# Course Lessons

## Hands-On Penetration Testing with the MMT-Pentester Toolbox

CyberSuite Academy — Text Lessons & Quizzes

Produced by Montimage · June 2026

*Each section is one Academy text-lesson page followed by its quiz. Answer keys are shown in green for the author and are not displayed to learners.*

# Introduction — Hands-On Penetration Testing with the MMT-Pentester Toolbox

*Estimated time: ~15 minutes · Text lesson*

## Welcome

Penetration testing is how organisations find their weaknesses before attackers do. In this course

you will learn the complete penetration-testing lifecycle — reconnaissance, controlled exploitation,

monitoring, and reporting — and how to carry it out safely and ethically using the Montimage

**MMT-Pentester** toolbox, a web platform that brings recon, attack simulation, monitoring, and

reporting together in one guided interface.

Everything is hands-on, but every exercise runs **only against provided, authorised lab targets** —

never against real or third-party systems. By the end you will be able to plan an engagement, run

reconnaissance and attack scenarios, interpret what a defender's monitoring would see, and produce a

clear, actionable report.

## What you will learn

- Describe the penetration-testing lifecycle and the legal and ethical rules of authorised testing.
- Perform reconnaissance with Nmap and map a target's attack surface.
- Configure and run controlled attack scenarios (SSH brute-force, HTTP Slowloris) in a safe lab.
- Monitor executions in real time and interpret intrusion-detection alerts.
- Generate professional reports and write actionable remediation recommendations.
- Use AI-assisted testing and understand the platform's integrated tool ecosystem.

## Who this course is for

This course suits IT and cybersecurity students, junior SOC analysts, system and network

administrators, and developers who want practical penetration-testing skills. A basic understanding

of networking (IP addresses, ports, HTTP, SSH) and comfort with a command line are recommended.

**No prior penetration-testing experience is required** — every concept is introduced before it is used.

## How the course is structured

The course is organised as five modules plus a final hands-on assessment. Each module is a short lesson followed by a quiz:

- **Module 1 — Pentesting Foundations, Ethics & Methodology**
- **Module 2 — Getting Started with MMT-Pentester**
- **Module 3 — Reconnaissance & Attack Scenarios**
- **Module 4 — Monitoring, Detection & Reporting**
- **Module 5 — Advanced: AI Agents & Tool Integration**
- **Final Hands-On Assessment (Capstone)**

You can follow the course at your own pace; the full effort is about 20 hours, ideally spread over three to four weeks.

## Assessment and certificate

Each module ends with a quiz; the pass mark is **7/10**. After the capstone you complete a final quiz that draws on all the modules. Complete every module and the final assessment with the required grades and you earn a **digital certificate of completion** from the CyberSuite Academy.

## A note on safety and ethics

The platform launches real attack tooling. You must only ever use it against the lab targets provided for this course, with explicit authorisation. Unauthorised testing of systems you do not own or have permission to test is illegal. Module 1 covers these boundaries in detail — they apply throughout.

## Meet the team

The MMT-Pentester toolbox and this course are produced by **Montimage**. Lead developer: Mohamed Hamdouni. For questions about the course, contact Maxime Vinh Hoa La (maxime.vinhhoa.la@montimage.eu).

# Module 1: Pentesting Foundations, Ethics & Methodology

*Estimated time: ~45 minutes · Text lesson + quiz*

## In this lesson

- Define penetration testing and explain why organisations pay for it.
- Distinguish vulnerability scanning, penetration testing, and red teaming, and choose a knowledge-depth model (black/grey/white box).
- Walk through the phases of an engagement, from pre-engagement scoping to reporting.
- Apply the legal and ethical rules — authorisation, scope, rules of engagement, and responsible disclosure — that make testing legitimate.
- Map the major methodologies and frameworks (PTES, OWASP, Cyber Kill Chain, MITRE ATT&CK) onto the MMT-Pentester recon → attack → report flow.

## What penetration testing is

A *penetration test* (or "pentest") is an **authorised, simulated attack** on a system, network, or application, carried out to find and demonstrate exploitable weaknesses before a real attacker does. It answers a blunt, practical question: *if someone tried to break in, could they — and how far would they get?*

The goal is constructive, not destructive. A pentest surfaces real risk, proves the impact of that risk, and hands defenders a prioritised list of fixes. Crucially, it produces **evidence rather than opinion**: a demonstrated exploit ("we logged in and read the customer table") is far more persuasive to management than a scanner label that simply says "high severity."

Hold on to one word above all others: **authorised**. The difference between a penetration tester and a criminal is not skill or tooling — the techniques can be identical. The difference is *permission and scope*. Every exercise in this course runs only against provided, isolated lab targets, with explicit authorisation, and never against real or third-party systems.

Organisations invest in pentesting for four recurring reasons:

- **Find before they do** — discover exploitable flaws while you still control the outcome.
- **Validate defences** — confirm that firewalls, patching, and monitoring actually hold up under attack.
- **Prioritise risk** — turn a long list of theoretical issues into a short list of things that truly matter.
- **Compliance and trust** — standards such as PCI-DSS, ISO 27001, and SOC 2 expect regular testing, and clients and regulators ask for proof.

## Scanning vs pentest vs red teaming

Beginners often blur three related security activities together. They sit on a spectrum of depth, realism, and cost — not three unrelated jobs.

| Activity | Driven by | Depth | Exploitation? | Primary question answered |
|---|---|---|---|---|
| Vulnerability scanning | Mostly automated tools | Broad, shallow | Rarely / safely | "What *potential* weaknesses exist?" |
| Penetration testing | Human + tools | Focused, deep | Yes, within scope | "Which weaknesses are *really* exploitable, and what's the impact?" |
| Red teaming | Human, goal-driven | Narrow, realistic | Yes, stealthy | "Can we be detected and stopped?" |

A common mistake is to treat a scan report as if it were a pentest. The distinction is concrete: a scanner says "port 22 *may* allow weak authentication"; a pentest *proves* it by logging in. Red teaming goes further still — it is goal-oriented adversary emulation that tests an organisation's detection and response across people, process, and technology, often stealthily and over weeks.

### How much the tester knows: black, grey, white box

A second axis describes how much inside information the tester is given before starting.

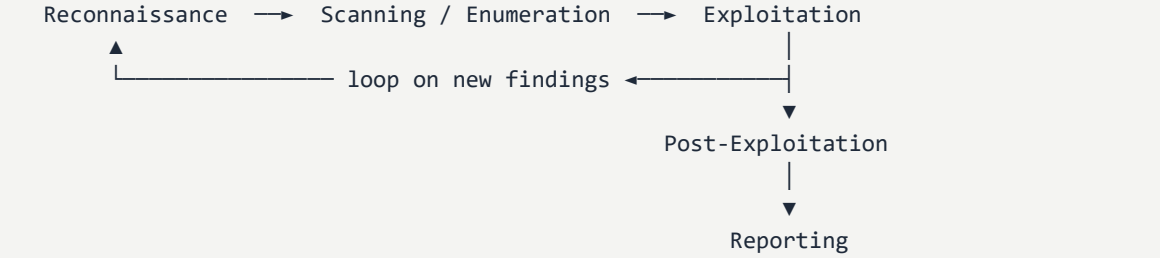
| Model | Tester knowledge | Simulates | Trade-off |
|---|---|---|---|
| Black-box | None | External attacker with no inside info | Realistic but slower; may miss internal issues |
| Grey-box | Partial (e.g., a user account, a diagram) | Insider or attacker with a foothold | Balanced coverage vs realism; common default |
| White-box | Full (code, configs, credentials) | Auditor with complete access | Maximum coverage; least about realism |

There is no "best" box colour — only the most appropriate one for the engagement's goal. Grey-box is the practical default because it spends time efficiently without pretending the attacker knows nothing.

## The phases of an engagement

A pentest follows a repeatable lifecycle. It is a *cycle*, not a strict one-way pipeline: findings in a later phase frequently send you back to an earlier one (new access reveals new things to scan).

```
Pre-engagement (scope + authorisation)
        │
        ▼
```




![figure: The penetration testing lifecycle](../assets/lifecycle.png)

- **Pre-engagement** always comes first: scoping, rules of engagement, and written authorisation. This is the non-technical foundation that makes everything legitimate.
- **Reconnaissance** — gather information about the target. *Passive* recon uses public data with no direct contact; *active* recon directly probes the target. Good targeting beats blind exploitation, so this is where engagement time is well spent.
- **Scanning and enumeration** — map the live attack surface: hosts, open ports, services, versions, and likely entry points.
- **Exploitation** — safely attempt to use a weakness to gain access or cause a defined effect, proving the risk is real.
- **Post-exploitation** — once in, assess what an attacker could *actually* do: pivot, escalate privileges, access data — all within scope. The aim is to demonstrate impact responsibly, not to maximise damage. Testers avoid damage, stay in scope, and document every action.
- **Reporting** — turn activity into value. This is often the most important phase for the client.

A pentest report typically contains:

- **Executive summary** — risk in plain language for decision-makers.
- **Technical findings** — each issue with evidence, severity, and reproduction steps.
- **Remediation guidance** — prioritised, actionable fixes (a finding without a fix is half-finished work).
- **Appendices** — scope, methodology, tools, and raw output.

In the MMT-Pentester Dashboard, the three guided scenarios map cleanly onto the early phases, and the platform auto-generates **PDF and Excel reports** from your executions to support the reporting phase:

- **Reconnaissance with Nmap** → recon + scanning/enumeration of a lab target.
- **SSH brute-force** → exploitation against a discovered SSH service.
- **HTTP Slowloris DoS** → exploitation (denial-of-service) against a discovered HTTP service.

![screenshot: MMT-Pentester Scenarios view with a guided scenario selected](../assets/scenarios-view.png)

## Legal authorisation, scope, and rules of engagement

Penetration testing is one of the few professions where the *same action* is either a paid service or a crime, decided entirely by permission. This is why the legal and ethical foundation matters more than any tool.

**Authorisation** means explicit, written permission from someone who actually owns or controls the target — an *authorised signatory* such as the system owner or CISO. A verbal "go ahead," or an email from the wrong person, is **not** authorisation. It must exist *before the first packet is sent* and must cover exactly what you intend to do. No authorisation, no test; there is no grey area. Unauthorised access — even "just scanning" — can breach computer-misuse laws such as the CFAA (US) or the Computer Misuse Act (UK), and "I was only curious" is not a defence.

A proper engagement is backed by a written agreement: a contract or Statement of Work plus an **authorisation letter** (sometimes called a "get-out-of-jail" letter) that names the systems, dates, and an emergency contact. You keep that letter at hand during testing as proof of permission. Consent can also be *withdrawn* — if the client says stop, you stop immediately.

**Scope** is exactly which assets you may touch: listed IP ranges, hosts, domains, applications, and accounts. Anything not listed as in-scope is **out-of-scope by default**. Discovering an out-of-scope asset (a shared-hosting neighbour, a third-party API, an SSH service that was never in the scope document) is a *finding to document* — never a new target to attack.

**Rules of engagement (RoE)** define the "how and when": permitted time windows, allowed techniques, rate limits, escalation contacts, and *prohibited actions*. Prohibited actions commonly include social engineering, physical intrusion, destructive payloads, denial-of-service, and touching real production data. Note that the RoE govern *when* as well as *what* — testing an in-scope target outside the agreed time window is still unauthorised.

The table below makes the line concrete:

| Situation | Authorised? | Why |
|---|---|---|
| Signed authorisation letter from the owner naming the host and dates | Yes | Explicit, written, from an authorised signatory, covers the action |
| A colleague verbally says "sure, scan our network" | No | Not written; may not be from an authorised signatory |
| You own the lab VM provided by this course | Yes | Provided to you explicitly for testing |
| You discover an interesting host online and "just scan" it | No | No permission; even scanning can breach the law |
| Target is in scope but tested outside the agreed window | No | RoE define *when*, not just *what* |
| You find an SSH service that was NOT listed in scope | No | Out-of-scope by default — report it, don't attack it |

In MMT-Pentester, scope maps directly onto the data model. Work is organised as **Project → Campaign → Scenario**, where each Scenario holds **targets {host, port, protocol}** and **attacks {params}**. Only ever load authorised, in-scope targets into a Scenario.

### Data handling and confidentiality

During a test you will see sensitive data — credentials, customer records, internal configs. Treat all of it as confidential. Collect only what you need to prove a finding: capture minimal evidence (screenshots, hashes, redacted samples) rather than exfiltrating real data. Store evidence and reports encrypted and access-controlled, and delete them per the agreed retention period.

The platform follows the same principle. The default super admin account (`admin` / `admin123456`) must be changed on first login; role-based access control (superadmin > admin > user) and audit logs support accountability; and you should run it behind HTTPS with a strong `JWT_SECRET` and network access restricted to controlled environments. The auto-generated PDF/Excel reports are confidential deliverables and must be handled as such.

### Responsible disclosure

If you find a genuine vulnerability — in MMT-Pentester or anywhere else — report it *responsibly*. Do not post it publicly. The platform's process, defined in `SECURITY.md`, is:

- **Do not** open a public GitHub issue for a vulnerability.
- Email **security@montimage.com** with a description, steps to reproduce, impact/severity, and optional suggested mitigations.
- Expect acknowledgement within 48 hours, an initial assessment within 7 days, critical fixes targeted within 30 days, and disclosure coordinated with you before anything is published.

The supported version line is 1.0.x, and reporters who follow the policy are credited in release notes unless they prefer to stay anonymous. Quoting `SECURITY.md` directly: *"This platform is designed for educational and research use in authorized, controlled environments only. Unauthorized use against systems you do not own or have explicit permission to test is illegal and outside the scope of this project."*

***Safety / Ethics rule:*** *All attacks in this course are run ONLY against provided, isolated lab targets, with explicit authorisation. Never against real or third-party systems. This is for education and authorised testing only.*

## Methodologies and frameworks

A *methodology* is a repeatable, agreed plan for *how* a test is run — not just which tools you use. Following one gives you full coverage (you don't forget steps), repeatability, comparable results between testers, and a shared vocabulary with clients and defenders. Without one, testing becomes "poke at random things until something breaks" — unscoped, unauditable, and hard to report.

Four frameworks recur in practice. They are complementary lenses, not competitors — a mature engagement borrows from all four.

| Framework | Type | Owner | Structure | Best used for | Example unit |
|---|---|---|---|---|---|
| **PTES** | Engagement process | PTES community | 7 phases (pre-engagement → reporting) | Planning/running a full test end-to-end | Phase 2: Intelligence Gathering |
| **OWASP Testing Guide (WSTG)** | Web testing checklist | OWASP | Categories of web tests; pairs with OWASP Top 10 | Deep testing of web applications | WSTG-ATHN (Authentication) |
| **Cyber Kill Chain** | Intrusion model | Lockheed Martin | 7 linear steps (Recon → Actions on Objectives) | Explaining/defending one intrusion | Step 1: Reconnaissance |
| **MITRE ATT&CK** | Adversary knowledge base | MITRE | Tactics → Techniques → Sub-techniques | Naming behaviours; mapping findings; detection | `T1110` Brute Force |

- **PTES** (Penetration Testing Execution Standard) defines seven phases: pre-engagement interactions → intelligence gathering → threat modeling → vulnerability analysis → exploitation → post-exploitation → reporting. It is process-oriented, telling you the *order* of work. Notice the bookends: it starts with authorisation/scope and ends with a report — exactly the discipline this course teaches.
- **The OWASP Testing Guide (WSTG)** is a structured catalogue of web-application tests (information gathering, configuration, authentication, session management, input validation, and more). It pairs with the OWASP Top 10: the guide is *how to test*, the Top 10 is *what commonly goes wrong*.
- **The Cyber Kill Chain** models a single linear intrusion in seven steps: Reconnaissance → Weaponization → Delivery → Exploitation → Installation → Command & Control → Actions on Objectives. It is defender-friendly — breaking *any* link stops the attack.
- **MITRE ATT&CK** is a living matrix built from observed real-world attacks. It separates *tactics* from *techniques*, and unlike the Kill Chain's fixed seven steps, it is granular and non-linear — a real attack jumps around the matrix.

### Tactics vs techniques, and reading a technique ID

This is the single most important distinction in ATT&CK:

- A **tactic** is the adversary's *goal* for a step — the "why." Examples: Reconnaissance, Initial Access, Credential Access, Impact.
- A **technique** is *how* they achieve that goal — the "how." A tactic contains many techniques, and a technique can serve more than one tactic.
- A **sub-technique** is a more specific variant of a technique.

Technique IDs are stable identifiers you cite in reports so defenders can look up exact detection and mitigation guidance:

```
T1110.001
| |    |
| |    └── Sub-technique number (.001 = Password Guessing) — optional
| └─────── Technique number (1110 = Brute Force)
└────────── "T" prefix = Technique
```

So `T1595` is **Active Scanning** (Reconnaissance), `T1110` is **Brute Force** (Credential Access, with sub-technique `T1110.001` Password Guessing), and `T1499` is **Endpoint Denial of Service** (Impact). Tactics have IDs too, prefixed `TA` (e.g. `TA0006` Credential Access), but in reports you most often cite the technique ID. Always include it so a defender can look it up directly on https://attack.mitre.org.

### Mapping MMT-Pentester to the frameworks

The platform's flow is **recon → attack → report**, which is simply PTES compressed: intelligence gathering → exploitation → reporting. PTES tells you the *order*, ATT&CK *names* what you did, and the report *communicates* it. The three guided scenarios are the spine of the hands-on labs:

| MMT-Pentester scenario | PTES phase | Kill Chain step | ATT&CK tactic | ATT&CK technique (ID) |
|---|---|---|---|---|
| 1. Reconnaissance with Nmap | Intelligence Gathering | Reconnaissance | Reconnaissance | Active Scanning (`T1595`) |
| 2. SSH brute-force | Exploitation | Exploitation | Credential Access | Brute Force (`T1110`, sub `T1110.001`) |
| 3. HTTP Slowloris DoS | Exploitation | Actions on Objectives | Impact | Endpoint Denial of Service (`T1499`) |

Reading across the table: recon to find the door, credential access to open it, impact to disrupt it — a mini kill chain in three labs. You run each Scenario from the **Scenarios** view (sequential or parallel executions, with live tool output streamed over WebSocket) and export the result from the **Reports** view as PDF or Excel. That report is your PTES phase-7 deliverable — annotate each finding with its ATT&CK technique ID so clients and blue teams can act on it.

![screenshot: Reports view with ATT&CK-tagged findings](../assets/reports-view.png)

## Summary

- A penetration test is an *authorised*, simulated attack that proves real, exploitable risk; permission and scope — not technique — separate it from a crime.
- Scanning, pentesting, and red teaming form a spectrum of depth and realism; the black/grey/white-box models describe how much the tester knows in advance, with grey-box the common default.
- Engagements follow an iterative lifecycle — pre-engagement → recon → scanning → exploitation → post-exploitation → reporting — and deliver an executive summary, technical findings, remediation, and appendices.
- Written authorisation from an authorised signatory, an explicit scope (out-of-scope by default), and rules of engagement (including time windows and prohibited actions) are mandatory before any test.
- Handle all data as confidential, minimise evidence, and disclose vulnerabilities responsibly — for this platform, email security@montimage.com rather than opening a public issue.
- PTES (process), OWASP WSTG (web checklist), the Cyber Kill Chain (intrusion model), and MITRE ATT&CK (tactics → techniques) are complementary lenses that map directly onto the MMT-Pentester recon → attack → report flow.

## Quiz: Pentesting Foundations, Ethics & Methodology

*Pass mark: 7/10. Choose the best answer.*

1. What single factor most clearly separates a penetration test from a criminal attack?

    - A) The tools used
    - B) Explicit authorisation and a defined scope
    - C) The skill of the tester
    - D) Whether any data is accessed

    **Answer:** B — Technique can be identical; legitimacy comes from written permission and an agreed scope.

2. Which activity is best described as automated, broad, and mostly *without* exploitation?

    - A) Red teaming
    - B) Penetration testing
    - C) Vulnerability scanning
    - D) Post-exploitation

    **Answer:** C — Scanning lists *potential* weaknesses quickly; it generally does not confirm them by exploiting.

3. A tester is given a low-privilege user account and a network diagram before starting. Which model is this?

- A) Black-box
- B) Grey-box
- C) White-box
- D) Red-box

**Answer:** B — Partial inside knowledge, such as a user account, defines grey-box testing.

4. What is the main purpose of the post-exploitation phase?

    - A) To list open ports and services
    - B) To gather public information about the target
    - C) To assess what an attacker could actually do after gaining access, within scope
    - D) To write the executive summary

    **Answer:** C — Post-exploitation measures real impact (pivoting, escalation, data access) responsibly and in scope.

5. A client's IT manager emails "feel free to test our web server whenever." Is this sufficient authorisation to begin?

    - A) Yes — it is in writing, so it counts
    - B) No — you need explicit written authorisation from an authorised signatory, covering specific scope and time
    - C) Yes — an email from any employee is legally binding
    - D) No — only verbal authorisation is valid

    **Answer:** B — Authorisation must be explicit, from someone with authority, and define scope and timing; a vague email is not enough.

6. During an authorised test you discover an SSH service on a host that was NOT listed in scope. What should you do?

    - A) Brute-force it — it's on the same network, so it's fair game
    - B) Ignore it entirely and never mention it
    - C) Document it as a finding and report it; do not attack it
    - D) Quickly test it before anyone notices

    **Answer:** C — Anything not listed as in-scope is out-of-scope by default; report the discovery rather than acting on it.

7. You find a genuine vulnerability in the MMT-Pentester platform itself. According to SECURITY.md, what is the correct first step?

    - A) Open a public GitHub issue so everyone is aware
    - B) Tweet the proof-of-concept to pressure a fast fix

- C) Email security@montimage.com privately with details and steps to reproduce
- D) Exploit it on a live deployment to confirm impact

**Answer:** C — Responsible disclosure means a private report to security@montimage.com, never a public issue or live exploitation.

8. Which framework is best described as an end-to-end *process* for running an engagement, from pre-engagement scoping to reporting?

    - A) MITRE ATT&CK
    - B) OWASP Testing Guide
    - C) PTES
    - D) Cyber Kill Chain

    **Answer:** C — PTES defines seven phases covering the whole engagement lifecycle.

# Module 2: Getting Started with MMT-Pentester

*Estimated time: ~45 minutes · Text lesson + quiz*

## In this lesson

- Describe what the MMT-Pentester platform is and the problem it solves, in plain terms.
- Sketch the platform's architecture — browser app, API, database, live updates, tools, and AI — and how the pieces fit together.
- Access the tool, log in for the first time, and change the default super-admin password.
- Navigate the five UI views (Dashboard, Projects, Scenarios, Agent, Reports) and say what each is for.
- Explain the Project → Campaign → Scenario data model and the three RBAC roles that govern access.

## What MMT-Pentester is, and the problem it solves

MMT-Pentester — also called the Montimage Pentesting Toolbox (version 1.0.0, by Montimage) — is a web platform for planning, orchestrating, and executing security tests and attack simulations. A *pentest* (penetration test) is an authorised, controlled exercise in which you behave like an attacker against a system you are allowed to test, so that real attackers cannot.

Traditionally you would run scanners and attack tools by hand, each in its own terminal, copy-pasting output between windows and stitching results together afterwards. MMT-Pentester replaces that scattered workflow with a single browser interface. You define the work, launch the tools, and watch their results stream back live — all in one place. Think of it as **mission control for an authorised engagement**: define the work, launch it, observe it, and report on it.

***Safety and ethics — read this first.*** *Everything in MMT-Pentester is for authorised testing against provided, isolated lab targets only. The platform can launch real attack tooling, so it must never be pointed at a real or third-party system. Throughout this course, every attack runs only against the lab targets you are explicitly given permission to test. This is for education and authorised testing.*

## The architecture in beginner terms

You do not need to install or configure anything to follow this lesson — the goal here is the *mental model*. MMT-Pentester is made of a few cooperating parts:

- **Front end (what you see):** a React 18 single-page application (SPA) that runs in your browser. It is fast and responsive and supports both dark and light mode. A single-page app loads once and then updates the screen in place, instead of reloading whole pages.
- **Back end (the brains):** an Express.js / Node.js API written in TypeScript. It handles logins, stores your work, and starts the security tools on your behalf.
- **Database (the memory):** MongoDB stores your projects, campaigns, scenarios, users, and audit logs so nothing is lost between sessions.

- **Live updates (the wire):** WebSocket channels push events and raw tool output to your screen the instant they happen — no page refresh needed.
- **The tools and the AI:** external security tools run as separate processes (called *child processes*); an AI assistant — provided through MCP and Together AI — can analyse the output and suggest next steps.

How do the pieces talk? Your browser sends ordinary requests to the **REST API** for things like "log me in" or "create a project." For anything that happens live, two **WebSocket** channels run in parallel: a Socket.io channel carries campaign and scenario *events*, while a separate raw ws channel carries the live *tool output*. The API starts each external tool as a child process and streams its output back over WebSocket so you watch it in near real time. The AI layer sits beside the API and powers the autonomous Agent.

You will later notice that "events" and "live tool logs" behave a little differently — that is the two-channel design at work. You do not need to memorise any port numbers; just remember there is a live feed coming back to your browser.

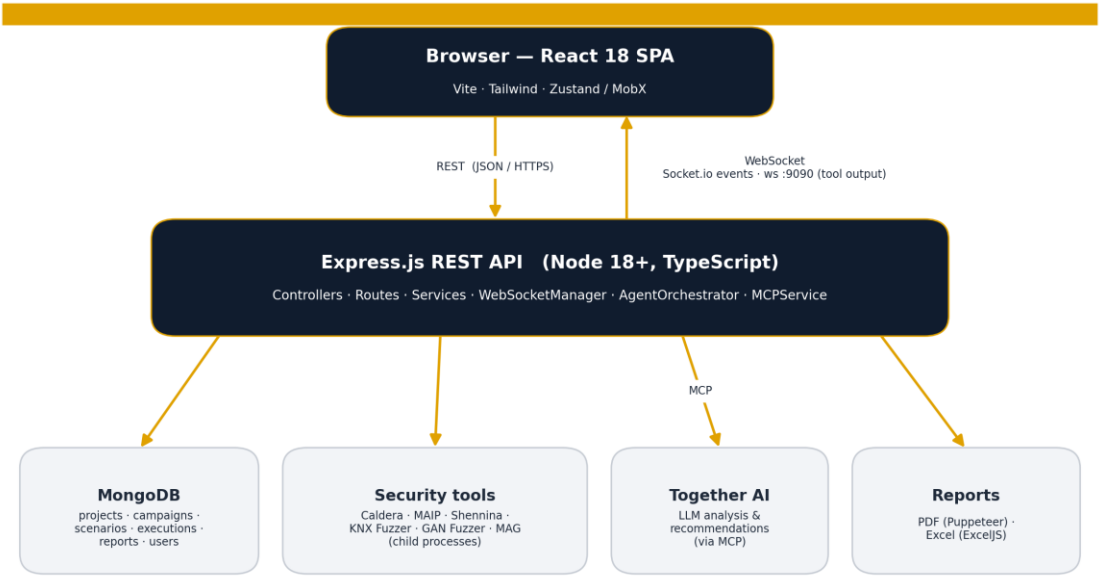



*Figure: MMT-Pentester architecture — browser SPA talking to the API, which talks to MongoDB and to the security tools, with live WebSocket feeds returning*

## Accessing the tool

There are two common ways to reach MMT-Pentester.

**The hosted instance.** Montimage runs a hosted copy you can sign in to directly at **https://mmt-pentester.montimage.eu/login**. This is the quickest way to try the platform with no setup.

**A local development install.** If you (or your instructor) run the platform yourself, the development front end lives at **`http://localhost:5173`** and its API at **`http://localhost:3000/api`**. The setup guide or your instructor handles getting it running.

Either way, the login screen is the same.

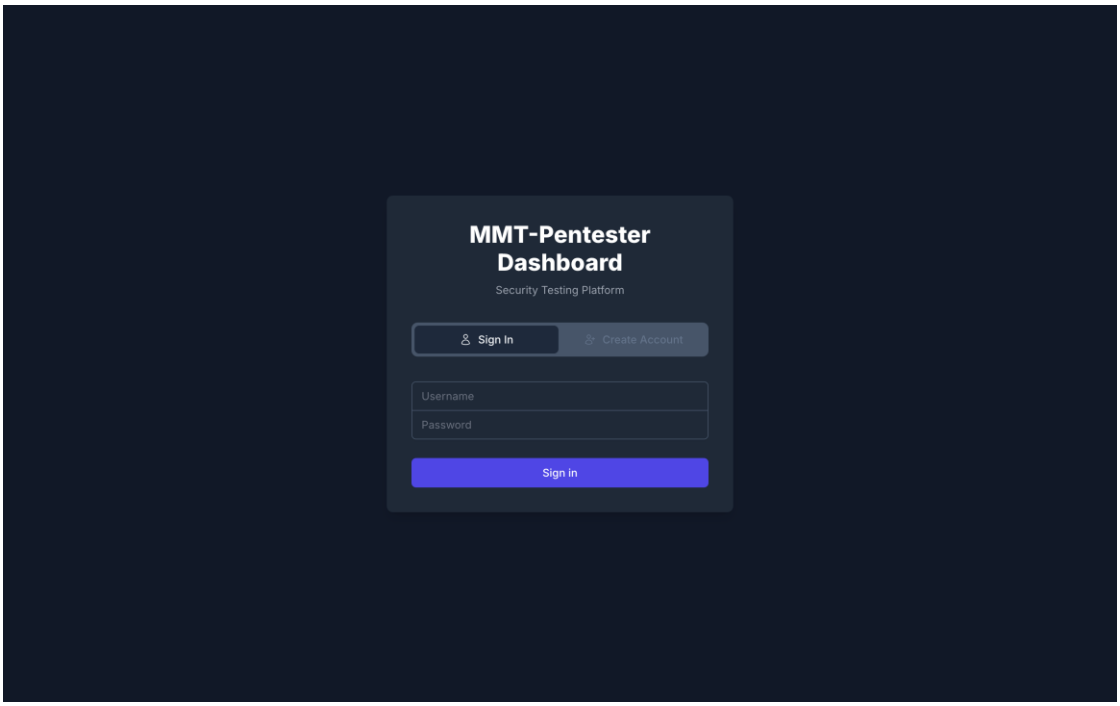



*Figure: MMT-Pentester login screen*

## First login and changing the password

Every fresh install of MMT-Pentester ships with a single **super admin** account:

```
Username: admin
Password: admin123456
```

These default credentials are *public knowledge* — they are documented and identical on every install. Leaving them unchanged is the same as having no password at all. For that reason, **the first thing you do on first login is change the default password**, and the platform enforces this: it prompts you for a new password before letting you continue.

Behind the scenes, the platform takes security seriously. Passwords are hashed with bcrypt (12 rounds), sessions are issued as JWTs and are IP-tracked, requests are rate-limited, and the account locks after **5 failed login attempts**. That lockout is worth remembering during the lab — type your new password carefully so you do not lock yourself out by retrying.

To log in and secure your account:

9. Open a browser and go to the hosted instance (`https://mmt-pentester.montimage.eu/login`) or your local front end (`http://localhost:5173`).
10. Enter the default super-admin credentials (`admin` / `admin123456`) and click **Log in**.
11. Because this is the default account, the platform prompts you to set a new password.
12. Choose a strong password, confirm it, and continue.
13. You land on the **Dashboard**.

## The five UI views

MMT-Pentester organises everything behind five views, reachable from the left-hand navigation. These five names are exact — you will see them in the interface.

| View | Open it when you want to… |
|---|---|
| **Dashboard** | See overall status, recent activity, and quick entry points — your home screen. |
| **Projects** | Create and organise engagements; manage campaigns and team access. |
| **Scenarios** | Define targets and attacks, then run and watch a test live. |
| **Agent** | Let the AI-assisted autonomous mode orchestrate steps for you. |
| **Reports** | Generate and download auto-built PDF and Excel reports of what was tested and found. |

As a beginner you will spend most of your early time in **Projects** and **Scenarios** — that is where engagements are built and tests are run. The Dashboard gives you a bird's-eye view, the Agent is for AI-assisted automation, and Reports turns your results into shareable documents.

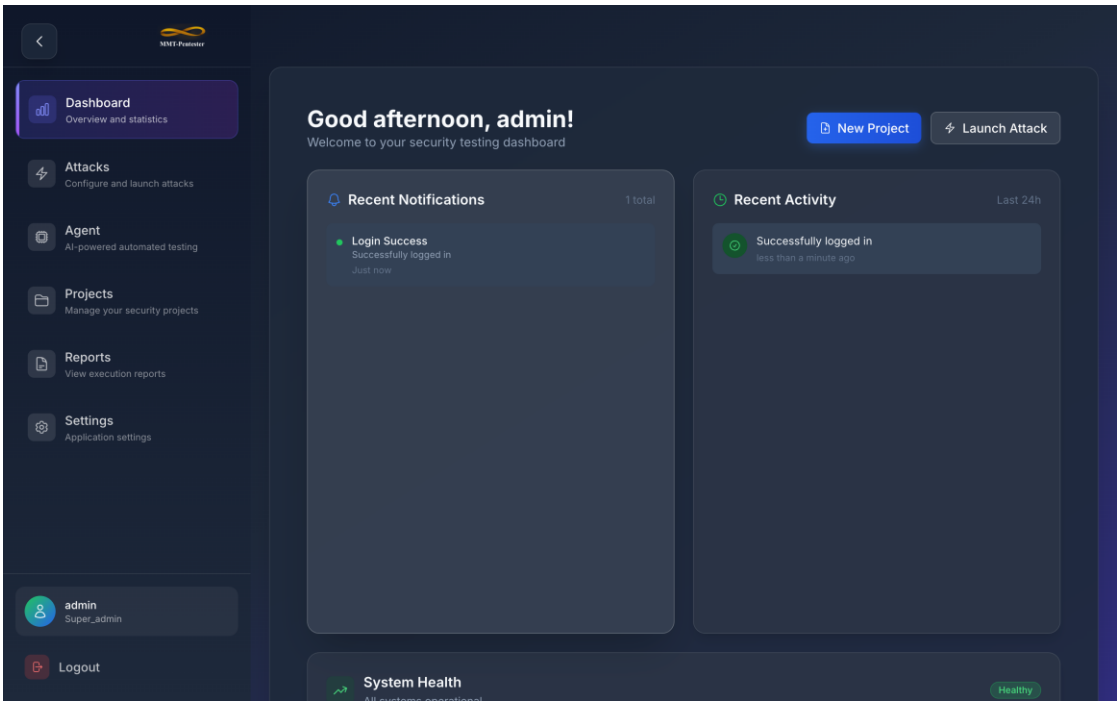



*Figure: MMT-Pentester Dashboard view with the left-hand navigation*

## The data model: Project → Campaign → Scenario

The single most important concept for finding your way around the tool is how work is organised. MMT-Pentester uses a three-level hierarchy:

```
Project            ("Acme Lab Assessment")
└── Campaign       ("External Recon")
    └── Scenario   ("Nmap scan of lab target")
        ├── targets:  { host, port, protocol }
        └── attacks:  { params }
```

- A **Project** is the top container for one engagement or client — for example, "Acme Lab Assessment."
- A **Campaign** is a phase or themed batch of work inside a project — for example, "External Recon."
- A **Scenario** is a single concrete, runnable test: one or more **targets** (each defined by a `host`, `port`, and `protocol`) plus one or more **attacks** (each with its own parameters).

Scenarios can execute **sequentially or in parallel**, and their output streams live over WebSocket so you watch the test unfold. Everything you do later in this course — the three guided scenarios (Nmap reconnaissance, SSH brute-force, and an HTTP Slowloris denial-of-service test) — lives inside this hierarchy. In this module's lab you will build the top two levels yourself: a Project containing one Campaign.

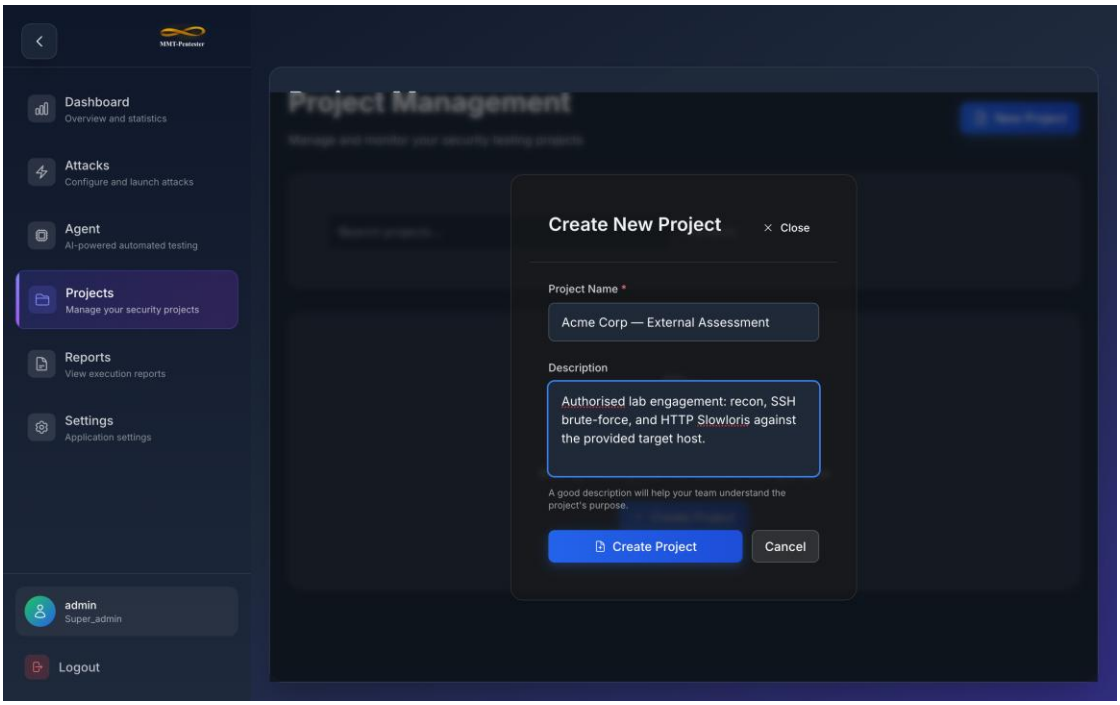



*Figure: Creating a new Project*

To create your first Project:

14. Open the **Projects** view.
15. Click **New Project**.
16. Enter a meaningful **name** (e.g. `My First Lab Project`) and a short **description**.
17. Set team access as needed — as super admin you already have access.
18. Save. The project appears in your Projects list.

Then create a Campaign inside it: open the project, create a **New Campaign**, give it a clear name (e.g. `Recon Phase`) and description, and save. The campaign now nests under the project, ready to hold scenarios later.

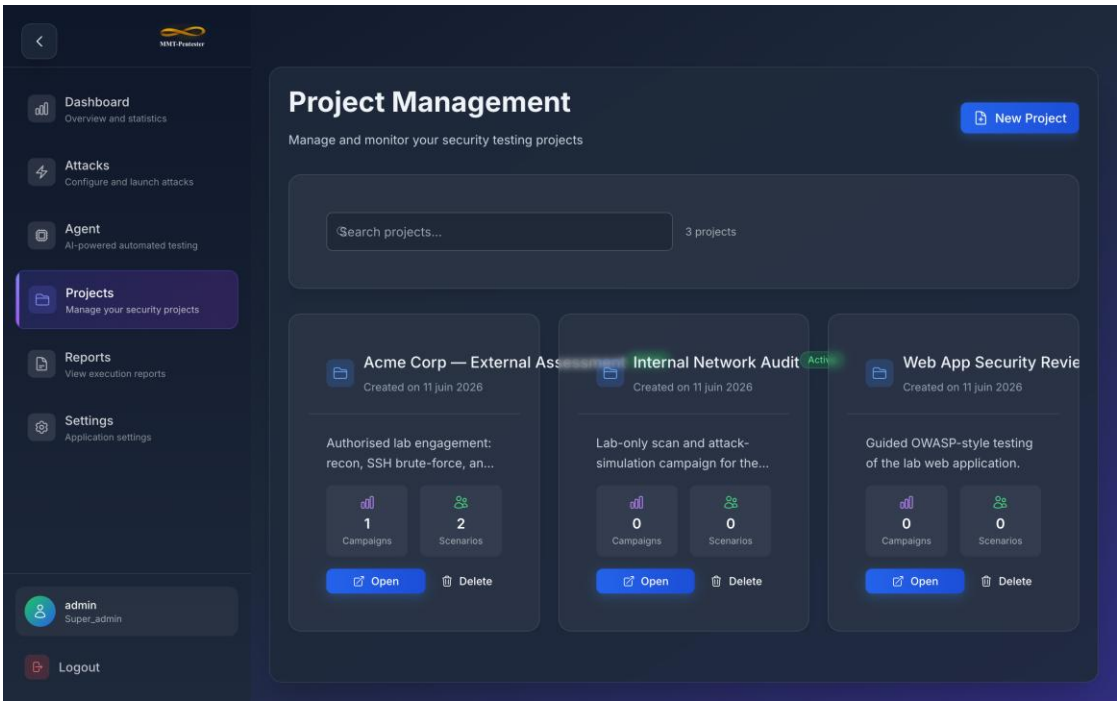



*Figure: Projects list showing a project and its nested campaign*

## Roles and access (RBAC)

MMT-Pentester uses **role-based access control (RBAC)** — your permissions depend on your assigned role. There are three roles, in order of power:

- **superadmin** — full control, including managing other admins and global settings. The default `admin` account is a superadmin.
- **admin** — manages users and projects within their scope.
- **user** — works inside the projects they have been granted access to.

The rule is simple: more senior roles can do everything a lower role can, plus more. And every action is written to **audit logs**, so who-did-what is always traceable. In real engagements that traceability protects both the client and the tester.

New accounts are created through **invite-code-gated registration**: a person needs a valid code (the `REGISTER_INVITE_CODE`) to sign up. This deliberately stops random strangers from creating accounts on a platform that can launch attack tooling. Admins can also disable open registration entirely and create accounts by hand for tighter control.

**As a course learner, register your own account** on the hosted instance (`https://mmt-pentester.montimage.eu/login` → *Create Account*) using the **invite code provided by Montimage / your instructor**. The `admin` account is reserved for platform administrators — don't use it for the labs. A reference project, *"CyberSuite — Pentesting Lab (Reference)"*, with worked recon / SSH brute-force / HTTP content-discovery scenarios against a provided lab target, is available on the platform for you to follow.

## Summary

- MMT-Pentester is a web "mission control" for *authorised* security testing — plan, launch, observe, and report on tests from one browser interface, against isolated lab targets only.
- Architecturally it is a React SPA in your browser, backed by an Express/Node API, a MongoDB database, two live WebSocket feeds, external tools run as child processes, and an AI assistant.
- Reach it via the hosted instance at `https://mmt-pentester.montimage.eu/login`; **course learners register their own account with the invite code** (the `admin` account is for administrators only).
- The interface has exactly five views: Dashboard, Projects, Scenarios, Agent, and Reports.
- Work is organised as Project → Campaign → Scenario, where a scenario holds targets (host, port, protocol) and attacks (params).
- Access is governed by three RBAC roles — superadmin > admin > user — with invite-code registration and full audit logging.

## Quiz: Getting Started with MMT-Pentester

*Pass mark: 7/10. Choose the best answer.*

19. In plain terms, what is MMT-Pentester?

    - A) A standalone antivirus product for end-user laptops
    - B) A web platform to plan, orchestrate, and execute authorised security tests from one interface
    - C) A spreadsheet add-in for tracking vulnerabilities
    - D) A firewall appliance you rack in a data centre

    **Answer:** B — It is a web "mission control" for planning, launching, observing, and reporting on authorised tests.

20. Which sequence describes the data model from largest container to smallest?

    - A) Scenario → Campaign → Project
    - B) Campaign → Project → Scenario
    - C) Project → Campaign → Scenario
    - D) Project → Scenario → Campaign

    **Answer:** C — A Project contains Campaigns, which contain Scenarios (which hold targets and attacks).

21. What must you do the very first time you log in with the default `admin` account?

    - A) Install the security tools
    - B) Disable the database
    - C) Create a report

- D) Change the default password

**Answer:** D — The default password (`admin123456`) is public, so the platform requires you to change it on first login.

22. Which list shows the five UI views exactly as they appear in MMT-Pentester?

    - A) Dashboard, Projects, Scenarios, Agent, Reports
    - B) Home, Tests, Tools, AI, Files
    - C) Overview, Campaigns, Attacks, Bot, Exports
    - D) Dashboard, Targets, Scenarios, Agent, Logs

    **Answer:** A — The exact views are Dashboard, Projects, Scenarios, Agent, and Reports.

23. How are live tool output and execution events delivered to the browser without refreshing the page?

    - A) By polling the database once a minute
    - B) Over WebSocket channels (Socket.io for events, a raw ws channel for tool output)
    - C) Through downloaded PDF reports
    - D) By email notifications

    **Answer:** B — Two WebSocket channels stream campaign/scenario events and raw tool output live.

24. Where can you reach the Montimage-hosted instance of the platform?

    - A) http://localhost:5173
    - B) https://github.com/Montimage/ai4sim
    - C) https://mmt-pentester.montimage.eu/login
    - D) http://localhost:3000/api

    **Answer:** C — The hosted instance login page is at `https://mmt-pentester.montimage.eu/login`; `localhost` addresses are for local development installs.

25. Which statement about the three RBAC roles is correct?

    - A) A `user` can manage other admins and global settings
    - B) Roles rank superadmin > admin > user, and higher roles include everything lower roles can do
    - C) There is only one role and everyone shares it
    - D) An `admin` outranks a `superadmin`

    **Answer:** B — Power runs superadmin > admin > user, with each higher role a superset of the one below it.

26. How is account creation controlled, and what happens after repeated failed logins?

- A) Anyone can self-register freely, and accounts never lock
- B) Invite-code-gated registration (which admins can disable), and the account locks after 5 failed logins
- C) Accounts are created automatically per IP address, and there is no lockout
- D) Only the database admin can add users via SQL, and lockout happens after 50 attempts

**Answer:** B — Registration is gated by an invite code (`REGISTER_INVITE_CODE`) and can be disabled; the account locks after 5 failed login attempts.

# Module 3: Reconnaissance & Attack Scenarios

*Estimated time: ~75 minutes · Text lesson + quiz*

## In this lesson

- Explain reconnaissance and distinguish passive from active information gathering.
- Use core Nmap flags to discover hosts, scan ports, and detect service versions, then read the output.
- Turn scan results into a minimal attack surface and configure scenario targets `{host, port, protocol}` and attacks `{params}` in MMT-Pentester.
- Run and observe the SSH brute-force scenario with Hydra (MITRE ATT&CK Credential Access) and the HTTP content-discovery scenario with dirb (MITRE ATT&CK Active Scanning / Brute Force).
- Choose between sequential and parallel execution and read live tool output streaming over WebSocket.

***Safety rule — read before you touch a tool.*** *Everything in this module runs* ***only*** *against the provided, isolated lab target, with explicit authorisation. Never scan, brute-force, or run a denial-of-service attack against real, production, or third-party systems — it is unlawful and harmful. The only difference between a penetration tester and an attacker is permission and scope, not technique. Keep every target you define inside the lab.*

## Reconnaissance: looking before you leap

Reconnaissance — "recon" — is the first phase of any engagement: you gather information about a target before you start testing it hard. The goal is to build an accurate picture of *what exists* (hosts, services, versions) so the later phases are targeted instead of blind. Good recon saves time, because you spend your effort on the soft spots you actually found rather than guessing. The broad, early sub-step is **footprinting**: sketching the outline of the target — address ranges, exposed services, technologies — before you zoom in. Think of recon as "measure twice, cut once."

There are two ways to gather that information:

- **Passive recon** collects data *without* directly touching the target — public DNS records, WHOIS, search engines, certificate logs, leaked data. It is low risk, low noise, and often legal even without scope.
- **Active recon** *directly probes* the target — pinging it, scanning its ports, grabbing service banners. It is faster and more accurate, but it generates traffic the target (and any monitoring) can see and log.

The moment you send a packet *to* the target, you are doing active recon and leaving a trace. Nmap scanning is active recon. The rule of thumb is to start passive to scope, then go active only inside an authorised, isolated lab — which is exactly where this course operates. In fact, the

lab's monitor and IDS *will* see your Nmap traffic, and that is the point: later in this module you will watch your own activity light up the detection view.

## Meet Nmap

**Nmap** (Network Mapper) is the standard open-source tool for active recon and enumeration. It answers three questions, in order:

27. **Host discovery** — which hosts are alive? ("Is it up?")
28. **Port scanning** — which TCP/UDP ports are open? ("What doors exist?")
29. **Service/version detection** — what software is behind each open port? ("Who answers the door?")

Beginners often jump straight to port scanning and skip whether the host is even up — keep the three-question framing. In MMT-Pentester, Nmap runs as a child process and its live output streams to the dashboard as it works.

### Core Nmap flags

| Flag | Name | What it does | When to use |
|---|---|---|---|
| `-sn` | Ping sweep | Host discovery only, no port scan | Find which hosts are alive |
| `-Pn` | No ping | Skip host discovery, assume host is up | When ping is blocked but the host is known up |
| `-p-` | All ports | Scan all 65535 TCP ports | Full coverage (slower) |
| `-p 22,80,443` | Port list | Scan only the listed ports | Fast, targeted scans |
| `-sV` | Version detection | Identify service software + version | Almost always — it drives the next steps |
| `-sC` | Default scripts | Run safe default NSE scripts | Quick extra enumeration |
| `-A` | Aggressive | `-sV` + `-sC` + OS detection + traceroute | Thorough one-shot (noisy) |
| `-T0`…`-T5` | Timing | Slow/stealthy → fast/noisy | `-T4` is a good lab default |
| `-oN file` | Output | Save normal output to a file | Record evidence for your report |

A thorough enumeration with versions and default scripts looks like this:

```
nmap -sC -sV -p- <lab-target>
```

A fast, targeted check for just the services this course uses:

```
nmap -sV -p 22,80 <lab-target>
```

Note that `-p-` combined with `-sV` can be slow — in a small lab it is fine, but on a wide network it is not subtle. Use `-Pn` when ping is blocked but you know the host is up.

### Reading the output

Each result line follows the pattern **PORT STATE SERVICE VERSION**:

```
PORT    STATE  SERVICE  VERSION
22/tcp  open   ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1
80/tcp  open   http     Apache httpd 2.4.52 ((Ubuntu))
443/tcp closed https
```

The **port state** is the first thing to read correctly:

- `open` — a service is accepting connections. These are the interesting ones.
- `closed` — the port is reachable but nothing is listening. Note it, but it is not a target.
- `filtered` — a firewall is blocking the probes, so Nmap cannot determine the true state. Worth recording.

The single most common beginner mistake is reading `filtered` as `closed` — they are not the same; `filtered` means "unknown." The **VERSION** column (produced by `-sV`) is gold: it tells you exactly what software and version to look up and test next. Always record open ports and their service/version strings verbatim — that record *is* your attack surface.

## From output to a minimal attack surface

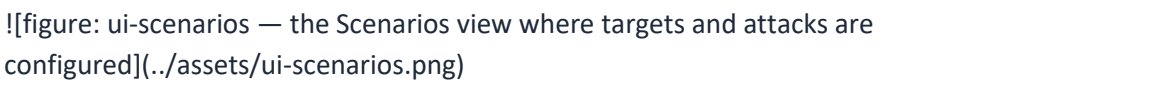
An **attack surface** is the set of exposed services an attacker could try to reach. For every `open` port, ask three questions: what is this service, what version is it, and is it a known entry point? Then prioritise the services that commonly allow access:

| Port | Service | Why it matters here |
|---|---|---|
| 22/tcp | SSH | Credential / brute-force target (Hydra) |
| 80/tcp | HTTP | Web content-discovery target (dirb) |
| 443/tcp | HTTPS | Encrypted web; same web surface |
| 21/tcp | FTP | Often weak/anonymous auth |
| 3306/tcp | MySQL | Database exposure |

In the example above, two services are `open`: **SSH on 22** and **HTTP on 80**. Those become your minimal attack surface and the targets of the two attack scenarios later in this module. A "minimal attack surface map" is not "all the data" — it is a short, ranked list of services worth testing, each with a one-line reason. Recon is about *targeting*.

## Running recon in MMT-Pentester

Recon runs as a **Scenario** inside the platform's data model: **Project → Campaign → Scenario → (targets + attacks)**. A Scenario holds one or more **targets** (`{host, port, protocol}` — the "where") and one or more **attacks** (`{params}` — the "how"), then executes them.




To run the recon scan:

30. Sign in to the MMT-Pentester Dashboard (dev URL `http://localhost:5173`). On a fresh install, change the default admin password (`admin` / `admin123456`) on first login.
31. Open the **Projects** view and create or open a project for this course; add a **Campaign**, then create a **Scenario** inside it.
32. In the Scenario, choose the **Reconnaissance with Nmap** template.
33. Set the **target** to the provided lab host only — never a real or third-party address.
34. Confirm the scan options (for example a `-sV` version scan over common ports, or `-p-` for full coverage).
35. **Execute** the scenario. A single sequential run is fine for one target.
36. Watch the **live tool output** stream to the dashboard over WebSocket as Nmap runs — you see results arrive in real time rather than waiting for a final report.
37. When the scan finishes, open the **Reports** view to review results and confirm expected vs actual behaviour on the MMT-Dashboard.
38. Record every open port, its service, and its version. Identify the SSH and HTTP services explicitly — they are the inputs to the attack scenarios below.

## Configuring targets and attacks for an attack scenario

Recon told you *what* is exposed; now you test *whether* those services are weak. The mental model stays crisp: **targets = where, attacks + params = how, the Scenario is the runnable unit.**

When you configure a **target**, set its three fields from your recon results:

- **host** — the lab target IP/hostname you discovered.
- **port** — the discovered service port (22 for SSH, 80 for HTTP).
- **protocol** — the service type (`ssh`, `http`, `tcp`…).

Targets are reusable across attacks within the same Scenario. The values must come from your recon output — do not guess ports. A wrong host or port means the attack silently does nothing.

When you configure an **attack**, you pick the tool and then fill its **params** — wordlists, thread count, send-rate, duration, and so on. Params control intensity and behaviour; sensible defaults are provided for the lab. Each attack is bound to a target so the tool knows where to point.

Beginners tend to over-tune: start with the lab defaults, then change one variable at a time so you can see what each one does.

### Sequential vs parallel execution

A Scenario can run its attacks in two modes:

- **Sequential** — attacks run one after another. This is predictable, the output is easier to read, and the load is lower.
- **Parallel** — attacks run at the same time. This is faster and simulates multi-vector pressure, but the output is noisier and harder to attribute.

Choose **sequential** while you are learning — and for both labs in this module — so each attack's live output is clean and attributable to a single tool. Reach for parallel later, when you deliberately want to model a realistic combined assault.

### Live output over WebSocket

The Dashboard uses two WebSocket channels. **Socket.io** carries campaign/scenario events, while a **raw WebSocket on port 9090** carries live tool stdout/stderr. "Live output" is literally the child process's output piped to your browser: you watch attempts, responses, and errors scroll in real time, with no need to wait for the final report. After the run, the same outcomes appear as results, alongside any monitor/IDS alerts. A common troubleshooting note: if you see no live output, check that port 9090 is reachable.

## Lab A — SSH brute-force with Hydra (Credential Access)

A **brute-force / credential attack** repeatedly tries username + password pairs until one authenticates — it is guessing, at machine speed. It targets weak, default, or reused credentials, here on the discovered SSH service (port 22). On MMT-Pentester this lab uses **Hydra**, a fast, widely-used network login cracker. It maps to **MITRE ATT&CK — Credential Access — Brute Force (T1110)**, including Password Guessing (T1110.001). (This is exactly why MMT-Pentester itself locks accounts after 5 failed logins.)

To run it:

39. In the **Scenarios** view, create a Scenario, e.g. `Lab A - SSH brute force`.
40. Add a **target**: `host` = the provided lab target, `port` = the SSH port you found during recon, `protocol` = `ssh`.
41. Add the **Hydra** attack and bind it to that target.
42. Fill the **params**: a username (the lab account, e.g. `root`), the provided lab password list (`wordlist`), the `service` = `ssh` and `port` = 22. Start with the lab defaults.
43. Set execution mode to **sequential** and launch the Scenario.
44. Watch the **live output** over WebSocket: Hydra prints each attempt, and reports a found `login: … password: …` line if a weak credential exists.
45. Switch to the **MMT-Dashboard** observation view and confirm the burst of failed authentications plus any IDS/monitor **alert on repeated login attempts**.

46. Capture screenshots of the live output and the detection alert.

```
# Example Hydra invocation and output (illustrative)
hydra -l root -P /home/ubuntu/lab-passwords.txt -s 22 <target> ssh
[DATA] attacking ssh://<target>:22/
[22][ssh] host: <target>   login: root   password: <found-or-none>
1 of 1 target completed
```

## Lab B — HTTP content discovery with dirb (Active Scanning / Brute Force)

***Note on the lab tool.*** *The platform does not ship a denial-of-service tool, so this lab uses* ***dirb****, an HTTP content/path brute-forcer, against the discovered web service. It is a real, observable web attack that generates a clear burst of HTTP requests — ideal for the detection work in the next module — without taking the service down.*

A **content-discovery / forced-browsing** attack requests many candidate paths (`/admin`, `/backup`, `/login`, …) from a web server to find hidden or unlinked resources. dirb walks a wordlist of common paths and reports which return valid responses. It maps to **MITRE ATT&CK — Reconnaissance / Active Scanning** and the web side of **Brute Force**.

To run it:

47. In the **Scenarios** view, create a Scenario, e.g. `Lab B - HTTP content discovery`.
48. Add a **target**: `host` = the provided lab target, `port` = the HTTP port you found during recon, `protocol` = `http`.
49. Add the **dirb** attack and bind it to that target.
50. Fill the **params**: the `target` and optionally a `wordlist` (the lab default `common.txt` is fine).
51. Set execution mode to **sequential** and launch the Scenario.
52. Watch the **live output** over WebSocket: dirb prints each path tested and flags found resources (`+ http://<target>/admin (CODE:200 …)`).
53. In the **MMT-Dashboard** observation view, confirm the spike of HTTP requests and any IDS/monitor **alert on web scanning / enumeration**.
54. Capture screenshots of the discovered paths and the detection alert.

```
# Example dirb invocation and output (illustrative)
dirb http://<target> /usr/share/dirb/wordlists/common.txt
+ http://<target>/index.html (CODE:200|SIZE:1024)
+ http://<target>/admin (CODE:301|SIZE:0)
==> DIRECTORY: http://<target>/uploads/
```

## Observing results and reporting

Penetration testing is half attack, half observation — the blue-team view (detection) is what makes this a learning exercise rather than just "running a tool." On the **MMT-Dashboard** you watch traffic seen, expected vs actual behaviour, and IDS/monitor alerts. For the Hydra brute-force run, look for failed-auth bursts, possible lockout or throttling, and an alert on repeated attempts. For the dirb run, look for the spike of HTTP requests to many distinct paths and any web-scanning / enumeration alert.

The platform auto-generates reports from the **Reports** view: **PDF** (via Puppeteer) and **Excel** (via ExcelJS). Use these alongside your own annotated screenshots to assemble an evidence bundle. A good run is reproducible: target, attack, params, observed output, and detection all recorded.

## Summary

- Recon comes first: gather information before testing. Passive recon never touches the target; active recon (including all Nmap scanning) does and leaves traces — so it runs only inside the authorised lab.
- Nmap's workflow is host discovery → port scan → service/version detection. Read results by **state** (`open`/`closed`/`filtered`) and **version**, and never confuse `filtered` with `closed`.
- Turn open services into a short, ranked attack surface; SSH (22) and HTTP (80) are the two services this course attacks.
- The data model is Project → Campaign → Scenario; a target is `{host, port, protocol}` (where) and an attack is `{params}` (how), with values taken from your recon output.
- SSH brute-force with **Hydra** is ATT&CK **Credential Access (T1110)**; HTTP content discovery with **dirb** is ATT&CK **Active Scanning / Brute Force**. Run both **sequentially** while learning for clean, attributable output.
- Live tool output streams over the raw WebSocket on **port 9090** (Socket.io carries scenario events); observe detection on the Dashboard and export PDF/Excel reports as evidence.

## Quiz: Module 3 — Reconnaissance & Attack Scenarios

*Pass mark: 7/10. Choose the best answer.*

55. Which activity is an example of **passive** reconnaissance?

    - A) Running an Nmap port scan against the target
    - B) Looking up the target's public WHOIS and DNS records
    - C) Grabbing a service banner with `-sV`
    - D) Sending a ping sweep with `-sn`

    **Answer:** B — WHOIS/DNS lookups never touch the target; the others all send packets to it, making them active recon.

56. What does the Nmap flag `-sV` do?

    - A) Scans all 65535 ports
    - B) Skips host discovery
    - C) Detects the service software and version on open ports
    - D) Runs the default NSE scripts

    **Answer:** C — `-sV` is service/version detection; `-p-` scans all ports, `-Pn` skips discovery, `-sC` runs default scripts.

57. In Nmap output, a port marked `filtered` means:

- A) A service is accepting connections
- B) Nothing is listening on that port
- C) A firewall is blocking the probes and Nmap can't determine the state
- D) The host is down

**Answer:** C — `filtered` means probes are being dropped (usually by a firewall), so the true state is unknown — not the same as `closed`.

58. In the MMT-Pentester data model, which fields define a scenario **target**?

- A) username, password, wordlist
- B) connections, interval, duration
- C) project, campaign, report
- D) host, port, protocol

**Answer:** D — A target is `{host, port, protocol}`; the rest are attack params or other levels of the model.

59. The SSH brute-force scenario maps to which MITRE ATT&CK tactic?

- A) Impact
- B) Reconnaissance
- C) Credential Access
- D) Exfiltration

**Answer:** C — Repeatedly guessing username/password pairs is Credential Access — Brute Force (T1110).

60. What does the dirb tool do in Lab B?

- A) Floods the server with traffic to take it offline
- B) Brute-forces HTTP paths from a wordlist to discover hidden/unlinked resources
- C) Cracks SSH passwords offline
- D) Captures wireless handshakes

**Answer:** B — dirb performs HTTP content discovery (forced browsing), requesting many candidate paths to find resources like /admin or /backup.

61. You want clean, attributable live output while learning both attack labs. Which execution mode should you choose?

- A) Parallel, so both finish faster
- B) Sequential, so each attack's output is separate and readable
- C) Neither — output mode has no effect
- D) Always parallel for any scenario

**Answer:** B — Sequential runs attacks one after another, keeping each tool's output clean; parallel is for modelling combined pressure.

62. During a run, where does the **live tool output** stream from?

- A) A nightly batch email
- B) The MongoDB change log only
- C) A raw WebSocket on port 9090 (with Socket.io carrying scenario events)
- D) The PDF report after the run finishes

**Answer:** C — Live stdout/stderr streams over the raw WebSocket on port 9090; Socket.io carries scenario/campaign events.

# Module 4: Monitoring, Detection & Reporting

*Estimated time: ~45 minutes · Text lesson + quiz*

## In this lesson

- Read the live MMT-Dashboard during and after an execution to follow what a test is actually doing.
- Compare expected versus actual behaviour to tell a successful test from a blocked or failed one.
- Explain at a high level what an IDS or monitor alert means and how detection works.
- Describe the five-part anatomy of a good finding: title, severity, evidence, impact, remediation.
- Generate and interpret the platform's PDF (Puppeteer) and Excel (ExcelJS) reports, and write actionable remediation.

## From running attacks to understanding them

In the previous modules you launched scenarios — Nmap reconnaissance, an SSH brute-force, an HTTP Slowloris denial-of-service — against the provided lab targets. This module is about the other half of the job: *reading the result*. Launching a tool is the easy part. The value of a penetration test lives in what you can say about it afterwards.

Every execution produces three things worth watching: the live tool output, the Dashboard's view of campaign and scenario state, and any detection alerts the lab's monitor raises. Learn to read all three and you stop being someone who operates a tool and start being someone who analyses an outcome.

Keep one principle in mind throughout: **nobody pays for "I ran a scan." They pay for "here is what is wrong, how I know, and how to fix it."** Evidence beats opinion, every time. And as always, everything here applies ONLY to the provided, isolated lab targets you have explicit authorisation to test — never to real or third-party systems.

## Reading the live MMT-Dashboard

![figure: dashboard-monitoring — MMT-Dashboard live execution with the log pane streaming tool output](#)

Open the **Dashboard** view at `http://localhost:5173` and make sure you are logged in. The live panes only update while you hold an active connection, because they are fed by WebSocket channels — close the tab and the stream stops.

The Dashboard is fed by two distinct live channels, and the difference matters:

- **Socket.io** carries campaign and scenario *lifecycle events* — the status that moves a run through `queued → running → completed` (or `failed`). This tells you *where* a run is in its life.

- A **raw WebSocket on port 9090** carries the tool's actual `stdout` and `stderr` — exactly what the tool prints to the terminal. This tells you *what the tool is saying*.

While an execution runs, watch three things at once: the **status badge** on the scenario or campaign, the **live log pane**, and the per-target progress. Both channels matter, and they can disagree. A green "completed" badge sitting on top of a log pane full of errors is a classic trap — the run finished, but it did not necessarily do anything useful. Always read the log, not just the badge.

When the run finishes, move to the **Reports** view to find the report record the platform generated for that execution.

## Expected versus actual behaviour

The single most valuable analytical habit in this course is simple: **write down what you expect to happen before you run the test, then compare it against what actually happened.**

Each guided scenario has an expected signature — a pattern of output and target response that indicates the attack landed:

| Scenario | Expected signature (success) | Common "actual" failure | What the mismatch means |
|---|---|---|---|
| Nmap recon | Open ports and services listed | Empty or all-filtered result | Target unreachable, wrong IP, or host down |
| SSH brute-force | Repeated auth attempts in the log; maybe a hit | Zero auth attempts | Service not running, port closed, or blocked upstream |
| HTTP Slowloris DoS | Response times climb; timeouts; an IDS alert | Target responds normally throughout | Connections refused, target hardened, or too few sockets |

The "actual" is whatever the live output and the target's real response tell you. When actual does not match expected, that gap is *information* — and usually a finding to investigate, not a reason to shrug and assume the target is secure.

Consider a brute-force that finishes "completed" but shows **zero authentication attempts** in the log. That does not mean the SSH service shrugged off your attack; it almost certainly means the attack never reached the service at all — a closed port, an unreachable host, or something blocking the traffic upstream. **Detection-free does not mean attack-free.** It may simply mean nothing landed.

## What IDS and monitor alerts mean

An **IDS** — Intrusion Detection System — is software that inspects network traffic or system behaviour and raises an alert when it sees something suspicious. Think of it as a defender's

smoke detector. You do not need to configure one for this course; you need to understand what an alert is telling you.

Detection works in two broad ways:

- **Signature-based detection** matches traffic against known-bad patterns. A burst of many SSH login attempts in a few seconds matches the "brute-force" signature and trips an alert.
- **Anomaly-based detection** learns a baseline of what "normal" looks like and flags deviations from it. A sudden flood of half-open HTTP connections during a Slowloris attack stands out from the baseline and trips an alert.

On the Dashboard, alerts appear alongside the execution output. Every alert answers one question: *would a defender have noticed this?* A test that trips loud alerts is **noisy**; a test that slips by quietly is **stealthy**. Both outcomes are worth reporting. "We compromised the host and the monitor never alerted" is a serious finding about the defender's blind spots, just as "any of these attacks triggers an immediate alert" is a useful reassurance.

## The anatomy of a good finding

This is the heart of the module. A finding is not "I think the SSH is weak." A finding has **five required parts**, and the platform's report data model stores each one:

| Part | The question it answers | Report field |
|---|---|---|
| **Title** | What is the problem, in one line? | `vulnerability` |
| **Severity** | How bad is it? (critical / high / medium / low) | `severity` |
| **Evidence** | How do we know? (logs, output, screenshots) | `evidence[]` |
| **Impact** | What could an attacker achieve? | `impact` |
| **Remediation** | What exactly fixes it? | `remediationInstructions` |

A finding missing any one of these is incomplete. **Evidence** is the part beginners most often skip, and it is precisely what separates a credible finding from a guess — a captured log line, a screenshot of the log pane, a recorded response. The data model above is not theoretical; it is exactly what the report stores.

Here is one complete finding, written the way it should be:

```
Title:        SSH service permits password authentication and is brute-forceable
Severity:     high
Service/Port: ssh / 22
Evidence:     - Live log: 312 authentication attempts accepted by the service in 40s
              - Screenshot: dashboard log pane showing repeated "Failed password" lines
Impact:       An attacker who guesses or cracks a weak password gains a shell on the
host,
```

```
              enabling lateral movement and data theft.
Remediation:  Disable password authentication (PasswordAuthentication no), enforce SSH
keys,
              install fail2ban with a 5-attempt / 15-minute lockout, and restrict SSH
to a
              management network.
```

### Severity and the overall risk score

Severity reflects how bad a *single* finding is, on the platform's four-tier scale of critical / high / medium / low. The report then rolls all findings into an overall `riskScore`, and a statistics block counts them by tier — `criticalFindings`, `highFindings`, `mediumFindings`, `lowFindings`.

Judge severity by **impact multiplied by ease of exploitation**, not by how impressive the attack felt. Resist severity inflation: labelling everything "critical" destroys the reader's trust and buries the things that truly matter. A medium-severity issue that is trivially exploitable can deserve more urgent attention than a critical one that requires physical access to the building. As with everything in this module — let the evidence, not the adrenaline, set the rating.

## Generating and interpreting the reports

![figure: ui-reports — the Reports view with a completed report and download options](#)

The platform produces two complementary deliverables from the **same** report record, so they never contradict each other.

**The PDF report (Puppeteer).** Puppeteer drives a headless Chrome to render a polished, client-ready document. This is the *human* deliverable — it goes to management and asset owners. It contains an executive summary written for non-technical readers, the attack narrative and methodology, per-finding detail, a remediation plan, and the statistics block. Look for the filename pattern `mmt-pentester-rapport-securite-<target>-<date>.pdf` and download it from the Reports view.

**The Excel report (ExcelJS).** ExcelJS exports structured data as an `.xlsx` file (CSV and JSON downloads are available too). This is the *machine and tracking* deliverable: one row per finding, sortable and filterable. Teams use it to triage — sort by severity, assign an owner to each row, and track remediation status over time. Rows paste straight into ticketing systems.

The download endpoint accepts a format (`json` | `csv` | `pdf`), and because both outputs come from one underlying record, the rule of thumb is simple: **pick the PDF to tell the story, pick the spreadsheet to drive the fix workflow.**

Whichever format you open, the report is organised the same way: metadata (target, scan type, tools, timing, linked project/campaign/scenario), the executive summary, the attack narrative and methodology, the statistics, the vulnerabilities (one block each, using the five-part anatomy), the prioritised remediation plan, and recommendations for next steps.

## Writing actionable remediation

A remediation that a busy sysadmin cannot act on without further research is barely a remediation at all. Compare:

- **Bad:** "Harden SSH."
- **Good:** "Disable password authentication, enforce key-based login, and add fail2ban with a 5-attempt lockout."

The good version is specific, verifiable, and prioritised. The platform's `remediationPlan` gives you a built-in checklist — fill in every field:

- `priority` — what should be fixed first.
- `fix` — the concrete steps.
- `estimatedEffort` — how much work it is.
- `businessImpact` — why it is worth doing.

Always pair the *what* (the fix) with the *why* (the impact), so the owner can justify the work to whoever signs off on it. This is where junior reporters most often fall short — push for instructions someone can execute today.

## Putting it together: the reporting loop

The full loop runs: **run → observe live → compare expected versus actual → capture evidence → write findings → generate the PDF and Excel → recommend fixes.** A good report is reproducible — someone else should be able to follow your evidence and see the same thing you saw.

And reporting is not the end of the engagement. It is the start of the fix. The platform automates the formatting; the *judgement* — what severity to assign, what impact to claim, what to fix first — is yours, and it is the skill that makes you valuable.

## Summary

- An execution gives you three things to read: the live tool output (raw WebSocket, port 9090), the Dashboard's lifecycle status (Socket.io), and any IDS/monitor alerts — never trust the green badge alone.
- Write your expected outcome before you run; when actual does not match expected, the gap is a finding to investigate, and detection-free does not mean attack-free.
- An IDS alert answers one question — would a defender have noticed? — and both "noisy" and "stealthy" results are reportable.
- A good finding has five parts: title, severity, evidence, impact, remediation; evidence is what turns a guess into a finding.
- The PDF (Puppeteer) tells the story for stakeholders; the Excel (ExcelJS) drives the triage workflow; both come from one record, so they never disagree.

- Remediation must be specific, verifiable, and prioritised — pair every fix with its business impact.

## Quiz: Module 4 — Monitoring, Detection & Reporting

*Pass mark: 7/10. Choose the best answer.*

63. During a live execution, which channel carries the raw tool stdout/stderr to the Dashboard?

    - A) Socket.io campaign/scenario events
    - B) The REST API on port 3000
    - C) The raw WebSocket on port 9090
    - D) MongoDB change streams

    **Answer:** C — raw tool output streams over the WebSocket on port 9090; Socket.io carries lifecycle events.

64. An SSH brute-force execution finishes "completed" but the live log shows zero authentication attempts. What is the best interpretation?

    - A) The target is secure and rejected everything silently
    - B) The attack likely never reached the service (closed port or unreachable host)
    - C) The IDS deleted the log entries
    - D) The report will automatically mark it critical

    **Answer:** B — no auth attempts means the attack never landed; "completed" status does not equal "attack succeeded."

65. Which set lists the five required parts of a good finding?

    - A) Title, Cost, Author, Date, Tool
    - B) Severity, CVE, Screenshot, Owner, Deadline
    - C) Title, Severity, Evidence, Impact, Remediation
    - D) Summary, Risk Score, Methodology, Narrative, Next Steps

    **Answer:** C — Title, Severity, Evidence, Impact, Remediation; evidence is the part beginners most often omit.

66. What is the main practical difference between the PDF (Puppeteer) and Excel (ExcelJS) reports?

    - A) The PDF is for stakeholders and storytelling; the Excel is structured data for triage and ticketing
    - B) The Excel contains real findings and the PDF contains placeholders
    - C) They are generated from different scans and often disagree
    - D) Only the PDF includes remediation

**Answer:** A — both come from the same report record; the PDF tells the story, the spreadsheet drives the fix workflow.

67. Which remediation recommendation is the most actionable?

    - A) "Improve SSH security."
    - B) "Disable password authentication, enforce key-based login, and add fail2ban with a 5-attempt lockout."
    - C) "Consider hardening the server at some point."
    - D) "Mark this finding as high severity."

    **Answer:** B — it is specific, verifiable, and tells the owner exactly what to do; A and C are vague, D is not a fix.

68. How does signature-based detection differ from anomaly-based detection?

    - A) Signature-based learns a baseline of normal; anomaly-based matches known-bad patterns
    - B) Signature-based matches known-bad patterns; anomaly-based flags deviations from a learned baseline
    - C) Both require manual configuration by the tester before each run
    - D) Anomaly-based only works on SSH and signature-based only on HTTP

    **Answer:** B — signatures match known patterns (e.g. a burst of SSH logins); anomaly detection flags departures from normal (e.g. a flood of half-open connections).

69. Which fields does the platform's remediation plan store to make a fix actionable?

    - A) priority, fix, estimatedEffort, businessImpact
    - B) author, date, cost, deadline
    - C) cve, cwe, references, screenshot
    - D) riskScore, criticalFindings, highFindings, lowFindings

    **Answer:** A — priority, fix, estimatedEffort, and businessImpact form a built-in checklist; pair every fix with its business impact.

70. A Slowloris test against a lab target triggers a loud IDS alert. What should you conclude when reporting?

    - A) The test failed and should be discarded
    - B) The result is not worth reporting because it was detected
    - C) The test was "noisy" — a defender would have noticed it, which is itself a reportable observation
    - D) The PDF report will automatically suppress the finding

**Answer:** C — an alert means a defender would notice; noisy and stealthy outcomes are both reportable, and severity is judged on impact and exploitability, not on whether it was detected.

# Module 5: Advanced – AI Agents & Tool Integration

*Estimated time: ~45 minutes · Text lesson + quiz*

## In this lesson

- Describe what the Agent view does and how AI-assisted, orchestrated test runs work in MMT-Pentester.
- Explain, at a conceptual level, how AgentOrchestrator coordinates a run and how MCP plus Together AI turn tool output into analysis and next-step recommendations.
- Identify each integrated tool — Caldera, MAIP, Shennina, KNX Smart Fuzzer, GAN Fuzzer, and MAG/MMT-Attacker — and recognise when each is the right choice.
- Apply human-in-the-loop judgement to AI recommendations within an authorised lab scope.

## From guided scenarios to assisted orchestration

In the earlier modules you drove three guided scenarios step by step: reconnaissance with Nmap, an SSH brute-force, and an HTTP Slowloris denial-of-service test. In each one, you decided what to run, watched the live output, and interpreted the results yourself.

This module looks at the **Agent** view, where the platform can coordinate a test run for you and use AI to suggest a sensible next step. The aim here is conceptual literacy — understanding what the agent actually does and what the integrated tool ecosystem is for — rather than completing a graded exercise. It is the most advanced part of the course, so we will be careful not to overstate what the AI can do on its own.

One thing does not change: everything described here runs **only** against provided, isolated lab targets, with explicit authorisation. Never against real or third-party systems. The AI does not relax that boundary.

## What "autonomous agent" means here

The word "autonomous" can sound like the platform hacks for you. It does not. In MMT-Pentester an **autonomous agent** is simply a loop:

71. **Run** a tool against the authorised target.
72. **Observe** the output it produces.
73. **Analyse** that output.
74. **Recommend** a reasonable next action.
75. **Decide** — you review the recommendation and approve, adjust, or stop.

The human stays in control throughout. The agent *proposes* and surfaces recommendations; you authorise meaningful steps. It does not replace the pentester. What it does is reduce the busywork between obvious steps — for example, noticing that a scan found an open SSH port and suggesting "consider a credential test against this service." Think of it as **assisted orchestration**, not hands-off hacking.

## The Agent view

The Agent view is one of the five platform views you already know: Dashboard, Projects, Scenarios, **Agent**, and Reports. It is where you start, watch, and review an AI-coordinated session against a lab target.

Crucially, the Agent view ties into the same data model as everything else. There is nothing new to learn structurally: a **Project** contains **Campaigns**, a campaign contains **Scenarios**, and a scenario points at **targets** (host, port, protocol) plus **attacks** (with parameters). The agent operates on those same primitives. Live tool output streams into the view over WebSocket, exactly as it did in the guided scenarios — so the experience of watching a run will already feel familiar.

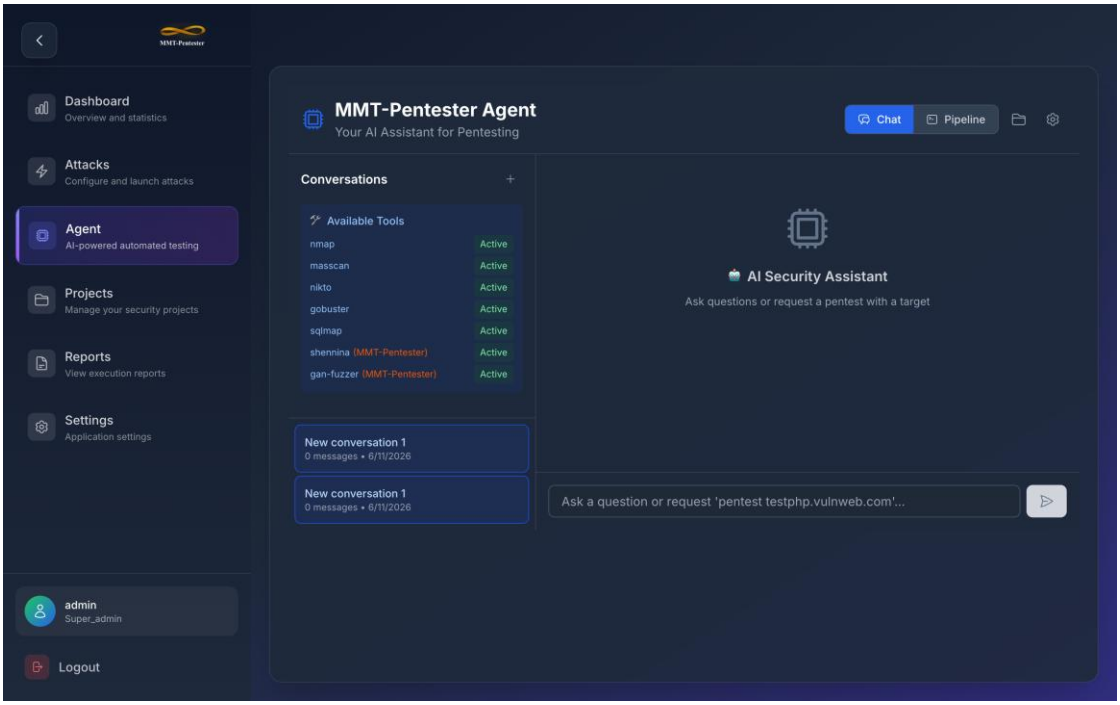



*Figure: Ui-agent*

## AgentOrchestrator: coordinating a run

Behind the Agent view sits a backend service called **AgentOrchestrator**. Conceptually it is the coordinator that connects three things: the tools, the AI layer, and the UI.

AgentOrchestrator is responsible for:

- **Launching and sequencing** tool executions against the authorised target.
- **Collecting** the output those tools produce as it streams back.
- **Routing** that output to the AI layer for analysis.
- **Managing run state** and feeding results — progress and recommendations — back into the platform so you see them in the Agent view.

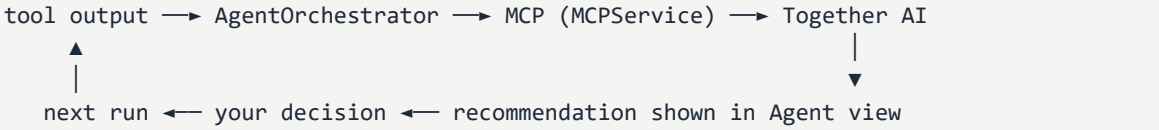
It uses the same execution machinery as the rest of the platform: tools run as child processes, executions can be sequential or parallel, and output streams live over WebSocket. You do not need to know its internals. The mental model is enough: *a coordinator that wires tools to the AI and the AI back to you.*

## MCP and Together AI: turning output into insight

Raw scan and attack output is not, by itself, advice. Something has to read it and reason about it. That is the job of the AI layer, and two pieces make it work.

**MCP (Model Context Protocol)** is a standard interface for handing structured context to an AI provider. In MMT-Pentester a component called `MCPService` connects to an MCP server, and the AI provider that performs the analysis is **Together AI**. MCP is best understood as the plumbing that carries context to the model — it is not where the "thinking" happens, and it does not store logs or replace WebSocket; it simply transports the tool output to the model in a structured form.

The flow looks like this:




So a stream of scan results enters one end, and a readable "here is what I see, here is what you might do next" comes out the other — surfaced to you in the Agent view. The AI reasons **only over the output it is given**. It cannot see your network beyond that output, and it can be wrong.

## Reading recommendations critically

AI recommendations are grounded in observed output, but grounded does not mean correct. A recommendation can be wrong, too generic to be useful, or incomplete because the AI only saw part of the picture.

Treat every recommendation as an *input to your judgement*, not an instruction to follow blindly. Validate it against what you can actually see:

- the live tool output in the Agent view,
- any IDS or monitor alerts,
- and the observations on the MMT-Dashboard (traffic seen, expected versus actual behaviour).

You decide whether to accept the suggestion, adjust it, or ignore it. And no recommendation ever overrides the authorised-scope and safety rules. If the AI suggests something outside your lab scope, the answer is simply no.

## The integrated tool ecosystem

MMT-Pentester runs a set of external security tools as child processes, and the agent can orchestrate them. Each tool answers a different question, so choosing the right one matters far more than running them all. The toolbox includes the following.

| Tool | Purpose | Typical use |
|---|---|---|
| **Caldera** | Adversary simulation / automated red-team TTP chains | Emulate a realistic attacker's tactics, techniques, and procedures against a lab host to see how they chain together |
| **MAIP** (Montimage Attack Injection Platform) | Inject attacks/anomalies into network traffic | Introduce malicious or malformed traffic to test detection and monitoring on the MMT-Dashboard |
| **Shennina** | Automated exploitation experiments | Explore automated exploitation flows against a discovered, vulnerable lab service |
| **KNX Smart Fuzzer** | Protocol fuzzing for KNX / building automation | Send unexpected or malformed KNX inputs to test the robustness of building-automation devices in the lab |
| **GAN Fuzzer** | ML (GAN)-driven fuzzing | Generate machine-learned input variations to probe a target for edge-case handling failures |
| **MAG / MMT-Attacker** (Montimage Attack Generator) | Generate attack traffic and payloads | Produce attack traffic or payloads to drive a scenario or feed detection testing |

### Choosing a tool — quick decision cues

A simple way to route your intent to the right tool:

- Want to mimic a full attacker playbook of chained tactics? → **Caldera**.
- Want to test whether your monitoring catches malicious traffic? → **MAIP**.
- Want to try automated exploitation of a known-vulnerable lab service? → **Shennina**.
- Testing a KNX or building-automation protocol for robustness? → **KNX Smart Fuzzer**.
- Want ML-generated, hard-to-anticipate inputs? → **GAN Fuzzer**.
- Need to generate attack traffic or payloads for a scenario? → **MAG / MMT-Attacker**.

You do not need deep expertise in any of these yet — a mental routing table is enough at this stage.

## Putting it together: a typical agent session

Here is how the pieces combine in one session, so you can picture the whole loop.

You pick a provided lab target — say, one of the hosts from the earlier Nmap recon scenario — and start a session in the Agent view. AgentOrchestrator runs an initial step, such as reconnaissance, and streams the output live. That output is sent via MCP to Together AI, which returns an analysis and a suggested next step, for instance "an SSH service was discovered on port 22; consider a credential test." The recommendation appears in the Agent view.

You review it. Does it match what you see in the live output and on the MMT-Dashboard? Is it inside your authorised scope? If yes, you let the loop continue; if not, you adjust or stop. As the session proceeds you observe results on the MMT-Dashboard and in the auto-generated Reports, just as with the guided scenarios.

Every action you can trigger from the UI is also available over the platform's REST API, which ships with an interactive Swagger reference. Advanced users automate scans, attacks, report retrieval, and agent runs by calling those endpoints directly — handy for CI pipelines or repeatable test harnesses.

## Safety and ethics — always

All tool runs and agent sessions execute **only** against provided, isolated lab targets, with explicit authorisation — never against real or third-party systems. AI recommendations do not change this boundary; they are suggestions you evaluate within your authorised scope. This platform is for education and authorised testing only.

## Summary

- The Agent view is one of the five platform views and runs AI-assisted sessions on the same Project → Campaign → Scenario data model, with live output over WebSocket.
- An autonomous agent here is a loop — run, observe, analyse, recommend, decide — in which the human always authorises meaningful steps.
- AgentOrchestrator coordinates a run by sequencing tools, collecting output, and routing it to the AI layer; MCP carries that output to Together AI, which returns analysis and a next-step recommendation.
- AI recommendations are grounded in observed output but can be wrong; validate them against the live output, IDS/monitor alerts, and the MMT-Dashboard, and accept, adjust, or ignore them.
- The tool ecosystem spans simulation (Caldera), traffic injection (MAIP), automated exploitation (Shennina), protocol and ML fuzzing (KNX Smart Fuzzer, GAN Fuzzer), and traffic/payload generation (MAG/MMT-Attacker) — pick by purpose.
- Human-in-the-loop and authorised-scope rules always apply; the AI never relaxes the safety boundary.

## Quiz: Module 5 – Advanced – AI Agents & Tool Integration

*Pass mark: 7/10. Choose the best answer.*

76. What best describes the role of AgentOrchestrator?

- A) It replaces the pentester and runs tests fully autonomously
- B) It coordinates an agent run — sequencing tool executions and routing output to the AI layer
- C) It is the database that stores scenarios
- D) It is the React component that renders the Agent view

**Answer:** B — AgentOrchestrator coordinates runs and connects tools, the AI layer, and the UI; the human still authorises steps.

77. In MMT-Pentester, an "autonomous agent" is best understood as:

- A) A system that finds and exploits vulnerabilities with no human involvement
- B) A loop of run → observe → analyse → recommend → decide, with the human authorising steps
- C) A separate paid AI product unrelated to the platform
- D) A scripted scenario that cannot be changed once started

**Answer:** B — It is assisted orchestration: the agent proposes, the human reviews and authorises.

78. What is the purpose of MCP (Model Context Protocol) in this platform?

- A) To encrypt traffic between the browser and the API
- B) To store audit logs of every agent action
- C) To carry structured context (tool output) to the AI provider for analysis
- D) To replace WebSocket for live tool output

**Answer:** C — MCP is the interface that hands tool output to Together AI so it can analyse and recommend a next step.

79. Which AI provider performs the analysis behind the Agent view?

- A) Together AI
- B) MongoDB
- C) Socket.io
- D) Puppeteer

**Answer:** A — `MCPService` connects to the MCP server and Together AI performs the analysis.

80. Which statement about AI recommendations is correct?

    - A) They are guaranteed correct and should be followed automatically
    - B) They override the authorised-scope rules when needed
    - C) They are grounded suggestions you must validate and can accept, adjust, or ignore
    - D) They are generated without using any tool output

    **Answer:** C — Recommendations are inputs to your judgement; the human stays in the loop and safety/scope rules always apply.

81. You want to emulate a realistic attacker's full tactics-and-techniques playbook against a lab host. Which tool fits best?

    - A) KNX Smart Fuzzer
    - B) Caldera
    - C) GAN Fuzzer
    - D) MAG / MMT-Attacker

    **Answer:** B — Caldera is the adversary-simulation tool for emulating chained attacker TTPs.

82. Which tool is purpose-built for fuzzing a building-automation protocol in the lab?

    - A) Shennina
    - B) MAIP
    - C) KNX Smart Fuzzer
    - D) Caldera

    **Answer:** C — The KNX Smart Fuzzer targets the KNX building-automation protocol with unexpected or malformed inputs.

83. You want to test whether your monitoring catches malicious or malformed network traffic. Which tool should you reach for?

    - A) MAIP (Montimage Attack Injection Platform)
    - B) GAN Fuzzer
    - C) Caldera
    - D) Shennina

    **Answer:** A — MAIP injects attacks and anomalies into network traffic to test detection and monitoring on the MMT-Dashboard.

# Final Hands-On Assessment (Capstone)

*Estimated time: ~60 minutes · Text lesson + final quiz*

## In this lesson

- Plan and run a complete, small-scale penetration test end to end against a fresh, authorised lab target using the MMT-Pentester Dashboard.
- Perform reconnaissance with Nmap and select two distinct services to test from the enumerated attack surface.
- Configure and run the SSH brute-force and HTTP Slowloris scenarios, then interpret their detections on the Dashboard.
- Produce a professional final report with a prioritised top-3 list of remediation recommendations.
- Pass the final summative quiz (≥70%) to complete the course and trigger the certificate flow.

## The capstone: putting it all together

This is the final module. Up to now each part of the course has taught one piece of the workflow in isolation — the engagement lifecycle, reconnaissance, attack execution, observation, and reporting. The capstone asks you to assemble all of those pieces into a single coherent mini-engagement against a fresh lab target, working under your own steam.

The arc you will follow is the same one a real junior pentester runs on a small scoped job:

**Recon (Nmap) → pick two services → two attacks (SSH brute-force + HTTP Slowloris) → observe detections → final report → final quiz → certificate.**

Everything you need you have already seen in the earlier modules. The capstone is not an exam trap; it is a confidence-builder that proves you can sequence the steps yourself and turn raw output into a useful deliverable. Because this lab is **graded against a rubric**, work methodically and document as you go — your screenshots and notes become the evidence in your report.

## Safety, authorisation, and scope (non-negotiable)

Even though you have heard this in every module, pause on it here, because the capstone is the moment you act most independently.

- Run everything **only** against the provided, isolated lab target — never against real or third-party systems.
- The lab target plus this exercise **is** your authorisation and scope. Stay inside it.
- Minimise harm, document every action, and stop if anything behaves unexpectedly outside the lab.
- Unauthorised testing is illegal regardless of intent. The single factor that separates a legitimate pentest from a crime is **explicit permission and an agreed scope** — not the tools, not the skill, not whether anything is actually exploited.

***Safety / Ethics rule:*** *All attacks in this course are run ONLY against provided, isolated lab targets, with explicit authorisation. Never against real or third-party systems. This is for education and authorised testing only.*

## Your mission brief

You are a junior penetration tester. A client has stood up a **fresh isolated lab host** and authorised a short, scoped test. Your job:

- **Target:** a fresh lab host provided for this capstone (host/IP given in the lab card on the Dashboard or your instructor's handout). If you are self-paced, use the default capstone target shipped with your lab pack.
- **Mission:** map the attack surface, demonstrate **two distinct weaknesses against two different services**, observe how the environment detects your activity, and report on risk with prioritised fixes.
- **Constraints:** use only the three guided scenario types you already know; observe results on the Dashboard; produce one report.
- **Output:** a final report (PDF/Excel export plus your written analysis) and a completed final quiz.

The two attacks must map to two *different* discovered services. This is deliberate — it forces you to actually read and use your recon results rather than guessing.

## Recap: the platform objects you will use

The MMT-Pentester data model nests like this:

```
Project  →  Campaign  →  Scenario  →  targets { host, port, protocol }
                                   →  attacks { params }
```

You will create one **Project** for this capstone, a **Campaign** inside it, and a **Scenario** for each guided activity. Executions stream live output over WebSocket; results and IDS/monitor alerts surface in the **Dashboard** and **Reports** views.

## Workflow at a glance

Keep this four-step checklist in view while you work:

84. **Recon** — create a Project → Campaign → Scenario, add the target, and run the Nmap recon scenario.
85. **Select** — read the enumeration; choose the SSH service and the HTTP service.
86. **Attack** — run SSH brute-force, then HTTP Slowloris, against the discovered services.
87. **Observe → Report** — read detections/alerts on the Dashboard, then auto-generate and annotate the report.

## Step-by-step walkthrough

### Phase 1 — Reconnaissance (Nmap)

88. Open the Dashboard at the lab URL (dev default `http://localhost:5173`) and log in.
89. Go to **Projects** and create a new project, e.g. `Capstone — <your name>`.
90. Inside it, create a **Campaign** (e.g. `Initial Assessment`).
91. Create a **Scenario** of type **Reconnaissance with Nmap**. Add a **target** with the provided host/IP; leave protocol/port at the recon defaults to sweep the service range.
92. Run the scenario and watch the live output stream. A typical recon invocation looks like:

```
nmap -sV -sC -p- <LAB_TARGET_IP>
```

93. When it finishes, read the enumeration: note every **open port**, its **service**, and its **version**. Capture a screenshot — this is the first piece of evidence for your report.

![figure: Scenarios view — Nmap recon results showing open ports, services and versions](../assets/nmap-results.png)

### Phase 2 — Select two services

94. From the enumeration, identify the **SSH service** (typically `22/tcp`) and the **HTTP service** (typically `80/tcp` or `8080/tcp`). Record host + port + version for each.
95. These two services are your two attack targets. Note *why* each is worth testing — for example, SSH exposed to password authentication, and an HTTP server that may be susceptible to connection exhaustion.

### Phase 3 — Run two attack scenarios

96. Create a **Scenario** of type **SSH brute-force**. Set the target to the discovered SSH host/port and supply the provided username/wordlist parameters. Run it and watch the live auth attempts stream over WebSocket. Note whether any credential succeeds and how the target responds (lockout, rate limiting, etc.).

```
# conceptual form of the SSH brute-force attack
hydra -l <user> -P <wordlist> ssh://<LAB_TARGET_IP>:22
```

97. Create a **Scenario** of type **HTTP Slowloris DoS**. Set the target to the discovered HTTP host/port and run it. Observe **service degradation** — the web service slowing or refusing new connections — and how long it takes to manifest.

```
# conceptual form of the Slowloris attack
slowloris <LAB_TARGET_IP> -p 80
```

![figure: live attack output streaming over WebSocket](../assets/attack-stream.png)

### Phase 4 — Observe detections

98. Go to the **Dashboard**. For each attack, compare **expected vs actual behaviour** and review the **IDS/monitor alerts** and the **traffic seen**.

99. For SSH, look for repeated failed-authentication events or a brute-force alert. For Slowloris, look for connection-exhaustion / DoS indicators and the degraded-service signal.
100. Screenshot each detection. These captures are your primary evidence for the report.




### Phase 5 — Produce the final report

101. Open the **Reports** view and auto-generate the report (**PDF via Puppeteer**, **Excel via ExcelJS**) from your executions.
102. Add your own written analysis on top of the auto-generated output:

    - **Executive summary** — risk in plain language for non-technical decision-makers.
    - **Findings** — one per attack, with evidence (screenshots/alerts), the affected service, and a severity.
    - **Top-3 prioritised recommendations** — ordered by risk-reduction impact (see below).

103. Export and submit per the submission instructions.

## Writing the top-3 recommendations

Prioritise by *impact × likelihood*, and make each recommendation **specific and actionable**. Strong capstone recommendations usually include, in some order:

| # | Example recommendation | Why it ranks high |
|---|---|---|
| 1 | Disable SSH password authentication; require key-based auth and enforce account lockout / fail2ban | Directly kills the brute-force path; high impact, low effort |
| 2 | Put the HTTP service behind a reverse proxy, rate-limit connections, and set connection timeouts | Mitigates Slowloris connection-exhaustion at the edge |
| 3 | Reduce attack surface — close/firewall unused ports found in recon; patch outdated service versions | Shrinks what an attacker can reach in the first place |

Tailor these to *your* actual findings — do not copy the table verbatim. The grader looks for recommendations that map directly to the evidence you gathered.

## Submission guidance

Submit a single package containing:

104. The **auto-generated report** export (PDF and/or Excel) from the Reports view.

105. Your **written analysis**: executive summary, per-finding evidence, and the **top-3 prioritised recommendations**.
106. **Screenshots** evidencing the Nmap results, both attack executions, and the dashboard detections/alerts.
107. A one-line statement confirming all activity was performed only against the provided lab target.

**Self-paced:** upload the package where your lab pack directs, then take the final quiz.

**Workshop:** hand the package to your instructor and present a 2-minute summary.

**Demo:** the instructor narrates one live end-to-end run covering all five phases.

After the deliverable is submitted, complete the **Final Quiz** below. A score of **≥70% (7/10)** passes. Passing mirrors the platform's challenge/certificate flow and issues your **certificate of completion**.

### Submission checklist (use before you hand in)

- [ ] Recon scenario run; all open ports/services/versions recorded.
- [ ] Two distinct services selected (SSH + HTTP) with justification.
- [ ] SSH brute-force scenario executed; outcome and target response noted.
- [ ] HTTP Slowloris scenario executed; service degradation observed.
- [ ] Dashboard detections/alerts reviewed and screenshotted for both attacks.
- [ ] Report auto-generated and annotated with executive summary + findings.
- [ ] Exactly three prioritised recommendations, each specific and tied to evidence.
- [ ] Safety/authorisation statement included.

## How you are graded

A rubric scores five areas — **Recon, Service selection, Execution, Observation, Report quality, Recommendations.** "Excellent" means complete, evidence-backed, and clearly communicated; "Missing" means absent or unsupported. The report and recommendations carry as much weight as the technical execution. Completion of the course requires the graded deliverable **plus** a passing final-quiz score (≥70%).

| Criterion | Excellent | Adequate | Missing |
|---|---|---|---|
| **Recon (Nmap)** | All open ports, services, and versions enumerated and recorded; attack surface clearly described | Main services found and listed, with minor gaps | No scan run, or results not recorded |
| **Service selection** | SSH and HTTP services correctly identified with host/port/version | Two services chosen but rationale thin or one detail missing | Services not selected or unjustified |

| | | | |
|---|---|---|---|
| | and a clear rationale for testing each | | |
| **Execution (two attacks)** | Both attacks run against the correct discovered services; parameters correct; outcomes captured | Both run but with a misconfiguration or missing outcome note | One or both attacks not executed |
| **Observation (detections)** | Dashboard traffic, expected-vs-actual behaviour, and IDS/monitor alerts reviewed and evidenced for both attacks | Detections noted for both but evidence partial | No observation of detections / no evidence |
| **Report quality** | Clear executive summary + per-finding evidence and severity; auto-generated export annotated; well structured | Report present and readable but missing a section or weak evidence | No report or unstructured notes only |
| **Recommendations (top-3)** | Exactly three, prioritised by impact, specific, actionable, tied directly to the evidence | Three present but generic or weakly prioritised | Fewer than three, or not tied to findings |

## Summary

- The capstone is one continuous engagement: recon → select two services → two attacks → observe detections → report, against a single fresh authorised lab target.
- Authorisation and scope are the only things that make the test legitimate; stay strictly inside the provided lab.
- The two attacks must hit two *different* enumerated services, forcing you to act on your recon results.
- Detections on the Dashboard (failed-auth/brute-force alerts for SSH; connection-exhaustion/degraded-service for Slowloris) are your evidence — screenshot them.
- Your report needs an executive summary, per-finding evidence, and exactly three prioritised, evidence-tied recommendations.
- Completing the course requires the graded deliverable **and** a final-quiz score of at least 7/10, which triggers the certificate flow.

## Final Quiz

*Final assessment — 10 questions drawn across the whole course. Pass mark: 7/10. Choose the best answer.*

108. What single factor most fundamentally separates an authorised penetration test from a criminal attack?

- A) The sophistication of the tools used
- B) Explicit authorisation and a defined scope
- C) Whether any vulnerability is actually exploited
- D) The seniority of the tester

**Answer:** B — Technique can be identical; legitimacy comes entirely from written permission and an agreed scope.

109. In the engagement lifecycle, which phase does the **Reconnaissance with Nmap** scenario primarily support?

- A) Reporting
- B) Post-exploitation
- C) Reconnaissance and scanning/enumeration
- D) Pre-engagement scoping

**Answer:** C — Nmap maps the live attack surface (hosts, ports, services, versions), which is the recon/scanning phase.

110. In the MMT-Pentester data model, what is the correct nesting order?

- A) Scenario → Campaign → Project → targets
- B) Project → Campaign → Scenario → targets + attacks
- C) Campaign → Project → Scenario → attacks
- D) Project → Scenario → Campaign → targets

**Answer:** B — A Project contains Campaigns, which contain Scenarios, which hold targets and attacks.

111. A tester is given a low-privilege user account and a network diagram before starting. Which knowledge-depth model is this?

- A) Black-box
- B) Grey-box
- C) White-box
- D) Red-box

**Answer:** B — Partial inside knowledge, such as a user account, defines grey-box testing.

112. During the SSH brute-force scenario, which Dashboard signal best confirms the attack is being detected?

- A) A drop in CPU temperature

- B) Repeated failed-authentication events / a brute-force alert
- C) A new open port appearing
- D) The PDF report regenerating itself

**Answer:** B — A brute-force surfaces as many failed auth attempts, which the monitor/IDS flags.

113. What is the characteristic effect of an HTTP Slowloris attack on the target service?

- A) It encrypts the web server's files
- B) It exhausts connection slots, degrading or denying service to legitimate users
- C) It elevates the attacker to root
- D) It rewrites the site's HTML

**Answer:** B — Slowloris holds many connections open slowly, exhausting the server's connection pool.

114. Which of these is the strongest, most directly relevant remediation for the SSH brute-force finding?

- A) Change the site's colour scheme
- B) Disable password authentication and require key-based auth, with lockout/fail2ban
- C) Increase the web server's connection timeout
- D) Buy a larger hard drive

**Answer:** B — Removing password auth and adding lockout directly closes the brute-force path.

115. Why must the capstone's two attacks target two *different* discovered services?

- A) To make the report longer
- B) To force you to actually use the recon results and demonstrate distinct weaknesses
- C) Because the platform forbids reusing a service
- D) To avoid generating any alerts

**Answer:** B — Tying each attack to a different enumerated service proves you read and acted on the recon.

116. Live tool output during a scenario execution reaches the Dashboard over which channel?

- A) A nightly batch email
- B) A WebSocket stream

- C) A printed log file only
- D) An FTP upload

**Answer:** B — Executions stream their live output to the UI over WebSocket in real time.

117. What is required to complete the course and receive the certificate of completion?

- A) Only running the Nmap scan
- B) Submitting the graded report deliverable and scoring ≥70% on the final quiz
- C) Scoring 100% on the final quiz
- D) Testing a real production system

**Answer:** B — Completion needs the graded deliverable plus a passing (≥70%) quiz score, which triggers the certificate flow.

# We Value Your Feedback!

*Text lesson · Course evaluation survey*

Congratulations on completing **Hands-On Penetration Testing with the MMT-Pentester Toolbox**!

Before you go, please take a few minutes to complete our short course-evaluation survey. Your honest

feedback helps us improve the lessons, labs, and the MMT-Pentester toolbox itself for future learners.

## What the survey asks about

- The clarity and usefulness of each module.
- Whether the hands-on labs were easy to follow and worked as expected.
- The pacing and overall difficulty of the course.
- Any technical issues you encountered with the platform.
- Suggestions and features you would like to see.

## Take the survey

Please complete the evaluation form here: **[survey link to be provided by the CyberSuite Academy]**.

The survey is anonymous and takes about five minutes. Thank you for helping us make the course better —

and good luck applying your new penetration-testing skills, responsibly and only on systems you are

authorised to test.