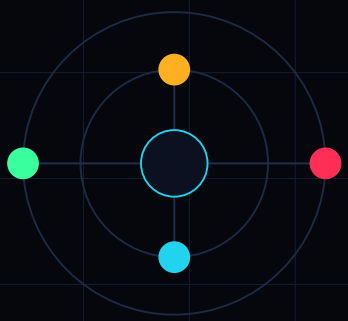
LIVE-TELEMETRY PURPLE RANGE

PROJECT AETHER

User & Setup Guide

Complete installation, configuration and operating instructions for the range server and the operator console.

TO START	`START-AETHER.bat / ./start-aether.sh`
CONSOLE	`http://localhost:4100`
STACK	`React 18 + Vite + Tailwind / Node + Express + Socket.io`
AUDIENCE	`Range operators, instructors, administrators`



SECTION 01

What AETHER Is

Project AETHER is an asymmetric, real-time cyber warfare simulator. A Blue Team defends a four-subnet corporate network while a Red adversary attempts to reach the Core Database. Unlike a scripted tabletop, every defensive control you buy changes the detection mathematics live, and the range reports the result as measured **MTTD** (Mean Time to Detect) and **MTTR** (Mean Time to Remediate).

The platform is two processes: an authoritative **range server** that owns all game state and orchestrates the AI adversary, and a browser-based **operator console**. They must both be running. This guide covers each in turn.

Architecture at a glance

COMPONENT	TECHNOLOGY	PORT	RESPONSIBILITY
Range Server	`Node + Express Socket.io`	`4100`	Authoritative tick engine, detection maths, fog of war, LLM adversary orchestration, config persistence.
Operator Console	`React 18 + Vite Tailwind CSS`	`5273`	Topology map, Blue SIEM panel, Red payload console, telemetry widget, admin panel.

Authoritative server model

The browser never decides outcomes. Clients send action *intents* only; the server validates every one against real state. This is also how fog of war is enforced - a Blue client physically cannot read undetected compromises out of the websocket frame.

The range topology

NODE	ZONE	ROLE IN THE SIMULATION
DMZ Web Server	`DMZ`	Internet-facing. Always visible to the adversary and a valid phishing entry point.
Corporate Subnet	`CORP`	User estate. Valid entry point; the usual first pivot.
Engineering Subnet	`ENG`	Interior. Reachable only by lateral movement.
Finance Subnet	`FIN`	Interior, off the Corporate estate.
Build Server	`ENG`	Interior, off Engineering. A common staging point.
Jump Host	`MGMT`	**Chokepoint.** The only route to the Core Database. Severing it contains the intrusion outright — at a real uptime cost.
Core Database	`CORE`	The crown jewel. Compromise here drains integrity three times faster and is the Red win condition.

The topology is randomised each match

Five edges are fixed, but two of the four possible approaches to the Jump Host are drawn at random per engagement, and a lateral Corporate-Engineering link appears about half the time. The shape of the intrusion therefore changes between matches — and because the Core sits behind a single chokepoint, micro-segmentation has a situation where it is decisively the right purchase.

SECTION 02

Prerequisites

REQUIREMENT	MINIMUM	NOTES
Node.js	`18.x or newer`	Node 20+ recommended. The server uses native fetch and ES modules.
npm	`9.x or newer`	Ships with Node.
Browser	`Any modern`	Chrome, Edge, or Firefox. WebSocket support required.
Disk	`~250 MB`	Almost entirely node_modules across the two packages.
LLM API key	`Optional`	Only affects Single Player. Without one the range runs a built-in heuristic adversary.

Verify your toolchain

```
node --version # expect v18.x or higher
npm --version # expect 9.x or higher
```

No internet? No problem.

AETHER has no runtime dependency on any external service. Once npm install has completed, the entire range - all three modes - runs fully offline. Only the optional LLM adversary reaches the internet, and it degrades gracefully when it cannot.

Project layout

```
Project-Aether/
START-AETHER.bat <-- double-click this on Windows
start-aether.sh <-- run this on macOS / Linux
launch.mjs One-command launcher (install, build, serve)
server/ Range server (Node)
index.js Express + Socket.io entry point
engine/constants.js ALL tunable balance numbers
engine/game.js Tick loop, detection, MTTD/MTTR, fog of war
engine/topology.js Node and edge definitions
llm/providers.js Multi-provider transport + timeout wrapper
llm/threatAgent.js Adversary turn, validation, heuristic fallback
llm/prompts.js Red Team Threat Actor system prompt
config/ Persisted LLM configuration (created at runtime)
client/ Operator console (React + Vite)
src/App.jsx Screen routing
src/components/ Topology, Blue panel, Red panel, admin, telemetry
docs/ This guide
```

SECTION 03

Quick Start

There is one step. You do not need to install anything by hand, edit any configuration, or open a terminal.

YOUR COMPUTER	WHAT TO DO
Windows	Double-click **START-AETHER.bat** in the Project-Aether folder.
macOS / Linux	Run `./start-aether.sh` from the Project-Aether folder.

The launcher does everything else: installs components, builds the console, finds a free port, starts the range, and opens your browser. When it is ready you will see:

```
AETHER IS RUNNING

On this computer : http://localhost:4100
For other players: http://192.168.0.50:4100
(same Wi-Fi / network only)

Press Ctrl+C to stop.
```

First run takes about a minute while it installs and builds. Every run after that starts in a couple of seconds.

> **The only prerequisite is Node.js**
> If Node.js is missing, the launcher stops and tells you exactly where to get it: go to nodejs.org, download the LTS installer, accept all the defaults, then start AETHER again. Nothing else needs installing.

Playing with someone else

The second URL - *"For other players"* - is the one you share. Anyone on the same Wi-Fi or office network opens it in a browser and they are in. They install nothing and configure nothing.

This works because, once built, the console is served by the range server itself: the interface, the API and the live connection all share **one address on one port**. There is no separate client process to start and no proxy to configure.

If something is already using the port

The launcher detects this and moves to the next free port automatically, reporting what it chose. No action needed.

```
! Port 4100 was busy - using 4101 instead.

On this computer : http://localhost:4101
```

Stopping it

Press **Ctrl+C** in the launcher window, or just close it. Matches are held in memory only, so nothing needs saving.

SECTION 04

How the Connection Works

You do not need this section to play. It is here for administrators deploying AETHER on a shared machine or troubleshooting a network.

Single-port mode (the default)

The launcher builds the operator console and the range server then serves it from its own process. The interface, the REST API and the live WebSocket all share one origin:

```
Browser ■■ http://:4100/ the console
■■ http://:4100/api/... admin configuration
■■ ws://:4100/socket.io live game traffic
```

Because everything is one origin there is no proxy, no CORS configuration, and no second process. The server binds on all network interfaces, which is what allows another machine to join by opening the same address.

What travels over the connection

DIRECTION	TRAFFIC
Console to server	Action *intents* only - create/join a range, and Blue or Red actions. The server validates every one against authoritative state. Nothing is trusted from the browser.
Server to console	One state frame per 3-second tick, plus one after every action.

Each player receives a different frame

The server sends every connected console its own filtered view rather than broadcasting one shared frame. A Blue player's frame has undetected compromises stripped out before transmission, so fog of war is enforced by the server and not by the interface. This is why a player cannot inspect network traffic to see where the adversary is.

Checking the range is healthy

```
curl http://localhost:4100/api/health
# {"ok":true,"service":"aether","tickMs":3000}
```

Developer mode

For working on the code, `node launch.mjs --dev` runs the server with watch-restart and the console on a separate Vite port with hot reload. In that mode Vite proxies the API and socket traffic through to the server. Use `--rebuild` to refresh the production build after code changes.

The manual two-process route still works if preferred: `npm start` in *server/* and `npm run dev` in *client/*, then open port 5273. Note that cross-machine play in that mode additionally needs `npm run dev -- --host`, which the launcher handles for you.

Persistent state

Matches live in memory only and are discarded when the server stops. The single persistent artefact is your saved LLM configuration in `server/config/llm-config.json`.

SECTION 05

Configuring the AI Threat Engine

Reach the admin panel from the main menu: **Threat Engine Configuration**. The engine drives **Single Player mode only**; both PVP modes use a human Red operator and ignore this configuration entirely.

Supported providers

PROVIDER	DEFAULT BASE URL	DEFAULT MODEL	KEY
Anthropic Claude	`https://api.anthropic.com`	`claude-opus-4-8`	Required
OpenAI GPT	`https://api.openai.com`	`gpt-4o`	Required
Google Gemini	`https://generativelanguage.googleapis.com`	`gemini-2.0-flash`	Required
Local Endpoint	`http://localhost:11434`	`llama3.1`	Optional

The Local Endpoint option targets Ollama and vLLM, both of which are called over the OpenAI-compatible `/v1/chat/completions` route. Leave the key blank unless your gateway enforces one.

Configuration fields

FIELD	PURPOSE
API Key	Stored server-side and never returned to the browser in full. Leave blank when re-saving to keep the existing key.
Target Model String	Overrides the provider default. Leave blank to use the default shown above.
Base Endpoint URL	Point at a proxy, gateway or self-hosted server. Leave blank for the provider default.
Max Tokens	Response ceiling. The adversary returns a small JSON object; 1024 is ample.
Temperature	0.0 gives a repeatable, predictable adversary - best for teaching. 0.7 to 1.0 gives varied, less predictable tradecraft.

The connection test loop

Test Connection sends a minimal probe asking the model to echo `{"status": "pong"}`, under a hard 5000 ms timeout. You can test a key typed into the form before saving it.

RESULT	MEANING	ACTION
Green + latency	Endpoint reachable and the handshake echoed correctly.	Save the configuration.
Green, unexpected payload	Endpoint reachable but the model did not echo the key. Common on small local models.	Usually still playable; the adversary tolerates loose JSON.
HTTP 401 / 403	Key rejected or missing.	Check the key and that it matches the selected provider.
HTTP 404	Model string or base URL wrong.	Verify the model name against the provider default.
HTTP 429	Rate limited or out of quota.	Wait, or lower the tick pressure by using a faster model.
ETIMEDOUT	No response inside 5000 ms.	Check network, proxy and firewall. Local models may simply be slow to load.

RESULT	MEANING	ACTION
fetch failed	Host unreachable.	For local endpoints, confirm the service is actually running on that port.

You do not need an API key to run the range

With no key configured - or on any transport error, timeout, malformed response, or out-of-budget move - AETHER falls back to a deterministic heuristic strategist that phishes, pivots toward the Core, switches to Living-off-the-Land when your sensors are hot, and detonates ransomware once it owns the crown jewel. Every mode is fully playable offline. Configure a key when you want less predictable adversary behaviour.

Provisioning by environment variable

For classroom or lab images, set the key before starting the server and skip the UI:

```
# PowerShell
$env:ANTHROPIC_API_KEY = 'sk-ant-...'; npm start

# bash
ANTHROPIC_API_KEY=sk-ant-... npm start
```

SECTION 06

Operating Modes

Single Player - choose your seat

Single Player asks which side you want. **The machine plays the other one.**

1. From the main menu, click **Single Player**.
2. Choose **BLUE TEAM** (defend) or **RED TEAM** (attack).
3. If you chose Red, pick how mature the AI defender should be.
4. Click **Begin engagement**.

YOUR SEAT	YOU DO	THE MACHINE
BLUE TEAM	Hold the SIEM. Triage alerts, evict intrusions, protect the Core Database.	Plays the LLM threat actor, adapting its tradecraft to your posture.
RED TEAM	Establish a foothold, pivot to the Core Database and collapse its integrity.	Plays an automated defender that triages, remediates and hunts.

Facing the AI Blue Team

When you take the Red seat, the defender's maturity is yours to set. Win rates below are for the adversary (you), measured over 80 runs each:

MATURITY	BEHAVIOUR	RED WIN
RECRUIT	Junior analyst. Responds to alerts but rarely triages first, buys no filter, never hunts. Acts only every other tick.	**75%**
OPERATOR	Competent SOC. Buys detection, triages before responding, evicts promptly.	**60%**
ELITE	Mature detection-and-response. Layered controls, deception, chokepoint containment and proactive hunting.	**13%**

The AI defender narrates its reasoning in the log feed, prefixed **[AI DEFENDER]** - so as the adversary you can watch exactly which of your moves it noticed and what it decided to do about it.

> **Red plays under fog of war too**
>
> As the adversary you only see hosts you have discovered. Everything else shows as an **Unmapped Host** and cannot be targeted until a stealth scan reveals it. You can see the defender's posture - which controls they have bought - because adapting to it is the point, but not their alert queue, their AP or their triage findings.

Threat engine telemetry states

STATE	INDICATOR	MEANING
IDLE	Soft green pulse	Monitoring subnets. No transaction in flight.
THINKING	Cascading amber lines	State has been sent to the model; awaiting its exploit decision.
EXECUTING	Red expanding shockwave	A move is landing. Round-trip latency is displayed when a live model was used.

Below the widget, the console prints the adversary's own *strategic assessment* - its rationale in its own words. This is the teaching payload of the mode: you can watch an attacker explain that it is going quiet *because* you tuned your SIEM.

Campaign — three linked operations

Three escalating engagements against a fixed sequence of adversaries, with estate damage carried forward between them. Losing a stage ends the campaign.

STAGE	OPERATION	ADVERSARY	PURPOSE
1	**QUIET LEDGER**	`IRON WRECKER`	A commodity ransomware crew. Loud, fast and catchable — learn the console.
2	**PALE SIGNAL**	`PALE OPPORTUNIST`	A capable adaptive operator that changes tradecraft against your posture.
3	**GLASS HARRIER**	`GHOST HARRIER`	State-grade espionage actor. Patient and quiet. Detection is everything.

Between stages the estate recovers 25% integrity but carries the rest of its damage forward. Finishing stage 1 at 40% integrity means starting stage 2 at 65%, not 100% — so early sloppiness compounds across the campaign.

Local Asymmetric PVP (split-screen)

Two players at one machine. The Blue SIEM and the Red payload console render side by side; the AI telemetry widget is hidden because there is no AI. Best on a wide or dual monitor.

1. Click **Local Asymmetric PVP**.
2. Player 1 takes the left panel (Blue), Player 2 the right (Red).
3. Both share one topology map. Red sees OWNED markers on captured nodes.

> **Fog of war does not apply in split-screen**
> Because both players share one screen, this mode shows full ground truth to both sides. It is an honest-information tactical exercise. For a genuine hidden-information game, use Remote PVP.

Remote PVP (two machines)

A human Red operator replaces the LLM entirely over a live WebSocket. Fog of war is fully enforced: the Blue player cannot see undetected compromises.

1. **Host:** click **Remote PVP**. The match opens in a lobby and displays a six-character range code in the command bar.
2. Share that code with the second player.
3. **Joiner:** from the main menu, type the code into the **RANGE CODE** box and click **Join as Red**.
4. The engagement goes live the moment the Red seat is filled and the tick loop starts.

Both machines must reach the same server. Across a LAN, the joining player browses to the host's address rather than localhost, and the Vite dev server must be started with `npm run dev -- --host` to bind beyond loopback.

SECTION 07

Blue Team Operations

You spend **Action Points**, accruing 4 per tick from a starting 20. Every control below changes the detection mathematics in a specific, visible way.

Action reference

ACTION	AP	EFFECT
Enable Proactive Logging	10	Purges the noise floor and **doubles sensor visibility**, roughly halving time-to-detect. But it also raises raw alert volume, so more false positives reach your queue. Still the strongest single purchase - just not a free one.
Tune SIEM Filter	14	Adds correlation signal, highlights compromise paths, and **cuts the false-positive rate by about 60%** - the direct answer to alert fatigue. Weak against Living-off-the-Land by design.
Deploy Honeypot	20	Injects a decoy node. Any attack against it is detected instantly, costs the adversary 15 AP and badly rattles them. Effectiveness depends on *who* you are fighting. Limit of two.
Micro-segment Subnet	30	Severs one edge. Decisive on the Jump Host chokepoint, marginal elsewhere. Costs 3% uptime immediately plus ongoing drain - containment is not free.
Investigate Node	6	**Triage.** Confirms a real intrusion or clears a benign alert. The cheapest way to find out what you are actually looking at.
Threat Hunt	16	**Proactive sweep** of a node and its neighbours. Surfaces hidden intrusions regardless of accrued signal, closes benign alerts, eradicates persistence and disrupts staged re-entry.
Remediate Node	16	Evicts the adversary, restores the node and recovers 3% integrity. Costs +10 AP against entrenched persistence - and is largely wasted on a false positive.

Using the panel

- **Proactive controls** are one-shot purchases. Once active they show ACTIVE and cannot be re-bought.
- **Micro-segmentation** asks for two endpoints: click one node chip, then a second, and the link between them is severed.
- **Investigate and Remediate** act on the currently selected node. Select it by clicking either the topology map or a node chip in the Response section.
- Remediate stays disabled until an event on that node is confirmed. If you believe something is there but cannot act, Investigate is the answer.

> **Recommended opening**
> Buy Proactive Logging on the first or second tick. At 10 AP it is the cheapest control in the game and it doubles visibility permanently - every later detection benefits. Players who bank AP for micro-segmentation instead almost always post worse MTTD and lose more integrity in the process.

Reading the topology map

COLOUR	STATE	MEANING
GREEN	**Secure**	No detected adversary presence.
AMBER	**Anomalous**	Unverified activity - signal is building but the event is not yet confirmed. Investigate.
RED	**Compromised**	Confirmed compromise. Remediation is now authorised.

A severed link is drawn dashed and marked SEVERED. The Core Database carries a violet ring marking it as the crown jewel. Honeypots display as DECOY.

SECTION 08

Alert Fatigue and Triage

Not every alert is an intrusion. The estate generates **benign anomalies** - backup jobs, patch orchestration, authorised scanners - that raise alerts indistinguishable from a real compromise until you triage them. This is the single largest difference between a detection *timer* and actual SOC work.

The alert queue

Every raised alert appears in the Blue panel queue with one of four states:

STATE	MEANING	WHAT TO DO
UNTRIAGED	An alert has crossed the correlation threshold. You do not yet know whether it is real.	Investigate (6 AP) to find out, or gamble on remediating blind (16 AP).
HOSTILE	Triage confirmed a genuine adversary presence.	Remediate. The MTTR clock is running.
ENTRENCHED	A confirmed intrusion that has established persistence.	Remediate at +10 AP, or threat hunt first to eradicate cleanly.
BENIGN	Triage cleared it. There was never an adversary.	Nothing. Responding would waste AP.

The economics of triage

This is the core decision loop of the Blue role:

- **Investigate first (6 AP).** You learn what the alert is. If benign it closes for free; if real, the forensic signal is boosted and you respond knowing it matters.
- **Remediate blind (16 AP).** Faster if you are right. If it was benign you abort partway and burn about half the cost for nothing - and the adversary got those ticks for free elsewhere.

> **Visibility without triage capacity is a trap**
> Proactive logging raises alert volume by roughly 60% alongside its detection benefit. A player who buys logging and then responds to every alert without triaging performs measurably WORSE than one who triages - 20% win rate versus 52%. Tuning the SIEM filter is the direct counter: it cuts false positives by about 60%.

Triage precision

The metrics bar tracks **Triage Precision** - of all response actions you took, the share that hit a real threat. It is scored in the after-action report. Precision below 50% means you are spending most of your response capacity on noise, which is exactly the failure mode alert fatigue describes.

SECTION 09

Dwell Time and Persistence

Breach damage is not linear. An adversary left alone for ten minutes is far more than twice as costly as one caught in five, because they spend that time entrenching. AETHER models this directly.

Entrenchment

Any intrusion that remains **un-evicted for 45 seconds** establishes persistence. Note that the clock runs to *eviction*, not to detection - knowing an adversary is present does not stop them digging in. Only removing them does. This is what makes both MTTD and MTTR matter super-linearly rather than additively.

CONSEQUENCE	EFFECT
Heavier damage	Integrity drain from that node increases by 70%.
Costlier eviction	Remediation costs +10 AP to dig out the persistence mechanisms.
Possible re-entry	A 45% chance that eviction was not clean. The adversary regains the host three ticks later **without a new intrusion** - and the replacement foothold starts almost silent.

Threat hunting is the counter

A hunt sweeps the target node and every immediate neighbour. Within that radius it:

- Injects heavy forensic signal into **every real intrusion**, surfacing adversaries that have not yet tripped a correlation threshold. This is the only way to find a genuinely quiet actor.
- Closes benign alerts for free, clearing your queue without spending triage AP on each.
- **Eradicates persistence outright**, so the subsequent eviction is clean and cannot re-enter.
- Disrupts any re-entry already staged in the swept area.

> **Hunting is what separates good play from expert play**
>
> Measured over 100 runs per policy, a Blue team that triages and remediates wins 43%. Adding proactive hunting and deception raises that to 88% - and the gap comes almost entirely from eradicating persistence before it forces a costly, unclean eviction. Against the patient GHOST HARRIER persona the difference is larger still.

SECTION 10

Red Team Operations

Available to human operators in both PVP modes. Red accrues 3 AP per tick from a starting 24. Every action carries a **noise** rating that drives how fast Blue detects it.

PAYLOAD	AP	NOISE	EFFECT
Phishing Campaign	12	1.0	Establishes an initial foothold. Valid only against internet-facing or user subnets - this is how you get in.
Stealth Network Scan	8	0.3	Maps a node and its adjacencies without compromising it. Very quiet; use it to bank AP while learning the terrain.
LotL Execution	20	0.35	Signed native tooling. Compromises like a pivot but receives only 35% of the SIEM filter bonus - the stealth play.
Lateral Pivot	18	0.8	Breaches a subnet adjacent to a node you already own. Faster and cheaper than LotL, but louder.
Ransomware Blitz	35	2.5	Heavy integrity damage against a node you own, doubled on the Core. Extremely loud - the endgame move.

Executing a payload

1. Select a payload from the inventory.
2. Choose a **target node**.
3. For anything except phishing and scanning, choose a **pivot node** - one you already own.
4. Click **Execute Payload**.

The server rejects moves that break the rules: pivoting from a node you do not own, crossing a severed link, targeting a non-adjacent node, or overspending AP. The reason is returned to you as an on-screen message.

The detection model - where MTTD comes from

Every compromise opens an **event** carrying a signal accumulator. On each 3-second tick the server advances it:

```
signal += action.noise x visibility

visibility = 1.0
+ 1.0 if proactive logging enabled
+ 0.8 if SIEM filter tuned (x 0.35 against LotL)

event is CONFIRMED when signal >= 6.0
```

MTTD is recorded as the interval from compromise to confirmation. MTTR is recorded from confirmation to successful remediation. Because proactive logging doubles the multiplier on a 1.0 base, it roughly **halves** the number of ticks required to cross the threshold - which is the entire thesis the range exists to demonstrate.

> **Why the SIEM filter alone is not enough**
>
> Living-off-the-Land receives only 35% of the filter bonus, because signature-based correlation is weak against signed native tooling. A Blue player who buys only the filter will watch a competent adversary go quiet and walk to the Core. Proactive logging closes that gap; the filter does not. This asymmetry is deliberate and is the most instructive single number in the simulation.

SECTION 11

Win Conditions and Scoring

OUTCOME	TRIGGER
RED VICTORY	Core Integrity reaches 0, **or** the Core Database is compromised while integrity is below 40%.
BLUE VICTORY	Blue holds the network for the full five-minute engagement window.

Live metrics

The bar above the topology map tracks Time Remaining, Business Uptime, Core Integrity, Avg MTTD, Avg MTTR and Detection Rate. MTTD and MTTR colour-code against the range benchmarks - green at or under 30s detect and 45s remediate, amber to double that, red beyond.

The after-action report

On completion AETHER produces a scorecard: average MTTD and MTTR, final uptime and integrity, undetected events, nodes remediated, honeypot trips, detection rate, and a weighted **Defensive Posture Score** out of 800.

COMPONENT	WEIGHT	BASIS
Core Integrity	`300`	Final integrity as a percentage.
Business Uptime	`200`	Final uptime - penalises over-aggressive micro-segmentation.
MTTD	`150`	Scored against the 30s benchmark. Full marks at or under; zero at double.
MTTR	`150`	Scored against the 45s benchmark, same curve.
Triage Precision	`100`	Share of response actions that hit a real threat rather than a false positive.

Benchmark grading

The debrief grades your measured MTTD and MTTR into bands rather than reporting a bare number, so the result is interpretable without prior context:

GRADE	MTTD	MTTR	INTERPRETATION
ELITE	`<= 15s`	`<= 20s`	Detection inside the adversary breakout window; eviction almost immediate on confirmation.
STRONG	`<= 30s`	`<= 45s`	Comfortably ahead of typical enterprise dwell time.
ADEQUATE	`<= 60s`	`<= 90s`	Detection occurred, but the adversary had time to establish.
AT RISK	`> 60s`	`> 90s`	Dwell long enough for entrenchment. This is where most real breaches sit.

Because uptime is weighted, containment is deliberately not free. A player who severs every link will hold the Core and still score poorly - which is the intended lesson about proportionate response.

Measured outcomes

Verified over 100 headless runs per policy against the real engine (`node tools/balance.mjs`). Outcome genuinely depends on how you play - neither side is scripted to win.

BLUE POLICY	WHAT IT DOES	BLUE WIN	MTTD
Passive	Nothing at all.	**0%**	21s
Reactive only	Responds to alerts; buys no proactive controls.	**1%**	25s
Proactive logging	Buys visibility, responds to every alert without triaging.	**44%**	13s
Layered defence	Logging + filter, triages before responding.	**43%**	13s
Expert play	Adds proactive threat hunting and deception.	**88%**	10s

Two clear results and one honest caveat. Doing nothing, or responding without any detection investment, loses almost every time. And **proactive threat hunting is the decisive skill** - it roughly doubles the win rate over competent reactive play.

Triage is currently a wash on average

At the present false-positive rate, triaging every alert before responding (43%) performs about the same as responding to everything blind (44%). Triage saves AP on benign alerts and makes eviction cheaper, but costs an extra tick of delay - and under time pressure those roughly cancel. It pays off most against the patient GHOST HARRIER and when the alert queue is busy. Raise FALSE_POSITIVES.baseRatePerTick if you want triage discipline to matter more.

SECTION 12

Dynamic Adversaries

One of three **personas** is drawn at the start of every match, so the same opening never produces the same engagement twice. The persona is named in the Attribution strip above the topology map.

PERSONA	DOCTRINE	BLUE WIN
GHOST HARRIER	Patient espionage actor. Stealth over speed; will sit quiet rather than risk disclosure. Hardest to detect - and because it enumerates carefully, the most likely to walk into a honeypot.	**46%**
PALE OPPORTUNIST	Adaptive operator. Reads your posture each tick and swings between stealth and force.	**39%**
IRON WRECKER	Smash-and-grab crew. Trades stealth for tempo and reaches for ransomware early - loud, and therefore catchable. Rushes past decoys without taking the bait.	**58%**

Deception is doctrine-dependent: honeypot susceptibility runs from 75% for the patient GHOST HARRIER down to 20% for IRON WRECKER. Buying a decoy against the wrong adversary is largely wasted AP.

How the adversary reacts to you

- **Heat** rises as you buy detection controls, pushing the adversary away from loud pivots toward Living-off-the-Land tradecraft.
- **Caution** rises when you evict it or when it trips a honeypot. A burned actor goes quiet, banks AP and rebuilds - and how badly it rattles depends on the persona.
- **Detection jitter** of plus or minus 35% means events do not confirm on an identical tick every run, so repeat plays stay decisions rather than memorisation.

Campaign phase and threat intel

The Attribution strip tracks the intrusion through **RECON, FOOTHOLD, EXPANSION, OBJECTIVE** and **IMPACT**. Alongside it, INTEL beats fire in the log feed off real state - whether you left sensors at baseline, whether containment is holding, whether you are recovering ground. Two matches read as different engagements.

> **The declassified decision log**
> The adversary's live reasoning is deliberately hidden during play - showing it would defeat fog of war. The complete decision log, including its rationale in its own words, is released in the after-action debrief. Read it back against your own timeline to see exactly which of your decisions changed its tradecraft.

Suggested teaching exercise

Run the same mode twice and compare the two after-action reports:

- **Round 1 - reactive.** Buy no proactive controls. Respond only once nodes turn red. Record the resulting MTTD.
- **Round 2 - proactive.** Buy Proactive Logging on tick one, then play identically.
- The MTTD delta between the two rounds is the argument for proactive security investment, expressed as a number the audience watched being generated rather than one asserted on a slide.

SECTION 13

Tuning and Troubleshooting

Every balance number lives in one file - `server/engine/constants.js`. No engine logic needs editing to re-balance the range. Restart the server to apply changes.

CONSTANT GROUP	CONTROLS
`TICK_MS`	Tick cadence. Default 3000 ms. Lower for a faster, more frantic exercise.
`MATCH_DURATION_S`	Engagement length. Default 300 s.
`AP`	Starting and per-tick Action Points for both sides, plus ceilings.
`BLUE_ACTIONS`	Cost of every defensive control.
`RED_ACTIONS`	Cost, noise and damage of every payload.
`DETECTION`	Threshold, visibility bonuses, filter-evasion factor, honeypot penalty.
`HEALTH`	Integrity and uptime drain rates, micro-segmentation penalties.
`SCORING`	Score weights and the MTTD/MTTR benchmarks.
`BENCHMARKS`	Grade bands shown in the after-action debrief.
`PERSONAS`	Adversary doctrines - stealth bias, patience, recon appetite, honeypot susceptibility.
`FALSE_POSITIVES`	Benign alert rate, how logging and filtering scale it, and the wasted-response refund.
`PERSISTENCE`	Dwell threshold, damage multiplier, eviction surcharge and re-entry chance.
`HUNT`	Threat hunt sweep radius and signal boost.
`CAMPAIGN`	Stage sequence, adversary per stage, and inter-stage recovery.

Verifying a balance change

After editing constants, re-check the outcome spread:

```
cd server
node tools/balance.mjs 100
```

The harness plays the real engine headlessly across five Blue skill levels and reports win rate, match length, MTTD and score, plus a per-persona breakdown. A healthy range shows a **gradient**. Two failure modes to watch for:

- **No gradient at the bottom** - even passive Blue wins. Blue's economy is too generous; raise remediation cost or lower AP accrual.
- **A cliff** - one purchase takes the win rate to 100%. The match is being decided by a single decision rather than by play.

The key ratio is Blue's eviction rate (remediate cost divided by AP per tick) against Red's compromise rate (pivot cost divided by AP per tick). Blue should win that race slightly, because Blue can only spend on *confirmed* events - which is what makes detection speed, rather than raw economy, the deciding factor.

Troubleshooting

SYMPTOM	CAUSE	RESOLUTION
Launcher says Node.js is not installed	Node.js missing or not on PATH.	Install the LTS build from nodejs.org, accept the defaults, then start AETHER again.
Another player cannot reach the range	Wrong URL, or a different network.	They must use the 'For other players' address, not localhost, and be on the same network. Check the host firewall allows the port.
Menu loads but mode cards do nothing	The server process stopped.	Check the launcher window for errors and start it again.
Port already in use	Another process holds the port.	The launcher moves to the next free port automatically and tells you which. Use the address it prints.
Code changes are not showing	Running the built console, not the source.	Use node launch.mjs --rebuild, or --dev for hot reload while developing.
Adversary never acts	No key and heuristic idle, or insufficient AP.	Normal early on - it banks AP. Check the log feed for an ENGINE fallback line.
Adversary is predictable	Running the heuristic fallback, or temperature 0.	Configure a working API key and raise temperature toward 0.7 to 1.0.
Cannot click Remediate	The event is not yet confirmed.	Investigate the node, or buy Proactive Logging to raise visibility.
Red seat is occupied	Someone already joined that range.	Host a fresh range to get a new code.
Nothing renders on join	Wrong or expired range code.	Codes are memory-only and die with the server or when all players leave. Re-host.

Range state is intentionally ephemeral

Matches live in server memory. Restarting the server, or every player leaving a range, destroys it. Capture the after-action report on screen before closing if you need the numbers for a debrief. The only persistent artefact is your LLM configuration.