

# Lenoos Net Audit

**v1.0.1** — Swiss Army Knife for Network Security & Diagnostics. A comprehensive, all-in-one Bash toolkit for network forensics, security auditing, censorship detection, vulnerability assessment, AI-powered pentesting, and performance stress testing.

## Table of Contents

- [Overview](#)
- [Features](#)
- [Requirements](#)
- [Installation](#)
- [Native Install](#)
- [Docker](#)
- [Usage](#)
- [Syntax](#)
- [All Flags & Parameters](#)
- [Examples & Scenarios](#)
- [Basic Network Audit](#)
- [Full Security Audit](#)
- [DNS & Certificate Analysis](#)
- [OWASP Penetration Test](#)
- [AI-Powered Pentest](#)
- [Stress / Load Testing](#)
- [Brute Force Simulation](#)
- [DDoS Resilience Simulation](#)
- [Full Port Scan & Vulnerability Check](#)
- [Data Breach & Sensitive Data Scan](#)
- [Censorship & DPI Detection](#)
- [Parallel Multi-Target Audit](#)
- [Export Results](#)
- [Custom Export Path](#)
- [Streaming Output](#)
- [Complete Kitchen-Sink Audit](#)
- [Docker Usage Examples](#)
- [Output & Reports](#)
- [Streaming Output](#)
- [Export Formats](#)
- [Architecture](#)
- [Troubleshooting](#)
- [Changelog](#)
- [License](#)

## Overview

---

**lenoos-net-audit.sh** is a single-file Bash script that performs deep-layer network diagnostics, security testing, and performance evaluation. It combines more than **20 specialized modules** into one cohesive tool — from DNS hijack detection and TLS certificate chain validation to OWASP-style penetration tests, AI-driven analysis (via Ollama), multi-vector DDoS simulations, and more.

Key design principles:

- **Zero external services required** — every test runs locally with standard Linux tools.
  - **Modular flags** — pick exactly the tests you need; skip everything else.
  - **Parallel execution** — scan hundreds of targets simultaneously with `-w`.
  - **Rich terminal output** — colour-coded sections, progress bars, histograms, and scorecards.
  - **Multi-format export** — save results as JSON, CSV, HTML, XML, YAML, or PDF (rich report with cover page, ToC, and page numbers).
  - **Custom export path** — specify custom file path and name with `-n` for any export format.
  - **Streaming output** — pipe structured JSON/YAML/XML/HTML/text to stdout for CI/CD, logging, and tool chaining.
  - **Prometheus exporter** — serve `/metrics` on a customizable port for Grafana dashboards with gauges, counters, histograms.
  - **Watch mode** — continuous re-audit at configurable intervals with live Prometheus metric updates.
- 

## Features

---

Category	Module	Flag	Description
Identity	Public IP Check	<code>-i</code>	Discover your public IPv4/IPv6 address
DNS	DNS Audit	<code>-d</code>	A/AAAA records, hijack detection, resolver analysis
DNS	DoH / DoT Audit	<code>-D</code>	DNS-over-HTTPS and DNS-over-TLS connectivity & latency
Routing	MTR Trace	<code>-r</code>	Full hop-by-hop route analysis with loss/latency stats
GeoIP	Geolocation	<code>-g</code>	IP geolocation and ASN lookup
TLS	Certificate Chain	<code>-c</code>	Full chain validation, expiry, OCSP, CT logs
TLS	SNI Audit	<code>-s</code>	SNI probing, ALPN, cipher suite, TLS version details
Censorship	DPI Detection	<code>-t</code>	Deep Packet Inspection fingerprinting (RST, injection, fragmentation)

Category	Module	Flag	Description
Censorship	Bypass Test	-b	Censorship bypass technique detection
Ports	Port Scan	-p <ports>	Targeted TCP/UDP port scan
Ports	Full Port Scan	-P	TCP SYN + UDP + OS detection (all 65535 ports)
Security	OWASP Pentest	-0	17-category OWASP Top-10 style web application pentest
Security	Vulnerability Check	-V	Nmap vuln scripts + online CVE database lookup
Security	Data Breach Audit	-B	5-phase breach exposure analysis
Security	Sensitive Data Scan	-S	JWT, localStorage, XSS, CSRF, PII leakage detection
AI	AI Pentest	-M <spec>	Ollama-powered (local/remote LLM) — configurable model, address, model path, CPU-only
Performance	Speed Test	(auto)	Download throughput measurement
Performance	Stress Test	-T <spec>	Configurable load test: requests, payload size, ramp mode, percentile analysis
Simulation	Brute Force Sim	-F <spec>	Login discovery, protection detection, credential spray, resilience grading
Simulation	DDoS Sim	-X <spec>	Multi-vector wave attacks, recovery test, protection analysis
Parallel	Multi-Target Workers	-W <cores>	Parallel background-job dispatch for multiple targets
Reporting	Advisory	-a	Risk scoring and conclusion matrix
Reporting	Action Plan	-A	Prioritised remediation action plan
Export	Export	-e <fmt>	JSON, CSV, HTML, XML, YAML, PDF
Export	Custom Export Path	-n <path>	Custom file path and name for export output (use with -e )
Stream	Stream Output	-o <fmt>	Real-time structured stream — JSON, YAML, HTML, XML, text (pipe-friendly)

Category	Module	Flag	Description
Monitoring	Prometheus Exporter	-E <port>	Serve <code>/metrics</code> endpoint for Grafana — gauges, histograms, counters
Monitoring	Watch Mode	-w <sec>	Re-run audit at interval — continuous Prometheus metric updates
Setup	Install Dependencies	-j	Auto-install required packages (apt / apk)

## Requirements

### System

- **OS:** Linux (Ubuntu/Debian, Alpine, RHEL/CentOS, Arch)
- **Shell:** Bash 4.0+
- **Privileges:** Root / sudo recommended for SYN scans and raw-socket tests

### Dependencies

Package (Debian/Ubuntu)	Package (Alpine)	Provides
curl	curl	HTTP requests, speed test
dnsutils	bind-tools	dig — DNS queries
knot-dnsutils	knot-utils	kdig — DoH/DoT queries
mtr-tiny	mtr	Route tracing with loss stats
openssl	openssl	TLS/SSL cert chain validation
nmap	nmap	Port scanning, OS detection, vuln scripts
jq	jq	JSON processing
whois	whois	WHOIS lookups
bc	bc	Floating-point maths
gawk	gawk	Advanced text processing
coreutils	coreutils	base64 , od , nproc , timeout
procps	procps	nohup , process tools

### Optional

Package	Purpose
ollama	Local CPU-based LLM for AI pentest (-M flag). Auto-installs at runtime if needed. Not required when using a remote Ollama server.

## PDF Backends (for -e pdf )

The script automatically detects and uses the best available PDF backend:

Backend	Priority	Notes
wkhtmltopdf	1st (preferred)	Best quality — headers, footers, page numbers. Auto-installed by -j .
google-chrome / chromium	2nd	Headless mode --print-to-pdf . Pre-installed in Docker image.
weasyprint	3rd	Python-based alternative.

If no PDF backend is found, the HTML source is preserved for manual conversion.

## Ollama Configuration

The -M flag accepts a configuration spec: MODEL[:URL[:PATH]]

- **MODEL** — Name of the LLM to use (default: `mistral` ). Must be a model available in the [Ollama library](#).
- **URL** — Address of the Ollama server (default: `http://127.0.0.1:11434` ). Set to a remote address to offload AI processing to another machine.
- **PATH** — Local directory for model storage (default: `~/ollama/models` ). Useful for persistent Docker volumes or shared NFS mounts.

All inference is **CPU-only**. The script sets `CUDA_VISIBLE_DEVICES=""` and `OLLAMA_NUM_GPU=0` to enforce this.

## Installation

### Native Install

```
# 1. Clone / download the script
git clone <repo-url> && cd lenoos-net-audit

# 2. Make executable
chmod +x lenoos-net-audit.sh

# 3. Auto-install dependencies (Debian/Ubuntu or Alpine)
sudo ./lenoos-net-audit.sh -j

# 4. Run an audit
sudo ./lenoos-net-audit.sh example.com -d -c -r -a
```

## Docker

```
# Build the image
docker build -t lenoos-net-audit .

# Run a scan
docker run --rm lenoos-net-audit example.com -d -c -r -a

# Run with full security audit
docker run --rm lenoos-net-audit example.com -d -c -r -0 -B -S -P -V -a -A -e json

# Interactive shell for manual runs
docker run --rm -it --entrypoint bash lenoos-net-audit
```

## Usage

### Syntax

```
lenoos-net-audit.sh [OPTIONS] <target1> [target2 ... targetN]
```

Targets can be domain names ( `example.com` ) or IP addresses ( `93.184.216.34` ). When multiple targets are given, they are processed sequentially — or in parallel if `-w` is specified.

### All Flags & Parameters

#### General

Flag	Argument	Description
<code>-i</code>	—	Show public IP address (IPv4 & IPv6)
<code>-j</code>	—	Install all required system dependencies
<code>-4</code>	—	Force IPv4 only
<code>-6</code>	—	Force IPv6 only
<code>-u</code>	—	Enable UDP protocol for applicable tests

#### Network Diagnostics

Flag	Argument	Description
<code>-d</code>	—	DNS audit — A/AAAA records, hijack & resolver analysis
<code>-D</code>	—	DoH (DNS-over-HTTPS) and DoT (DNS-over-TLS) audit

Flag	Argument	Description
-r	—	MTR route trace — hop-by-hop loss & latency
-g	—	GeolP and ASN lookup
-c	—	TLS certificate chain validation, expiry, OCSP
-s	—	SNI audit — ALPN negotiation, cipher suite, TLS version
-t	—	DPI (Deep Packet Inspection) detection & fingerprinting
-b	—	Censorship bypass technique detection
-p	<port_list>	Scan specific ports (comma-separated, e.g. 80,443,8080 )

## Security & Pentesting

Flag	Argument	Description
-0	—	OWASP-style pentest (17 categories)
-B	—	Data breach exposure audit (5 phases)
-S	—	Sensitive data scan (JWT, XSS, CSRF, PII, localStorage)
-P	—	Full port scan — TCP SYN + UDP + OS fingerprint (all 65535 ports)
-V	—	Vulnerability check — nmap NSE scripts + online CVE lookup
-M	<model[:url[:path]]>	<b>AI pentest</b> — model = LLM name, url = Ollama server address, path = model storage dir. CPU-only.

## Performance & Simulation

Flag	Argument	Default	Description
-T	<N[:L[:M]]>	100	<b>Stress test</b> — N requests, L payload KB, M mode: fixed , random , ramp
-F	<A[:D[:W]]>	20:100:50	<b>Brute force sim</b> — A attempts per phase, D delay ms, W workers
-X	<W[:C[:S]]>	5:50:30	<b>DDoS sim</b> — W wave count, C concurrent connections, S sustain seconds
-W	<cores>	nproc	Parallel worker threads for multi-target dispatch

## Reporting & Export

Flag	Argument	Description
-a	—	Risk advisory with conclusion matrix
-A	—	Prioritised remediation action plan
-e	<format>	Export results — json , csv , html , xml , yaml , pdf
-n	<path>	Custom export file path/name (use with -e ). Auto-appends extension if missing. Creates parent directories.
-o	<format>	Stream output — json , yaml , html , xml , text (pipe to stdout or file)
-E	<port>	Prometheus metrics exporter on given port (default 9101 ). Serves /metrics
-w	<seconds>	Watch mode — re-run audit every N seconds (minimum 5). Use with -E for Grafana

## Examples & Scenarios

### 1. Basic Network Audit

Quick DNS + routing + certificate check with advisory:

```
sudo ./lenoos-net-audit.sh example.com -d -r -c -a
```

### 2. Full Security Audit

Comprehensive security audit of a web application:

```
sudo ./lenoos-net-audit.sh example.com -d -c -s -0 -B -S -P -V -a -A
```

This runs: DNS audit, cert chain, SNI analysis, OWASP pentest, breach audit, sensitive data scan, full port scan, vulnerability check, advisory, and action plan.

### 3. DNS & Certificate Analysis

Deep DNS and TLS inspection including DoH/DoT:

```
sudo ./lenoos-net-audit.sh example.com -d -D -c -s
```

### 4. OWASP Penetration Test

Run only the 17-category OWASP-style pentest:

```
sudo ./lenoos-net-audit.sh https://target-app.com -0
```

Covers: Broken Access Control, Cryptographic Failures, Injection, Insecure Design, Security Misconfiguration, Vulnerable Components, Authentication Failures, Data Integrity, Logging Failures, SSRF, and more.

## 5. AI-Powered Pentest

Use a local or remote Ollama LLM for intelligent security analysis.

The `-M` flag accepts a spec: `MODEL[:URL[:PATH]]`

Component	Description	Default
MODEL	LLM model name	<code>tinyllama</code>
URL	Ollama server address	<code>http://127.0.0.1:11434</code>
PATH	Directory to store/read models	Ollama default ( <code>~/.ollama/models</code> )

**Popular CPU-optimised models:** `tinyllama`, `mistral`, `llama3`, `phi3`, `gemma2`, `qwen2`

```
# Default: local Ollama, tinyllama model (lightweight, fast)
sudo ./lenoos-net-audit.sh -M tinyllama target-app.com

# Use a larger model for deeper analysis
sudo ./lenoos-net-audit.sh -M mistral target-app.com

# Use llama3
sudo ./lenoos-net-audit.sh -M llama3 target-app.com

# Remote Ollama server on a different machine
sudo ./lenoos-net-audit.sh -M tinyllama:http://10.0.0.5:11434 target-app.com

# Remote server + custom model storage path
sudo ./lenoos-net-audit.sh -M llama3:http://10.0.0.5:11434:/data/ai-models target-app.com

# Custom local model storage directory
sudo ./lenoos-net-audit.sh -M mistral:/mnt/ssd/ollama-models target-app.com
```

The AI pentest includes: - Automatic API endpoint discovery - File upload vulnerability testing - Brute force detection - OWASP A01–A10 scenario analysis - AI-generated scorecard and recommendations - **CPU-only enforcement** — GPU is automatically disabled  
`( CUDA_VISIBLE_DEVICES="" , OLLAMA_NUM_GPU=0 )`

**Note:** When using a remote URL, the script communicates via Ollama's HTTP API — no local installation is needed. The model will be pulled on the remote server if not already available.

## 6. Stress / Load Testing

**Basic** — 100 requests with defaults:

```
sudo ./lenoos-net-audit.sh https://api.example.com -T
```

**Custom** — 500 requests, 64 KB payload, ramp mode:

```
sudo ./lenoos-net-audit.sh https://api.example.com -T 500:64:ramp
```

**Heavy load** — 2000 requests, 128 KB payload, random mode:

```
sudo ./lenoos-net-audit.sh https://api.example.com -T 2000:128:random
```

The stress test produces:

- Response time percentiles (p50, p90, p95, p99)
- ASCII histograms
- Throughput (RPS) analysis
- 6-phase detailed report with grade (A+ to F)

**7. Brute Force Simulation**

Test login endpoint resilience against credential attacks:

**Default** (20 attempts, 100 ms delay, 50 workers):

```
sudo ./lenoos-net-audit.sh https://target-app.com -F
```

**Aggressive** (100 attempts, 50 ms delay, 100 workers):

```
sudo ./lenoos-net-audit.sh https://target-app.com -F 100:50:100
```

Phases: Login form discovery → Protection detection → Credential spray → Results analysis → Grade & advisory.

**8. DDoS Resilience Simulation**

Evaluate target's resilience to distributed denial-of-service:

**Default** (5 waves, 50 concurrent, 30 s sustain):

```
sudo ./lenoos-net-audit.sh https://target-app.com -X
```

**Heavy simulation** (10 waves, 200 concurrent, 60 s sustain):

```
sudo ./lenoos-net-audit.sh https://target-app.com -X 10:200:60
```

Phases: Baseline → Multi-vector waves → Recovery test → Protection analysis → Resilience grade → Comprehensive advisory.

**9. Full Port Scan & Vulnerability Check**

Scan all 65535 ports and check for known vulnerabilities:

```
sudo ./lenoos-net-audit.sh example.com -P -V
```

Or scan specific ports only:

```
sudo ./lenoos-net-audit.sh example.com -p 22,80,443,3306,5432,8080 -V
```

## 10. Data Breach & Sensitive Data Scan

Check for data exposure and sensitive information leakage:

```
sudo ./lenoos-net-audit.sh https://target-app.com -B -S
```

Detects: Exposed credentials, JWT tokens, localStorage/sessionStorage secrets, XSS vectors, CSRF tokens, PII patterns.

## 11. Censorship & DPI Detection

Detect deep packet inspection and censorship mechanisms:

```
sudo ./lenoos-net-audit.sh blocked-site.com -t -b -d -D
```

Tests for: TCP RST injection, HTTP response injection, packet fragmentation, and censorship bypass techniques.

## 12. Parallel Multi-Target Audit

Audit 5 targets simultaneously using 4 worker threads:

```
sudo ./lenoos-net-audit.sh -W 4 site1.com site2.com site3.com site4.com site5.com -d -c -r -a
```

## 13. Export Results

Export a full audit to different formats:

```
# JSON
sudo ./lenoos-net-audit.sh example.com -d -c -r -a -e json

# HTML report
sudo ./lenoos-net-audit.sh example.com -d -c -0 -a -A -e html

# CSV data
sudo ./lenoos-net-audit.sh example.com -d -c -r -e csv

# XML
sudo ./lenoos-net-audit.sh example.com -d -c -a -e xml
```

```
# YAML
sudo ./lenoos-net-audit.sh example.com -d -c -a -e yaml

# PDF – rich report with cover page, table of contents, page numbers
sudo ./lenoos-net-audit.sh example.com -drtcabgsA0BSPV -e pdf
```

## 14. Custom Export Path

Use `-n` to specify a custom file path and name for any export:

```
# Export JSON to a specific path (auto-appends .json)
sudo ./lenoos-net-audit.sh example.com -d -c -e json -n /tmp/reports/myreport

# Export PDF with a custom name
sudo ./lenoos-net-audit.sh example.com -drtcabgs -e pdf -n ./reports/audit-2026

# Export to exact filename (extension already present)
sudo ./lenoos-net-audit.sh example.com -d -e csv -n output.csv

# Creates parent directories automatically
sudo ./lenoos-net-audit.sh example.com -d -e html -n /opt/audits/weekly/report.html
```

Without `-n`, files are saved to the `exports/` subdirectory as `lenoos-audit-YYYYMMDD_HHMMSS.  
<format>`.

## PDF Branding ( `pdf.conf` )

Customize PDF report appearance by creating a `pdf.conf` file. The script searches these locations (in order):

1. `./pdf.conf` (current working directory)
2. `<script_dir>/pdf.conf` (same folder as `lenoos-net-audit.sh`)
3. `~/.config/lenoos/pdf.conf`

### Supported fields:

Key	Description
<code>PDF_LOGO</code>	Path to a PNG or SVG logo (embedded as base64 in the cover page)
<code>PDF_BRAND</code>	Brand / company name displayed on cover page and headers
<code>PDF_AUTHOR</code>	Report author name
<code>PDF_FILENAME</code>	Custom filename template. Placeholders: <code>{targets}</code> , <code>{date}</code>
<code>PDF_WEBSITE</code>	Company website URL
<code>PDF_EMAIL</code>	Contact email address
<code>PDF_PHONE</code>	Contact phone number
<code>PDF_CONTACT_PERSON</code>	Name of the contact person

Key	Description
PDF_TEST_ENV	Description of test environment (e.g., "Production", "Staging")
PDF_LAB_DETAILS	Lab / infrastructure details
PDF_REF_BASE_URL	Base URL for QR code — encodes <base_url>/<filename> on the cover page

A sample `pdf.conf` is included in the repository.

## 15. Streaming Output

Stream structured output in real-time as each module completes — ideal for piping into other tools, logging systems, or CI/CD pipelines.

**Piping** — when `stdout` is a pipe, the stream goes to `stdout` and terminal output goes to `stderr`:

```
# Stream JSON and pipe to jq for pretty-printing
sudo ./lenoos-net-audit.sh -o json -d -c example.com | jq .

# Stream YAML to a file
sudo ./lenoos-net-audit.sh -o yaml -d -r example.com > audit-stream.yaml

# Stream XML and grep for failures
sudo ./lenoos-net-audit.sh -o xml -d -0 example.com | grep '<severity>fail</severity>'

# Plain text stream (ANSI stripped) for logging
sudo ./lenoos-net-audit.sh -o text -d -c -r example.com > audit.log 2>/dev/null

# JSON stream to jq, extract only fail events
sudo ./lenoos-net-audit.sh -o json -d -0 -c example.com | jq '.events[] | select(.severity=="fa'
```

**File mode** — when `stdout` is a terminal, the stream is saved to a file automatically:

```
# Terminal shows colored output; stream saved to lenoos-stream-* .json
sudo ./lenoos-net-audit.sh -o json -d -c -r example.com

# Combine with export: -e for summary file, -o for detailed stream
sudo ./lenoos-net-audit.sh -o json -e html -d -c -0 example.com
```

### Stream formats:

Format	Flag	Description
JSON	<code>-o json</code>	Array of event objects with seq, timestamp, target, module, severity, message
YAML	<code>-o yaml</code>	Structured event list — human-readable and machine-parseable

Format	Flag	Description
HTML	-o html	Styled event cards — open in browser for live review
XML	-o xml	XML events — integrate with SIEM, Splunk, ELK pipelines
text	-o text	Plain text (ANSI stripped) — clean logs for grep, awk, sed

### Event structure (JSON example):

```
{
  "stream": true,
  "generated": "2026-02-19T10:30:00+00:00",
  "version": "v1.0.1",
  "events": [
    {
      "seq": 1,
      "timestamp": "2026-02-19T10:30:01+00:00",
      "target": "example.com",
      "module": "dns",
      "type": "block",
      "data": "DNS audit results..."
    }
  ],
  "total_events": 12,
  "completed": "2026-02-19T10:31:45+00:00"
}
```

## 16. Complete Kitchen-Sink Audit

Run every module on a target with action plan and HTML export:

```
sudo ./lenoos-net-audit.sh https://example.com \
-i -d -D -r -g -c -s -t -b \
-p 22,80,443,8080 -P -V \
-O -B -S -M tinyllama \
-T 200:32:ramp \
-F 30:100:50 \
-X 5:50:30 \
-a -A -e html -n full-audit-report
```

---

## Docker Usage Examples

---

```
# Build once
docker build -t lenoos-net-audit .

# Quick DNS check
docker run --rm lenoos-net-audit example.com -d

# Full security audit with HTML export (mount volume for exports)
docker run --rm -v $(pwd)/exports:/opt/lenoos-net-audit/exports \
```

```

lenoos-net-audit example.com -d -c -O -B -S -V -a -A -e html

# Stress test
docker run --rm lenoos-net-audit https://api.example.com -T 500:64:ramp

# AI pentest with custom model and model volume
docker run --rm -v ollama-models:/opt/lenoos-net-audit/models \
    lenoos-net-audit -M tinyllama target-app.com

# AI pentest pointing to a remote Ollama server
docker run --rm lenoos-net-audit -M tinyllama:http://192.168.1.100:11434 target-app.com

# Stream JSON output from Docker to host pipe
docker run --rm lenoos-net-audit -o json -d -c example.com | jq .

# Stream YAML to a file on the host
docker run --rm lenoos-net-audit -o yaml -d -r -O example.com > audit-stream.yaml

# Parallel scan of multiple targets
docker run --rm lenoos-net-audit -W 4 sitel.com site2.com site3.com -d -c -r

# Interactive mode
docker run --rm -it --entrypoint bash lenoos-net-audit

# With network host mode (recommended for accurate results)
docker run --rm --net=host lenoos-net-audit example.com -d -r -c -P -V -a

```

**Tip:** Use `--net=host` for the most accurate network measurements and to avoid Docker's NAT layer affecting MTR traces and port scans.

## Output & Reports

The script produces colour-coded terminal output with:

- **Section headers** — clearly delineated test phases
- **Progress indicators** — real-time feedback during long-running scans
- **Scorecards** — letter grades (A+ through F) for security and performance modules
- **Conclusion matrix** — dynamic aggregated summary showing only enabled module columns
- **Histograms** — ASCII bar charts for latency distribution (stress test)
- **Percentile tables** — p50/p90/p95/p99 response times

## Streaming Output

The `-o` flag enables **real-time structured streaming**. Unlike `-e` (which exports a summary at the end), `-o` captures each module's full output as it runs and emits it as structured events.

Feature	<code>-e</code> (Export)	<code>-o</code> (Stream)
When	After all scans complete	Real-time, per-module

Feature	-e (Export)	-o (Stream)
Content	Summary metrics only	Full module output
Pipe-friendly	No (writes to file)	Yes (stdout when piped)
Formats	json, csv, html, xml, yaml, pdf	json, yaml, html, xml, text
Use case	Final report	CI/CD, logging, live monitoring

Both can be used together: `-o json -e html` streams JSON in real-time AND saves an HTML summary.

---

## Export Formats

---

Format	Flag	Description
JSON	<code>-e json</code>	Machine-readable structured data — ideal for CI/CD pipelines
CSV	<code>-e csv</code>	Comma-separated values — import into Excel or databases
HTML	<code>-e html</code>	Styled report with embedded CSS — share via browser
XML	<code>-e xml</code>	XML document — integrate with enterprise SIEM tools
YAML	<code>-e yaml</code>	Human-friendly structured format — for config management
PDF	<code>-e pdf</code>	Rich report with cover page, ToC, page numbers, captured console output, system info. Supports <code>wkhtmltopdf</code> , Chrome/Chromium headless, and <code>weasyprint</code> backends (auto-detected).

All formats support custom output path via `-n <path>`.

---

## Prometheus Exporter & Watch Mode

---

The `-E <port>` flag starts a lightweight Prometheus-compatible HTTP metrics exporter. Combined with `-w <seconds>`, the suite enters **watch mode** — continuously re-running audits and updating metrics for Grafana dashboards.

### Quick Start

```
# One-shot audit with Prometheus metrics on port 9101
sudo ./lenoos-net-audit.sh -E 9101 -drc google.com

# Watch mode: re-audit every 5 minutes, serve metrics on port 9200
sudo ./lenoos-net-audit.sh -E 9200 -w 300 -drtcabgs google.com youtube.com

# Verify metrics (from another terminal)
curl -s http://localhost:9101/metrics | head -30
```

## Exposed Metrics

Metric	Type	Labels	Description
lenoos_audit_info	Gauge	version , hostname , os , kernel	Suite metadata (always 1)
lenoos_audit_runs_total	Counter	—	Total audit cycles completed
lenoos_audit_duration_seconds	Gauge	—	Duration of last audit run
lenoos_audit_targets	Gauge	—	Number of targets
lenoos_audit_workers	Gauge	—	Parallel workers configured
lenoos_dns_ok	Gauge	target	DNS OK=1, FAIL/HIJACK=0
lenoos_mtr_loss_percent	Gauge	target	MTR packet loss %
lenoos_cert_days_remaining	Gauge	target	TLS cert days to expiry
lenoos_speed_mbps	Gauge	target	Download speed MB/s
lenoos_dpi_score	Gauge	target	DPI score (0=clean)
lenoos_dpi_RST	Gauge	target	TCP RST detected (0/1)
lenoos_dpi_inject	Gauge	target	HTTP inject (0/1)

Metric	Type	Labels	Description
lenoos_dpi_frag	Gauge	target	Frag interference (0/1)
lenoos_dpi_status_info	Gauge	target , level	DPI status label
lenoos_ports_open	Gauge	target	Open ports found
lenoos_ports_closed	Gauge	target	Closed/filtered ports
lenoos_sensitive_score	Gauge	target	Sensitive data score
lenoos_fullscan_ports	Gauge	target	Full scan discovered ports
lenoos_vuln_hits	Gauge	target	CVE hits
lenoos_ai_score	Gauge	target	AI risk score
lenoos_ai_grade_info	Gauge	target , grade	AI grade label
lenoos_stress_rps	Gauge	target	Stress test RPS
lenoos_stress_grade_info	Gauge	target , grade	Stress grade label
lenoos_brute_score	Gauge	target	Brute force score
lenoos_brute_grade_info	Gauge	target , grade	Brute force grade label
lenoos_ddos_score	Gauge	target	DDoS resilience score
lenoos_ddos_grade_info	Gauge	target , grade	DDoS grade label
lenoos_bypass_ok	Gauge	target	Bypass succeeded (0/1)
lenoos_sni_info	Gauge	target , tls_version , alpn , cipher , status	SNI metadata

Metric	Type	Labels	Description
lenoos_audit_duration_seconds_histogram_*	Histogram	—	Audit duration distribution
lenoos_dpi_score_histogram_*	Histogram	—	DPI score across targets
lenoos_mtr_loss_percent_histogram_*	Histogram	—	MTR loss distribution
lenoos_cert_days_remaining_histogram_*	Histogram	—	Cert days distribution
lenoos_speed_mbps_histogram_*	Histogram	—	Speed distribution

## Grafana Integration

1. Add Prometheus data source pointing to your Prometheus server
2. Configure Prometheus `scrape_configs` to include the exporter:

```
# prometheus.yml
scrape_configs:
  - job_name: 'lenoos-audit'
    scrape_interval: 60s
    static_configs:
      - targets: ['<host>:9101']
```

1. Import or create dashboards using `lenoos_*` metrics
2. Suggested panels:
3. **Gauge**: `lenoos_cert_days_remaining` (cert expiry countdown)
4. **Stat**: `lenoos_dns_ok` (DNS health per target)
5. **Graph**: `lenoos_mtr_loss_percent` over time (watch mode)
6. **Heatmap**: `lenoos_dpi_score` across targets
7. **Bar Chart**: `lenoos_ports_open` per target
8. **Table**: `lenoos_sni_info` with all TLS labels

## Docker with Prometheus

```
docker run -d --name lenoos-prom -p 9101:9101 \
lenoos-net-audit -E 9101 -w 300 -drc google.com
```

## Architecture

```
lenoos-net-audit.sh (single file, ~7700+ lines)
|
├── Configuration          (pdf.conf branding, exports/ directory)
└── Identity & Setup     (-i, -j)
```

DNS Module	( -d, -D)
Routing Module	( -r, -g)
TLS/SSL Module	( -c, -s)
Censorship Module	( -t, -b)
Port Scanning	( -p, -P)
Security Testing	
OWASP Pentest	( -0) 17 categories
Vuln Check	( -V) NSE + CVE
Breach Audit	( -B) 5 phases
Sensitive Scan	( -S) JWT/XSS/CSRF/PII
AI Pentest	( -M) Ollama LLM
Performance Testing	
Speed Test	( auto)
Stress Test	( -T) 6-phase analysis
Brute Force Sim	( -F) 5-phase simulation
DDoS Sim	( -X) 6-phase simulation
Reporting	
Advisory	( -a) Dynamic conclusion matrix
Action Plan	( -A) Remediation steps
Export	( -e) 6 formats + custom path ( -n)
Stream Output	( -o) 5 formats, pipe-friendly
Monitoring	
Prometheus	( -E) /metrics endpoint
Watch Mode	( -w) Continuous re-audit
Parallel Dispatch	( -W) Multi-target workers

## Troubleshooting

Problem	Solution
command not found: dig	Run <code>sudo ./lenoos-net-audit.sh -j</code> or install <code>dnsutils</code> manually
command not found: kdig	Install <code>knot-dnsutils</code> (Debian) or <code>knot-utils</code> (Alpine)
Nmap requires root	Run with <code>sudo</code> for SYN scans and OS detection
AI pentest fails ( - M )	Ensure Ollama is installed and running: <code>ollama serve &amp;</code> , or use a remote URL: <code>-M tinyllama:http://remote:11434</code>
AI model download slow	Default <code>tinyllama</code> is fast; for larger models pre-pull: <code>ollama pull mistral</code>
Custom model path denied	Ensure the directory exists and is writable; use <code>-M model:/writable/path</code>
Stream output empty	Ensure <code>-o</code> has a valid format: <code>json</code> , <code>yaml</code> , <code>html</code> , <code>xml</code> , <code>text</code>
Stream + pipe not working	Pipe captures stream; terminal output goes to stderr. Use <code>2&gt;/dev/null</code> to suppress terminal

Problem	Solution
Stress test timeouts	Increase timeout or reduce request count in <code>-T</code> spec
Docker: MTR shows ???	Use <code>--net=host</code> for accurate route tracing
Export file not created	Check write permissions in the current directory, or use <code>-n</code> to specify a writable path
Custom export path fails	Ensure parent directory exists or is creatable; the script runs <code>mkdir -p</code> automatically
PDF generation failed	Install a PDF backend: <code>wkhtmltopdf</code> (preferred), <code>chromium</code> , or <code>weasyprint</code> . Run <code>-j</code> to auto-install.
IPv6 tests fail	Ensure IPv6 is enabled on your network, or use <code>-4</code> to force IPv4

## Changelog

---

See [CHANGES.md](#) for a detailed changelog following [Semantic Versioning](#).

**Current version:** v1.0.1

---

## License

---

This project is provided for **educational and authorized security testing purposes only**. Always obtain proper authorization before scanning or testing targets you do not own.

---

*Built with standard Linux tools — no cloud dependencies, no API keys, no subscriptions.*