



Tugas 3 - Tes penetrasi pada DVWA

PT. Mencari Cinta Sejati

ANDI AHMAD RAMADHAN M.P. - 1301213166

JOSUA PANE - 1301210254

MURSYID NAJIB MUHANA - 1301210411

RAIHAN ABDURRAHMAN - 1301210340

SEAGATA ADE PRATAMA BARUS - 1301210371



Copyright © 2021 Offensive Security Ltd. All rights reserved.

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from Offensive Security.



Daftar Isi

1. DVWA	3
1.1. Pengenalan	3
1.2. Tujuan	3
1.3. Requirements	3
2. Melakukan serangan Brute Force	4
2.1. Pengenalan Metode	4
2.2. Mencari Informasi & Vulnerabilities	4
2.3. Vulnerabilities	4
2.4. Penetration Testing	4
2.5. Analisis	12
3. Melakukan serangan Command Execution	13
3.1. Pengenalan Metode	13
3.2. Mencari Informasi & Vulnerabilities	13
3.3. Vulnerabilities	14
3.4. Penetration Testing	14
3.5. Analisis	17
4. Melakukan serangan CSRF	18
4.1. Pengenalan Metode	18
4.2. Mencari Informasi & Vulnerabilities	18
4.3. Vulnerabilities	18
4.4. Penetration Testing	18
4.5. Analisis	18
5. Melakukan serangan File Inclusion	19
5.1. Pengenalan Metode	19
5.2. Mencari Informasi & Vulnerabilities	19
5.3. Vulnerabilities	19
5.4. Penetration Testing	20
5.5. Analisis	22
6. Melakukan serangan SQL Injection	23
6.1. Pengenalan Metode	23
6.2. Mencari Informasi & Vulnerabilities	23
6.3. Vulnerabilities	24
6.4. Penetration Testing	27
6.5. Analisis	28



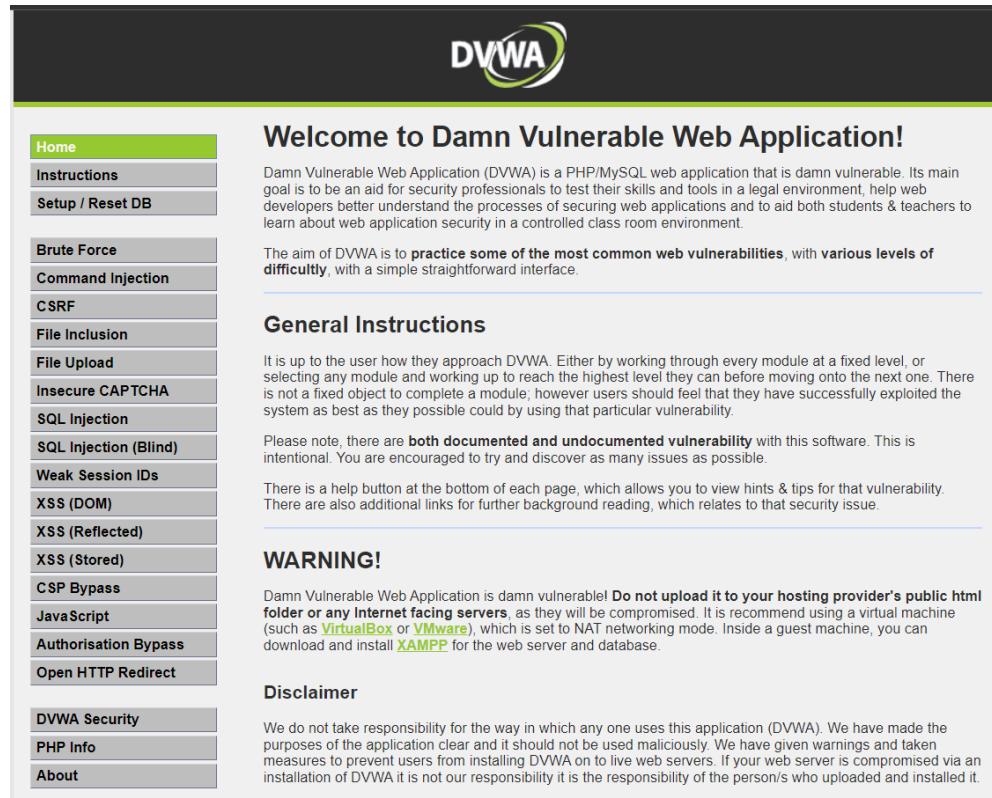
7. Melakukan serangan SQL Injection (Blind)	29
7.1. Pengenalan Metode	29
7.2. Mencari Informasi & Vulnerabilities	29
7.3. Vulnerabilities	30
7.4. Penetration Testing	30
7.5. Analisis	32
8. Melakukan serangan Upload	33
8.1. Pengenalan Metode	33
8.2. Mencari Informasi & Vulnerabilities	33
8.3. Vulnerabilities	33
8.4. Penetration Testing	37
8.5. Analisis	38
9. Melakukan serangan XSS Reflected	39
9.1. Pengenalan Metode	39
9.2. Mencari Informasi & Vulnerabilities	39
9.3. Vulnerabilities	39
9.4. Penetration Testing	40
9.5. Analisis	40
10. Melakukan serangan XSS Stored	41
10.1. Pengenalan Metode	41
10.2. Mencari Informasi & Vulnerabilities	41
10.3. Vulnerabilities	41
10.4. Penetration Testing	41
10.5. Analisis	41



1. DVWA

1.1. Pengenalan

DVWA (Damn Vulnerable Web Application) merupakan sebuah aplikasi web yang sengaja dibuat dengan kelemahan keamanan. Tujuan utamanya adalah untuk menjadi alat pembelajaran bagi para ahli keamanan informasi dan pembuat web. DVWA memberikan kesempatan bagi mereka untuk berlatih dan meningkatkan kemampuan mereka dalam menemukan dan menangani kelemahan keamanan web dalam sebuah lingkungan yang aman dan sesuai hukum. Aplikasi ini memuat berbagai jenis kelemahan keamanan yang sering terjadi, seperti SQL Injection, Cross-Site Scripting (XSS), CSRF, dan lainnya. Ini memudahkan pengguna untuk belajar dan menguji berbagai kerentanan tersebut secara aman.



The screenshot shows the main interface of the DVWA application. At the top, there's a dark header bar with the DVWA logo. Below it is a light-colored main content area. On the left side, there's a vertical sidebar menu with various exploit modules listed: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, DVWA Security, PHP Info, and About. The central part of the page has a heading "Welcome to Damn Vulnerable Web Application!" followed by a brief description of what DVWA is and its purpose. It also includes sections for "General Instructions", "WARNING!", and "Disclaimer".

(gambar 1 : tampilan halaman awal DVWA)



1.2. Tujuan

Tujuan dari laporan ini adalah untuk memberikan analisis terhadap keamanan aplikasi web, khususnya dengan fokus pada pengujian penetrasi dan kerentanan CSRF di DVWA. Laporan ini bertujuan untuk mendemonstrasikan bagaimana DVWA dapat digunakan sebagai alat latihan untuk mengidentifikasi dan memitigasi risiko keamanan dalam pengembangan web. Tujuan DVWA sendiri adalah untuk mendidik pengguna tentang pentingnya praktik keamanan yang baik dalam pengembangan web dan memberikan pengalaman praktis dalam menghadapi berbagai serangan web.

1.3. Requirements

- Sistem operasi : Kali Linux
- Aplikasi Web : DVWA
- Versi DVWA : 1.10
- Server Web : Apache
- Database : MySQL
- Browser : Mozilla Firefox
- Tools : Burp Suite



2. Melakukan serangan Brute Force

2.1. Pengenalan Metode

Serangan brute force adalah metode peretasan yang dilakukan dengan cara mencoba semua kemungkinan kombinasi kata sandi, kredensial login, atau kunci enkripsi hingga menemukan yang benar. Istilah brute force sendiri mengacu kepada upaya paksa yang dilakukan secara berlebihan untuk mendapatkan akses ke suatu akun.

2.2. Mencari Informasi & Vulnerabilities

informasi mengenai serangan brute force attack didapatkan dengan link yang ada dalam DVWA. berikut linknya:

More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>

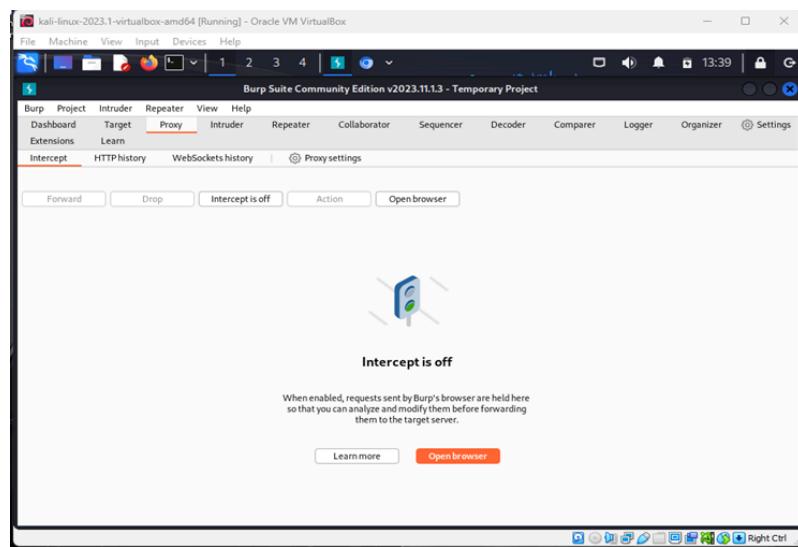
untuk mencari informasi vulnerabilities kami langsung menggunakan burpsuite.

2.3. Vulnerabilities

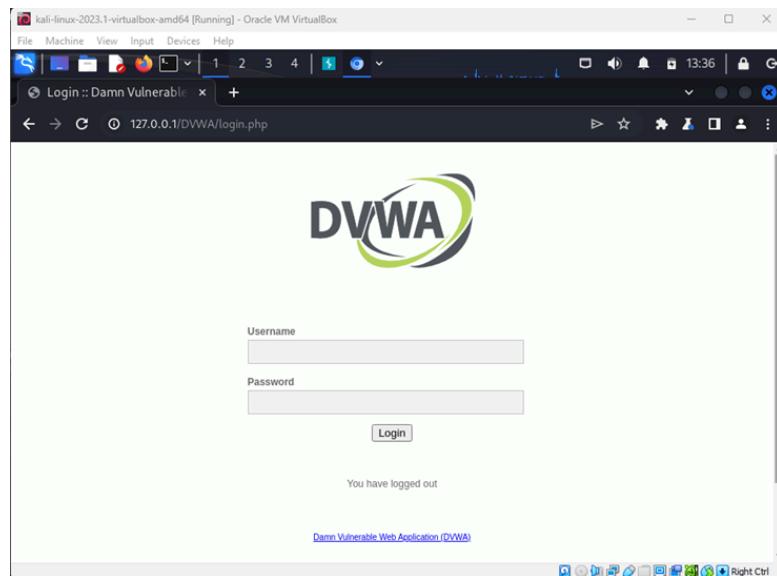
tidak ada Batasan kesalahan dalam menginputkan username dan password. Jika dilihat dari sisi user maka username dan password mudah ditebak atau terdapat pada dictionary penyerang.

2.4. Penetration Testing

Masuk ke aplikasi Burp Suite kemudian masuk ke bagian proxy, intercept, kemudian open browser



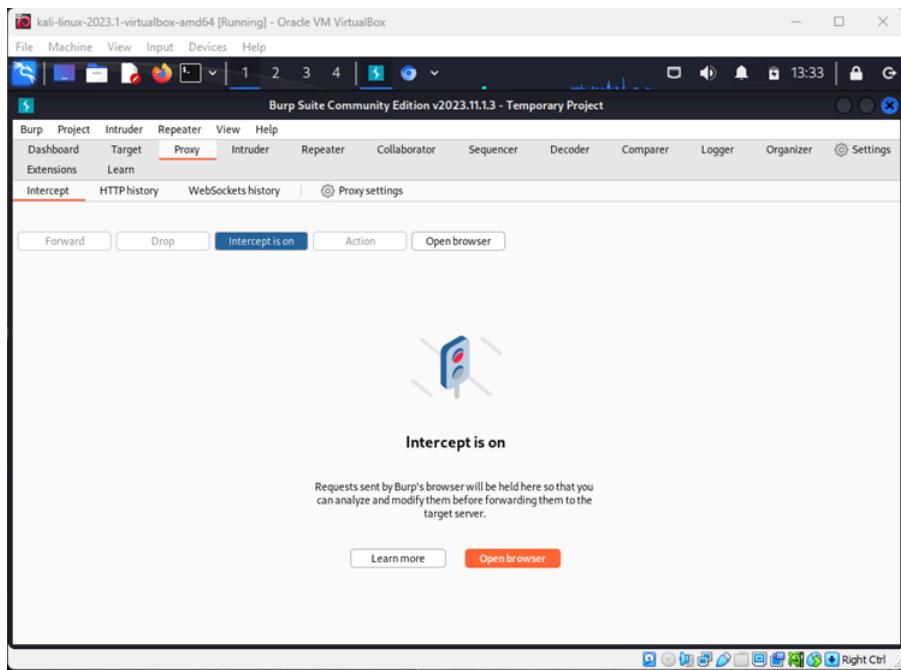
Login ke DVWA





OFFENSIVE[®]
security

Turn on intercept



Klik bagian brute force kemudian Masukan password asal karena
asumsinya kita tidak mengetahui password dan username yang benar



OFFENSIVE[®] security

The screenshot shows a web browser window for 'kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox'. The URL is '127.0.0.1/DVWA/vulnerabilities/brute/'. The DVWA logo is at the top. The main content is 'Vulnerability: Brute Force' with a 'Login' form. The sidebar has tabs for Brute Force (selected), Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), and XSS (Reflected). The login form has 'Username: halo' and 'Password: bandung4'. Below the form is a 'More Information' section with three links.

- https://owasp.org/www-community/attacks/Brute_force_attack
- <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>

Setelah tertangkap kirim hasil tangkapan ke intruder dengan klik kanan pada bagian kode kemudian pilih send to intruder. Setelah itu turn off intercept nya.

The screenshot shows a web browser window for 'kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox'. The URL is '127.0.0.1/DVWA/vulnerabilities/brute/'. The Burp Suite interface is overlaid. The 'Intercept' tab is selected. A context menu is open over a captured request, with 'Send to Intruder' highlighted. The request details show a GET to '/DVWA/vulnerabilities/brute/?username=halo&password=bandung4' with various headers and a JSON payload.

```
1 GET /DVWA/vulnerabilities/brute/?username=halo&password=bandung4
HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Not A Brand";v="8", "Chromium";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
like Gecko) Chrome/120.0.6099.71 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/DVWA/vulnerabilities/brute/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=elrnrvsl2el5f5jib050219dc; security=medium
Connection: close

```



OFFENSIVE[®] security

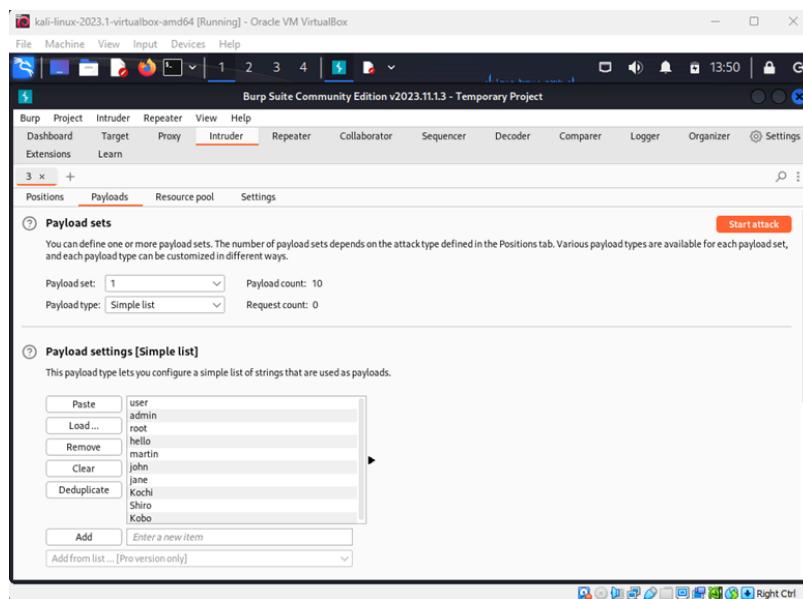
Setelah itu kita ke menu intruder kemudian pilih attack type menjadi cluster bomb, dan mark username dan password asal yang kita masukan tadi dengan mem-block bagian username dan password asal yang kita masukan kemudian klik add secara bergantian.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Attacktype' dropdown, 'Cluster bomb' is chosen. Under 'Payload positions', there are two entries: 'Target: http://127.0.0.1' and '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=1'. Both entries have the 'Update Host header to match target' checkbox checked. A 'Start attack' button is visible at the top right.

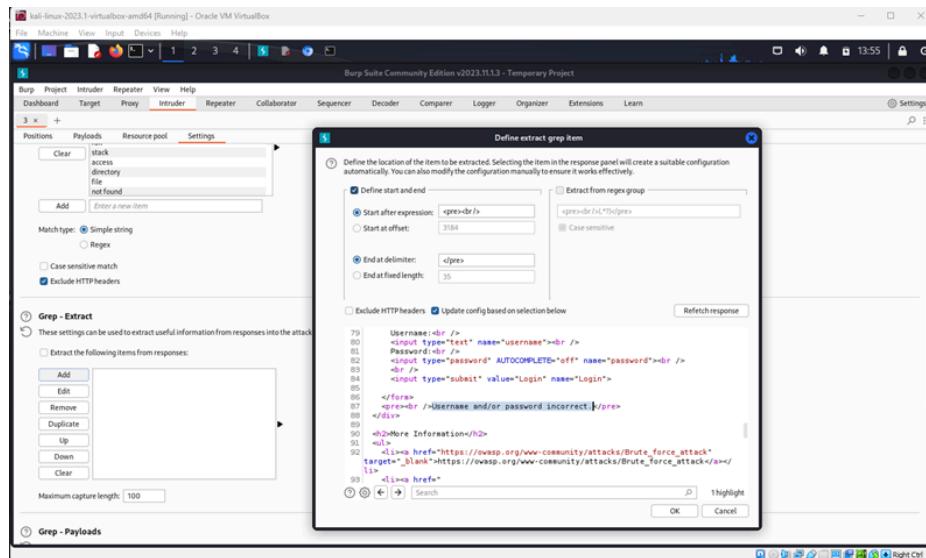
Kemudian masuk ke menu payloads dan masukan list username pada payloads 1 dan password pada payloads 2



OFFENSIVE® security



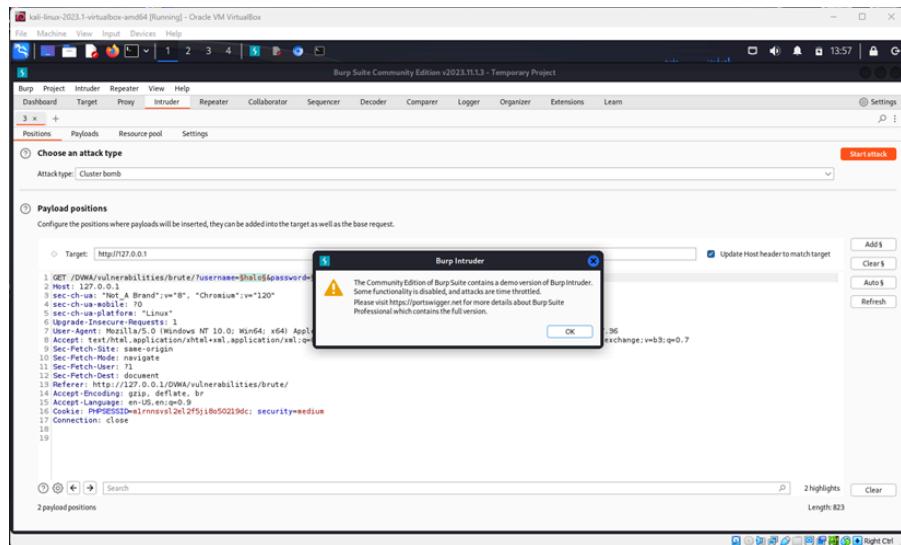
Kemudian masuk ke menu setting, pada bagian Grep-Extract kita tambahkan untuk menandakan username dan password salah kemudian klik ok





OFFENSIVE® security

Kemudian Kembali ke menu positions dan klik start attack. Akan muncul peringatan karena menggunakan community version akan lebih lama dalam melakukan brute force. Klik ok



Tunggu hingga brute force attack selesai

2. Intruder attack of http://127.0.0.1 - Temporary attack - Not saved to project file						
Attack	Save	Columns	Results	Positions	Payloads	Resource pool
<input type="button" value="Filter: Showing all items"/>						
Request	Payload1	Payload 2	Status code	Error	Timeout	Length
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4627
1	user	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
2	admin	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
3	root	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
4	hello	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
5	martin	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
6	john	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
7	jane	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
8	Kochi	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
9	Shiro	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
10	Kobo	user	200	<input type="checkbox"/>	<input type="checkbox"/>	4627
11	user	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4626



Setelah selesai lihat pada bagian <pre>
 cari yang tidak ada tulisan Username and/or password incorrect. Pada bagian payload 1 merupakan username dan pada payloads 2 merupakan password yang benar
Pada kasus ini:

username : admin
password : password

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	<pre> 	Comment
25	martin	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
26	john	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
27	jane	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
28	Kochi	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
29	Shiro	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
30	Kobo	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
31	user	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
32	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4669	<pre> 	
33	root	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
34	hello	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
35	martin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.
36	john	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4626	<pre> 	Username and/or password incorrect.

Request Response

Pretty Raw Hex

```
1 GET /DWA/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
4 sec-ch-ua-mobile: ?
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71
Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-excha
nge;v=wb3;q=0.7
9 Sec-Fetch-Site: same-origin
```

0 highlights

Finished

Kita coba login dengan username dan password yang didapat



The screenshot shows a web browser window titled 'Vulnerability: Brute Force'. The URL is 127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=password&Login=Login#. The DVWA logo is at the top. On the left is a sidebar with various attack types: Home, Instructions, Setup / Reset DB, Brute Force (highlighted in green), Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, and DVWA Security. The main content area has a 'Login' form with fields for 'Username' and 'Password', and a 'Login' button. Below it, a message says 'Welcome to the password protected area admin' with a small profile picture. A 'More Information' section lists three links: https://owasp.org/www-community/attacks/brute_force_attack, <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>, and <https://www.goinuxcloud.com/brute-force-attack-web-forms>.

Berhasil login!!

2.5. Analisis

menu login ini tidak memiliki batasan kesalahan dalam menginputkan username dan password sehingga serangan seperti brute force attack dapat dilakukan. jika dilihat dari sisi pengguna, kelemahan terdapat pada password yang mudah ditebak dan tidak unik, sehingga dapat dilakukannya brute force attack.



3. Melakukan serangan Command Execution

3.1. Pengenalan Metode

Serangan Command Execution, juga dikenal sebagai Command Injection, adalah serangan keamanan di mana tujuannya adalah untuk menjalankan perintah sembarang pada sistem operasi host melalui aplikasi yang rentan. Serangan ini mungkin terjadi ketika sebuah aplikasi mengirimkan data pengguna yang tidak aman (seperti form, cookies, header HTTP, dll.) ke shell sistem.

Dalam serangan ini, perintah sistem operasi yang disuplai oleh penyerang biasanya dieksekusi dengan hak istimewa dari aplikasi yang rentan. Serangan Command Injection dimungkinkan terutama karena validasi input yang tidak memadai.

Serangan ini berbeda dari Code Injection, di mana Code Injection memungkinkan penyerang untuk menambahkan kode mereka sendiri yang kemudian dieksekusi oleh aplikasi. Dalam Command Injection, penyerang memperluas fungsi default dari aplikasi, yang mengeksekusi perintah sistem, tanpa perlu menyuntikkan kode.

Contoh sederhana dari Command Execution adalah ketika sebuah program menerima nama file sebagai argumen baris perintah, dan menampilkan isi file tersebut kembali ke pengguna. Jika program tersebut dijalankan dengan hak istimewa root, maka panggilan ke sistem() juga akan dijalankan dengan hak istimewa root.

3.2. Mencari Informasi & Vulnerabilities

informasi mengenai serangan Command Execution attack didapatkan dengan link yang ada dalam DVWA. berikut linknya:



More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

untuk menemukan vulnerabilities command injection, kami mencoba satu per satu command untuk mencari kerentanan.

3.3. Vulnerabilities

Pada bagian command injection dengan Tingkat security medium terdapat filterisasi string “&&” dan “;” sehingga command yang tidak mengandung string tersebut dapat dieksekusi. kami mendapatkan vulnerability ini dengan mencoba beberapa kemungkinan command yang dapat dieksekusi.

3.4. Penetration Testing

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Command diatas akan melakukan ping ke Alamat ip dan akan menampilkan nama dari user.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Tidak terjadi apa-apa



Vulnerability: Command Injection

Ping a device

Enter an IP address:

Sekarang kita menggunakan command yang berbeda

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Hasilnya masih sama

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Kita gunakan command yang lain

Vulnerability: Command Injection

Ping a device

Enter an IP address:

www-data

Command berhasil dieksekusi



Vulnerability: Command Injection

Ping a device

Enter an IP address:

Kita coba dengan command lain

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
www-data
PING 120.0.0.1 (120.0.0.1) 56(84) bytes of data.
64 bytes from 120.0.0.1: icmp_seq=1 ttl=243 time=94.0 ms
64 bytes from 120.0.0.1: icmp_seq=2 ttl=243 time=93.9 ms
64 bytes from 120.0.0.1: icmp_seq=3 ttl=243 time=93.5 ms
64 bytes from 120.0.0.1: icmp_seq=4 ttl=243 time=94.0 ms

--- 120.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3013ms
rtt min/avg/max/mdev = 93.536/93.883/94.030/0.203 ms
```

Command bisa dieksekusi

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Coba command lain



Vulnerability: Command Injection

Ping a device

Enter an IP address:

[help](#) [index.php](#) [source](#)

Command berhasil dieksekusi

3.5. Analisis

Berdasarkan hasil percobaan, website tersebut memiliki filterisasi string “&&” dan “;” sehingga command yang tidak mengandung string tersebut dapat dieksekusi.



4. Melakukan serangan CSRF

4.1. Pengenalan Metode

Cross-Site Request Forgery (CSRF) adalah serangan keamanan di mana penyerang memaksa pengguna terotentikasi untuk mengirimkan request yang tidak diinginkan ke aplikasi web tempat mereka telah otentikasi. Ini terjadi ketika aplikasi web tidak memverifikasi apakah request yang diterima benar-benar dimaksudkan oleh pengguna. Serangan ini memanfaatkan kepercayaan yang diberikan situs kepada browser pengguna.

4.2. Mencari Informasi & Vulnerabilities

Note: Browsers are starting to default to setting the [SameSite cookie](#) flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:

- [Chromium](#)
- [Edge](#)
- [Firefox](#)

As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, you could try using other vulnerabilities in this app to store them, the Stored XSS lab would be a good place to start.

More Information

- <https://owasp.org/www-community/attacks/csrf>
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Pengaturan SameSite Cookie:

Browser web sedang beralih untuk mengatur flag cookie Same Site ke "Lax" secara default, yang akan mengurangi efektivitas beberapa jenis serangan CSRF. Pengaturan Same Site membatasi kapan cookie dikirim dengan permintaan lintas-situs, yang meningkatkan keamanan terhadap serangan CSRF. Chromium, Edge, dan Firefox adalah browser yang diumumkan akan menerapkan perubahan ini

Alternatif untuk Serangan CSRF:

Karena pengaturan SameSite yang diperketat akan menurunkan peluang berhasilnya serangan CSRF, disarankan untuk mencari alternatif lain



seperti serangan Stored Cross-Site Scripting (XSS) yang bisa digunakan sebagai titik awal untuk menemukan kerentanan lain di aplikasi tersebut.

Untuk informasi yang lebih lanjut dapat klik laman pada gambar 4.2.

4.3. Vulnerabilities

String kata sandi baru terlihat pada textfield laman url saat pengguna berhasil mengubah kata sandi.

4.4. Penetration Testing

1. Dalam hal ini kita akan mencoba untuk mengubah kata sandi dalam lama DVWA. Ada dua input text field yang dapat diisi.

Vulnerability: Cross Site Request Forgery (CSRF)

The screenshot shows a web page with a light gray background. At the top, it says "Change your admin password:". Below this is a "Test Credentials" button. Underneath are two input fields: one labeled "New password:" and another labeled "Confirm new password:", both with placeholder text. At the bottom is a "Change" button.

2. Masukkan string apa saja untuk mencoba melakukan perubahan kata sandi. Pada kesempatan ini saya mengubah kata sandi menjadi 'josuapane'. Dan



berikut tampilan setelah kata sandi berhasil diubah.

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

3. Terdapat vulnerable yang dapat dimanfaatkan oleh attacker/hacker, dimana pada saat kita berhasil mengubah kata sandi, terlihat kata sandi baru tidak dalam bentuk token.

```
localhost/dvwa/vulnerabilities/csrf/?password_new=josuapane&password_conf=josuapane&Change=Change#
```

Hal ini tentunya dapat dimanfaatkan oleh attacker untuk melakukan aksi merugikan seperti mengubah kata sandi atau melakukan transaksi keuangan tanpa pengetahuan atau persetujuan pengguna.

4. Saya melakukan percobaan untuk mengubah kata sandi, melalui laman url tersebut. Dalam hal ini saya ubah kata sandinya menjadi 'percobaan2'

```
localhost/dvwa/vulnerabilities/csrf/?password_new=percobaan2&password_conf=percobaan2&Change=Change#
```

Muncul Message Berikut



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

That request didn't look correct.

Ternyata hal ini tidak berhasil. Saya tidak berhasil login menggunakan kata sandi 'percobaan2'

Damn Vulnerable Web Application (DVWA)Test Credentials - Google Chrome

localhost/dvwa/vulnerabilities/csrf/test_credentials.php

Test Credentials

Vulnerabilities/CSRF

Wrong password for 'admin'

Username

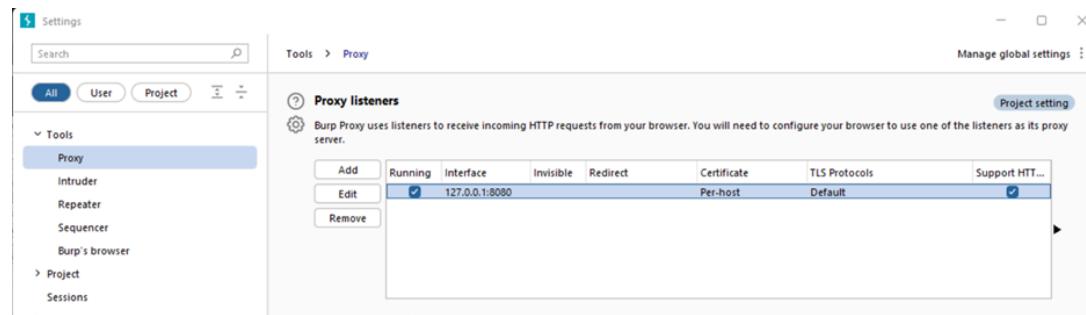
Password

Hal ini terjadi karena laman DVWA mencoba mengambil data referensi dari browser, tetapi browser tidak menyediakan informasi itu. Ini biasa terjadi jika pengguna mengakses halaman langsung atau jika pengaturan privasi browser



mencegah informasi referer dikirim. Hal ini dapat di cegah jika kita menggunakan referer yang sama ketika kita mencoba login, yaitu local host kita.

5. Disini saya menggunakan tool interceptor 'Burp'. Dan mengubah referer saya menjadi local host yaitu : '127.0.0.1'



Dan ketika saya reload ulang halaman web untuk mengubah kata sandinya, saya berhasil mengubah kata sandi menjadi 'percobaan2'

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

4.5. Analisis

Berdasarkan percobaan diatas, dapat ditarik kesimpulan bahwa aplikasi ini rentan terhadap serangan Cross-Site Request Forgery karena ketidadaan token CSRF saat mengubah kata sandi. Tanpa token ini, operasi seperti perubahan kata sandi dapat dimanipulasi oleh penyerang. Percobaan awal



mengganti kata sandi gagal karena DVWA memvalidasi referer, namun dengan menggunakan Burp Suite untuk mengubah referer menjadi localhost, perubahan kata sandi berhasil. Ini menunjukkan bahwa meskipun ada beberapa langkah keamanan, sistem masih dapat dieksloitasi jika penyerang mengetahui cara mengelabui mekanisme validasi referer. Penting untuk memperkuat mekanisme keamanan seperti implementasi token CSRF dan validasi referer yang ketat untuk mencegah serangan semacam ini.



5. Melakukan serangan File Inclusion

5.1. Pengenalan Metode

A file inclusion vulnerability adalah jenis kerentanan web yang paling sering ditemukan untuk memengaruhi aplikasi web yang mengandalkan waktu eksekusi script. Masalah ini disebabkan ketika aplikasi membangun jalur ke kode yang dapat dieksekusi menggunakan variabel yang dikendalikan penyerang dengan cara yang memungkinkan penyerang mengontrol file mana yang dieksekusi pada saat dijalankan. ([wikipedia](#))

5.2. Mencari Informasi & Vulnerabilities

Medium Level

The developer has read up on some of the issues with LFI/RFI, and decided to filter the input. However, the patterns that are used, isn't enough.

Developer telah memperbaiki issue untuk mengakses file pada server dengan cara melakukan filter agar input yang dimasukkan untuk nama page tidak mengakses folder lain selain yang diberikan detail dari hal tersebut dapat dibaca pada link dibawah

More Information

- [Wikipedia - File inclusion vulnerability](#)
- [WSTG - Local File Inclusion](#)
- [WSTG - Remote File Inclusion](#)

5.3. Vulnerabilities

Jikalau mengganti cara mengakses folder menjadi // karena filternya mengurangi / yang awalnya/ akan ter filter menjadi namun jika dijadikan 2(//) maka yang awalnya ../// akan menjadi/ maka kita dapat mengakses file yang terdapat pada server biarpun terdapat filter



5.4. Penetration Testing

Screenshot of DVWA showing a successful file inclusion exploit. The URL is `localhost/DVWA/vulnerabilities/fi/?page=file3.php`. The page displays the content of `file3.php`, which includes the DVWA footer: "Damn Vulnerable Web Application (DVWA)".

Screenshot of DVWA showing a hidden file inclusion exploit. The URL is `localhost/DVWA/vulnerabilities/fi/?page=file4.php`. The page displays the content of `file4.php`, which includes the DVWA footer: "Damn Vulnerable Web Application (DVWA)".

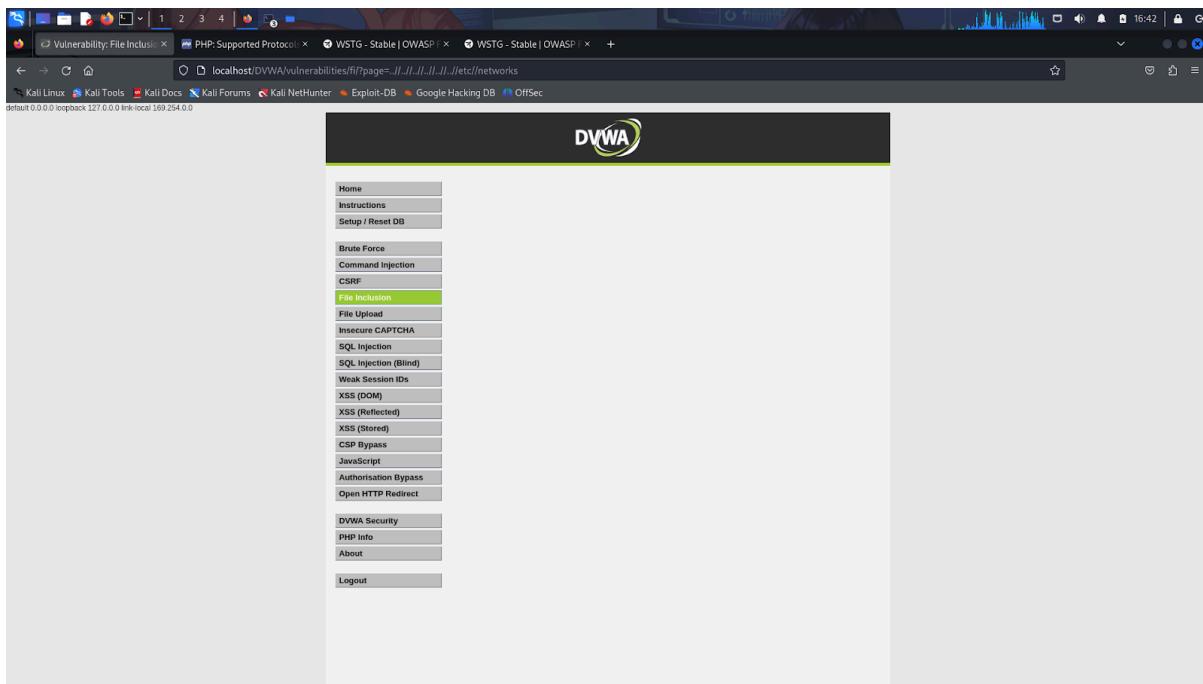


A screenshot of a Kali Linux desktop environment with multiple browser tabs open. The active tab shows a DVWA (Damn Vulnerable Web Application) interface with a sidebar menu. The main content area displays a terminal window with a very long list of file paths, likely demonstrating a file inclusion exploit. The paths include various system files like /etc/passwd, /etc/hosts, and /etc/shadow, along with logs and configuration files from various services.

mengakses file /etc/passwd

A screenshot of a Kali Linux desktop environment with multiple browser tabs open. The active tab shows a DVWA (Damn Vulnerable Web Application) interface with a sidebar menu. The main content area displays a terminal window showing the contents of the /etc/passwd file, which lists user accounts on the system.

mengakses file /etc/hosts untuk mengetahui ip dari user pada server



mengakses /etc/networks untuk mengetahui ip server yang dipakai pada jaringan server

5.5. Analisis

Seperti yang telah saya jelaskan pada poin diatas Vulnerabilities ini dapat terjadi dikarenakan metode akses file dapat terlihat pada address bar tanpa mengganti nama file yang asli dengan link menuju file tersebut ataupun memakai metode lainnya dan hal ini umum terjadi pada website yang tidak di amankan secara penuh dalam kasus ini hanya melakukan filter namun hal tersebut dapat diatasi dengan hanya memberikan akses pada file tertentu dan melakukan cek apakah file yang diakses boleh diakses atau tidak dan mengecek isi file tersebut sebelum di kirim ke client



6. Melakukan serangan SQL Injection

6.1. Pengenalan Metode

SQL Injection adalah sebuah teknik yang menyalahgunakan sebuah celah keamanan yang terjadi dalam lapisan basis data sebuah aplikasi. Cela ini terjadi ketika masukan pengguna tidak disaring secara benar dari karakter-karakter pelolos bentukan string yang diimbuhkan dalam pernyataan SQL atau masukan pengguna tidak bertipe kuat dan karenanya dijalankan tidak sesuai harapan. ([wikipedia](#))

6.2. Mencari Informasi & Vulnerabilities

Medium Level

The medium level uses a form of SQL injection protection, with the function of "`mysql_real_escape_string()`". However due to the SQL query not having quotes around the parameter, this will not fully protect the query from being altered.

The text box has been replaced with a pre-defined dropdown list and uses POST to submit the form.

Dikarenakan input yang diberikan adalah dropdown maka kita tidak dapat menulis script namun hal tersebut dapat diatasi dengan melakukan inspect element dan mengubah value yang dikirimkan dan value yang akan dikirim dan pada hal ini kita mau melakukan SQL inject untuk mencari info apa yang ada pada database, dan untuk informasi lebih lanjut dapat dibaca pada link yang tertera dibawah

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://bobby-tables.com/>



6.3. Vulnerabilities

The screenshot shows a Firefox browser window with the address bar set to `localhost/DVWA/vulnerabilities/sql/`. The main content area displays the DVWA logo and the title "Vulnerability: SQL Injection". On the left, a sidebar menu lists various attack types, with "SQL Injection" currently selected. Below the menu is a "More Information" section containing several links related to SQL injection. The bottom portion of the screenshot shows the browser's developer tools, specifically the "Elements" tab, inspecting the HTML code of the dropdown menu. The code includes a `<select name='id'>` element with multiple options, one of which contains a SQL injection payload (`1 or 1=1 UNION SELECT 1,2;-- $Submit=Submit`). The developer tools also show the CSS styles applied to the select element.

dengan memakai perintah union kita bisa melakukan print semua entry yang terdapat pada table



The screenshot shows a web browser window with the URL localhost/DVWA/vulnerabilities/sql/. The page title is "Vulnerability: SQL Injection". On the left, there's a sidebar with a navigation menu including "Home", "Instructions", "Setup / Reset DB", "Brute Force", "Command Injection", "CSRF", "File Inclusion", "File Upload", "Insecure CAPTCHA", "SQL Injection" (which is highlighted in green), "SQL Injection (Blind)", "Weak Session IDs", "XSS (DOM)", "XSS (Reflected)", "XSS (Stored)", "CSP Bypass", "JavaScript", "Authorisation Bypass", "Open HTTP Redirect", and "DVWA Security". The main content area displays a form with "User ID:" dropdown set to "1" and a "Submit" button. Below the form, several UNION SELECT queries are listed, each showing a different row of data from the database:

- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: admin
Surname: admin
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: John
Surname: Brown
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: Hack
Surname: Me
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: Pablo
Surname: Bob
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: Bob
Surname: Smith
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
First name: 1
Surname: 2

Below the queries, there's a "More Information" section with links to external resources:

- http://en.wikipedia.org/wiki/SQL_injection
- https://www.owasp.org/index.php/SQL_injection_cheat_sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://dbaby-tables.com>

The bottom of the browser window shows the developer tools' DOM tab, highlighting the body element. The CSS tab shows some styles applied to the body:

```
body {  
    margin: 0;  
    color: #000;  
    font: 12px/1.5px Arial, Helvetica, sans-serif;  
    min-width: 980px;  
    height: 100%;  
    position: relative;  
}
```

namun karena tidak menemukan data yang menarik saya mencoba membuka database dan mengetahui bahwa pada database tersebut terdapat table users maka saya mencoba melakukan print table tersebut sebagai uji coba

Name	Type	Schema
Tables (2)		
guestbook		CREATE TABLE
users		CREATE TABLE
user_id	int	"user_id" int I
first_name	text	"first_name" t
last_name	text	"last_name" t
user	text	"user" text D
password	text	"password" te
avatar	text	"avatar" text I
last_login	datetime	"last_login" d
failed_login	int	"failed_login"
Indices (0)		
Views (0)		
Triggers (0)		



Vulnerability: SQL Injection

User ID: 1

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://bobby-tables.com/>

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Surmae: admin

ID: 1 or 1=1 union select user, password from users#

First name: Gordon

Surname: Brown

ID: 1 or 1=1 union select user, password from users#

First name: Hack

Surname: Me

ID: 1 or 1=1 union select user, password from users#

First name: Pablo

Surname: Picasso

ID: 1 or 1=1 union select user, password from users#

First name: Bob

Surname: Smith

ID: 1 or 1=1 union select user, password from users#

First name: admin

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 or 1=1 union select user, password from users#

First name: gordonb

Surname: e99a18c428cb38d5f260853678922e03

ID: 1 or 1=1 union select user, password from users#

First name: 1337

Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 or 1=1 union select user, password from users#

First name: pablo

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 or 1=1 union select user, password from users#

First name: smithy

Surname: 5f4dcc3b5aa765d61d8327deb882cf99



sayangnya passwordnya berada dalam hash sebenarnya masih ada cara lain yaitu memakai database dumping agar password dapat diakses sebagai text dan ada cara lain lagi namun saya disini hanya mau menunjukan vulnerability nya

6.4. Penetration Testing

The screenshot shows a web browser window with the URL `localhost/DVWA/vulnerabilities/sql/`. The page title is "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various exploit categories, and the "SQL Injection" option is highlighted. Below the menu, there is a form with a dropdown labeled "User ID" set to "1" and a "Submit" button. To the right of the form, there is a "More Information" section with several links related to SQL injection. The bottom half of the screen shows the browser's developer tools (Inspector) with the "Elements" tab selected. The DOM tree is expanded to show the injected SQL code: `<option value="1 or 1=1>`. The developer tools also show the CSS styles applied to the page elements.



The screenshot shows a web browser window with the URL localhost/DVWA/vulnerabilities/sql/. The main content area displays a form titled "Vulnerability: SQL Injection" with a dropdown menu set to "1". Below the dropdown is a "Submit" button. To the right of the dropdown, there are several SQL injection payloads:

- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: admin
Surname: admin
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: John
Surname: Brown
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: Hack
Surname: Me
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: Pablo
Surname: Gómez
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: Bob
Surname: Smith
- ID: 1 or 1=1 UNION SELECT 1,2;-- &Submit=Submit.
- First name: 1
Surname: 2

Below the payloads, a "More Information" section lists several links:

- http://en.wikipedia.org/wiki/SQL_injection
- https://www.owasp.org/index.php/Category:SQL_injection
- https://www.owasp.org/www-community/attacks/SQL_Injection
- <https://dbfiddle.uk.com/>

The browser's developer tools are visible at the bottom, showing the HTML structure and CSS styles being applied to the page elements.

6.5. Analisis

Hal ini adalah hal yang normal terjadi pada web server jika database tidak diamankan dengan bagus dan terdapat cara lain untuk melakukan request ke server yaitu dengan cara tidak secara langsung melakukan GET request ke server memakai query ataupun dengan mengatur agar user hanya dapat melakukan query yang telah di define oleh developer



7. Melakukan serangan SQL Injection (Blind)

7.1. Pengenalan Metode

SQL Injection (blind) digunakan ketika aplikasi web rentan terhadap injeksi SQL tetapi hasil injeksi tidak terlihat oleh penyerang. Halaman dengan kerentanan mungkin bukan halaman yang menampilkan data, tetapi akan ditampilkan secara berbeda tergantung pada hasil pernyataan logis yang diinjeksikan ke dalam pernyataan SQL yang sah yang dipanggil untuk halaman tersebut. Jenis serangan ini secara tradisional dianggap memakan banyak waktu karena sebuah pernyataan baru harus dibuat untuk setiap bit yang dipulihkan, dan tergantung pada strukturnya, serangan ini mungkin terdiri dari banyak permintaan yang gagal.[\(wikipedia\)](#)

7.2. Mencari Informasi & Vulnerabilities

Medium Level

The medium level uses a form of SQL injection protection, with the function of "[mysql_real_escape_string\(\)](#)". However due to the SQL query not having quotes around the parameter, this will not fully protect the query from being altered.

The text box has been replaced with a pre-defined dropdown list and uses POST to submit the form.

Dikarenakan ini adalah blind maka query yang dikirimkan tidak akan di return dengan msg yang dapat dipakai oleh hacker maka satu-satunya cara untuk mengetahui apakah terjadi respond atau tidak adalah dengan memakai time based injection untuk mengetahui apakah query berhasil atau tidak, dan informasi lebih lanjut dapat dibaca pada link yang tertera pada dibawah

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- <https://bobby-tables.com/>



7.3. Vulnerabilities

untuk ini saya menyarankan membaca bagian upload terlebih dahulu agar jelas namun intinya adalah sama yaitu mengirimkan request palsu yang telah dimodif dalam hal ini request query dikirim dengan memakai time based command contohnya sleep() untuk mengetahui berapa lama perintah masuk ke server jikalau terjadi sleep contohnya 3 detik maka dapat dipastikan query berhasil oleh karena itu pada blind SQL injection dikarenakan tidak ada return maka proses hacking dilakukan untuk melakukan sesuatu pada database tanpa meminta return dan sleep() dapat digunakan untuk mempermudah proses tersebut sebagai penanda bahwa query berhasil untuk screenshot dapat dilihat pada penetration testing

7.4. Penetration Testing

The screenshot shows a web browser window with the title 'Vulnerability: SQL Injectio...'. The URL is 'localhost/DVWA/vulnerabilities/sql_injection/'. The main content area displays the DVWA logo and the heading 'Vulnerability: SQL Injection (Blind)'. A form field labeled 'User ID' contains the value '1 OR 1=1' and has a 'Submit' button. Below the form, a message box says 'User ID exists in the database.' To the right of the message box, there is a 'More Information' section with a bulleted list of links related to SQL injection. At the bottom left, it shows 'Username: admin', 'Security Level: medium', 'Locale: en_US', and 'SQLi DB: mysql'. At the bottom right, there are 'View Source' and 'View Help' buttons.



OFFENSIVE[®] security

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Scope settings

Filter: Hiding found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Request Response Inspector

Host Method URL Params Status code Length MIME type Title Notes Time requested

http://localhost GET /DVWA/vulnerabilities/sql_injection/ 200 4576 HTML Vulnerability SQL injectio... 17:48:03 30 D.

http://localhost POST /DVWA/vulnerabilities/sql_injection/ 200 4645 HTML Vulnerability SQL injectio... 17:48:03 30 D.

http://localhost GET /DVWA/vulnerabilities/sql_inject... 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/about.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa.css/main.css 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/index.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/login.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/logout.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/page_error.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/recent_logos... 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

http://localhost GET /DVWA/dvwa/testimonials.php 200 4587 HTML Vulnerability SQL injectio... 17:48:04 30 D.

Request Response Inspector

Pretty Raw Hex Render

```

1: POST /DVWA/vulnerabilities/sql_injection/ HTTP/1.1
2: Host: localhost
3: Content-Length: 10
4: Content-Type: application/x-www-form-urlencoded
5: Sec-Ch-Ua: "Chromium";v="119", "Not %A_Brand";v="24"
6: Sec-Ch-Ua-Mobile: ?0
7: Sec-Ch-Ua-Platform: "Linux"
8: Upgrade-Insecure-Requests: 1
9: Origin: http://localhost
10: Content-Type: application/x-www-form-urlencoded
11: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/119.0.6065.199 Safari/537.36
12: 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
13: Sec-Fetch-Site: same-origin
14: Sec-Fetch-Mode: navigate
15: Sec-Fetch-Dest: document
16: Sec-Fetch-User: ?1
17: Referer: http://localhost/DVWA/vulnerabilities/sql_injection/
18: Accept-Encoding: gzip, deflate, br
19: Accept-Language: en-US,en;q=0.9
20: Cookie: PHPSESSID=3t9zohd4gh2f1g; security=medium
21: Connection: close
22: 
```

Submit

Request Response Inspector

Pretty Raw Hex Render

```

1: POST /DVWA/vulnerabilities/sql_injection/ HTTP/1.1
2: Host: localhost
3: Content-Length: 10
4: Content-Type: application/x-www-form-urlencoded
5: Sec-Ch-Ua: "Chromium";v="119", "Not %A_Brand";v="24"
6: Sec-Ch-Ua-Mobile: ?0
7: Sec-Ch-Ua-Platform: "Linux"
8: Upgrade-Insecure-Requests: 1
9: Origin: http://localhost
10: Content-Type: application/x-www-form-urlencoded
11: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/119.0.6065.199 Safari/537.36
12: 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
13: Sec-Fetch-Site: same-origin
14: Sec-Fetch-Mode: navigate
15: Sec-Fetch-Dest: document
16: Sec-Fetch-User: ?1
17: Referer: http://localhost/DVWA/vulnerabilities/sql_injection/
18: Accept-Encoding: gzip, deflate, br
19: Accept-Language: en-US,en;q=0.9
20: Cookie: PHPSESSID=3t9zohd4gh2f1g; security=medium
21: Connection: close
22: 
```

Selected text: `|d=1 AND sleep(3)| AND Submitt=Submit`

Decoded from: URL Encoding

Selected text: `|d=1 AND sleep(3)| AND Submitt=Submit`

Request attributes Request body parameters Request cookies Request headers Response headers

Target: http://localhost



A screenshot of the Burp Suite Community Edition interface. The Request tab shows a POST request to the URL /vulnerabilities/sql_blind/. The request body contains several parameters, including 'id=1 AND sleep(3) AND substr(Substr' and '1 OR 1'. The Response tab shows the server's response, which includes a date header (Date: Sat, 30 Dec 2023 10:48:59 GMT), a status code (HTTP/1.1 200 OK), and a message (There was an error.). The Inspector panel on the right displays various request and response attributes like headers, cookies, and bodies.

7.5. Analisis

Saya kurang tahu banyak mengenai hal ini dan apa yang dapat dilakukan dengannya namun perbedaannya hanya ada atau tidaknya return maka cara mengatasinya sama saja dengan SQL Injection biasa yaitu dengan cara tidak secara langsung melakukan GET request ke server memakai query ataupun dengan mengatur agar user hanya dapat melakukan query yang telah di define oleh developer



8. Melakukan serangan Upload

8.1. Pengenalan Metode

File Upload Vulnerabilities adalah ketika server web mengizinkan pengguna mengunggah berkas ke sistem berkas tanpa memvalidasi hal-hal seperti nama, jenis, isi, atau ukurannya secara memadai. Kegagalan dalam menegakkan pembatasan dengan benar pada hal ini dapat berarti bahwa bahkan fungsi pengunggahan gambar dasar dapat digunakan untuk mengunggah file yang sewenang-wenang dan berpotensi berbahaya. Hal ini bahkan dapat mencakup file skrip sisi server yang memungkinkan eksekusi kode jarak jauh. ([source](#))

8.2. Mencari Informasi & Vulnerabilities

Medium Level

When using the medium level, it will check the reported file type from the client when its being uploaded.

Developer melakukan pemeriksaan tipe file pada client sebelum di upload maka kita dapat mengubah informasi yang akan dikirimkan dikarenakan proses cek tidak dalam server dan server tidak mencoba mengecek gambar yang dikirim dengan cara melakukan pengecekan format kembali di server ataupun melakukan resize image atau pun juga di cek menggunakan algoritma lainnya dan informasi lebih lanjut dapat dibaca pada link yang tertera di bawah

More Information

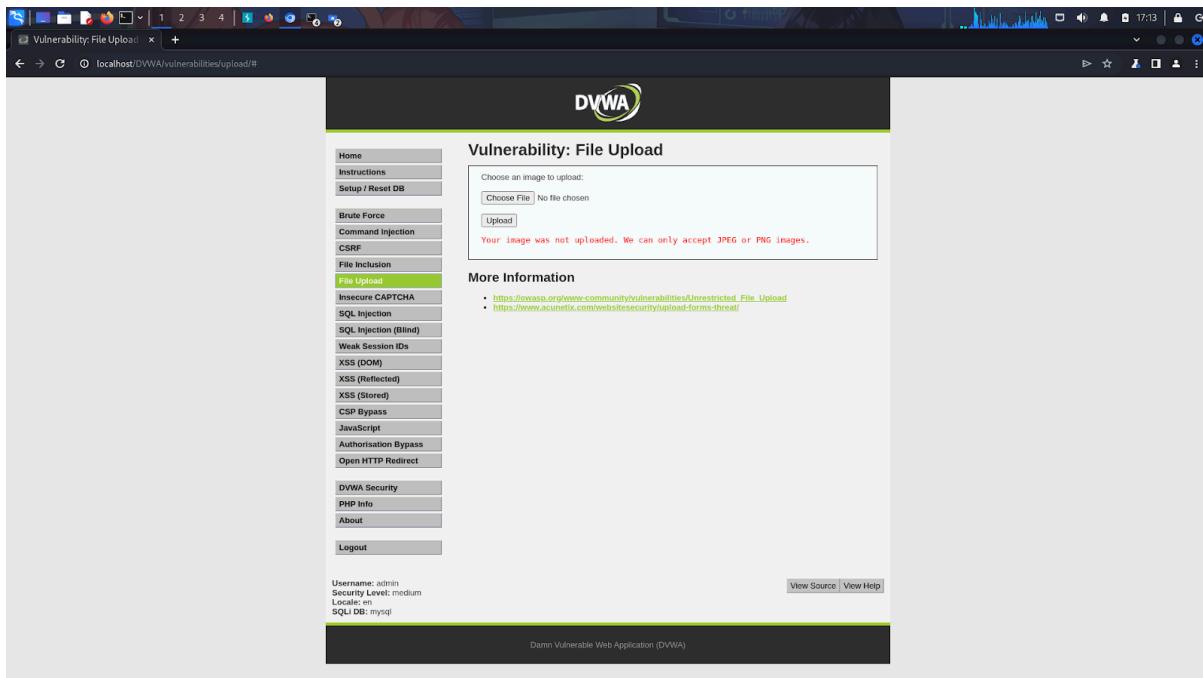
- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- <https://www.acunetix.com/websitetecurity/upload-forms-threat/>

8.3. Vulnerabilities

Maka yang perlu dilakukan hanya mengirimkan request palsu kepada server kalau file telah di cek dan sesuai guidelines dari server yaitu berupa image padahal sebenarnya filenya masih berupa .php dan dapat diakses oleh



client atau dengan kata lain di run oleh client pada server dengan cara di open namun saya tidak melakukannya anggap saja file yang berisi trojan adalah virus.php



A screenshot of a web browser showing the DVWA (Damn Vulnerable Web Application) File Upload page. The URL in the address bar is `localhost/DVWA/vulnerabilities/upload/#`. The page title is "Vulnerability: File Upload". On the left, there's a sidebar menu with various exploit categories like Home, Instructions, Setup / Reset DB, Bruteforce, Command Injection, CSRF, File Inclusion, File Upload (which is highlighted in green), Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorization Bypass, Open HTTP Redirect, DVWA Security, PHP Info, About, and Logout. The main content area has a form titled "Choose an image to upload:" with a "Choose File" button and an "Upload" button. A message at the bottom of the form says "Your image was not uploaded. We can only accept JPEG or PNG images." Below the form, there's a "More Information" section with two links: https://owasp.org/www-community/vulnerabilities/Restricted_File_Upload and <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>. At the bottom right of the content area are "View Source" and "View Help" buttons. At the very bottom of the page, it says "Damn Vulnerable Web Application (DVWA)".

pertama kita perlu melakukan upload file virus agar file request dapat di generate



A screenshot of the Burp Suite Community Edition interface. The top navigation bar includes Project, Intruder, Repeater, View, Help, Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn, and Settings. The main window shows a list of captured requests under the "Target" tab, with a filter applied to hide CSS, binary content, and empty responses. The list includes various requests to "http://localhost" such as "/DVWA/vulnerabilities/upload", "/DVWA/vulnerabilities/fileUpload", and "/DVWA/vulnerabilities/fileUpload". The "Request" section shows the raw HTTP request, and the "Response" section shows the raw HTTP response. To the right, the "Inspector" panel displays detailed information about the selected request, including Request attributes, Request body parameters, Request cookies, Request headers, and Response headers.

berikut adalah file requestnya, yang kita catat pada burpsuite, hal tersebut dapat dilakukan pada wireshark juga, namun burpsuite memiliki fitur untuk melakukan repeat request kepada server, maka berikutnya kita hanya perlu mengubah requestnya agar memenuhi permintaan server yaitu berupa image



A screenshot of the Burp Suite Community Edition interface. The "Repeater" tab is selected. The "Request" pane shows a POST request to "/vulnerabilities/upload" with various headers and a multipart form-data body containing a file named "virus.png". The "Response" pane displays the server's response, which includes an HTML page with a message about the file being successfully uploaded. The "Inspector" pane on the right shows the selected file upload message. The status bar at the bottom indicates "4,440 bytes | 3 millis".

lalu send request dan voila, file virus.php sudah terupload untuk lebih lanjutnya seperti cara membuat backdoor dari trojan dll bukanlah tugas saya karena saya karena hanya diminta untuk melakukan upload



OFFENSIVE security®

8.4. Penetration Testing

The screenshot shows the DVWA (Damn Vulnerable Web Application) File Upload page. The URL is `localhost/DVWA/vulnerabilities/upload/#`. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command injection, CSRF, File Inclusion, File Upload (highlighted in green), Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, DVWA Security, PHP Info, About, and Logout.

The main content area has a title "Vulnerability: File Upload". It contains a form for uploading an image. The "Choose File" input field shows "No file chosen". Below it is an "Upload" button. A message at the bottom states: "Your image was not uploaded. We can only accept JPEG or PNG images."

A "More Information" section provides two links:

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- <https://www.acunitix.com/websitedevelopment/upload-forms-threat/>

At the bottom, there are "View Source" and "View Help" links. The footer says "Damn Vulnerable Web Application (DVWA)".

A screenshot of the Burp Suite Community Edition interface. The "Repeater" tab is selected. A request is shown in the "Request" pane, which is a POST to "/vulnerabilities/upload". The response is displayed in the "Response" pane, showing a successful file upload message. The "Inspector" pane on the right shows the selected response body: ".../vulnerable/uploads/virus.png.php successfully uploaded". The status bar at the bottom indicates "4,440 bytes | 3 millis".

8.5. Analisis

Hal ini dapat terjadi dikarenakan pengecekan file hanya terjadi pada client dan tidak pada server oleh karena itu modified request dapat dikirim, hal ini dapat diatasi dengan 2 cara yaitu melakukan pengecekan nama file sekali lagi pada server dan juga melakukan resize image agar file tersebut benar benar berupa image namun ada langkah extreme untuk ini yaitu melakukan re-encode pada image agar dapat menjadi image baru dikarenakan kode malware masih tetap dapat dimasukkan pada file image normal menggunakan tool seperti GIF_injector([Go, do a crime](#)) ataupun BMPinjector([Go, do a crime](#)) dan masih banyak tool lain untuk melakukannya untuk berbagai format dan inti dari proses upload ini adalah awal dari XSS



9. Melakukan serangan XSS Reflected

9.1. Pengenalan Metode

XSS (Cross-Site Scripting) adalah serangan keamanan web yang memungkinkan penyerang menyisipkan skrip berbahaya ke dalam halaman web yang dilihat oleh pengguna lain. Metode XSS Reflected adalah salah satu bentuk XSS di mana skrip berbahaya disisipkan dan dieksekusi langsung oleh browser pengguna saat membuka halaman yang mengandung payload XSS tersebut.

9.2. Mencari Informasi & Vulnerabilities

Melihat dari textbox yang mengembalikan apapun yang di tulis di situ.

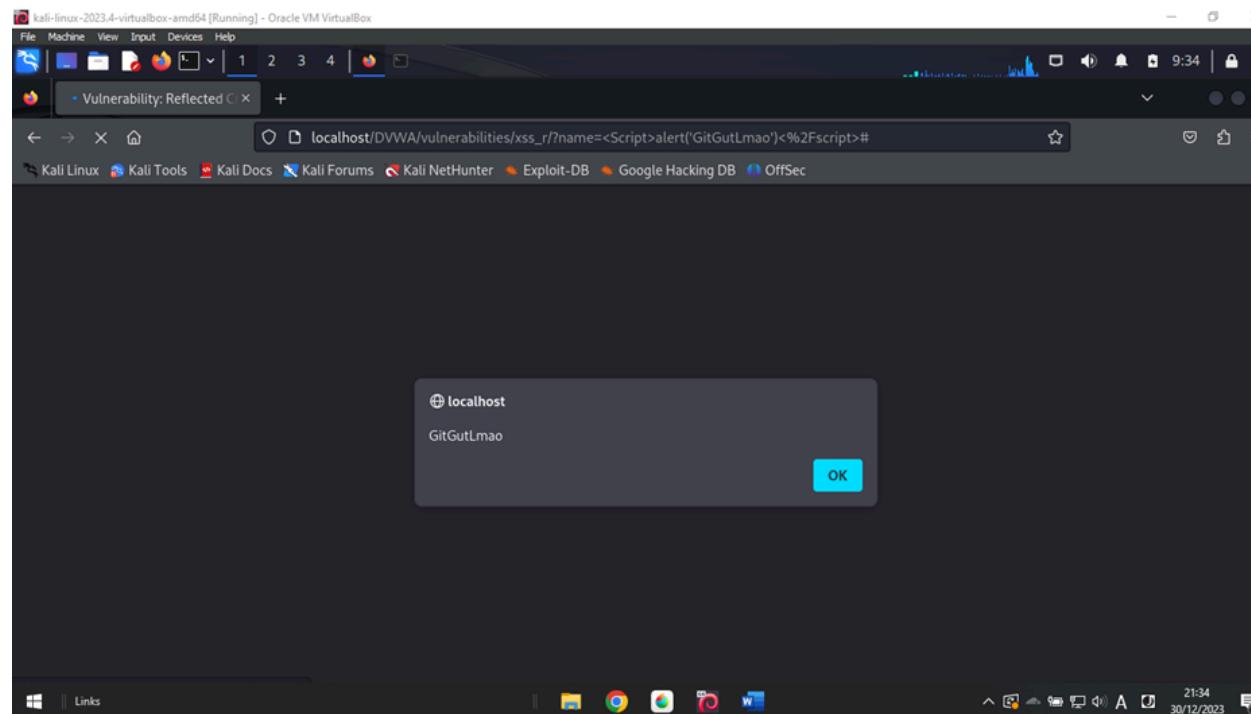
9.3. Vulnerabilities

Dikarenakan kita bisa menulis apa pun di textbox tersebut dan Website hanya melakukan filtering seadanya maka kita bisa memasukan script cth `<Script>alert('GitGutLmao')</script>` ke textbox tersebut dengan mengtweak sedikit dari filter yang di berikan pada level medium di DVWA.



9.4. Penetration Testing

Memasukan Script yang bisa merusak di textbox



9.5. Analisis

Walaupun sudah diberikan filtering kepada textbox yang digunakan itu tetap tidak cukup jika kita cukup mengubah case diluar dari filtering yang diberikan *Level medium.



10. Melakukan serangan XSS Stored

10.1. Pengenalan Metode

XSS Stored adalah jenis serangan keamanan pada aplikasi web di mana penyerang menyisipkan skrip berbahaya ke dalam server dan disimpan di database atau server web. Skrip tersebut kemudian diambil dan dieksekusi setiap kali halaman yang terkait diunduh oleh pengguna yang mengaksesnya. Penyisipan skrip biasanya terjadi melalui formulir atau input pengguna yang tidak diperiksa dengan benar.

10.2. Mencari Informasi & Vulnerabilities

Medium Level

The developer had added some protection, however hasn't done every field the same way.

Ada beberapa bagian atau input dalam aplikasi sudah diterapkan beberapa langkah perlindungan atau validasi tetapi belum diaplikasikan dengan cara yang sama atau seragam di seluruh bagian aplikasi tersebut.

10.3. Vulnerabilities

Lemahnya sanitasi pada form nama.



10.4. Penetration Testing

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: raihan
Message: alert('halo')

Dari gambar diatas bisa diambil bahwa saat menyisipkan script ke dalam form message ,script tersebut tidak berjalan,selanjutnya bagaimana jika menyisipkan kode html ke dalam form message?



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: raihan

Message: alert(\"halo\")

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: adit

Message: halo

Dan dapatkan kode html juga tidak berfungsi ,asumsi yang bisa didapatkan terdapat filtrasi untuk mencegah script/kode injection didalam form message,filter yang sering digunakan adalah kata kunci yang dilarang seperti<script> namun filter tersebut bisa dikelabui dengan mengubah besar/kecil huruf



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: rolan
Message: alert('halo')

Dan hasil yang di dapat script tetap tidak berfungsi ,analisis yang di diperoleh adanya sanitasi pada form message yang membersihkan atau memfilter data masukan pengguna agar tidak mengandung script/kode berbahaya yang dapat di eksekusi pada halaman web

bagaimana dengan form name?



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Saat ingin menginjeksi kode html ke dalam form name ,ada masalah yang di hadapi,yaitu form diatur max length dengan 10 karakter ,untuk menangani hal ini kita dapat merubah ketentuannya dengan membuka inspect element

The screenshot shows the browser's developer tools with the 'Inspector' tab selected. It displays the HTML structure of a guestbook form. The 'Name' field is highlighted with a blue selection bar, showing the input code: <input name="txtName" type="text" size="30" maxlength="100">. The 'Message' field is also visible below it.

```
<div class="vulnerable_code_area">
  <form method="post" name="guestform" "="">
    <table width="550" cellspacing="1" cellpadding="2" border="0">
      <tbody>
        <tr>
          <td width="100">Name *</td>
          <td>
            <input name="txtName" type="text" size="30" maxlength="100">
          </td>
        </tr>
        <tr>
          <td width="100">Message *</td>
          <td>
            <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
          </td>
        </tr>
      </tbody>
    </table>
  </form>
</div>
```

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *



Setelah diubah ketentuan di dengan cara membuka inspect element,kita dapat menginjeksi kode html yang panjangnya lebih dari 10 karakter

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: **halo**
Message: halo bang

Dan hasil yang didapat kode html berfungsi,pada guestbook kata 'halo' dicetak tebal,asumsi yang kita ambil sanitasi pada form name lemah

Mari kita coba menginjeksikan script js

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: **halo**
Message: halo bang



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: halo

Message: halo bang

Name: alert('hai')

Message: halo bang

Namun untuk form name terdapat filter kata kunci yang dilarang , sama halnya seperti form message kita bisa mencoba dengan mengganti dengan huruf besar

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

<Script>alert('halo')</script>

Message *

halo bang

Name: halo

Message: halo bang



Vulnerability: Stored Cross Site Scripting (XSS)

A screenshot of a web application interface. On the left, there is a form with two fields: "Name *" containing "<Script>alert('halo')</script>" and "Message *" containing "halo bang". Below the form is a dark modal window with the text "localhost" and "halo" inside. A blue "OK" button is at the bottom right of the modal. A horizontal line connects the "Message *" field to the "localhost" text in the modal.

Dan dipatkan script js berfungsi.

10.5. Analisis

Analisis yang diperoleh pada form name terdapat sanitasi yang lemah sehingga dapat meninggalkan celah atau rentang untuk serangan keamanan , sebaliknya pada form message terdapat sanitasi yang kuat sehingga pengujian tidak bisa menginjeksikan skrip/kode berbahaya ke dalam database.