Range42 — Semi-Technical Overview

Modular Cyber Range Platform for Real-World Readiness

NC3 / Range42 Team

September 30, 2025

Proxmox   Ansible   Orchestration   Telemetry

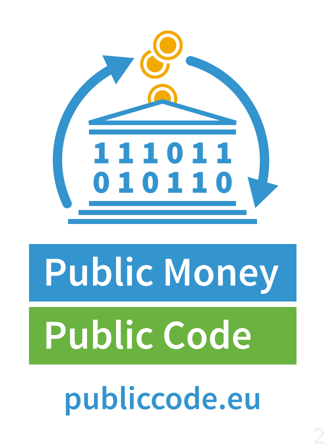
Public money, public code. Let’s go all the way, shall we?

Current Team

- Core development team of 3-5 contributors
- Mix of security researchers, DevOps engineers, and platform architects
- Collaborative model with NC3 and academic partners

Current Funding

- Public sector grant funding for cyber training infrastructure
- Open-source model: no licensing fees, transparent development
- Investment in reusable, community-owned tooling



Agenda

1. What Range42 is & why it matters
2. Competitive landscape
3. Architecture at a glance
4. Repository audit findings (13 repositories analyzed)
5. Risks, governance, and quality gates
6. 90-day roadmap & demo plan

TLP:CLEAR — Information may be shared freely without restriction.

What is Range42?

- **Modular cyber range platform** for offensive, defensive, and hybrid training.
- **Reproducible IaC**: build, deploy, document labs via Proxmox, Ansible, Docker.
- **Private APIs** for orchestration & telemetry; developer toolkits for pipelines.

ℹ Built to simulate *real-world incidents* safely, with isolation, snapshots, and telemetry.

Range42 vs. Other Cyber Ranges

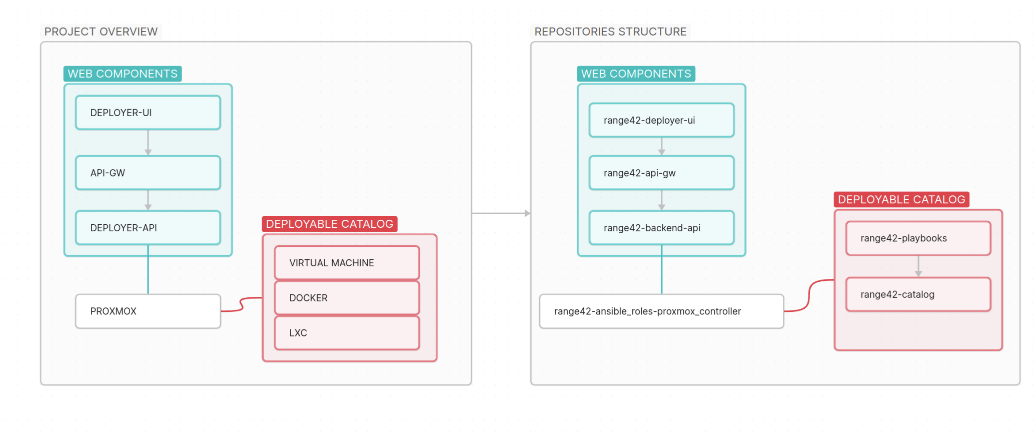
Feature	Range42	Commercial SaaS	Cloud Native	Traditional
Open Architecture	✓	×	×	×
IaC/GitOps	✓	~	✓	×
Private Deployment	✓	×	~	✓
Cost Control	✓	×	~	✓
Full Data Custody	✓	×	×	✓
API Orchestration	✓	✓	✓	×
Rapid Reset/Snapshots	✓	✓	~	~
Custom Scenarios	✓	×	~	✓

Range42's Edge: Full control, reproducibility, and cost-effectiveness without vendor lock-in.

Architecture at a Glance

- **Hypervisor layer**: Proxmox VMs/LXCs; snapshots; network segments.
- **Automation layer**: Ansible roles orchestrate lifecycle, network, firewall, images.
- **Control plane**: Backend API (routes for VM/Net/Runner); Kong gateway.
- **UX**: Deployer UI (visual design), EMP mockup (exercise mgmt), trainee access.
- **Observability**: Wazuh for logs/alerts; structured telemetry.

Architecture: Logical Components → Repository Mapping



ℹ **Left:** Logical architecture flow. **Right:** Actual GitHub repository structure.

Repository Audit Findings

Organization-Wide Audit Results

13 Repositories Analyzed (2 public, 11 private)

Critical Findings

- 9/13 repositories have LICENSE template placeholders (`<year>`, `<name of author>`)
- 13/13 repositories missing SECURITY.md vulnerability disclosure policy
- 12/13 repositories have no CI/CD pipeline
- 0 repositories have all required governance files

✓ **Good News:** Zero high-severity security findings across all code scans (bandit, pip-audit, npm audit).

Automation

range42-ansible_roles-private-devkit >_

Status: Active **Commits:** 179 **Lang:** Shell

Purpose: Helper scripts for Proxmox and Ansible operations including VM/LXC management, firewall rules, and JSON transformations.

Key Findings

- DevKit provides 100+ helper scripts following strict naming convention for Proxmox automation.
- Zero Bandit findings; actively used with 179 commits but lacks CI pipeline.
- Critical need: finalize LICENSE placeholders and add governance documentation for contributors.

range42-ansible_roles-proxmox_controller ⚙️

Status: Active **Commits:** 108 **Lang:** Ansible/YAML

Purpose: Ansible role for managing Proxmox nodes via API: VMs, LXC containers, networking, storage, firewall, and snapshots.

Key Findings

- Comprehensive Ansible role managing full Proxmox lifecycle via API with 108 commits.
- Broad functionality coverage but lacks CI pipeline for ansible-lint and idempotence tests.
- Immediate actions: finalize LICENSE, add CI, and document variables with example playbooks.

Control Plane

range42-backend-api

Status: Active **Commits:** 125 **Lang:** Python/Shell/YAML

Purpose: FastAPI backend orchestrating Proxmox deployments via Ansible, with routes for VM control, networking, and bundle execution.

Security/Quality

- FastAPI backend with 125 commits; clean security scans (bandit, pip-audit, safety).
- Comprehensive route structure for Proxmox control and bundle orchestration via Ansible.
- Production hardening needed: add CI pipeline, unit tests, finalize LICENSE, and SECURITY policy.

TLP:CLEAR — Information may be shared freely without restriction.

range42-api-definitions

Status: Stale **Commits:** 5 **Lang:** JSON

Purpose: Placeholder repository for OpenAPI/Swagger specifications defining the Range42 backend API contracts.

Decision Point

- Placeholder repository with OpenAPI specs but minimal content and only 5 commits.
- Missing LICENSE file entirely; needs decision on archive versus active development.
- If kept: seed with comprehensive API definitions and add CI for spec validation.

range42-api-gw

Status: Active **Commits:** 5 **Lang:** N/A

Purpose: Kong API Gateway configuration for authentication, ACL, and access control policies in front of backend API.

Key Findings

- Kong API Gateway repository for authentication and access control with minimal current content.
- LICENSE has template placeholders; needs Kong declarative configs or deployment manifests.
- Add documentation for Kong setup, plugin configuration, and integration with backend API.

UX Layers

range42-deployer-ui

Status: Prototype **Commits:** 28 **Lang:** Vue/TS

Purpose: VueFlow-based visual orchestrator for designing, validating, and deploying Range42 infrastructure through node-based interface.

Key Findings

- VueFlow-based UI with node-based infrastructure design; unit/e2e tests exist but no CI.
- Good UX foundation with i18n support and localStorage data management for offline capability.
- Wire CI for automated testing, stabilize API contracts with backend, and resolve npm advisory.

range42-emp-mockup

Status: Prototype **Commits:** 1 **Lang:** Vue/TS

Purpose: Exercise Management Platform scaffold with basic Vue 3 routing, TypeScript setup, and unit test foundation.

Key Findings

- Exercise Management Platform seed project with 1 commit; basic Vue 3 scaffold only.
- Missing LICENSE entirely; has 2 low npm advisories and no CI pipeline.
- Define MVP scope and architecture first, then add CI and resolve dependency vulnerabilities.

Infrastructure & Content

range42-playbooks

Status: Active **Commits:** 75 **Lang:** Ansible/YAML

Purpose: Centralized Ansible playbooks organizing scenarios and bundles for CLI or backend API orchestration of infrastructure deployments.

Key Findings

- Central playbook repository with 75 commits organizing scenarios and reusable bundles.
- Clear structure with test scripts but no CI for ansible-lint or syntax validation.
- Add CI pipeline, introduce scenario taxonomy with tags, and document compatibility matrices.

TLP:CLEAR — Information may be shared freely without restriction.

range42-catalog

Status: Active **Commits:** 91 **Lang:** Ansible/YAML

Purpose: Collection of reusable Ansible roles and Docker/Compose stacks for deploying vulnerable scenarios and infrastructure bundles.

Key Findings

- Catalog of Ansible roles and Docker stacks for deploying training scenarios with 91 commits.
- Volatile tree structure organizing CVEs and misconfigurations by technology type.
- Add CI for ansible-lint, introduce scenario taxonomy with tags, and finalize LICENSE.

Documentation & Governance

.github — Community Health Files

Status: Active **Commits:** 13 **Lang:** Markdown/YAML

Purpose: Organization-wide default community health files to standardize CODE_OF_CONDUCT, CONTRIBUTING, SECURITY, issue/PR templates, and support docs across public repositories.

Key Findings

- Centralizes community health files for all public repos.
- Add CI to lint templates and validate YAML.
- Document private-repo strategy and license requirements.

⚠ **Critical Limitation**: Defaults apply *only* to public repos; private repos need local copies. LICENSE files cannot be inherited—each repo must add its own.

TLP:CLEAR — Information may be shared freely without restriction.

range42-documentation-private-obsidian 📖

Status: Active **Commits:** 3 **Lang:** Markdown/Shell

Purpose: Private Obsidian vault containing architecture canvases, meeting notes, API drafts, and internal documentation.

Key Findings

- Private Obsidian vault with architecture canvases and internal documentation; 3 commits only.
- Contains valuable schemas and meeting notes but lacks public documentation export pipeline.
- Create automated export to public docs, version control architecture canvases, and add pruning cadence.

Repository Management

gh-repo-organizer

Status: Active **Commits:** 18 **Lang:** Bash

Purpose: Bash script for mass cloning and auditing GitHub organization repositories with comprehensive standards compliance checking.

Key Features

- Production-ready tool for mass repository cloning and standards auditing across GitHub organizations.
- Detects LICENSE template placeholders, missing documentation, and CI/CD configurations automatically.
- Zero security findings; needs CI pipeline and contributor documentation to reach maturity.

Repository Standards Compliance

📋 Automated Sanity Checks Performed

- **License Validation**: LICENSE files with template placeholder detection.
- **Documentation**: README, CHANGELOG, CONTRIBUTING, SECURITY policies.
- **Git Configuration**: .gitignore, .editorconfig standards.
- **CI/CD Detection**: GitHub Actions, GitLab CI, Jenkins, Travis, etc.
- **Templates**: Issue/PR templates, CODE OF CONDUCT.

ℹ Addresses governance gaps: *license placeholders, missing SECURITY policies, CI/CD standardization.*

Integration with Range42 Roadmap

🚀 Supports 90-Day Goals

- **Week 1-3**: Identify repos needing CI foundations via automated audit.
- **Week 2-5**: Batch detection of missing LICENSE and SECURITY policies.
- **Week 4-8**: Track backend-UI contract compliance across repos.
- **Ongoing**: Monthly audits for standards drift prevention.

💡 **Measurable Progress**: Generate compliance dashboards and track improvement over time.

Risks & Governance

Key Risks & Gaps ⚠

Governance & Quality

- LICENSE placeholders in 9/13 repos: Replace `<year>` and `<name of author>` with actual values.
- No SECURITY.md in any repo: Define vulnerability disclosure and triage process.
- No CI/CD in 12/13 repos: Enable lint, tests, security scans, release checks.
- Missing contributor docs: Add CONTRIBUTING.md, .editorconfig standards.

Technical Debt

- API contract formalization needed between backend and UI.
- Scenario taxonomy and compatibility matrix documentation.
- Stale repositories need archive-or-activate decisions.

Roadmap

90-Day Roadmap 🛣️

Week 1–3: Governance Foundations

- Fix all LICENSE template placeholders across 9 repositories.
- Add SECURITY.md with vulnerability disclosure process to all repos.
- Add CONTRIBUTING.md and .editorconfig to active repositories.

Week 2–5: CI/CD Foundations

- Enable GitHub Actions CI for ansible-lint, shellcheck, pytest, npm audit.
- Add automated security scans (bandit, pip-audit, safety, npm audit).
- Implement pre-commit hooks for code quality.

Week 4–8: API & Contract Stability

- Backend–UI contract tests; stabilize API definitions.
- Document all Ansible role variables with examples.

90-Day Roadmap (continued) 🛣

Week 6–12: Content & Architecture

- Inventory taxonomy and scenario coverage matrix.
- Decision on api-definitions: archive or activate with OpenAPI specs.
- Decision on emp-mockup: define MVP or archive.
- Export pipeline from private Obsidian docs to public documentation.

Continuous: Monitoring & Metrics

- Monthly automated compliance audits using gh-repo-organizer.
- Track metrics: test coverage, vulnerability response time, CI green rate.
- Generate quarterly compliance reports for stakeholders.

Demo Plan ▶

Demo Scenario: Deploy vulnerable lab environment end-to-end

Steps

1. Deploy demo lab via playbooks (create-vms-admin, create-vms-vuln, create-vms-student).
2. Control VMs via backend API routes (start, stop, pause, resume).
3. Visualize infrastructure in Deployer UI with node-based editor.
4. Create snapshots and demonstrate reset-to-clean-state capability.
5. Show baseline telemetry collection in Wazuh dashboard.

Success Criteria

- Green CI badges on all participating repositories.
- Idempotent Ansible runs with zero drift.
- Complete infrastructure lifecycle: deploy → observe → reset.

Takeaways ✓

- Range42 = modular, reproducible cyber range for realistic training.
- Strong security baseline: Zero high-severity findings across all code.
- Governance is the multiplier: LICENSE, SECURITY, CI/CD, and contributor docs.
- Clear 90-day path to production-grade readiness with measurable milestones.

💡 **Next Steps**: Fix LICENSE placeholders, add SECURITY.md, enable CI across all active repos.