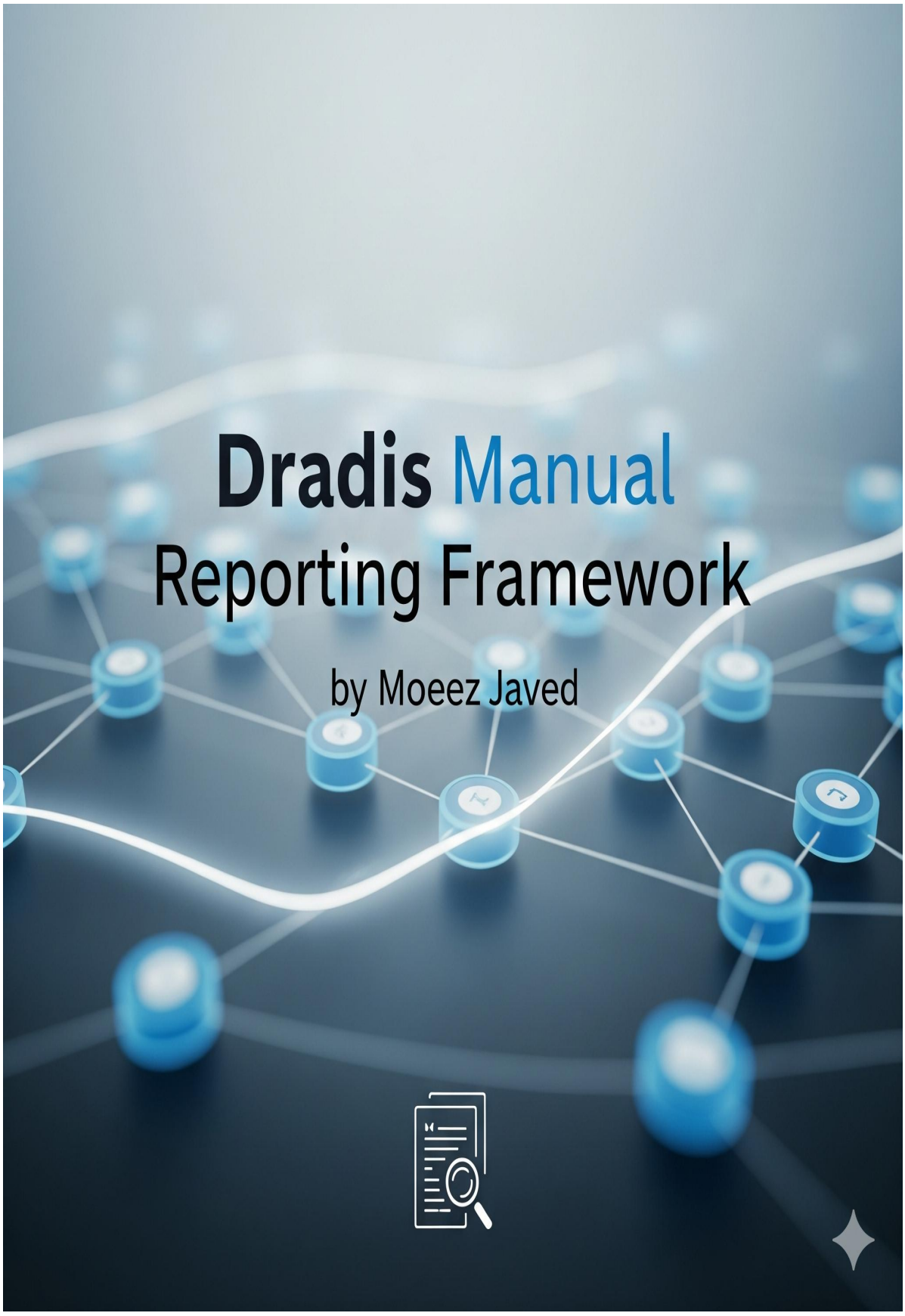

Dradis Manual
Reporting Framework
by Moeez Javed

Dradis Manual

Audience: Students, interns, and junior security practitioners learning collaborative reporting and evidence consolidation for security assessments.

Purpose: This manual teaches what Dradis is, why teams use it, how to install and run Dradis (Community Edition) in a lab, and how to use it step-by-step to import scanner output, collaborate on findings, and generate consistent reports. It focuses on safe, authorized lab usage and defender-friendly workflows.

1 — Short introduction

Dradis is an open-source collaboration and reporting framework for security teams. It lets you centralize scanner outputs (Nmap, Nessus, Burp, etc.), manage issues and evidence, collaborate with teammates, and generate consistent reports (Word, HTML, CSV, PDF) quickly.

Why Dradis is important for students: - Saves time: automates repetitive reporting tasks so teams can focus on analysis.
- Reproducibility: ensures consistent deliverables across engagements.
- Collaboration: shared project workspace for notes, evidence, and QA before reporting.
- Learning: connects raw scanner data to findings, remediation, and report templates.

Safety & ethics: Only deploy and use Dradis on lab systems or for engagements where you have express written permission. Don’t ingest production-sensitive data into public or shared demo instances.

2 — Learning objectives

After this manual students will be able to: 1. Install Dradis CE (Kali, Docker, or from Git) in an isolated lab.
2. Start/stop the Dradis server and access the web UI.
3. Create projects, import scanner outputs and manual findings, and manage evidence.
4. Use the Export Manager to generate reports and export data.

5. Back up Dradis data, troubleshoot common issues, and automate simple tasks.

3 — Supported install paths (choose one for your lab)

Three common ways to run Dradis Community Edition (CE):

- **Kali package** (quick, easier for student VMs) — installs from Kali repositories.
- **Docker image** (fast, reproducible, isolated).
- **Git install (from source)** — best for developers or contributors, gives full control.

Choose the method that matches your lab requirements.

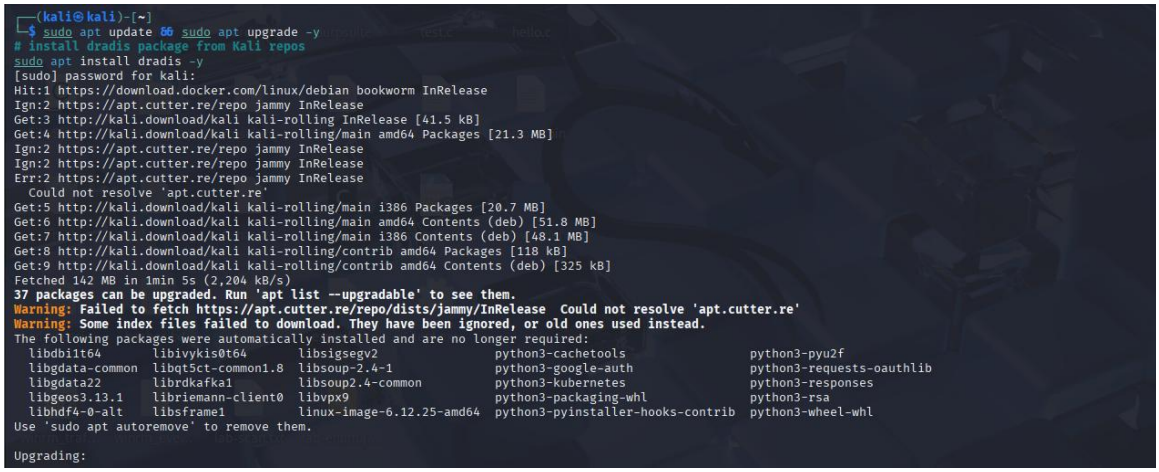
4 — Prerequisites

- A lab VM or host with internet access (for downloads) — or a pre-built offline image for air-gapped labs.
- Ruby and bundler (for Git installs).
- Docker (optional, for Docker installs).
- Basic Linux administration skills (systemd services, ports, file permissions).

5 — Step-by-step installation (commands)

Option A — Install from Kali package (fast)

```
# update packages first
sudo apt update && sudo apt upgrade -y
# install dradis package from Kali repos
sudo apt install dradis -y
```



After installation, check for a service or instructions printed by the package. Some Kali packages create a dradis user and a systemd service; if so, start the service with:

sudo systemctl start dradis
sudo systemctl enable dradis

If the package didn't create a service, see the Git or Docker methods below to run the server manually.

Option B — Run with Docker (recommended for class demos)

 # pull the latest community edition image
docker image pull dradis/dradis-ce:latest

 # run the container, mapping port 3000 on the host
docker run -d --name dradis-ce -p 3000:3000 dradis/dradis-ce:latest

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run -d --name dradis-ce -p 3000:3000 dradis/dradis-ce:latest
e7c96b7ba32a96016326e8cbcfb4c05352d9fcd6cf5d80c8884d7ab573f710af
```

Open the web UI at: http://<host-ip>:3000 (replace <host-ip> with localhost if running locally).

To stop/remove:

docker stop dradis-ce
docker rm dradis-ce

Option C — Install from Git (source install; more control)

 # install system deps (example on Debian/Ubuntu/Kali)
*sudo apt update **&&** sudo apt install -y git ruby build-essential libsqlite3-dev libpq-dev nodejs yarn*

 # clone the repo
git clone https://github.com/dradis/dradis-ce.git
cd dradis-ce

 # run the setup script (this installs gems and creates sample data)
./bin/setup

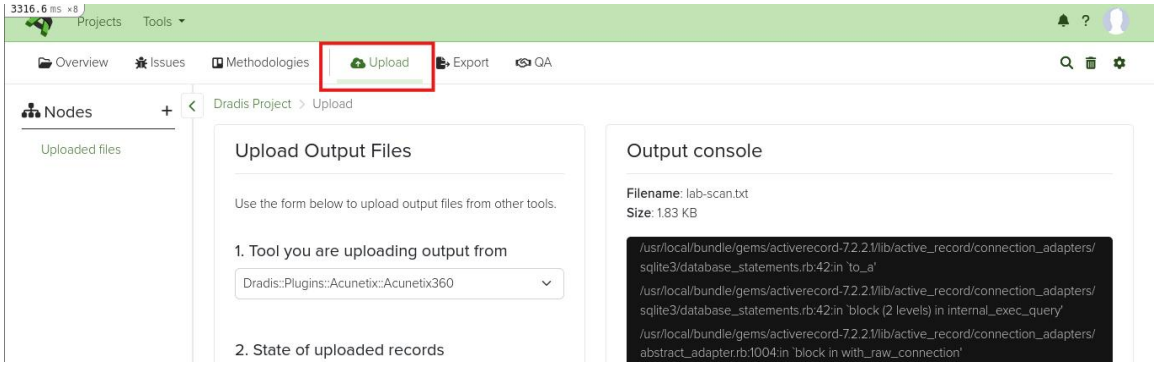
 # start the Rails server (development)
bundle exec rails server -b 0.0.0.0 -p 3000

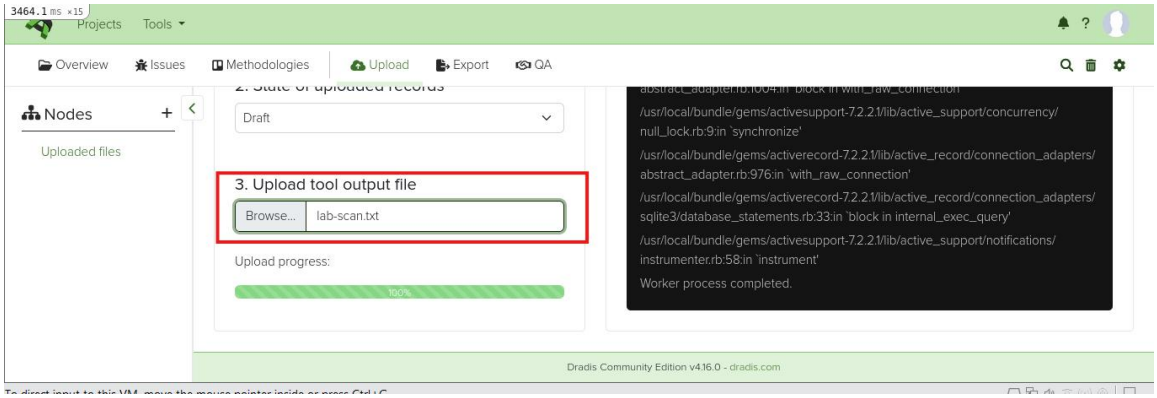
Notes and troubleshooting: - If bundle or rails are not found, run gem install bundler and bundle install inside the project directory.
- ./bin/setup may take several minutes while gems install and assets compile.
- For production-like installations you'll typically configure a reverse proxy (nginx) and a background worker (Redis + Sidekiq) per the official docs.

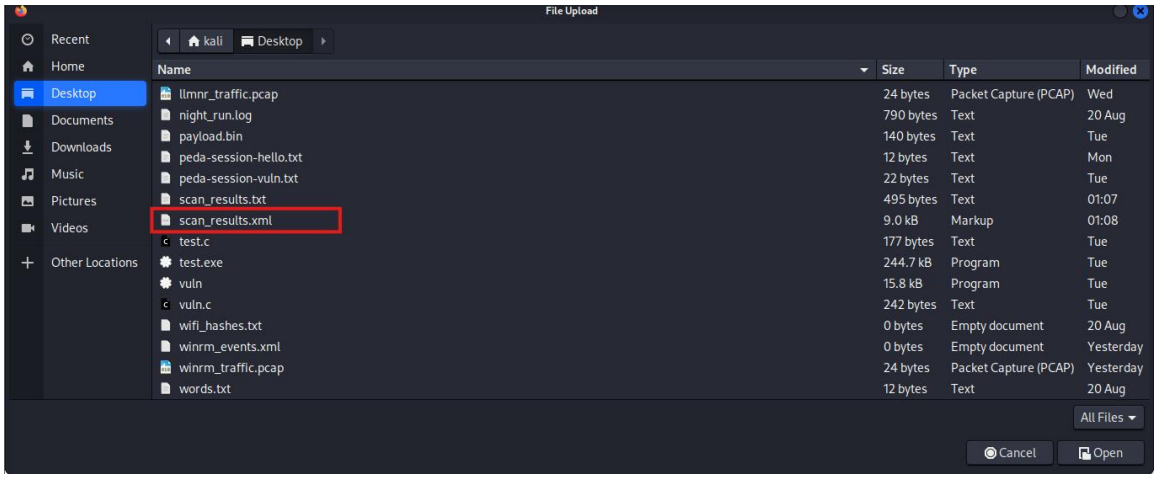
6 — First run & accessing the UI

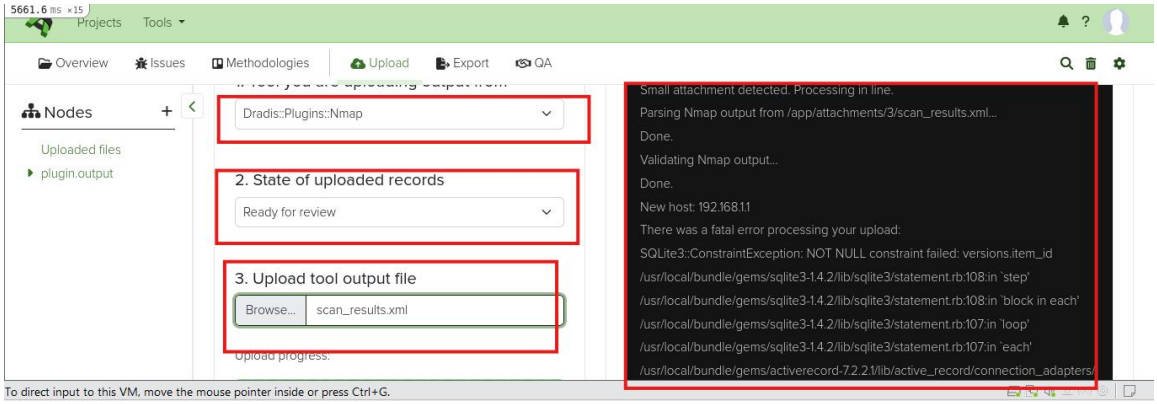
1. With the server running (Docker/Kali/package/git), open a browser to: http://localhost:3000 or http://<server-ip>:3000.
2. Create an instructor/admin account when prompted (or follow your lab's provided credentials).

3. Create a new project: **Projects → New Project** and give it a name (e.g., "Lab – AcmeCorp Test").


3316.6 ms ×8
Projects
Tools
Overview
Issues
Methodologies
Upload
Export
QA
Nodes
Uploaded files
Dradis Project > Upload
Upload Output Files
Use the form below to upload output files from other tools.
1. Tool you are uploading output from
Dradis::Plugins::Acunetix::Acunetix360
2. State of uploaded records
Output console
Filename: lab-scan.txt
Size: 1.83 KB
/usr/local/bundle/gems/activerecord-7.2.2.1/lib/active_record/connection_adapters/sqlite3/database_statements.rb:42:in `to_a'
/usr/local/bundle/gems/activerecord-7.2.2.1/lib/active_record/connection_adapters/sqlite3/database_statements.rb:42:in `block (2 levels) in internal_exec_query'
/usr/local/bundle/gems/activerecord-7.2.2.1/lib/active_record/connection_adapters/abstract_adapter.rb:1004:in `block in with_raw_connection'


3464.1 ms ×15
Projects
Tools
Overview
Issues
Methodologies
Upload
Export
QA
Nodes
Uploaded files
Draft
3. Upload tool output file
Browse...
lab-scan.txt
Upload progress:
100%
/usr/local/bundle/gems/activesupport-7.2.2.1/lib/active_support/concurrency/null_lock.rb:9:in `synchronize'
/usr/local/bundle/gems/activerecord-7.2.2.1/lib/active_record/connection_adapters/abstract_adapter.rb:976:in `with_raw_connection'
/usr/local/bundle/gems/activerecord-7.2.2.1/lib/active_record/connection_adapters/sqlite3/database_statements.rb:33:in `block in internal_exec_query'
/usr/local/bundle/gems/activesupport-7.2.2.1/lib/active_support/notifications/instrumenter.rb:58:in `instrument'
Worker process completed.
Dradis Community Edition v4.16.0 - dradis.com


File Upload
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Other Locations
kali
Desktop
Name
Size
Type
Modified
llmnr_traffic.pcap 24 bytes Packet Capture (PCAP) Wed
night_run.log 790 bytes Text 20 Aug
payload.bin 140 bytes Text Tue
peda-session-hello.txt 12 bytes Text Mon
peda-session-vuln.txt 22 bytes Text Tue
scan_results.txt 495 bytes Text 01:07
scan_results.xml 9.0 kB Markup 01:08
test.c 177 bytes Text Tue
test.exe 244.7 kB Program Tue
vuln 15.8 kB Program Tue
vuln.c 242 bytes Text Tue
wifi_hashes.txt 0 bytes Empty document 20 Aug
winrm_events.xml 0 bytes Empty document Yesterday
winrm_traffic.pcap 24 bytes Packet Capture (PCAP) Yesterday
words.txt 12 bytes Text 20 Aug
All Files
Cancel
Open



Quick UI navigation: - **Projects:** container for findings, notes, nodes, evidence.
- **Issues:** findings with severity, remediation and references.
- **Nodes:** hosts/targets (e.g., 10.0.0.5 or web.example.com).
- **Evidence:** attachments from scanner outputs (imported XMLs, screenshots).
- **Export / Export Manager:** create deliverables (Word, HTML, CSV, PDF).

7 — Importing scanner output (step-by-step)

Dradis accepts many scanner formats (Nmap XML, Nessus, Burp XML, OpenVAS, Nikto, etc.). There are two main ways to import:

A — From the project UI (recommended for students) 1. Open your project.
2. Click **Import/Upload** (or **Tools → Upload** depending on UI).
3. Select the scanner type (if prompted) or use the appropriate import plugin (e.g., Nmap importer).
4. Upload the XML/CSV file (for example nmap.xml from nmap -oX nmap.xml).
5. Dradis will parse and create Nodes, Evidence, and preliminary Issues or data points to review.

B — Command-line upload (automation) - The Community Edition supports add-ons and upload plugins; for automation, you can use the Dradis API or write a small script to POST files to the import endpoint. See the project's addons and upload documentation for details.

Practical step: generate an nmap XML on your Kali host:

nmap -sC -sV -oX lab-nmap.xml 10.0.2.0/24

Then in Dradis, upload that lab-nmap.xml via the Import UI. Nodes and ports will be populated automatically.

8 — Managing findings and evidence (best practices)

- **Create Issues** for confirmed findings — include title, description, severity, evidence, and recommended remediation.

- **Attach Evidence** (screenshots, raw XML snippets) to Issues — this keeps the finding reproducible and auditable.

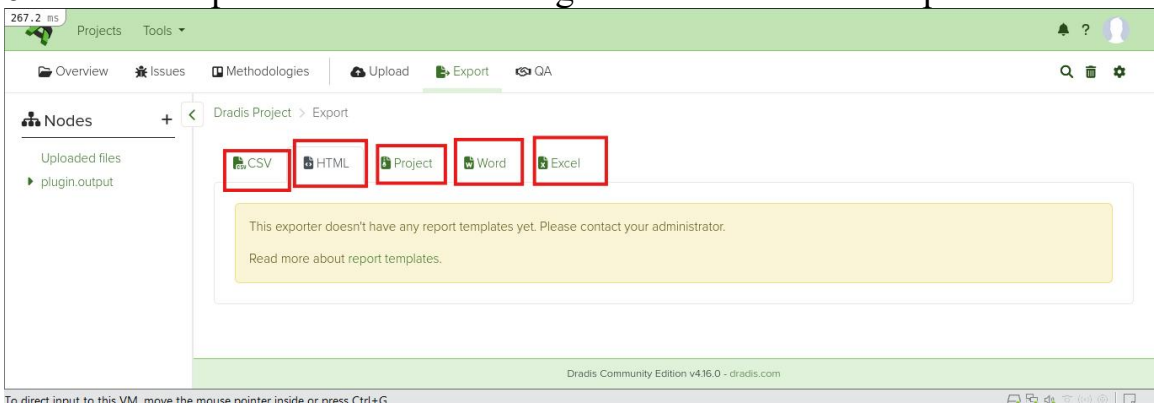
- **Use States** (Draft / Ready / Published) to implement a QA workflow before export.

- **Add Notes** for reviewers and link Issues to Nodes.

9 — Exporting reports (Export Manager)

Use the Export Manager inside a Project to create Word, HTML, CSV or PDF reports.

Step-by-step (UI): 1. Open your project.
2. Click **Export** in the project header.
3. Choose an Export Plugin (Word, HTML, CSV, PDF).
4. Choose a template if the plugin supports templates.
5. Choose whether to export Published records only, or All records.
6. Start the export and download the generated file when complete.



Command-line export (advanced): some exporters provide rake tasks or CLI commands for automated exports; consult the Dradis docs for the exporter you installed and the project ID you want to export.

10 — Backups & maintenance (quick commands)

Backup strategy: regularly export the Dradis DB and configuration, and back up uploads/ and any custom templates.

- If using **SQLite** (development), copy the DB file:

```
cp path/to/dradis-ce/db/development.sqlite3 /root/dradis-backups/dradis-$(date +%F).sqlite3
```

- If using **PostgreSQL**, use pg_dump:

```
pg_dump -U <dbuser> -h <dbhost> <dbname> > dradis_backup_$(date +%F).sql
```

- Back up the public/uploads/ or storage/ folder (attachments) and any Gemfile.plugins or custom templates.

Restart the server (Git install example):

```
# inside dradis-ce folder
bundle exec rails server -b 0.0.0.0 -p 3000
# or use systemd/service wrapper if you created one
sudo systemctl restart dradis
```

11 — Troubleshooting common issues

- **bundle/rails not found**: install bundler (gem install bundler) and run bundle install inside the project directory.

- **Missing native dependencies**: install libsqlite3-dev, libpq-dev (for PostgreSQL), nodejs, and yarn before bundle install.

- **Assets or export failures**: precompile assets in production mode: RAILS_ENV=production bundle exec rails assets:precompile.

- **Server not reachable**: confirm the Rails server or Docker container is running and port 3000 is accessible (open firewall or use -p 3000:3000 with Docker).

12 — Automation & CLI tips

- Use **PyDradis** (community wrappers) to script interactions with Dradis via the API.

- Use the **Rails console** to inspect or manipulate data (advanced / instructor use):

```
# inside project folder
RAILS_ENV=production bundle exec rails console
# then in console: Dradis::Project.find(1) etc.
```

- Create a systemd service to run the Rails server in the background (for Git installs) so students don't need to start it manually each session.

13 — Classroom exercises (suggested)

Exercise 1 — Import & map: - Generate nmap -oX, nikto -output or Burp XML test files on Kali.
- Import them into a new Dradis Project.
- Create Issues and attach evidence, then mark them Ready for review.

Exercise 2 — Report generation: - Use the Export Manager to generate a Word and HTML report from the project.
- Validate that exported findings match the Issues and Evidence in Dradis.

Exercise 3 — QA workflow: - Assign roles: Reporter, Reviewer, Approver. Use states to move Issues from Draft to Published and verify exports include only Published items.

Exercise 4 — Automation: - Write a small script (or use PyDradis) to upload an nmap.xml file programmatically and trigger a report export.

14 — Security & operating guidance

- Run Dradis on a restricted VM that only team members can access.

- Change default OS and administrative passwords on any production or shared images.

- Limit access to exported reports and avoid storing unredacted production data in shared instances.

- Keep Dradis and its plugins updated; monitor the project's issue tracker for security advisories.

15 — Further reading & resources

- Dradis GitHub (source & issues) — good for latest code and troubleshooting.

- Official Dradis documentation & guides — installation, importers, exporters, and admin topics.

- Dradis community forums for common installation fixes and community add-ons.

Closing notes

This manual gives students a practical, step-by-step guide to installing, using, and maintaining Dradis in an ethical lab environment. If you want, I can:

- Produce a **one-page cheat sheet** (PDF/Word) with the core commands.

- Create a **student worksheet** with checkboxes and spaces to paste outputs.

- Build a **Docker Compose** example to deploy Dradis + Redis + Postgres for the class.

End of manual.