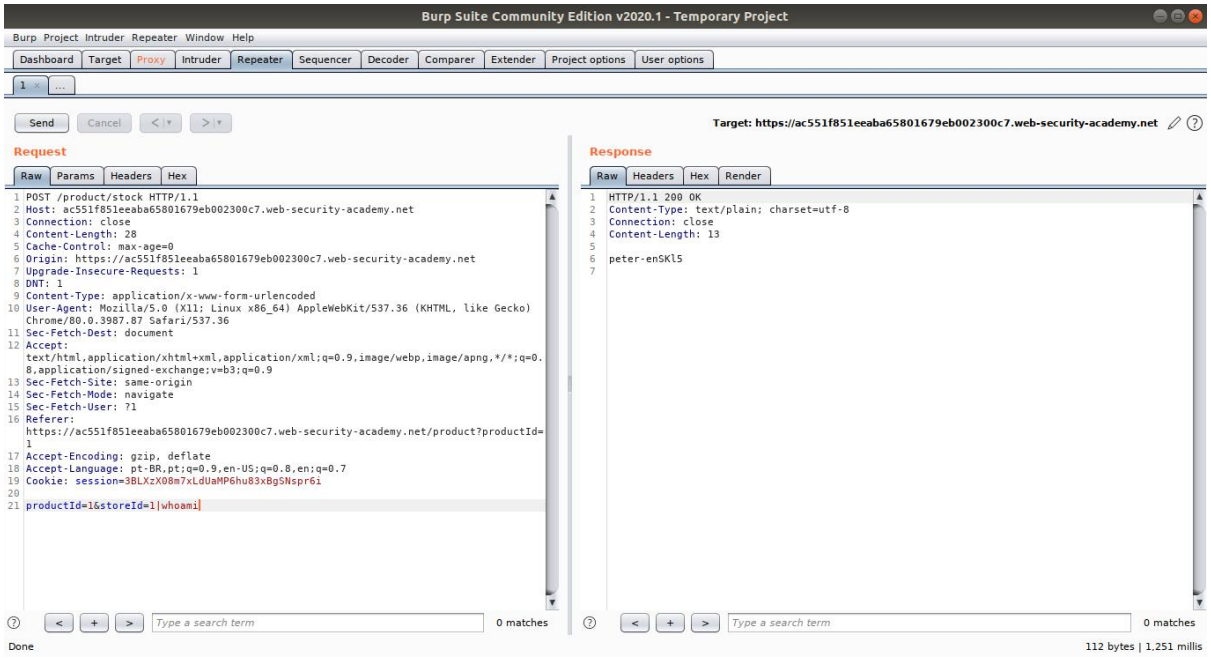
c) O SSRF é uma vulnerabilidade de segurança que permite que um atacante induza a aplicação no lado do servidor a fazer solicitações HTTP para um domínio arbitrário de sua escolha, podendo ser o da própria aplicação através de um endereço de loopback. Também, a exploração dessa vulnerabilidade permite o acesso a dados de sistemas com os quais a aplicação se comunica, parecendo que as requisições são feitas pela organização hospedeira da aplicação vulnerável.

e) Um ataque de CSRF pode ser evitado através do uso deum token CSRF. Ele possui um valor único e secreto, gerado pelo aplicativo do servidor e transmitido ao cliente, devendo ser incluído em uma solicitação HTTP subsequente feita pelo cliente. Assim, a aplicação do servidor valida a solicitação quando o token esperado está presente e compatível. A existência de um token dificulta a elaboração de uma solicitação HTTP totalmente válida, ou seja, com todos os parâmetros necessários para a aplicação atender à solicitação.

[9-b)]

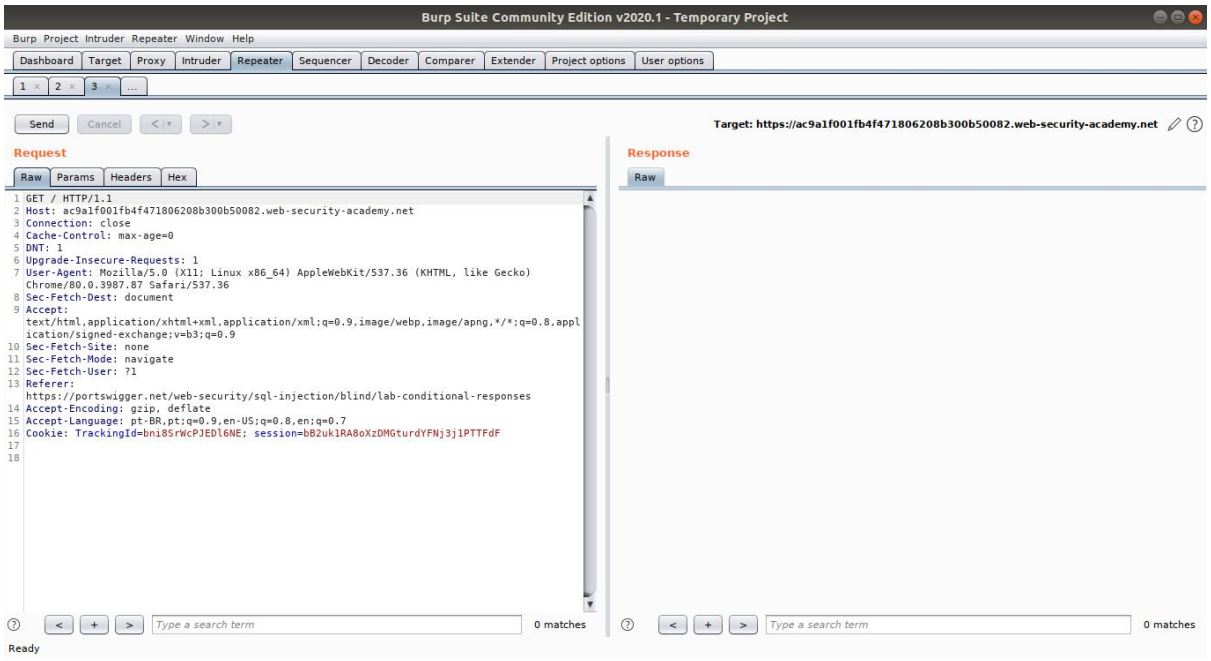
https://portswigger.net/web-security/os-command-injection/lab-simple

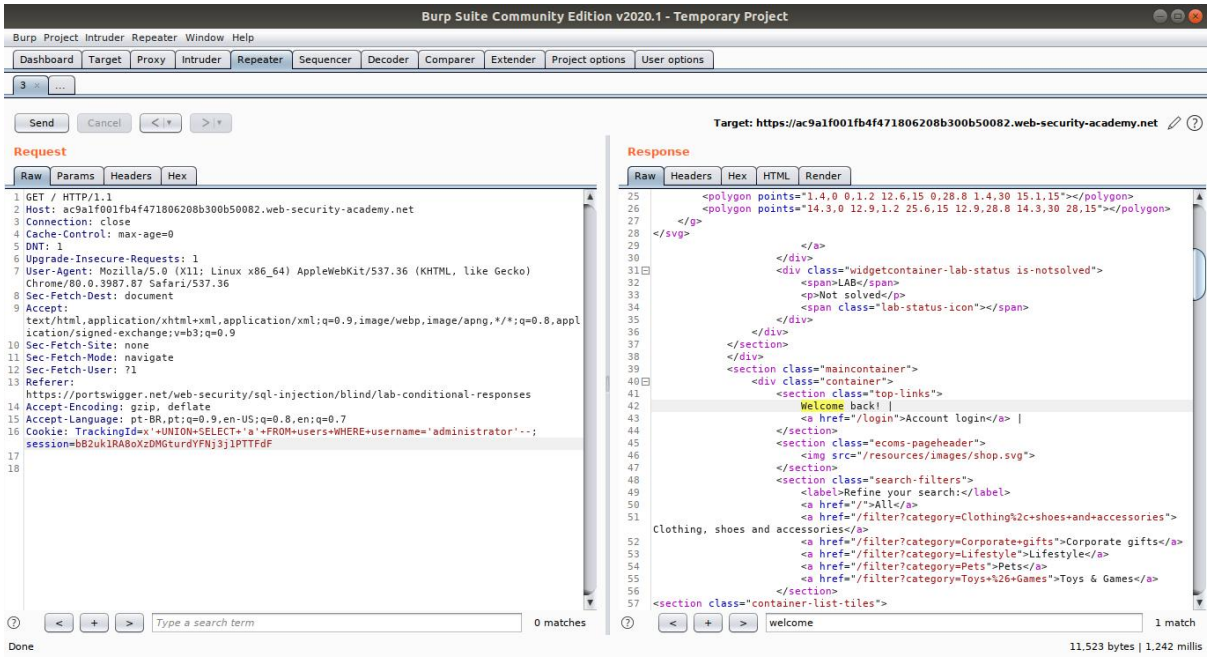


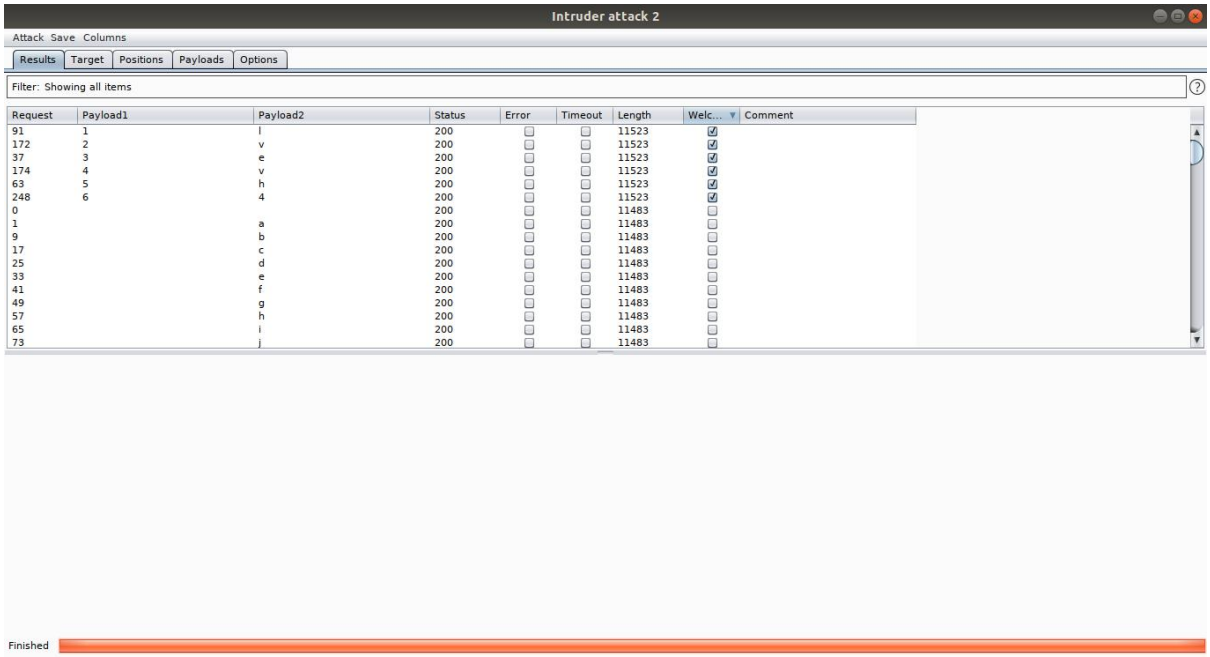


[10-d)]

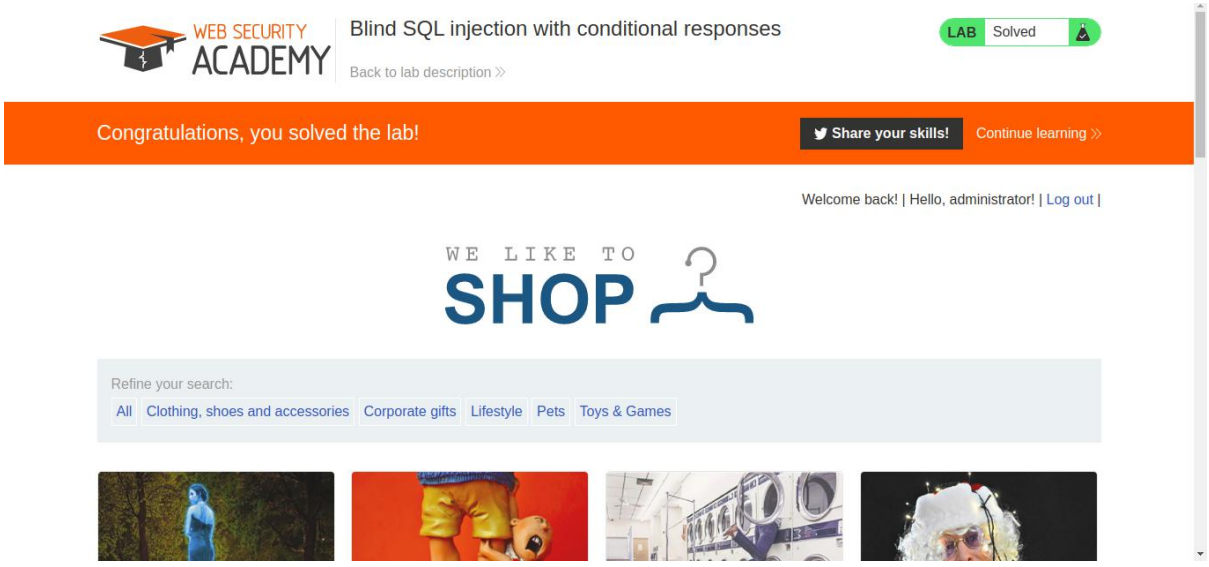
https://portswigger.net/web-security/sql-injection/blind/lab-conditional-responses



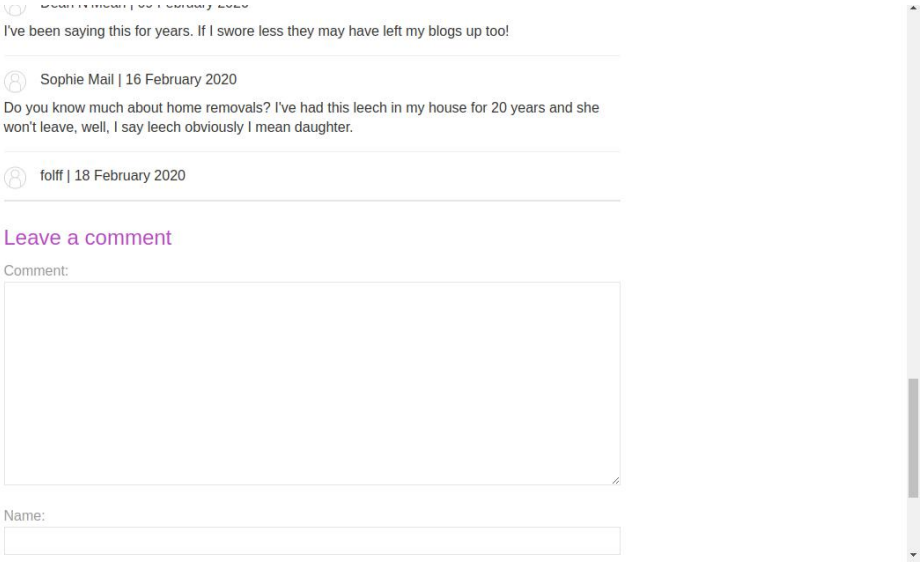




Username: administrator; Password: lvevh4

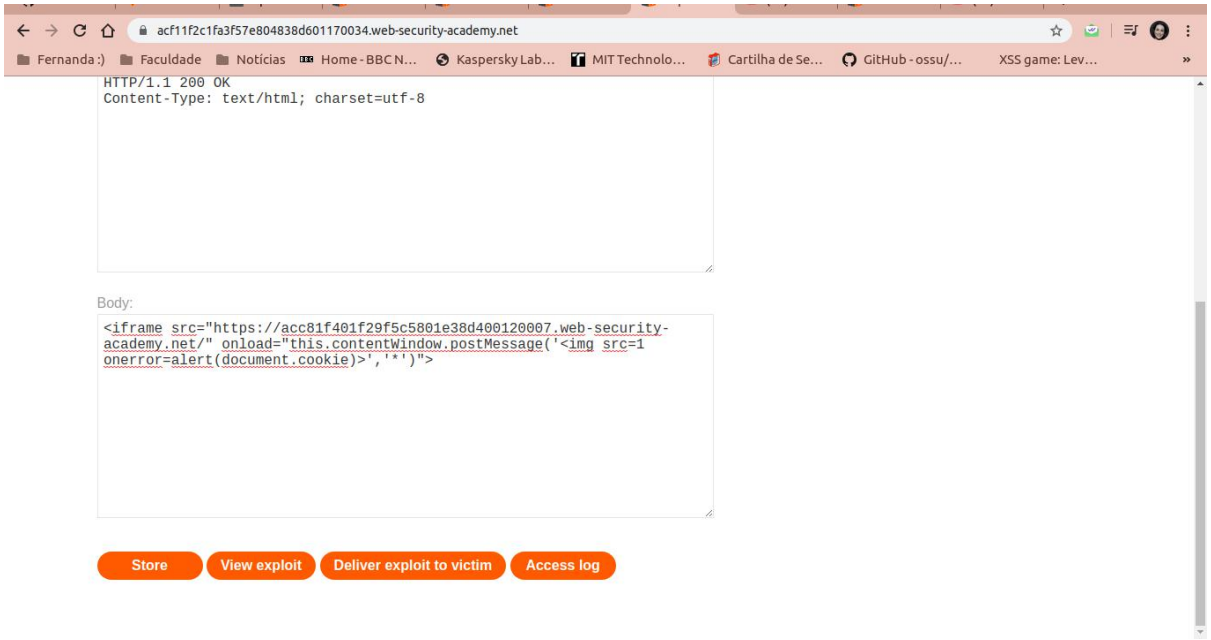


[11-c)]
https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded



[11-d)]

https://portswigger.net/web-security/dom-based/controlling-the-web-message-source/lab-dom-xss-using-web-messages

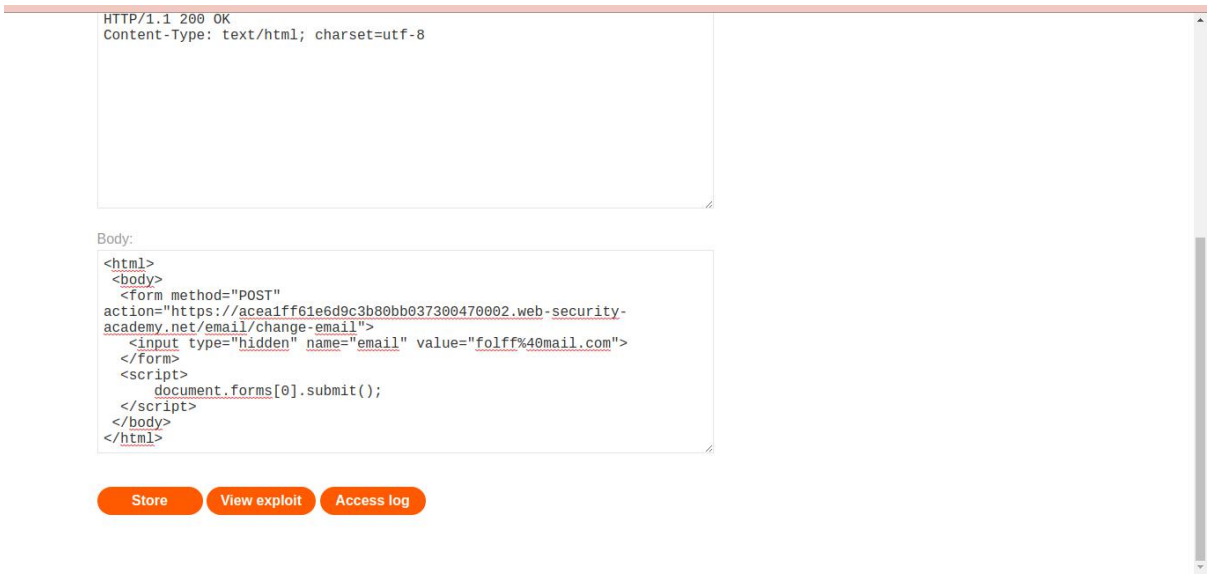


O postMessage () envia uma mensagem Web para a página inicial, o EventListener pega seu conteúdo e insere na div. Porém, como img contém um atributo src inválido, ocorre um erro e a carga útil é executada.

[12-e)]

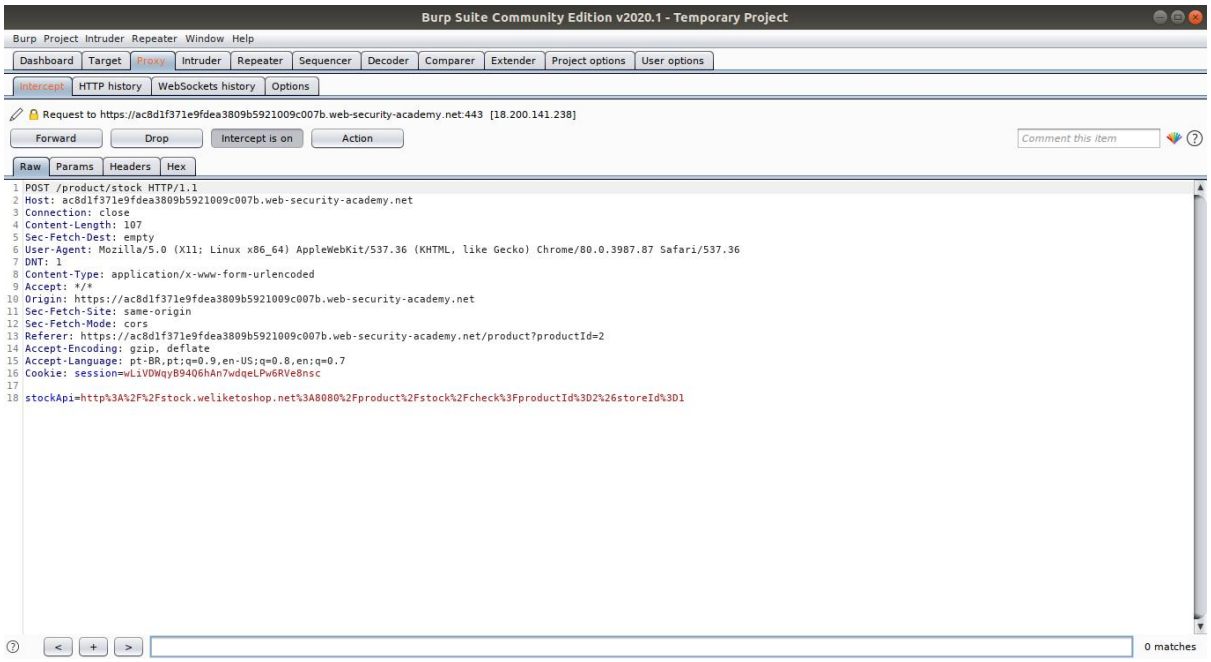
[13-b)]

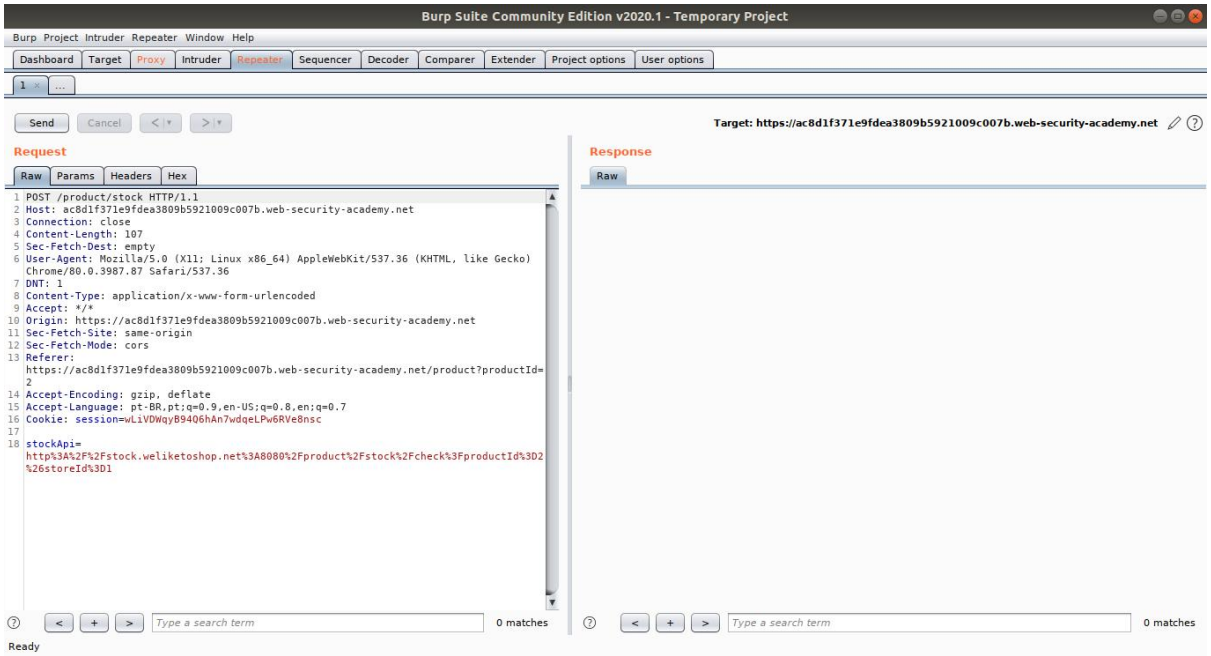
https://portswigger.net/web-security/csrf/lab-no-defenses

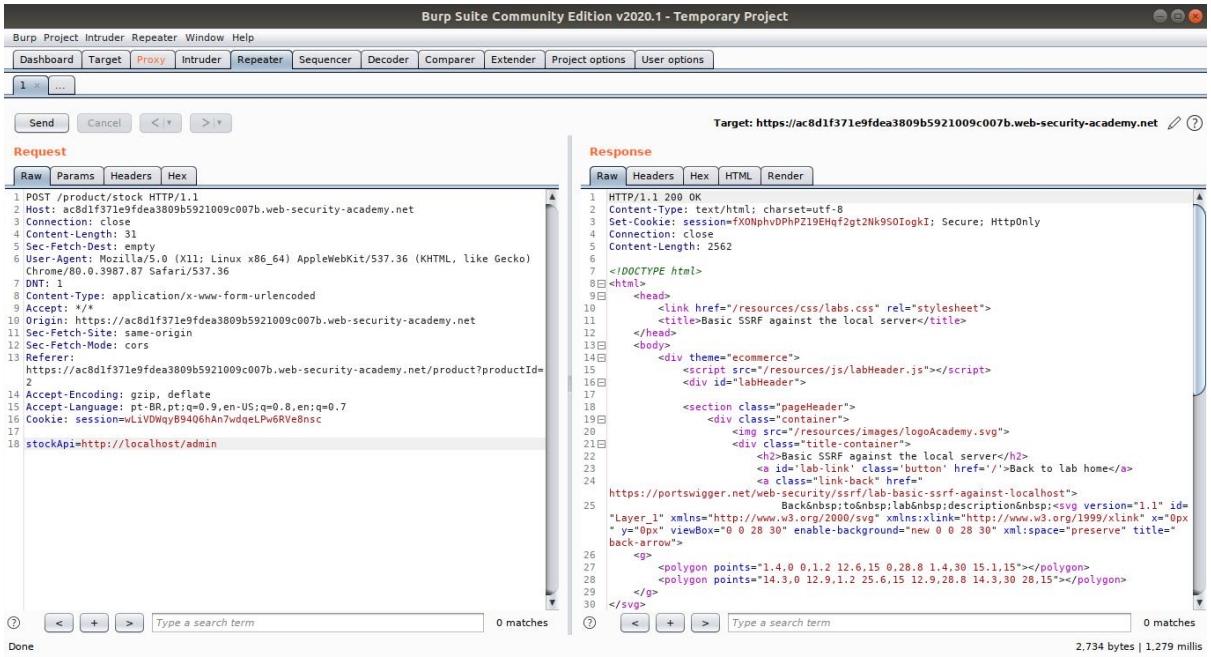


[13-d)]

https://portswigger.net/web-security/ssrf/lab-basic-ssrf-against-localhost





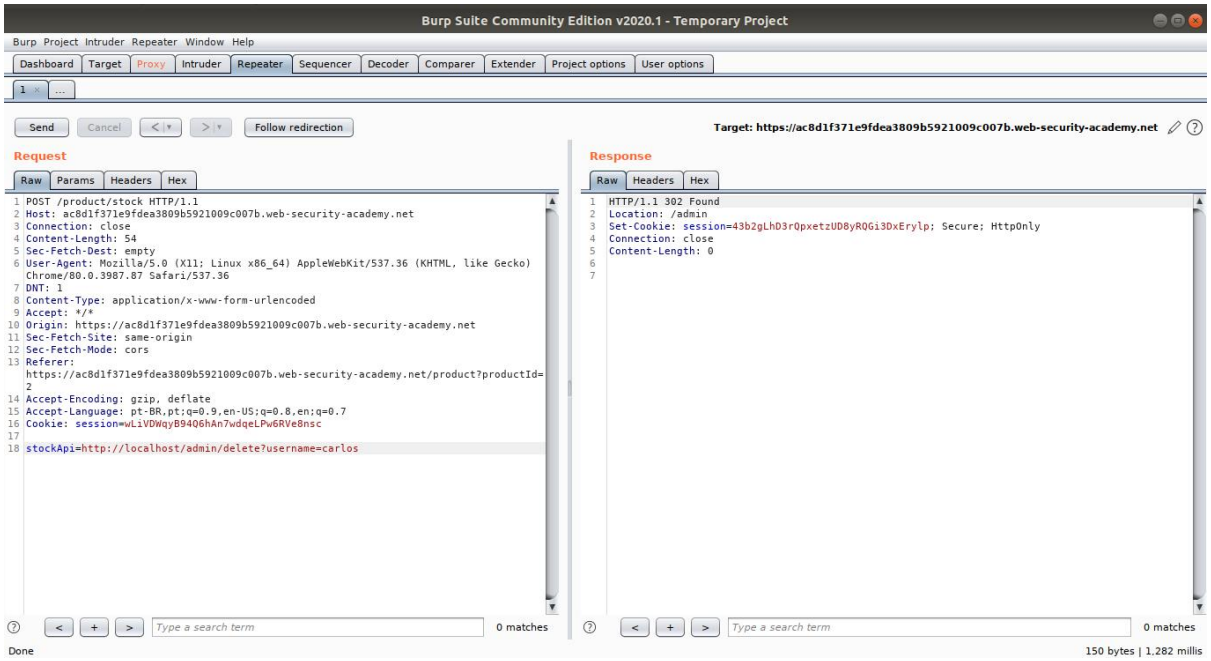

Burp Suite Community Edition v2020.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Request
Raw Params Headers Hex
POST /product/stock HTTP/1.1
Host: ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Connection: close
Content-Length: 31
Sec-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.87 Safari/537.36
DNT: 1
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Referer: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net/product?productId=2
Accept-Encoding: gzip, deflate
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: session=wLiVDWqyB94Q6hAn7wdqeLPw6RVe8nsc
stockApi=http://localhost/admin
Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Set-Cookie: session=fXONphvDPhPZ19EHqf2gt2Nk9SOIogkI; Secure; HttpOnly
Connection: close
Content-Length: 2562
<!DOCTYPE html>
<html>
<head>
<link href="/resources/css/labs.css" rel="stylesheet">
<title>Basic SSRF against the local server</title>
</head>
<body>

<script src="/resources/js/labHeader.js"></script>

<section class="pageHeader">

<img src="/resources/images/logoAcademy.svg">

<h2>Basic SSRF against the local server</h2>
Back to lab home
<a class="link-back" href="https://portswigger.net/web-security/ssrf/lab-basic-ssrf-against-localhost">
Back to lab description <svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0 28 30" enable-background="new 0 0 28 30" xml:space="preserve" title="back-arrow">
<g>
<polygon points="1.4,0 0,1.2 12.6,15 0,28.8 1.4,30 15.1,15"></polygon>
<polygon points="14.3,0 12.9,1.2 25.6,15 12.9,28.8 14.3,30 28,15"></polygon>
</g>
</svg>
Type a search term
0 matches
Done
2,734 bytes | 1,279 millis


Burp Suite Community Edition v2020.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel Follow redirection
Target: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Request
Raw Params Headers Hex
POST /product/stock HTTP/1.1
Host: ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Connection: close
Content-Length: 54
Sec-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.87 Safari/537.36
DNT: 1
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Referer: https://ac8d1f371e9fdea3809b5921009c007b.web-security-academy.net/product?productId=2
Accept-Encoding: gzip, deflate
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: session=wLiVDWqyB94Q6hAn7wdqeLPw6RVe8nsc
stockApi=http://localhost/admin/delete?username=carlos
Response
Raw Headers Hex
HTTP/1.1 302 Found
Location: /admin
Set-Cookie: session=43b2gLhD3rOpxetzUD8yRQGi3DxErylp; Secure; HttpOnly
Connection: close
Content-Length: 0
Type a search term
0 matches
Done
150 bytes | 1,282 millis