MODULE 01 FIELD EXERCISE

Prove Your Node: Surface, Ingress, Break, and Repair

TASK: Audit your Module 01 node, operate it from off-site, repair an injected fault, and recover it after a reboot.

CONDITION: Given your operational Module 01 node (Synapse, AdGuard Home, and the Headscale tailnet), your GL.iNet router, your VPS, a hardened phone and laptop, an off-site location on cellular or other Wi-Fi, a second device or assistant, and about two hours. You work only against your own node, router, VPS, and home network.

STANDARD: Record a Baseline and an After score for each vector on the scorecard. The task is met when: the LAN scan shows only the intended ports; the internet shows no open ports at your home address and only ports 22, 80, and 443 on the VPS; no device on the LAN resolves around AdGuard; the services answer over the tailnet and nowhere else; the required fault is injected and repaired; and the node restores its services on its own after a reboot with the backup restore confirmed.

> SAFETY FIRST
>
> Keep a keyboard and monitor on the Pi while you work, because a bad ufw or SSH change can drop your connection over the network. Back up a file before you edit it, and inject only the faults listed here, never a rule on port 22.

PHASE 1 · RECON: MAP YOUR NODE'S SURFACE

Task 1 · Know the terrain

1. On the Pi, run `ip -4 addr` and confirm its LAN address is the reserved `192.168.8.152`. The router is `192.168.8.1` and the LAN is `192.168.8.0/24`.
2. Find the tailnet address with `tailscale ip -4`. It is `100.64.0.1`, the address Synapse binds to.
3. Find your home public address: run `curl ifconfig.me` from a device on the LAN. Note it and your VPS address for Task 5.

Task 2 · Scan the Pi from the LAN

1. On the laptop, install the scanner if needed with `brew install nmap`, then run `nmap -sV 192.168.8.152`.
2. Expect `22` (SSH), `53` (AdGuard DNS), and the AdGuard web UI on `80` or `3000`. Port `8008` must not appear, because Synapse binds to the tailnet address, not the LAN. Task 9 makes this break.
3. Score: count the ports answering on the LAN. Write it in the Node surface (LAN) row, Baseline.

Task 3 · Confirm the listeners, firewall, and SSH auth

1. List what is listening with `ss -tlnp`. Synapse should show on `100.64.0.1:8008`, AdGuard DNS on `0.0.0.0:53`, and SSH on `22`. Nothing should bind to a public address.
2. Read the firewall with `sudo ufw status verbose`. The default should be deny incoming, with LAN-scoped allows for `22` and `53` from `192.168.8.0/24`. Note anything else. Docker publishes its container ports (`53`, `80`, `3000`) with its own rules that bypass ufw, so ufw does not list them and the LAN scan in Task 2 is the true picture of what is reachable.
3. Confirm SSH refuses passwords: `sudo sshd -T | grep -Ei 'passwordauthentication|permitrootlogin'` must read `passwordauthentication no`. Then test from the laptop: `ssh -o PreferredAuthentications=password user@192.168.8.152` must fail without prompting.
4. Score: note the auth method in the Ingress control row, Baseline.

Task 4 · Check DNS filtering and hunt for leaks

1. Confirm AdGuard resolves and blocks: `dig +short @192.168.8.152 example.com` returns a normal address, and `dig +short @192.168.8.152 doubleclick.net` returns `0.0.0.0`.
2. Open the AdGuard dashboard, confirm both lookups appear in the Query Log with the device address, and read the block rate.
3. Hunt for leaks. A device can bypass AdGuard with its own encrypted DNS. On each device, check that no browser Secure DNS setting and no phone Private DNS setting points anywhere but the Pi. A device that resolves `doubleclick.net` to a real address is leaking.
4. Score: count the leaking devices and note the block rate in the DNS integrity row, Baseline.

PHASE 2 · OFF-SITE: PROVE THE BOUNDARY AND OPERATE IT

Task 5 · See what the internet can reach

1. Go off-site and off your home network, on cellular or other Wi-Fi. Do not run this from your own LAN, because that tests the router's inside, not what the internet sees.
2. Scan your home public address with `nmap -Pn <your-WAN-address>`. The Pi has no port forwards, so nothing should answer. If a port answers, remove the forward on the router at `192.168.8.1`.
3. Scan your VPS with `nmap -Pn -sV <your-VPS-address>`. Only `22`, `80`, and `443` should answer. Anything else is a gap.
4. Without a laptop, use an online checker against your own address only, such as `canyouseeme.org`.
5. Score: count ports answering from the internet across your home address and VPS in the Node surface (WAN) row, Baseline.

Task 6 · Prove the only way in is the tailnet

1. Off the tailnet, try to shell in over the internet: `ssh user@<your-WAN-address>` must be refused or time out.
2. Bring up Tailscale and try again over the tailnet: `ssh user@100.64.0.1` is also refused. SSH is allowed only from `192.168.8.0/24`, so the Pi is administered from the LAN, and the tailnet carries services, not a shell.
3. Confirm the service answers over the tailnet: `curl -o /dev/null -w '%{http_code}\n' http://100.64.0.1:8008/_matrix/client/versions` returns `200`. Off the tailnet it returns `000`.
4. Score: the only things reachable are the services on the tailnet. Confirm the Ingress control row, Baseline reads zero shells or services reachable without the tailnet.

Task 7 · Operate the services and check roaming DNS

1. Confirm the tunnel: `tailscale status` lists the Pi with a recent handshake, and `tailscale ping 100.64.0.1` succeeds.
2. Open Element, signed in to your homeserver at `100.64.0.1`, and send a message. Delivery confirms the chain works over the tunnel.
3. Check roaming DNS: a phone on cellular usually uses the carrier's DNS, not the Pi, so it is not filtered off-site unless you point it back. Record which off-site devices are filtered and which are not.
4. Score: add any unfiltered off-site device to the DNS integrity row, Baseline.

Task 8 · Optional: onboard a teammate from the field

1. Shell into the VPS, whose SSH is reachable from anywhere and key-only: `ssh user@<your-VPS-address>`.
2. Mint a key with `sudo headscale preauthkeys create --user 1 --reusable --expiration 24h` and copy the `hskey-...` value.
3. On the new device, enroll with `sudo tailscale up --login-server https://headscale.your-domain --authkey <the key>`, and confirm it with `sudo headscale nodes list`.
4. Install Element on that device, sign in at `100.64.0.1`, and message your other account.

PHASE 3 · BREAK IT AND REPAIR IT

Task 9 · Required inject: misbind Synapse, then fix it

1. Back up the file first: `cp ~/synapse/compose.yaml ~/synapse/compose.yaml.bak`.
2. Break it: edit `~/synapse/compose.yaml`, change the port line from `100.64.0.1:8008:8008` to `0.0.0.0:8008:8008`, then run `cd ~/synapse && docker compose up -d`.
3. Detect it: re-run `nmap -sV 192.168.8.152` from the laptop. Port `8008` now answers on the LAN, so Synapse is exposed on the wrong interface.
4. Fix it: `cp ~/synapse/compose.yaml.bak ~/synapse/compose.yaml`, run `docker compose up -d`, and re-scan to confirm `8008` is gone from the LAN.

Task 10 · Diagnose off-site, repair on the LAN

1. With LAN access or a teammate on the LAN standing by, stop a service: `docker stop synapse`.
2. From off-site, Element will not connect. Diagnose without a shell on the Pi: `tailscale ping 100.64.0.1` still succeeds, so the tunnel is fine, but `curl -o /dev/null -w '%{http_code}\n' http://100.64.0.1:8008/_matrix/client/versions` returns `000`. The fault is the service, not the network.
3. Fix it on the LAN, since the Pi has no remote shell. You, or the teammate on the LAN, run `docker start synapse`. A reboot also restores it, because the restart policy is `unless-stopped`.
4. Confirm the curl probe returns `200` again and Element reconnects.

Task 11 · Optional injects: DNS leak and dropped peer

1. DNS leak: set your phone's Private DNS to a public resolver. Detect it: the device now resolves `doubleclick.net` to a real address and stops showing in the AdGuard Query Log. Fix it by setting Private DNS back to Automatic.
2. Dropped peer: run `sudo tailscale down` on a client. Detect it: `tailscale status` shows the peer offline and Element cannot reach `8008`. Fix it with `sudo tailscale up`, or run `tailscale logout` and re-enroll with a fresh key as in Task 8.

PHASE 4 · CLOSE, RECOVER, AND SCORE

Task 12 · Close the gaps and re-scan

1. Fix what the recon turned up: remove any unexplained ufw allow, turn off password auth if Task 3 caught it prompting, remove any router port forward from Task 5, and shut off any device DNS setting that leaks around AdGuard.
2. Confirm nothing broke: re-run `sudo ufw status verbose` and `ss -tlnp`, and check that the Task 9 fix left Synapse on `100.64.0.1:8008`.
3. Re-scan the LAN, and the WAN and VPS from off-site. Write the new counts in the Node surface rows, After, and re-count DNS leaks into the DNS integrity row, After.

Task 13 · Run the recovery drill

1. Note the time and run `sudo reboot`.
2. When it returns, run `docker ps` to confirm every container is Up. A container can be Up while the service is not ready, so also re-run the Task 2 scan and the Task 4 `dig` block test.
3. Write the minutes from reboot to services answering in the Recovery row, After.
4. Test the backup: run `./backup.sh` and confirm it prints `Restore test: OK`.

Task 14 · Total the score and set the cadence

1. Add up the Baseline and After columns and write the difference. Lower is better on every row.
2. Write your adversary summary below. You pass if nothing unexpected answers from the internet, no LAN device leaks around AdGuard, services reach only over the tailnet, you injected and repaired the required fault, and the node returned on its own after the reboot.
3. Set a reminder to re-run the external scan and the DNS leak hunt every 3 to 4 months, and write the first re-check date in the cadence log.

Task 15 · Adversary summary and field notes

In three sentences, describe your node as an adversary would: what they can reach from the internet, what they must still defeat to get in (the tailnet, and key-only SSH that answers only on the LAN), and which threat from your Module 00 threat model this node answers. Log which fault you injected, how you detected it, where you fixed it, and the reboot recovery time.

Node exposure scorecard

Fill in Baseline during Phases 1 and 2, and After during Phase 4. Lower is better on every row. You pass when the internet-facing surface is only what you intended, LAN DNS leaks reach zero, services answer only over the tailnet, and the node recovers on its own after a reboot.

Vector	What you count	Baseline	After
Node surface (LAN)	Ports answering on the LAN scan (expect 22, 53, 80/3000; not 8008)		
Node surface (WAN)	Ports answering from the internet, home address plus VPS (home: 0; VPS: 22/80/443)		
DNS integrity	Devices leaking around AdGuard (goal: 0 on the LAN)		
Ingress control	Shells or services reachable without the tailnet (goal: 0)		
Recovery	Minutes from reboot to services answering again		
TOTAL	Add up the rows above		
DELTA	Baseline total minus After total (write it in the After box)		

#	Task	Where / Tool	Standard (done =)	Date	Finding
1	Map the addresses	`ip -4 addr` / `tailscale ip -4` / `curl ifconfig.me`	LAN, tailnet, WAN, and VPS addresses recorded		
2	LAN port scan	`nmap -sV 192.168.8.152`	22, 53, 80/3000 only; 8008 absent		
3	Listeners + firewall + SSH auth	`ss -tlnp` / `ufw status` / `sshd -T`	Bindings correct; default-deny; password auth off		
4	DNS filter + leak hunt	`dig @192.168.8.152` / AdGuard Query Log	Blocks confirmed; leaking devices counted		
5	External surface scan	`nmap -Pn` (WAN + VPS), off-site	Home: nothing; VPS: 22/80/443 only		
6	Ingress boundary test	`ssh` (WAN + tailnet) / `curl :8008`	Both shells refused; service 200 over tailnet only		
7	Off-site operate + DNS check	`tailscale status` / Element / `dig`	Message sent; roaming DNS state recorded		
8	Field onboarding (optional)	`ssh VPS` / `headscale preauthkeys` / `tailscale up`	New device enrolled; two accounts messaging		
9	Misbind + repair Synapse	`compose.yaml` (with `.bak`) / `nmap`	8008 appears on LAN, then removed after fix		
10	Diagnose off-site, repair on LAN	`docker stop` / `tailscale ping` / `curl` / `docker start`	Fault isolated to service; restored to 200		
11	Optional injects (DNS + peer)	Private DNS / `tailscale down|logout`	Each fault detected and reversed		
12	Close and re-scan	`ufw` / router / device DNS + `nmap`	Gaps closed; LAN, WAN, and DNS re-measured		

#	Task	Where / Tool	Standard (done =)	Date	Finding
13	Reboot + backup recovery	`sudo reboot / docker ps / backup.sh`	Services auto-return; Restore test: OK		
14	Score, summary, cadence	`scorecard + threat model`	Delta computed; summary written; re-check set		

Re-scan cadence log

Surface drifts: a firmware update reopens a port, a new device joins the LAN with its own DNS, a container changes how it binds. Re-run the external scan and the DNS leak hunt every 3 to 4 months and log each pass here.

Check	Run (date)	Re-check due	Re-checked (date)	Clean? (Y/N)