My Notebook
Complete Notebook

PDF Version generated by

Om Shah (osshah@uts.edu.au)

on

Sep 19, 2025 @02:59 PM AEST

Table of Contents

Log of Activities

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 1:55 PM GMT+10

admin availability team brief research setup planning ilc install tooling documentation team-support verification quiet-recon mapping target-selection light-scans spoofing Windows-SMB rebuild virtualbox vmware-troubleshooting drivers display-fix arp validation

Week 1 – not present (work commitments)

Intention: n/a
What I did:

-
- Unavailable due to work commitments.
  Outcome:
- No activity this week.
  Tags: admin, availability
  Keywords: week1, work-commitments

Week 2 – joined group, brief received, start research and setup

Intention: join team, understand assignment, begin tooling research and VM setup.

What I did:

-
- Met the group and read the project brief and PDF.
- Researched Metasploitable 3 and identified required tools: Packer, Vagrant, Git for Windows.
- Attempted to get MS3 prebuilt (VMware) but links had moved or were unavailable.
-
- Looked for alternative sources, noted issues but paused due to other assessments.

Outcome:

-
- Correct tools were identified but downloads were not finalised this week.

Tags: team, brief, research, setup
Keywords: metasploitable3, prebuilt-missing, packer-build, vagrant, git-for-windows

---

Week 3 – no lab progress (uni workload)

Intention: planned out an approach when time allowed.
What I did:

-
- Focused on other assessments but planned out and allocated time for week 4.

Outcome:

-
- No material progress this week.
  Tags: admin, availability
  Keywords: week3, assessment-load

Week 4 – team meeting, roles, ILC questions

Date: Tue 19-08-2025 (started ~3:00 pm)
Intention: assign product proposal roles and clarify ILC tasks.
Hours spent: 4h (1h pre-class, 2h in class, 1h ILC at home)
What I did:

-
- Met with the team, assigned proposal roles and attack responsibilities.
-
- Asked ILC questions and confirmed activities and outputs.

Outcome:

-
- Roles locked and ILC expectations clear.
  Tags: team, planning, ILC
  Keywords: roles, proposal, ilc-clarification

Week 4 – full install push, documentation, ping verification, team handover

Date: Thu 21-08-2025 (started ~12:00 am)
Intention: complete VM installs, set up toolchain, document steps for team, verify basic connectivity.
Hours spent: 15h (7h install and documentation, 5h ILC, 4h product proposal)

What I did:

-
- Installed tools: Packer, Vagrant, Git for Windows.
- Moved work out of System32 folder to a user Tools folder for builds.
- Installed Packer plugins: virtualbox, chef, vagrant.

- Added VirtualBox to PATH and verified plugins.
- Built MS3 Ubuntu 14.04 with Packer (virtualbox-iso).
- Exported VM as OVA and imported into VMware where required.
- Attempted Windows Server on VMware (not stable), switched Windows Server to VirtualBox.
- Performed simple ping checks between Kali, Ubuntu and Windows to verify basic reachability.
- Wrote install-prep notes and shared on Microsoft Teams for the group.

Outcome:

-
- Working lab: Windows on VirtualBox, Ubuntu and Kali on VMware.
- Toolchain confirmed and documented.
- Basic connectivity verified by pings.
- Ready to proceed to next tasks when scheduled.

Tags: install, tooling, documentation, team-support, verification
Keywords: virtualbox, vmware, PATH, packer-plugins, ova-export, sha1-check, packer-build, vagrant, tools-folder, ping-check

---

Week 5 – passive recon, target selection, first spoofing (Windows)

Date: Wed 27-08-2025 (started ~7:00 am)
Intention: figure out who's on the network without making noise, and prove I can fake one device's identity on the Windows PC (spoofing only, no MITM).

Hours spent: (7 hours)
What I did:

- Watched the network with Wireshark instead of blasting scans. I sent a couple of pings to the gateway just to wake things up.
- From the traffic I spotted three devices:
  • 10.20.0.101 looked like Windows
  • 10.20.0.102 looked like Linux (ubuntu)
  • 10.20.0.103 was present but wouldn't answer pings
- Did a very small, quiet check of common services:
  • .102 had SSH and a web page open
  • .101 had Windows file sharing (SMB) open
  • .103 didn't show anything useful
- Chose .101 (Windows/SMB) and .102 (Linux [ubuntu] SSH/HTTP) as the main targets and saved .103 for later.
- Took baselines: on Kali I wrote down my network card's hardware address (MAC); on Windows I saved the current network "address book" (arp -a).

- Ran a one-off ARP spoof against Windows: I told Windows that Ubuntu's IP (10.20.0.102) was at Kali's MAC. Checked arp -a on Windows and saw the fake entry.
  (I did not turn on traffic forwarding. This was spoofing only, not MITM)

Outcome:

- I now have a clean, low-noise map of the subnet and which services are open.
- Spoofing worked: Windows showed the fake IP→MAC match.
- Ready to do MITM later as a separate step (poison both sides + allow forwarding).

Tags: quiet-recon, mapping, target-selection, light-scans, spoofing, Windows-SMB
Keywords: Wireshark, ping, SSH, HTTP, SMB, ARP, MAC, arp -a

Week 6 – rebuild, fix-ups, and MAC spoofing proof (still no MITM)

Date: Thu–Fri 04–05 Sep 2025
Intention: get Windows working again, keep the lab stable, and re-do MAC spoofing on the new Windows box to show clear proof.
Hours spent: ~7h (licensing 10m, black-screen fix 2h, tools/drivers 3.5h, driver follow-up 20m, spoofing 1h)

What I did:

- Rebuilt Windows. The old server hit the license wall and went "not genuine," so I rebuilt it clean.
- Fixed the black screen. In VirtualBox I upped the video memory, which finally let it boot.
  Tried moving to VMware (turned on 3D accel and more graphics memory), but…
- VMware Tools headache. Old VirtualBox drivers clashed with VMware's. The ISO I had was a "lite" one, so key drivers (network/graphics) kept failing. After too much time, I stuck with VirtualBox where everything already worked.
- New IP. The rebuilt Windows came up as 10.20.0.105 (not .101). I updated my notes.
- MAC spoofing (redo, in plain steps):
  - Wrote down Kali's real hardware address (MAC).
  - Turned Kali's network card off, changed the MAC to match Ubuntu's, then turned it back on.
  - Got a duplicate-MAC warning (expected, because Ubuntu is still alive with that same MAC). Windows pings to Kali timed out which is a normal "who's who?" confusion.
  - To prove the spoof clearly, I sent a gratuitous ARP (Kali "shouts" to the network: *"10.20.0.100 is at this MAC"*).
  - Checked Windows' network address book (ARP table): it now showed Kali's IP using Ubuntu's MAC.
  - I then pinged Ubuntu from Windows so it refreshed that entry too and now Windows saw both .100 and .102 with the same MAC, which is exactly what a MAC clone looks like.

Outcome:

- Lab is stable on VirtualBox; VMware put aside to save time.
- MAC spoofing proven on the new Windows build (clear ARP table evidence).
- Duplicate-MAC effects understood (lost pings until ARP is updated).
- No MITM yet (I did not turn on packet forwarding or poison the gateway). That stays as a separate next step.

Tags: rebuild, virtualbox, vmware-troubleshooting, drivers, display-fix, spoofing, arp, validation
Keywords: video memory, 3D acceleration, VMware Tools, VSock/VMCI, balloon driver, gratuitous ARP, duplicate MAC, ARP table, 10.20.0.105

Week 7 – Spoofing (DNS, DNS+ARP, DHCP)

Date: Mon–Fri 08–12 Sep 2025
Intention: Prove spoofing capabilities without full MITM by redirecting name resolution and (attempting) lease control, to lay groundwork for later MITM.

Hours spent: ~7h (DNS spoof 2h, DNS+ARP 2h, DHCP spoof 3h)

What I did:

- **DNS Spoofing (dnsmasq):**
  - Confirmed Kali lab IP and **disabled IP forwarding** to keep this distinct from MITM.
  - Verified port 53 was free; installed and configured **dnsmasq** to resolve any query to **10.20.0.100**.
  - Tested locally to confirm spoof responses.
- **DNS + ARP Spoofing:**
  - ARP-poisoned the gateway so Windows would query **Kali's fake DNS** without touching victim settings.
  - Added an IP alias and rebound **dnsmasq** so it could answer on the expected address.
  - Confirmed DNS redirection worked transparently.
- **DHCP Spoofing (dnsmasq → Yersinia):**
  - Configured a rogue DHCP (gateway + DNS pointing to Kali) to race the legit server.
  - Legitimate DHCP consistently won; switched to **Yersinia** in DHCP mode on the correct interface and re-tried.
  - Rogue server still lost the race—marked for follow-up.

Outcome:

- **DNS Spoofing** successfully demonstrated with **dnsmasq**.
- **DNS + ARP Spoofing** worked after IP aliasing, achieving transparent DNS redirection.

- **DHCP Spoofing** attempted with **dnsmasq** and **Yersinia**; not yet successful due to the legitimate server responding faster—needs more tuning.
- Groundwork established for upcoming **MITM** phases.

Tags: Week 7, Spoofing, DNS Spoofing, ARP Spoofing, DHCP Spoofing, Penterrant Lab, Kali, Yersinia, dnsmasq
Keywords: network spoofing, DNS redirection, ARP spoofing, DHCP rogue server, Kali Linux, VMware lab, victim redirection, network security testing, MITM preparation, dnsmasq, Yersinia

---

Week 8 – Sentinel MVP coding + lab workflow

Date: Wed 17 Sep 2025
Intention: Build a **minimum viable product** of Sentinel focused on safe scanning and clean reporting; finalize shared-folder workflow for fast iteration.

Hours spent: ~8h (coding 6.5–7h, workflow polish 1–1.5h)

What I did:

- **Sentinel MVP (scan mode):**
  - Created `config.yaml` (app name, default subnet, allowlist) + `config.py` loader.
  - Added allowlist safety helper in `utils.py` to block scans outside the lab.
  - Wrote `discovery.py` using **python-nmap** with `-T4 --top-ports 20 -sV`; structured results (IP, ports, service, product/version).
  - Built `main.py` CLI (modes: scan/attack/mitigate—**scan implemented**, others stubbed).
  - Used **Rich** to render clean result tables; added `__init__.py` for package structure.
- **Workflow polish:**
  - Ensured the **shared folder** mounts automatically (fstab) and symlinked it to Desktop for quick access.
  - Verified the code runs consistently on Kali with local venv and shared sources.

Outcome:

- **Sentinel MVP** built with a working **scan mode**.
- Shared folder integrated and **auto-mounted** for smooth host↔VM workflow.
- Config + loader enable quick subnet/allowlist edits **without code changes**.
- Safety check restricts scans to the **lab subnet** only.
- Discovery module scans top-20 TCP ports and outputs structured host/service data.
- **CLI (`main.py`)** in place (scan functional; attack/mitigate stubbed).
- Results displayed in clear **Rich** tables.

- Version control/structure ready for expansion in later weeks.

Tags: Sentinel, Week 8, MVP, Network Scanning, Config, Automation, Kali, VMware
Keywords: network discovery, Nmap, Python, YAML config, allowlist, VM shared folder, fuse.vmhgfs-fuse, CLI, Rich library, Penterrant, subnet scanning, MVP development, version control

Ideas and Designs

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 2:07 PM GMT+10

tooling workflow ilc design recon stealth target-selection platform-choice stability spoofing arp validation

Week 4-

Design choice – mixed hypervisors by role

What I designed/considered
Keep Windows Server on VirtualBox and Ubuntu + Kali on VMware Workstation Pro.

Why

- **Reliability:** Windows was unstable on VMware (boot loop and black screen) but ran cleanly on VirtualBox.
- **Build simplicity:** MS3 build path already aligns with Packer + VirtualBox on Windows.
- **Efficiency:** Linux guests are smooth on VMware on my PC therefore fewer tweaks were needed.
- **Risk reduction:** Avoids sketchy ISO workarounds and fragile re-imports and keeps a predictable workflow.
- **Portability option:** If needed later, i can always export from VirtualBox as OVA for VMware.

Outcome/decision

- Adopted the split: Windows = VirtualBox, Ubuntu/Kali = VMware.
- Define roles per platform, took base build snapshots after first clean boot, and document steps for teammates.

Links to Log/Evidence
ILC/Week 4

Tags: design, tooling, workflow
Keywords: virtualbox, vmware, metasploitable3, packer-build, snapshot, tools-folder

---

Week 5 –
Design choice – quiet recon and careful target picking

What I designed/considered

- Instead of blasting the whole subnet with a loud ARP sweep, I chose to sit back and listen using Wireshark.
- To avoid getting nothing at all, I planned to create just a little safe traffic (pings to the gateway) so machines would “talk” and reveal themselves.
- For hosts that stayed quiet, I would only do the bare minimum checks (like top 10 ports with nmap -Pn) to see if they were worth the effort.
- Targets would be picked based on value of services, not just who’s alive e.g. SMB on Windows or SSH/HTTP on Linux are more useful than a mystery host with nothing showing.

Why

- Stealth: Less chance of setting off alarms or “looking noisy” on the network.
- Clarity: Focus time on machines that actually expose useful services.
- Efficiency: Saves effort by ignoring dead ends like .103 that are alive at L2 but useless higher up.
- Realism: Attackers in real life don’t waste time hammering dead targets; they go where the payoff is bigger.

Outcome/decision

- Picked .101 (Windows with SMB open) and .102 (Linux with SSH + HTTP) as main targets.
- Parked .103 for later since it didn’t respond beyond ARP.
- Proved the concept by successfully spoofing Windows’ ARP table once, but kept MITM for a separate phase.

Links to Log/Evidence
ILC / Week 5

Tags: design, recon, stealth, target-selection
Keywords: Wireshark, ping, TTL, SMB, SSH, HTTP, ARP, nmap, quiet-scan

---

Week 6 –
Design choice – stick with VirtualBox and prove MAC spoofing

What I designed/considered

- After hitting licensing and driver problems with Windows on VMware, I weighed two options:
    1. Keep fighting VMware Tools and broken drivers,
    2. Roll back to VirtualBox where things already worked.
- For spoofing, I decided to redo MAC spoofing with the rebuilt Windows (now at .105) so I had a clean demonstration.
- To show clear proof, I planned to use a gratuitous ARP after the spoof so Windows’ ARP table would visibly update, even with duplicate MACs on the network.

Why

- Stability: VirtualBox gave me a working Windows machine right away, no hours lost on drivers.
- Time-saving: Avoided chasing VMware “lite ISO” issues and rollbacks.
- Clarity of demo: Gratuitous ARP gives visible evidence of spoofing success, instead of relying on broken pings.
- Separation of phases: Kept spoofing limited to identity changes only; saved MITM for later.

Outcome/decision

- Committed to VirtualBox for Windows going forward.

- Redid MAC spoofing against .105 and proved it by forcing Windows' ARP table to show Kali's IP with Ubuntu's MAC.
- Understood duplicate-MAC effects (lost pings, network confusion), but used gratuitous ARP and pings to refresh tables for proof.

Links to Log/Evidence
ILC / Week 6

Tags: design, platform-choice, stability, spoofing, arp, validation
Keywords: VirtualBox, VMware Tools, drivers, gratuitous ARP, duplicate MAC, ARP table, 10.20.0.105

Week 7 –

Design choice – spoofing paths without MITM

What I designed/considered

- **DNS Spoofing** with `dnsmasq` bound to Kali's lab IP (`10.20.0.100`), keeping **IP forwarding disabled** to avoid MITM.
- Verifying **port 53** wasn't already in use before starting the fake DNS.
- **DNS + ARP Spoofing**: ARP-poison the gateway so Windows/Ubuntu send DNS queries to Kali **without changing victim settings**; add an **IP alias** if needed so `dnsmasq` can bind on the address victims expect.
- **DHCP Spoofing**: first with `dnsmasq` (offer options **3=router** and **6=DNS** pointing at Kali), then with **Yersinia** in DHCP mode to race the legitimate server.
- Keep each spoof **separate from MITM**, documenting steps and evidence for later escalation.

Why

- **Transparency:** Redirect traffic/DNS without touching victim configs.
- **Separation of concerns:** Spoofing now, MITM later — cleaner evidence.
- **Realism:** Rogue DNS/DHCP are common attacker techniques.
- **Repeatability:** Simple configs and clear binding/aliasing make the demo reliable.

Outcome/decision

- **DNS Spoofing** worked with `dnsmasq`.
- **DNS + ARP Spoofing** succeeded after IP aliasing, giving transparent DNS redirection.
- **DHCP Spoofing** not yet successful (legit server won races); to be tuned in Week 9.
- Solid groundwork laid for later **MITM** demonstrations.

Links to Log/Evidence
ILC / Week 7

Tags: design, spoofing, dns, arp, dhcp, lab
Keywords: dnsmasq, ARP poisoning, IP alias, port 53, IP forwarding, DHCP option 3/6, Yersinia, rogue DHCP, transparent redirection

---

Week 8 –

Design choice – Sentinel MVP coding and lab workflow setup

What I designed/considered

- **Sentinel MVP (scan-first):**
  - `config.yaml` + `config.py` (app name, default subnet, **allowlist**).
  - `utils.py` safety check to **block non-allowlisted** subnets.
  - `discovery.py` using **python-nmap** with `-T4 --top-ports 20 -sV` and structured results (IP, port, proto, service, product/version).
  - `main.py` CLI (modes: **scan** implemented; **attack/mitigate** stubbed).
  - Clean **Rich** tables for terminal output; package structure with `__init__.py`.
- **Workflow setup:**
  - VMware **shared folder** mounted via `vmhgfs-fuse`, **symlink** on Desktop, **fstab** auto-mount for repeatability.
  - Kept the Python **venv on local disk** (not hgfs) for reliability.

Why

- **Modularity & safety:** Config-driven + allowlist reduces risk and eases changes.
- **Clarity:** Clear CLI + formatted tables = demo-ready.
- **Portability:** Works offline in the lab; quick to retarget by editing YAML.
- **Repeatability:** Auto-mounted shared folder speeds iteration across machines.

Outcome/decision

- **Sentinel MVP (scan mode)** working end-to-end.
- Allowlist enforcement active; scans constrained to lab subnet.
- Results rendered in clean **Rich** tables; technical/executive outputs ready.
- Shared folder auto-mount + Desktop shortcut in place; venv stable on local FS.
- Codebase structured and version-controlled; ready to add vuln mapping and attack/mitigate orchestration next.

Links to Log/Evidence
ILC / Week 8

Tags: design, coding, scanning, Sentinel, config, automation, workflow

Keywords: python-nmap, pyyaml, Rich, allowlist, YAML config, CLI, vmhgfs-fuse, auto-mount, venv, MVP, subnet scanning

Things that Worked

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 2:13 PM GMT+10

success research planning environment portability networking firewall team documentation ops recon stealth scanning efficiency targeting spoofing validation rebuild stability display troubleshooting platform-choice arp

Week 2 -

Requirement discovery – Vagrant/Packer identified

What worked
Early research clarified that MS3 builds expect Packer + VirtualBox builder with a Vagrant post-processor, plus Git for Windows.

Why it worked
Reading the brief/docs first saved time later and set the correct toolchain plus chatgpt helped simplify my understanding of the process.

Outcome/decision
Adopted Packer (virtualbox-iso) + Vagrant in the workflow; planned to export to OVA when VMware was needed.

Links to Log/Evidence
ILC/Week 2

Tags: success, research, planning
Keywords: packer-build, virtualbox-iso, vagrant, git-for-windows

---

Week 4 -

Toolchain configured – PATH and Packer plugins

What worked
Adding VirtualBox to PATH and installing Packer plugins (virtualbox, chef, vagrant) made builds possible.

Why it worked
Packer could find `VBoxManage` and recognise builder/provisioner/post-processor types.

Outcome/decision
Stable local build environment; verified with `packer plugins list`.

Links to Log/Evidence
ILC/Week 4

Tags: success, tooling, environment
Keywords: PATH, packer-plugins, virtualbox-iso

MS3 build via Packer (VirtualBox → OVA → VMware)

What worked
Built Ubuntu 14.04 MS3 with Packer (virtualbox-iso), then exported the VM as OVA and imported into VMware.

Why it worked
Using the builder the template expects, then a standard appliance export, avoids manual reconfiguration.

Outcome/decision
Consistent VM produced and portable between VirtualBox and VMware.

Links to Log/Evidence
ILC/Week 4

Tags: success, build, portability
Keywords: packer-build, ova-export, cross-hypervisor

Integrity gate – rejected mismatched ISO

What worked
Caught a SHA-1 mismatch on a web-archive Windows ISO and discarded it.

Why it worked
Hash verification flagged integrity risk before install.

Outcome/decision
Stuck to trusted media.

Links to Log/Evidence
ILC/Week 4

Tags: success, security
Keywords: sha1-check, official-sources

Network recovery – bidirectional ping across VMs

What worked
Aligned networks and allowed ICMP Echo in Windows so Ubuntu (VMware) and Windows (VirtualBox) could ping both ways.

Why it worked
Fixed subnet mismatch and firewall rule blocked replies.

Outcome/decision
Stable L2 connectivity across Kali/Ubuntu/Windows; ready for MITM.

Links to Log/Evidence
ILC/Week 4

Tags: success, networking, firewall
Keywords: subnet-fix, icmp-allow, cross-hypervisor

Team handover – install-prep shared on Teams

What worked
Wrote install-prep notes and posted to Microsoft Teams so the group could reproduce builds and networking fixes.

Why it worked
Reduced repeated troubleshooting; aligned everyone on the same steps.

Outcome/decision
Faster onboarding for teammates; fewer setup questions.

Links to Log/Evidence
ILC/Week 4

Tags: success, documentation, team
Keywords: install-prep, teams-post, handover

Team handover – install-prep shared on Teams

What worked
Wrote **install-prep** notes and posted to Microsoft Teams so the group could reproduce builds and networking fixes.

Why it worked
Reduced repeated troubleshooting; aligned everyone on the same steps.

Outcome/decision
Faster onboarding for teammates; fewer setup questions.

Links to Log/Evidence
ILC/Week 4

Tags: success, documentation, team
Keywords: install-prep, teams-post, handover

Base snapshot captured

What worked
Took a Base snapshot after import/install so I can revert quickly.

Why it worked
Gives a clean restore point before experiments.

Outcome/decision
Safer testing; faster recovery from mistakes.

Links to Log/Evidence
ILC/Week 4

Tags: success, ops
Keywords: snapshot, revert-point

Week 5 -

Passive recon

What worked
Listening with Wireshark and sending just a couple of pings to the gateway let me discover active hosts without scanning the whole subnet.

Why it worked
Even minimal traffic triggered ARP replies, which gave me IP ↔ MAC details of the machines.

Outcome/decision
I got a host inventory quietly: `.101` (Windows), `.102` (Linux), `.103` (unknown/filtered).

Links to Log/Evidence
Log of Activities / Week 5

Tags: success, recon, stealth
Keywords: Wireshark, ARP, ping, passive-capture

Service checks

What worked
Using `nmap -Pn` with only the top 10 ports showed me which services were open without blasting the whole network.

Why it worked
Skipping ping tests avoided wasted probes, and limiting to a few ports reduced “noise.”

Outcome/decision
Confirmed `.102` had **SSH + HTTP**, and `.101` had **SMB**. Marked both as high-value targets.

Links to Log/Evidence
Log of Activities / Week 5

Tags: success, scanning, efficiency
Keywords: nmap, -Pn, ssh, http, smb

Target choice

What worked
I focused on `.101` (Windows with SMB) and `.102` (Linux with SSH/HTTP) as the main targets.

Why it worked
These services are easier to explore and historically vulnerable, unlike `.103` which was silent beyond ARP.

Outcome/decision
Two clear targets identified for later spoofing and exploitation attempts.

Links to Log/Evidence
Log of Activities / Week 5

Tags: success, targeting, planning
Keywords: smb, ssh, http, prioritisation

Spoofing test

What worked
Ran an ARP spoof so Windows thought Ubuntu's IP belonged to Kali's MAC. Windows' `arp -a` showed the poisoned entry.

Why it worked
ARP has no built-in validation, so Windows simply believed the fake mapping.

Outcome/decision
Proved I can poison Windows' ARP table. Kept MITM separate for next phase.

Links to Log/Evidence
Log of Activities / Week 5

Tags: success, spoofing, validation
Keywords: arp, spoof, Windows, arp -a

---

week 6 -

Windows rebuild

What worked
Rebuilt the Windows Server VM after the license expired and caused “not genuine” mode.

Why it worked
A clean reinstall was faster and more reliable than trying to fix licensing issues.

Outcome/decision
Got a fresh Windows VM at `.105`, stable and ready for spoofing tests.

Links to Log/Evidence
Log of Activities / Week 6

Tags: success, rebuild, stability
Keywords: windows-server, license, rebuild

Black screen fix – video memory increase

What worked
In VirtualBox, raising video memory solved the black screen boot loop.

Why it worked
The default video memory setting wasn’t enough to start Windows; more memory allowed the VM to display properly.

Outcome/decision
VM booted normally, no more stuck black screen.

Links to Log/Evidence
Log of Activities / Week 6

Tags: success, display, troubleshooting
Keywords: video-memory, boot, virtualbox

Platform choice – sticking with VirtualBox

What worked
After hitting VMware Tools errors and driver rollbacks, I stayed with VirtualBox where everything already worked.

Why it worked
Avoided wasting hours on broken drivers and “lite” ISOs; VirtualBox already had networking and display stable.

Outcome/decision
Windows will stay on VirtualBox; Ubuntu + Kali remain on VMware.

Links to Log/Evidence
Log of Activities / Week 6

Tags: success, stability, platform-choice
Keywords: virtualbox, vmware-tools, drivers

MAC spoofing

What worked
Changed Kali’s MAC to match Ubuntu’s, then sent a gratuitous ARP so Windows updated its ARP table.

Why it worked
Gratuitous ARP “shouts” the fake mapping, and Windows accepted it without question.

Outcome/decision
Windows showed Kali’s IP using Ubuntu’s MAC, clear proof the spoof succeeded.

Links to Log/Evidence
Log of Activities / Week 6

Tags: success, spoofing, arp
Keywords: mac-spoof, gratuitous-arp, arp-table, validation

Week 7

DNS spoofing (dnsmasq)

What worked
Configured `dnsmasq` on Kali to answer all DNS queries with `10.20.0.100` after confirming port 53 was free and **IP forwarding was disabled** (keep spoofing separate from MITM).

Why it worked
DNS trust is unauthenticated in this lab; victims accept the first valid DNS response. Binding `dnsmasq` to the lab IP ensured replies came from the expected host.

Outcome/decision
DNS spoofing confirmed; use as a building block for later demos and combine with ARP where victim-side changes aren't desired.

Links to Log/Evidence
ILC / Week 7

Tags: success, spoofing, dns
Keywords: dnsmasq, port 53, IP binding, IP forwarding off

DNS + ARP spoofing

What worked
Used ARP poisoning of the gateway so Windows/Ubuntu sent DNS to Kali automatically; added an **IP alias** so `dnsmasq` could bind where victims expected.

Why it worked
ARP has no authentication; poisoning updates the victim's ARP cache, so DNS queries route via Kali without touching victim configs.

Outcome/decision
Transparent DNS redirection achieved; keep aliasing step in the runbook for reliability.

Links to Log/Evidence
ILC / Week 7

Tags: success, spoofing, arp, dns
Keywords: ARP poisoning, IP alias, gateway spoof, transparent DNS

DHCP spoof groundwork (dnsmasq → Yersinia)

What worked
Set up a rogue DHCP (options **3=router**, **6=DNS** pointing to Kali) and tested with **Yersinia** in DHCP mode on the correct interface.

Why it worked / didn't fully work
Rogue offers were sent, but the legitimate DHCP server answered faster, so clients preferred the legit lease.

Outcome/decision
Not yet successful; plan to tune timing/rates and consider DHCP starvation or isolation for Week 9.

Links to Log/Evidence
ILC / Week 7

Tags: attempt, spoofing, dhcp
Keywords: rogue DHCP, lease race, Yersinia, options 3 & 6

Week 8 –

Sentinel MVP (scan mode)

What worked
Built **config-driven** scanner: `config.yaml` + loader (`config.py`), allowlist guard (`utils.py`), Nmap discovery (`discovery.py` with `-T4 --top-ports 20 -sV`), and CLI entrypoint (`main.py`).

Why it worked
Modular design keeps logic clean; allowlist prevents off-scope scans; version detection adds useful context to findings.

Outcome/decision
MVP scanner complete; attack/mitigate remain as stubs for later milestones.

Links to Log/Evidence
ILC / Week 8

Tags: success, coding, scanning, Sentinel
Keywords: python-nmap, YAML, allowlist, `-sV`, top-20 ports

Reporting & UX polish

What worked
Used **Rich** to print clear tables; structured results (IP, port, proto, service, product/version) suitable for technical/executive outputs.

Why it worked
Readable CLI output speeds interpretation and is demo-friendly; structured data makes later reporting easy.

Outcome/decision
Adopt Rich tables and structured findings as the default presentation layer.

Links to Log/Evidence
ILC / Week 8

Tags: success, reporting, ux
Keywords: Rich tables, structured findings, exec/tech outputs

---

Workflow reliability (shared folder + venv)

What worked
Kept code in **VMware shared folder** (auto-mounted via `fstab`, Desktop symlink) and created the Python **venv on local disk** (not `hgfs`) to avoid symlink/permissions issues.

Why it worked
hgfs syncs code seamlessly across host/VM, while a local venv ensures package installs and tooling behave correctly.

Outcome/decision
Fast iteration across machines; stable runtime environment for further development.

Links to Log/Evidence
ILC / Week 8

Tags: success, workflow, environment
Keywords: vmhgfs-fuse, auto-mount, symlink, venv, shared folder

---

Things that Failed

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 2:36 PM GMT+10

failure downloads tooling install security workflow environment recon targeting scanning prioritisation tradeoff licensing rebuild display vmware drivers vmware-tools spoofing arp

Week 2 -

MS3 prebuilt images (VMware) – links moved/removed

What failed

- Rapid7/official links for Metasploitable 3 prebuilt VMware images were gone/redirected; download path dead.

Likely cause

- Repo restructure/deprecation of prebuilt artefacts.

Fix / decision

- Switched to build-from-source: git clone MS3, used vagrant post-processor, packer and VirtualBox builder to produce the VM.

Links to Log/Evidence

- ILC/Week 2
- ILC/Week 4

Tags: `failure`, `downloads`, `tooling`
Keywords: `metasploitable3`, `prebuilt-missing`, `packer-build`

Week 2–3

Time/availability – progress stalled

What failed
Other assignments and work commitments meant no useful output by end of Week 2 and no lab work in Week 3.

Likely cause
Conflicting deadlines and limited on-campus time.

Fix / decision
Scheduled a long day of work in Week 4 and caught up: full install, networking fix, recon, and documentation for the team.

Links to Log/Evidence
ILC/Week 2
ILC/Week 3
ILC/Week 4

Tags: failure, admin, time
Keywords: work-commitments, assessment-load, catch-up-session

---

Week 4 -

Windows Server on VMware – boot loop / black screen

What failed

- Windows Server 2008 R2 on VMware Workstation Pro kept black-screening and rebooting in a loop.

Likely cause

- Virtual hardware / driver mismatch during install or post-export import.

Fix / decision

- Abandoned VMware build for Windows.
- Kept Windows Server on VirtualBox (stable there).
- For VMware use, rely on OVA export only if needed later.

Links to Log/Evidence

- ILC/Week 4

Tags: `failure`, `install`, `tooling`
Keywords: `vmware-loop`, `virtualbox`, `ova-export`

Untrusted Windows ISO – integrity check failed

What failed

- Web-archive Windows Server ISO SHA-1 didn’t match Microsoft reference.

Likely cause

- Tampered/corrupt mirror; not an official distribution.

Fix / decision

- Deleted the ISO and refused to use unverified ISO.
- Continued with VirtualBox build path that worked.

Links to Log/Evidence

- ILC/Week 4

Tags: `failure`, `security`
Keywords: `sha1-check`, `tamper-risk`, `official-sources`

Packer in `C:\Windows\System32` – protected directory

What failed

- Tried running `git clone`/Packer commands under System32; permissions and path issues blocked progress.

Likely cause

- System-protected directory; non-portable working dir.

Fix / decision

- Created a dedicated Tools folder (user-writable).
- Re-ran all commands there; documented the correct pathing.

Links to Log/Evidence

- ILC/Week 4

Tags: `failure`, `workflow`
Keywords: `sys32`, `tools-folder`, `pathing`

Packer/VirtualBox not found – PATH & plugins missing

What failed

- Packer couldn't find VBoxManage / builder types; builds wouldn't start.

Likely cause

- VirtualBox wasnt on PATH; required Packer plugins (virtualbox-iso, chef, vagrant) not installed.

Fix / decision

- Added `C:\Program Files\Oracle\VirtualBox` to PATH.
- Installed plugins: `virtualbox`, `chef`, `vagrant`.
- Verified with `packer plugins list`.

Links to Log/Evidence

- ILC/Week 4

Tags: `failure`, `tooling`, `environment`
Keywords: `PATH`, `virtualbox-iso`, `packer-plugins`

Cross-hypervisor networking – Ubuntu couldn't ping Windows

What failed
Ubuntu (VMware) couldn't ping Windows (VirtualBox). Mixed subnets and Windows firewall blocked replies.

Likely cause
Different default virtual networks (VMware vs VirtualBox) and Windows ICMP Echo blocked for that subnet.

Fix / decision
Changed Windows firewall to allow ICMP Echo from the VMware subnet and aligned adapter settings. Result: bidirectional ping OK across all VMs.

Links to Log/Evidence
ILC/Week 4

Tags: failure, networking, firewall
Keywords: subnet-fix, icmp-allow, cross-hypervisor

---

week 5 -

Host `.103` – alive but useless

What failed
`10.20.0.103` answered ARP but gave no ping replies and all Nmap probes came back filtered.

Likely cause
Probably a firewall blocking ICMP and common ports, or a machine with no real services running.

Fix / decision
Marked `.103` as a low-value target and moved on to `.101` (Windows/SMB) and `.102` (Linux/SSH+HTTP).

Links to Log/Evidence
Log of Activities / Week 5

Tags: failure, recon, targeting
Keywords: filtered, arp-only, 10.20.0.103

Lack of service confirmation on `.103`

What failed
Even with `-Pn`, Nmap couldn't show any open services on `.103`.

Likely cause

Firewall rules silently dropped packets, giving “filtered” state across all tested ports.

Fix / decision
Deprioritised `.103` since it offered no attack surface; focused efforts on other hosts.

Links to Log/Evidence
Log of Activities / Week 5

Tags: failure, scanning, prioritisation
Keywords: nmap, -Pn, filtered, no-open-ports

Noisy scan avoided (trade-off)

What failed
Did not run a fast ARP sweep, which meant discovery was slower and dependent on triggering traffic.

Likely cause
Deliberate choice to stay stealthy, but the downside was less immediate visibility.

Fix / decision
Accepted the trade-off: slower recon but much quieter. Used Wireshark ARP capture + selective pings instead.

Links to Log/Evidence
Log of Activities / Week 5

Tags: failure, tradeoff, recon
Keywords: arp-sweep, stealth, passive-capture

Week 6 -

Windows licensing expired

What failed
The original Windows Server VM hit the end of its license period and went into “not genuine” mode.

Likely cause
Time-limited evaluation license expired.

Fix / decision
Rebuilt the Windows Server VM from scratch instead of trying to bypass licensing.

Links to Log/Evidence
Log of Activities / Week 6

Tags: failure, licensing, rebuild
Keywords: windows-server, eval-license, rebuild

Black screen boot in VMware

What failed
After exporting Windows to VMware, it booted into a black screen loop.

Likely cause
Default video memory settings too low; graphics hardware mismatch.

Fix / decision
Raised video memory in VirtualBox to solve the issue there, but VMware remained unreliable.

Links to Log/Evidence
Log of Activities / Week 6

Tags: failure, display, vmware
Keywords: black-screen, video-memory, vmware-loop

VMware Tools install errors

What failed
VMware Tools kept rolling back with driver errors (VSock/VMCI, balloon). Even when forcing the MSI, it gave error 2711.

Likely cause
Leftover VirtualBox Guest Additions conflicting with VMware drivers, and the mounted ISO was a “lite” version missing key drivers.

Fix / decision
Stopped chasing VMware Tools issues and stayed with VirtualBox for Windows.

Links to Log/Evidence
Log of Activities / Week 6

Tags: failure, drivers, vmware-tools
Keywords: vmware-tools, driver-conflict, error-2711

Duplicate MAC conflict during spoofing

What failed
When Kali copied Ubuntu’s MAC, the network got confused and dropped pings from Windows.

Likely cause
Two live machines were claiming the same MAC at the same time.

Fix / decision
Used a gratuitous ARP from Kali to force Windows to update its ARP table and show the spoofed mapping.

Links to Log/Evidence
Log of Activities / Week 6

Tags: failure, spoofing, arp
Keywords: duplicate-mac, gratuitous-arp, arp-table

Week 7: Things that failed

DNS plus ARP initial DNS binding failure

What failed
dnsmasq would not answer on 10.20.0.1 and nslookup from the victim kept failing.
Why it failed
Kali did not own 10.20.0.1 so dnsmasq could not bind to that address.
Outcome or decision
Added an IP alias for 10.20.0.1 on Kali and restarted dnsmasq. Documented add alias then bind then test as a required step.
Links to log or evidence
ILC Week 7
Tags
failure, binding, dns

Keywords
dnsmasq, IP alias, bind error, nslookup

ARP target confusion with wrong gateway

What failed
ARP spoofing against 10.0.3.2 produced no useful DNS redirection.
Why it failed
That gateway is on the internet or NAT side not the lab subnet. The correct lab gateway for spoofing is 10.20.0.1.
Outcome or decision
Standardised a pre attack check to confirm the victim interface and gateway on the lab network using ipconfig on Windows and ip route on Kali. Use only the lab gateway for ARP attacks.
Links to log or evidence
ILC Week 7
Tags
failure, targeting, arp
Keywords
gateway mismatch, NAT, host only, route check

DHCP spoof rogue server lost the race

What failed
Rogue DHCP using dnsmasq and then Yersinia did not override leases. Victims accepted offers from the legitimate server.
Why it failed
The legitimate DHCP server responded faster and more consistently so my offers arrived second.
Outcome or decision
Move to Week 9 for tuning. Increase packet rate and adjust timeouts. Consider DHCP starvation if allowed or segment isolation to suppress the legitimate server. Keep Yersinia on the correct interface and pre set options for router and DNS.
Links to log or evidence
ILC Week 7
Tags
failure, dhcp, race
Keywords
rogue DHCP, lease timing, Yersinia, options 3 and 6

Week 8: Things that failed

Virtual environment creation inside shared folder

What failed

Creating a Python virtual environment inside mnt hgfs Capstone failed with an operation not supported error.

Why it failed

The vmhgfs fuse filesystem does not provide all features required by virtual environments such as symlinks.

Outcome or decision

Create the virtual environment on local disk in the home directory and keep project code in the shared folder. Added this rule to the setup guide.

Links to log or evidence

ILC Week 8

Tags

failure, environment, venv

Keywords

vmhgfs fuse, filesystem limitation, venv location

Module import error for core

What failed

Running python minus m sentinel dot main crashed with ModuleNotFoundError for core.

Why it failed

core is a subpackage of sentinel. Absolute imports to core did not resolve when running as a module.

Outcome or decision

Switched imports to sentinel dot core or used relative imports. Added a quick import test to the runbook.

Links to log or evidence

ILC Week 8

Tags

failure, packaging, imports

Keywords

package layout, absolute import, relative import

Dependency installs blocked with no internet

What failed

pip install from requirements failed on Kali when isolated to the lab network.

Why it failed

There was no internet route from the lab network so pip could not reach package repositories.

Outcome or decision

Two path solution. Preferred is adding a second NAT network adapter to Kali for updates. Air gapped method is to download wheels on the host, copy them into the shared folder, then install using a local directory without contacting the internet.

Links to log or evidence

ILC Week 8

Tags

failure, dependencies, offline

Keywords
offline install, wheel cache, dual NIC, NAT

Shared folder not visible at first boot

What failed
mnt hgfs was empty even though shared folders were enabled.
Why it failed
The hgfs client was not mounted automatically at boot.
Outcome or decision
Verified exposure with vmware hgfsclient, mounted with vmhgfs fuse to mnt hgfs with allow other, and added a correct fstab entry for auto mount on reboot.
Links to log or evidence
ILC Week 8
Tags
failure, workflow, mount
Keywords
vmware hgfsclient, vmhgfs fuse, fstab, auto mount

What I learned or would do better

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 2:58 PM GMT+10

reflection security networking tooling workflow reflections recon stealth fingerprinting methodology stability platform-choice validation

Week 4 -

Integrity-first VM– verify before install

What I learned
Hash-check ISOs (prefer SHA-256, accept SHA-1 for legacy) against a trusted Microsoft checksum before use.

Why it matters
Prevents wasting hours on corrupt/tampered media and avoids risky installs.

What I'd do better next time
Only use official Microsoft sources/tools; verify hashes immediately after download and document the result.

Links to Log/Evidence
ILC/Week 4

Tags: reflection, security
Keywords: sha1-check, sha256, official-sources

Design for L2 first – confirm adjacency before attacks

What I learned
Proving 1-hop (same L2) and ARP behaviour early makes MITM feasible and saves chasing routing issues.

Why it matters
Removes network ambiguity and focuses effort on the actual attack steps.

What I'd do better next time
Baseline with ping on all VMs, then proceed to MITM only after both directions are clean.

Links to Log/Evidence
ILC/Week 4

Tags: reflection, networking
Keywords: 1-hop, arp, mitm-feasible

Environment hygiene – PATH + plugins + writable workspace

What I learned

Add VirtualBox to Windows PATH, install required Packer plugins, and avoid System32 as a working dir.

Why it matters
Removes “tool not found/unknown builder” errors and prevents permission roadblocks.

What I’d do better next time
Work from a user Tools folder, script PATH/plug-in checks, and record them in the prep notes.

Links to Log/Evidence
ILC/Week 4

Tags: reflection, tooling, workflow
Keywords: PATH, packer-plugins, tools-folder

---

week 5 -

Value of Stealth

What I learned
Watching ARP traffic in Wireshark plus sending a couple of pings gave me enough info to map the subnet without noisy scans.

Why it matters
Keeps recon quiet — no big ARP sweeps that would stand out if the network was monitored.

What I’d do better next time
Still start passive, but prepare small “nudge” traffic (like gateway pings) earlier so I don’t sit waiting for random chatter.

Links to Log/Evidence
Log of Activities / Week 5

Tags: reflection, recon, stealth
Keywords: wireshark, arp, passive-capture

TTL values are a quick OS clue

What I learned
Ping replies gave me OS hints: TTL=128 pointed to Windows, TTL=64 pointed to Linux.

Why it matters
Gives me fast confirmation of host type without extra tools.

What I'd do better next time
Note TTLs right away as part of baseline mapping so I don't need to double-check later.

Links to Log/Evidence
Log of Activities / Week 5

Tags: reflection, fingerprinting
Keywords: ttl, windows, linux, icmp

Pick targets based on service value, not just presence

What I learned
`.101` (Windows/SMB) and `.102` (Linux/SSH + HTTP) were worth pursuing; `.103` wasn't.

Why it matters
Avoids wasting time on filtered/no-service hosts; keeps focus on exploitable points.

What I'd do better next time
Run a quick top-ports check earlier on each host to confirm value before diving deeper.

Links to Log/Evidence
Log of Activities / Week 5

Tags: reflection, targeting, efficiency
Keywords: smb, ssh, http, prioritisation

Spoofing vs MITM

What I learned
I successfully poisoned Windows' ARP table (spoofing), but did not enable forwarding or target the gateway, so it wasn't MITM yet.

Why it matters
Keeps the lab work clear and structured — spoofing is identity trickery, MITM is full interception.

What I'd do better next time
Clearly label each stage in notes (spoof-only vs MITM) and take ARP table screenshots as proof right after each step.

Links to Log/Evidence
Log of Activities / Week 5

Tags: reflection, spoofing, methodology
Keywords: arp, spoofing, mitm, separation

Week 6 -

Licensing can kill momentum

What I learned
When the Windows eval license expired, it stopped being reliable and forced a rebuild.

Why it matters
Losing a working VM mid-lab wastes time and breaks continuity.

What I'd do better next time
Take a clean snapshot right after install so I can roll back quickly if licensing runs out.

Links to Log/Evidence
Log of Activities / Week 6

Tags: reflection, stability
Keywords: windows-server, snapshot

Not every hypervisor is worth fixing

What I learned
VMware gave me driver and tools headaches, but VirtualBox worked fine with Windows straight away.

Why it matters
Time spent troubleshooting platform quirks doesn't add value to the spoofing/MITM tasks.

What I'd do better next time
Pick the platform that's stable early, and only switch if there's a clear need.

Links to Log/Evidence

Log of Activities / Week 6

Tags: reflection, platform-choice
Keywords: virtualbox, vmware, drivers

Duplicate MACs cause confusion

What I learned
When Kali and Ubuntu both used the same MAC, Windows got confused — pings failed until I forced an ARP update.

Why it matters
Shows the risk of cloning an active machine’s MAC; it creates conflicts and dropped traffic.

What I’d do better next time
Either spoof a unique locally-administered MAC to avoid conflicts, or take the real host offline first.

Links to Log/Evidence
Log of Activities / Week 6

Tags: reflection, spoofing, networking
Keywords: mac-spoof, duplicate, arp-table

Gratuitous ARP is a good proof generating tool

What I learned
Sending a gratuitous ARP from Kali made Windows instantly update its ARP table to show the spoofed mapping.

Why it matters
It’s an easy, reliable way to prove spoofing success even with duplicate MACs.

What I’d do better next time
Use gratuitous ARP sooner to get visible proof without waiting for normal traffic.

Links to Log/Evidence
Log of Activities / Week 6

Tags: reflection, spoofing, validation

Keywords: gratuitous-arp, arp, spoofing-proof

Week 7 -

DNS spoofing is powerful when kept separate from MITM

What I learned
Keeping IP forwarding disabled and binding dnsmasq to the correct lab IP let me demonstrate DNS spoofing cleanly without turning it into a man in the middle.
Why it matters
Clear separation of spoofing from interception makes evidence easier to understand and reduces unintended routing changes.
What I would do better next time
Add a pre run checklist that includes verifying port 53 is free, confirming Kali’s lab IP, and explicitly switching IP forwarding off before starting.
Links to Log or Evidence
ILC Week 7
Tags
reflection, spoofing, methodology
Keywords
dnsmasq, port 53, IP forwarding off

Gateway targeting must be on the lab side

What I learned
Spoofing the NAT side gateway did not help the lab attack path. Targeting the lab gateway and the victim’s active interface is essential.
Why it matters
Correct adjacency is a prerequisite for reliable redirection and avoids time lost chasing the wrong path.
What I would do better next time
Verify the victim interface and default gateway on the lab subnet with ipconfig and ip route before starting any spoof.
Links to Log or Evidence
ILC Week 7
Tags
reflection, networking, targeting
Keywords
gateway, host only, adjacency, route check

Binding to an address you do not own will fail

What I learned
dnsmasq could not bind until I added the IP alias to Kali.
Why it matters
Services must bind to an address that exists on the host, otherwise clients will never see the response.
What I would do better next time
Create the alias first, then start or restart the service, then test with nslookup from both attacker and victim.
Links to Log or Evidence
ILC Week 7
Tags
reflection, configuration
Keywords
IP alias, bind, nslookup

Losing the Rogue DHCP race

What I learned
The legitimate DHCP server consistently answered faster than my rogue server.
Why it matters
Timing and interface selection decide who hands out leases, which directly affects whether traffic is redirected.
What I would do better next time
Tune packet rates and timeouts, consider DHCP starvation if permitted, or isolate the segment so the legitimate server cannot respond.
Links to Log or Evidence
ILC Week 7
Tags
reflection, dhcp
Keywords
rogue DHCP, lease race, interface, timing

Week 8 -

Config driven design saves time

What I learned
Putting app name, default subnet, and allowlist in a YAML file with a small loader kept settings out of code.
Why it matters
Faster retargeting and less chance of mistakes when environments change.
What I would do better next time
Document a standard config template and validate it at startup so errors are caught early.
Links to Log or Evidence
ILC Week 8
Tags

reflection, design, configuration

Keywords

YAML config, allowlist, loader

Scan first with clear outputs

What I learned

A simple scan mode that uses python nmap and prints structured results with Rich is enough to demonstrate value early.

Why it matters

Early working output builds momentum and provides evidence while attack and mitigation modes are still being built.

What I would do better next time

Add a scope line and nmap arguments to every report and include a note that top ports does not mean exhaustive coverage.

Links to Log or Evidence

ILC Week 8

Tags

reflection, scanning, reporting

Keywords

python nmap, Rich tables, scope

Keep the virtual environment on local disk

What I learned

Creating a virtual environment on the shared folder failed. Keeping the venv on local disk avoided filesystem limitations.

Why it matters

Prevents installation errors and keeps the development environment stable across reboots.

What I would do better next time

Include a one line activation step in the runbook and add an offline install path using pre downloaded wheels when there is no internet.

Links to Log or Evidence

ILC Week 8

Tags

reflection, environment, reliability

Keywords

venv, vmhgfs, offline install

Workflow automation matters

What I learned

Auto mounting the shared folder with fstab and adding a desktop symlink reduced friction when switching between host and VM.

Why it matters

Less setup time means more time coding and testing.

What I would do better next time

Add a quick health check section to the runbook that confirms the mount, symlink, and permissions before starting development.

Links to Log or Evidence

ILC Week 8

Tags

reflection, workflow

Keywords

fstab, symlink, shared folder

Week 1

Om Shah (osshah@uts.edu.au) - Aug 25, 2025, 10:47 PM GMT+10

admin availability

Date: 29th July Week 1

Intention: n/a

What I did:

- I was unavailable due to work commitments.

Outcome:

- No activity this week.

Tags: `admin, availability`
Keywords: `week1, work-commitments`

Week 2

Om Shah (osshah@uts.edu.au) - Aug 25, 2025, 10:47 PM GMT+10

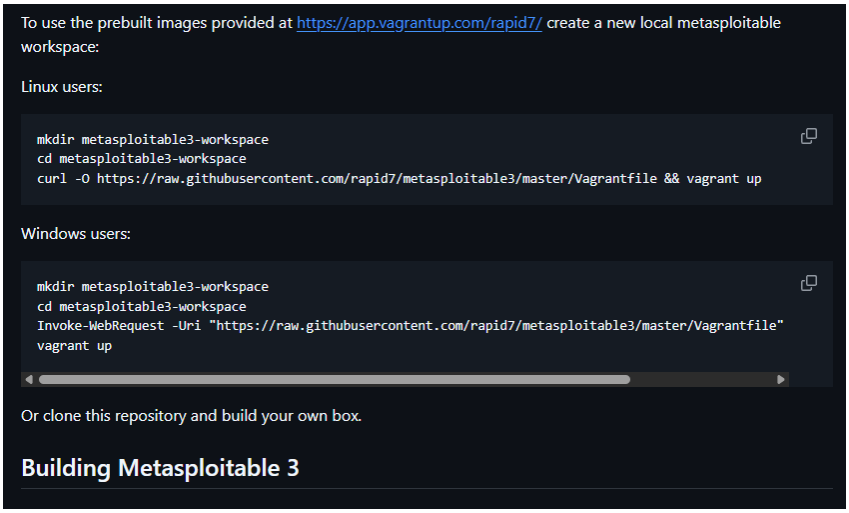
team brief research setup

Week 2 – joined group, brief received, start research and setup

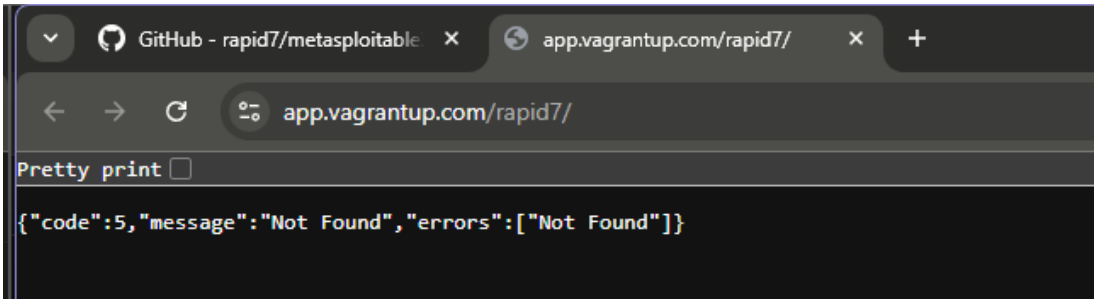
Date: 05/08/2025
Intention: join a team, understand the assignment, begin tooling research and VM setup.

What I did:

- Met the group and read the project brief and the PDF with project details.
- Started researching Metasploit/Metasploitable/Kali and lab setup steps and learned i needed vagrant and packer for builds.
- I tried to download Metasploitable 3 as it seemed like the best option since ms2 didnt have windows server which limited the attack surface which may result in a lack of learning opportunities across different platforms but had issues with ms3 pulling from Rapid7 GitHub or official links pointing to removed or changed resources.



the link for the prebuilt images compatible with vmware led to this



- I had to look for alternative sources however other assignments slowed progress resulting in nothing valuable found.

Outcome:

- I learned what didnt work which was valuable, but ultimately downloads were not finalised by the end of Week 2.

- **Tags:** `team, brief, research, setup`
- **Keywords:** `metasploitable3, prebuilt-missing, packer-build, vagrant, git-for-windows`

Week 3

Om Shah (osshah@uts.edu.au) - Aug 25, 2025, 10:48 PM GMT+10

admin availability

Week 3 – no lab progress (uni workload)

Date: 12/08/2025
Intention: continue setup if time allowed.
What I did:

- Was swamped with other assessments and could not go to campus on tuesday and the assignments hindered any further research but did allow me to plan out the following week so that i could get the neccessary work done.

Outcome:

- No material progress this week other than a rough plan for week 4 and optimised research plan to finally get the virtual machines up and running.

Tags: `admin`, `availability`
Keywords: `week3`, `assessment-load`

Week 4

Om Shah (osshah@uts.edu.au) - Aug 25, 2025, 10:49 PM GMT+10

team planning ilc install tooling documentation team-support verification

Week 4 – full install push, documentation, sharing with team

Date: Tue 19/08/2025 (started ~3:00 pm) (team meeting and class)

Intention: Assign tasks for product proposal and discuss ILC

Hours Spent: 4 hours (1 hour prior to class, 2 hours in class, 1 hour at home on the ILC)

Outcome:

Roles were assigned for the product porposal and questions i had regarding the ILC were discussed

---

Date: Thu 21/08/2025 (started ~12:00 am)

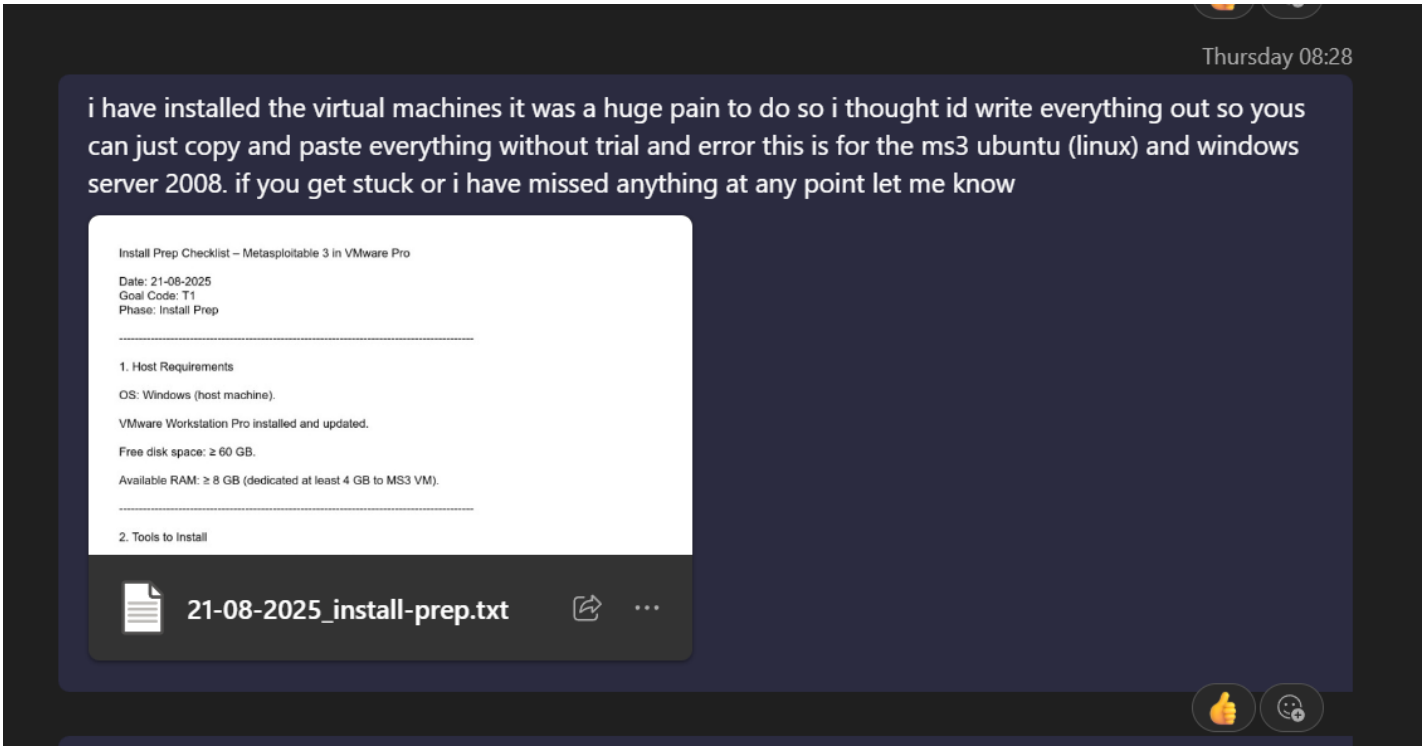
Intention: finish VM installs, fix issues, document steps for team, complete recon baseline.

Hours Spent: 15 hours (7 hours for full install and documentation, 5 hours for ILC, and 4 hours for product proposal)

What I did:

- Installed VMs: Ubuntu (VMware), Windows Server 2008 R2 (VirtualBox), Kali (VMware).

- Through my research on week 2 i was aware that vagrant, packer and git for windows was required to run ms3 on vmware workstation pro provided by the university so i installed those first and verified using the following commands i documented in my install-prep.txt file which was shared with the team to ease the installation process




```
2. Tools to Install

Vagrant → https://developer.hashicorp.com/vagrant/downloads

Packer → https://developer.hashicorp.com/packer/install

Git for Windows → https://git-scm.com/download/win
```

```
3. Verification Commands (PowerShell)

Run after installation:

vagrant --version
packer --version
git --version
```

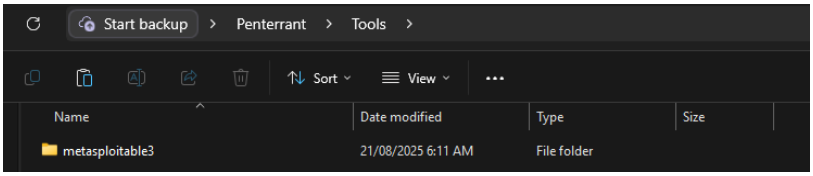
```
PS C:\WINDOWS\system32> git --version
git version 2.51.0.windows.1
PS C:\WINDOWS\system32> vagrant --version
Vagrant 2.4.8
PS C:\WINDOWS\system32> packer --version
Packer v1.14.1
PS C:\WINDOWS\system32>
```

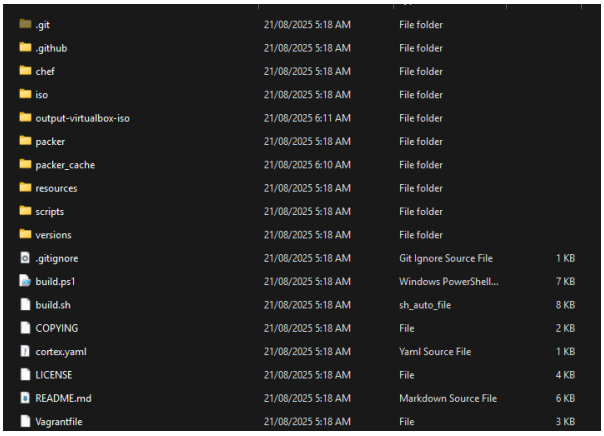
- after installing and verifying these tools i had to install ms3 and found a working link from rapid7's github however not the prebuilt image. i had to learn how to build the image myself using virtualbox and the above tools using the following documented process

```
4. Metasploitable3 Install (PowerShell)

run - git clone https://github.com/rapid7/metasploitable3.git
run - cd metasploitable3

*do not install on sys32 dir as it is protected*
```

```
PS C:\Users\omssh\Desktop\Penterrant\Tools>
PS C:\Users\omssh\Desktop\Penterrant\Tools> git clone https://github.com/rapid7/metasploitable3.git
Cloning into 'metasploitable3'...
remote: Enumerating objects: 4985, done.
remote: Counting objects: 100% (79/79), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 4985 (delta 61), reused 48 (delta 48), pack-reused 4906 (from 3)
Receiving objects: 100% (4985/4985), 248.19 MiB | 19.35 MiB/s, done.
Resolving deltas: 100% (2473/2473), done.
PS C:\Users\omssh\Desktop\Penterrant\Tools> cd metasploitable3
PS C:\Users\omssh\Desktop\Penterrant\Tools\metasploitable3>
```

- the commands written down clone ms3 from the rapid7 github however i was a little confused as to why it wasnt working and realised sys32 is a protected directory and therefore ran everything in a tools folder as shown below





- as packer requires the virtualbox builder plugin i had to install virtualbox alongside the packer plugins needed which are shown in the following. these are necessary as:

1. the chef plugin is so that a build can run chef inside the VM during image creation so that it can run the "recipes" needed to install services and vulnerable configurations.
2. ms3 needs the virtualbox-iso builder otherwise packer wouldnt be able to talk to virtualbox making it impossible to communicate for VM creation and this one was chosen rather than another builder so that it would keep packer clean on windows and avoid version conflicts
3. the vagrant command installs the vagrant post-processor so Packer can package the finished VM into a .box file for Vagrant.

4. finally the packer plugins installed is just a sanity check to ensure everything is installed as required

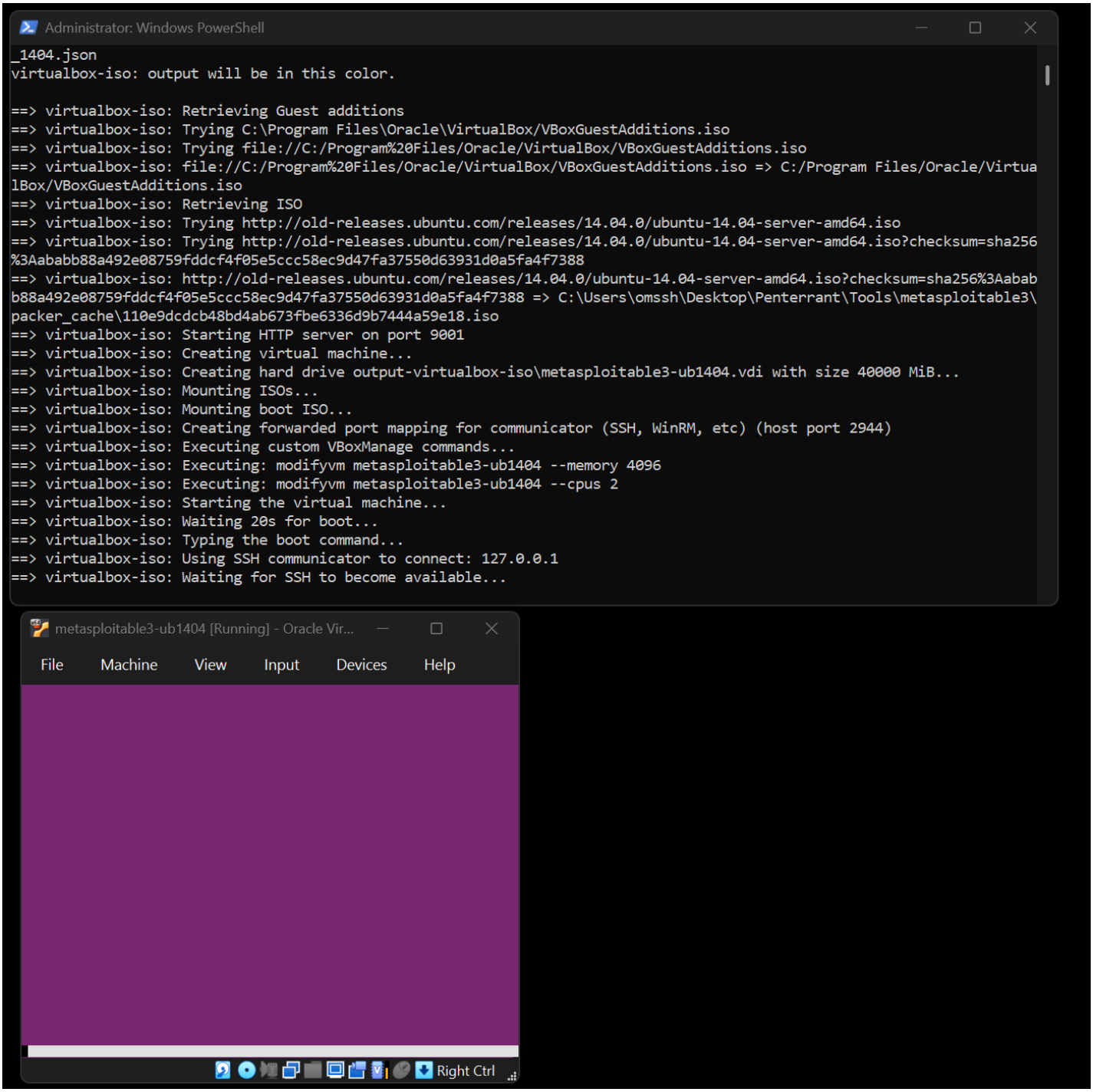
```
VirtualBox Setup (NECESSARY)

*you must have VirtualBox installed for the build itself, however we can use VMWARE PRO later on*

*Packer (v1.14.x) needs the VirtualBox builder plugin*

packer plugins install github.com/hashicorp/chef
packer plugins install github.com/hashicorp/virtualbox
packer plugins install github.com/hashicorp/vagrant
packer plugins installed

*add to path permanently*

[Environment]::SetEnvironmentVariable(
  "Path",
  $env:Path + ";C:\Program Files\Oracle\VirtualBox",
  "User"
)
```

- at the end it was important to add virtualbox to the path i used chatgpt for this command as i didnt understand the error i was being thrown but unfortunately i did not document the error at the time. regardless the reasoning for adding virtualbox to the path is essentially so that it is accessible from any shell i was provided the option for temporarily adding it to the "current" powershell terminal but chose to permanently add it to simplify running commands in the future without having to go through this process again.

- Once this was completed i had to build the actual machines again with the help of the metasploitable3 folder we previously made using the git clone from rapid7 and used the follwing command to build the machine

```
5. Build Metasploitable 3 UBUNTU (PowerShell)

cd ...metasploitable3 (goto folder)

Ubuntu 14.04 (Linux target) Installation:

packer build --only=virtualbox-iso .\packer\templates\ubuntu_1404.json

-- will open virtualbox

login: vagrant
password: vagrant
```

packer build --only=virtualbox-iso .\packer\templates\ubuntu_1404.json

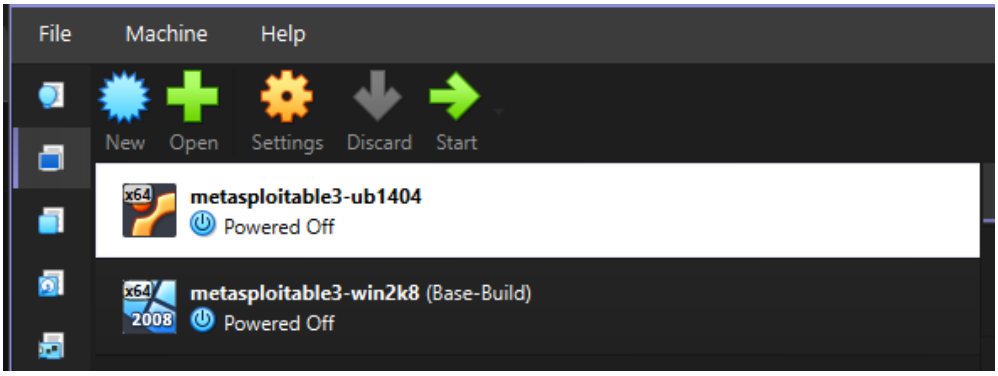


- the above image documents the build-in-progress step and the below image shows the ubuntu machine once created in virtualbox. however to export to vmware workstation there were extra steps that were needed as shown in section 5.1 of the install-prep.txt.

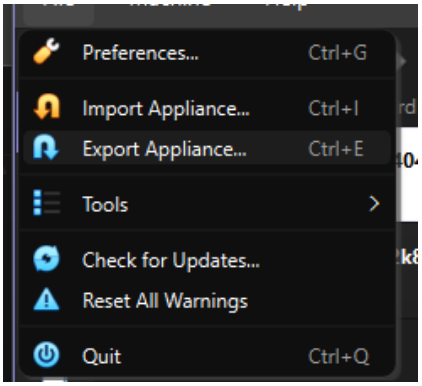
```
Ubuntu 14.04 LTS ubuntu tty1

ubuntu login:vagrant
Password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
vagrant@ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
0
    link/ether 08:00:27:0e:97:cb brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fd17:625c:f037:2:9d2e:bbf5:5dc8:f864/64 scope global temporary dynamic
       valid_lft 86332sec preferred_lft 14332sec
    inet6 fd17:625c:f037:2:a00:27ff:fe0e:97cb/64 scope global dynamic
       valid_lft 86332sec preferred_lft 14332sec
    inet6 fe80::a00:27ff:fe0e:97cb/64 scope link
       valid_lft forever preferred_lft forever
vagrant@ubuntu:~$
```

```
5.1. Export from VirtualBox

go to machines (main VirtualBox Manager window) --> File --> Export Appliance…

Select your metasploitable3-ub1404 VM

once VirtualBox closes go to Documents and move the .ova file to you preferred location

open the OVA file in VMWare and name it

*VMWare will have restrictions on if the file can be imported due to strict restrictions but
clicking retry will install open the VM with loosened restructions*
```

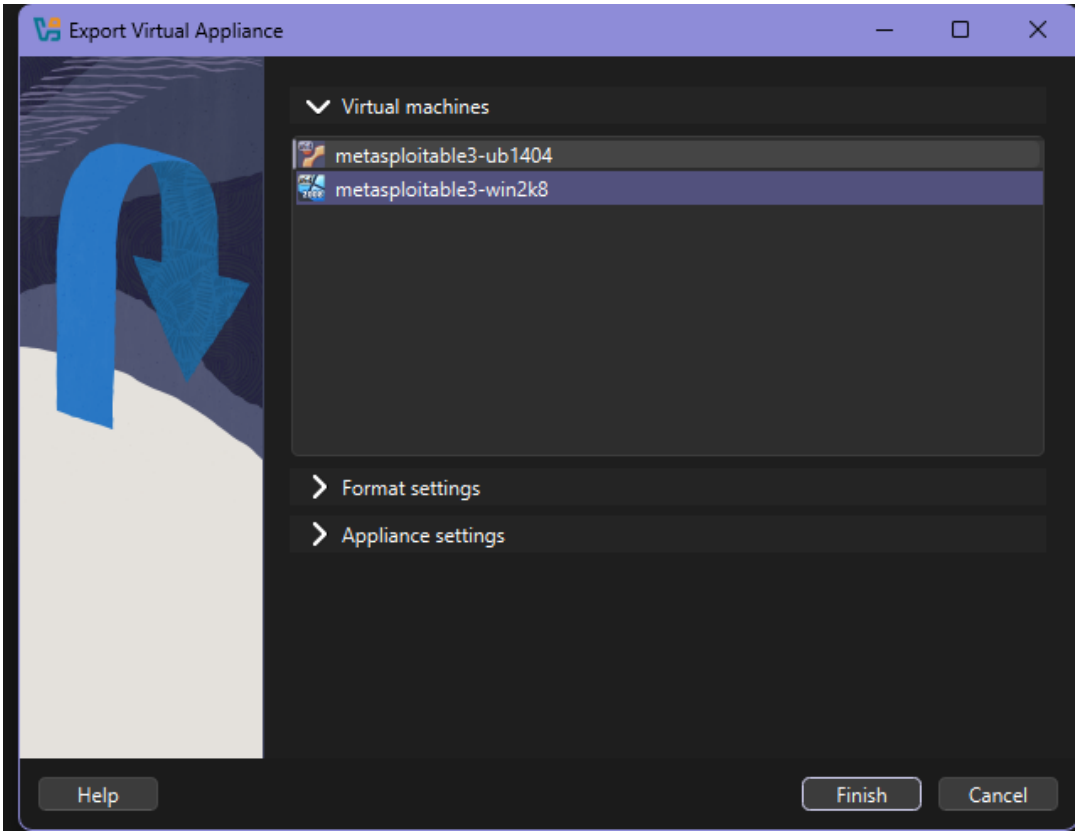
- i did not take screenshots of the process at the time however for documentation purposes the below images will depict the process. note: these images are taken after the fact for documentation in LabArchives and are not from the original export from virtualbox.



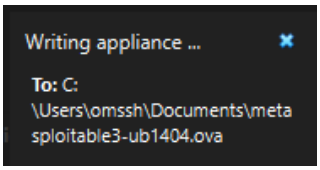
- this is the machine manager for virtualbox showing the VMs



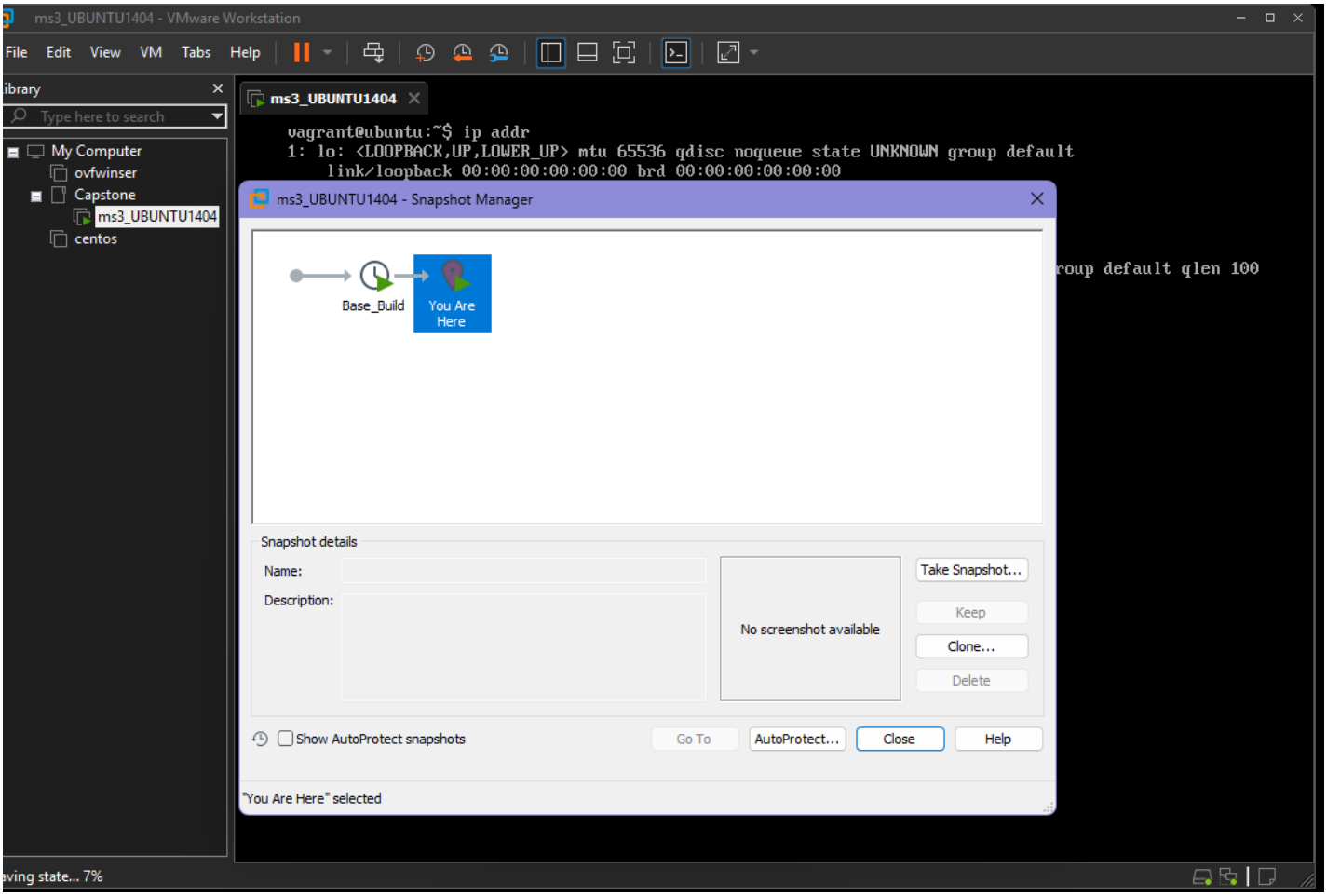
- Ctrl + E or Export Appliance opens a wizard as shown below on which the machine needed must be selected



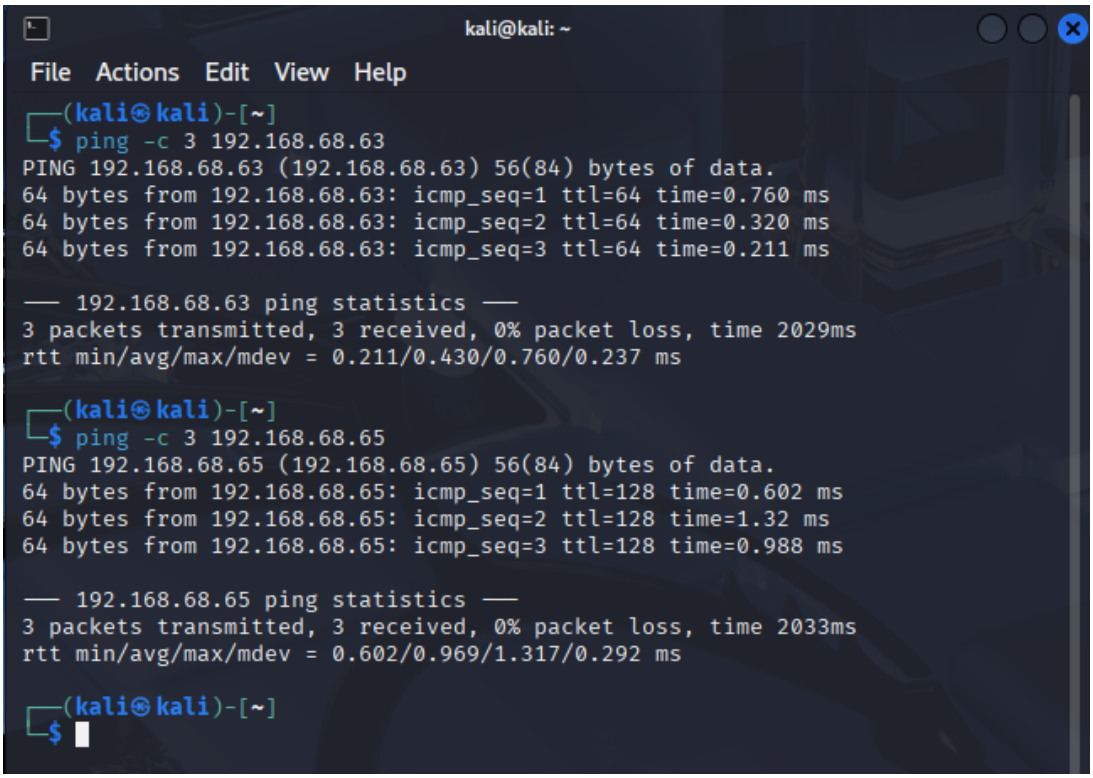
- upon clicking finish the appliance will be written and saved as a .ova file in your target folder

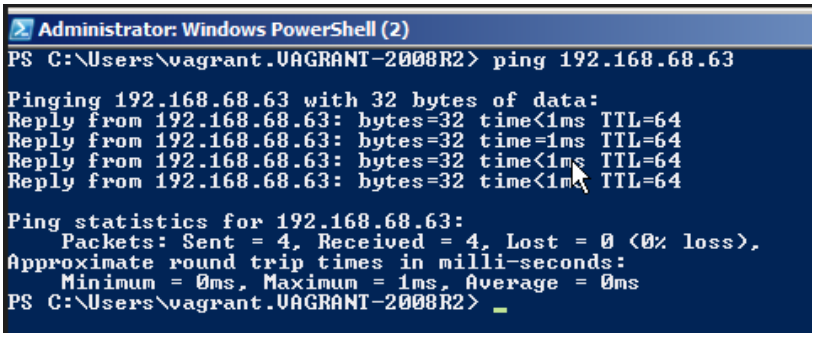


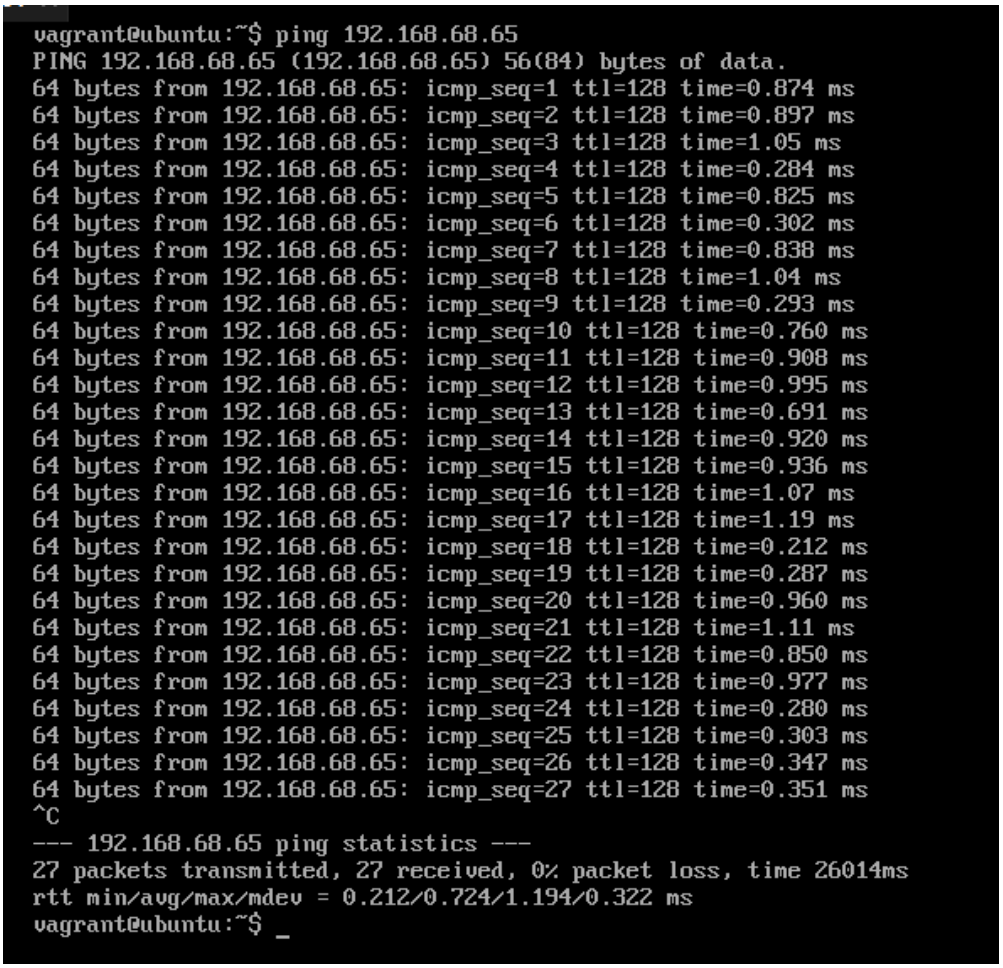
- after this all i did was open the ova in VMWare workstation and take a snapshot of the base build.



- the most difficult part of the installation process was getting windows server to work on vmware as i followed the previous process exactly and was still having issues. i attempted to remove the harddrive as that may be causing issues as well as the CD/DVD and even creating a new hard drive for the VM but no amount of changes altered the blackscreen and reboot loop error i was encountering. i even installed a ISO separately from web archives since windows no longer uploads their windows server however the SHA-1 integrity check failed to ensure the file is bit for bit identical to the original and therefore I had to remove the ISO as it had been tampered with or modified and was not worth the risk. ultimately i decided to just use virtualbox with the windows server VM as after some trial and error i determined the issue was actually the appliance creation process so it was just an easier solve to remain with virtualbox.
- Continuing with virutalbox however caused its own issues as after a ping check i realised it wasnt working from the ubuntu machine so i changed the nat to host only which didnt solve the issue, and then realised different virtual machines run off of different subnets which i had to figure out how to fix and then finally i changed the windows permission to allow incoming traffic from the ubuntu machine which finally resolved the issue. the following images are pings run from the kali (VMWare), ubuntu (VMWare) and windows (VirtualBox) machines.


kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ping -c 3 192.168.68.63
PING 192.168.68.63 (192.168.68.63) 56(84) bytes of data.
64 bytes from 192.168.68.63: icmp_seq=1 ttl=64 time=0.760 ms
64 bytes from 192.168.68.63: icmp_seq=2 ttl=64 time=0.320 ms
64 bytes from 192.168.68.63: icmp_seq=3 ttl=64 time=0.211 ms
— 192.168.68.63 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.211/0.430/0.760/0.237 ms
(kali㉿kali)-[~]
$ ping -c 3 192.168.68.65
PING 192.168.68.65 (192.168.68.65) 56(84) bytes of data.
64 bytes from 192.168.68.65: icmp_seq=1 ttl=128 time=0.602 ms
64 bytes from 192.168.68.65: icmp_seq=2 ttl=128 time=1.32 ms
64 bytes from 192.168.68.65: icmp_seq=3 ttl=128 time=0.988 ms
— 192.168.68.65 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.602/0.969/1.317/0.292 ms
(kali㉿kali)-[~]
$


Administrator: Windows PowerShell (2)
PS C:\Users\vagrant.VAGRANT-2008R2> ping 192.168.68.63
Pinging 192.168.68.63 with 32 bytes of data:
Reply from 192.168.68.63: bytes=32 time<1ms TTL=64
Reply from 192.168.68.63: bytes=32 time=1ms TTL=64
Reply from 192.168.68.63: bytes=32 time<1ms TTL=64
Reply from 192.168.68.63: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.68.63:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PS C:\Users\vagrant.VAGRANT-2008R2>



- After ensuring the machines can communicate with one another i stopped here to work on the team product proposal

Outcome:

- Installation: Windows on VirtualBox, Ubuntu and Kali on VMware, all able to communicate. Documented steps for the team. Recon ready for MITM and spoofing.
- Product Proposal: Assigned roles and attack roles on the product proposal and did the team description section. this took a long time due to communication issues.

Tags: `team, planning, ilc, install, tooling, documentation, team-support, verification`

Keywords: `virtualbox, vmware, PATH, packer-plugins, ova-export, sha1-check, packer-build, vagrant, tools-folder, ping-check`

Week 5

Om Shah (osshah@uts.edu.au) - Aug 29, 2025, 8:52 PM GMT+10

Om Shah (osshah@uts.edu.au) - Sep 05, 2025, 9:38 PM GMT+10

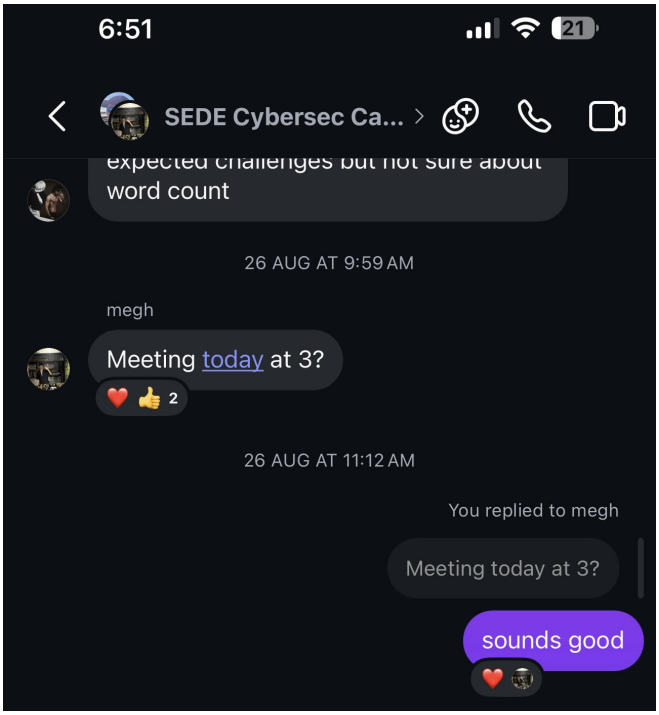
recon stealth quiet-recon mapping scanning light-scans target-selection spoofing arp Windows-SMB

Week 5 -

Date: Tue 26/08/2025 (team meeting ~3:00 pm).

Intention: Meet with the group to check progress, assign tasks, and confirm everyone's role in ongoing work.

Hours Spent: 3 hours(1 meeting before class + 2 hours in class)



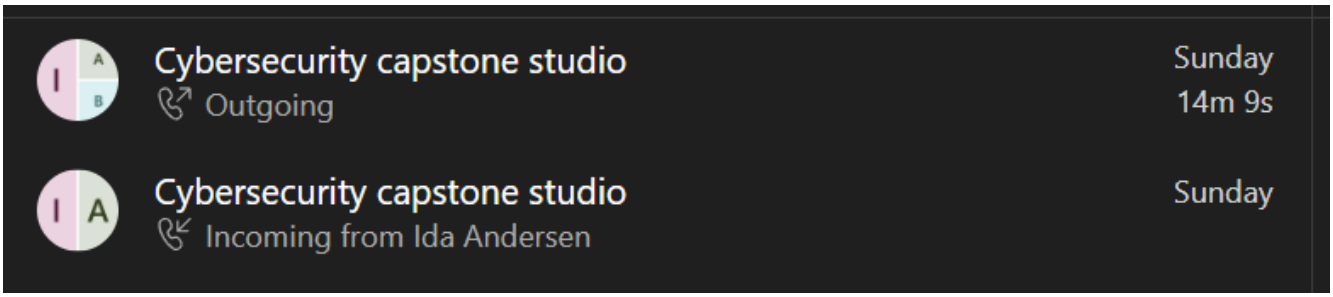
What we did:

- Discussed the upcoming roadshow presentation and prepped for the project to get a headstart
- Noted information provided by tutors in class
- discussed the roadshow with our product owner online

Outcome:

- Team gained clarity on what the roadshow needed to cover.
- Roles and early tasks were assigned, giving us a head start.
- Clearer path forward for week 6 preparation.

Date: Sun 31/08/2025 (teams call).



Intention: Polish off any rough edges/misunderstandings for the roadshow

Hours Spent: 0.4 hours (~15 mins)

What we did:

- Quick sync on the roadshow presentation content.
- Cleared up any misunderstandings from earlier discussions.
- Made final notes to align on who covers what during the presentation.

Outcome:

- Everyone on the same page before the roadshow.
- Presentation content and speaker roles clarified.
- Confidence that no major gaps were left going into the week.

Tags: group-effort, presentation, collaboration, planning
Keywords: teams-call, roadshow, prep, final-check

Date: Thu 28/08/2025

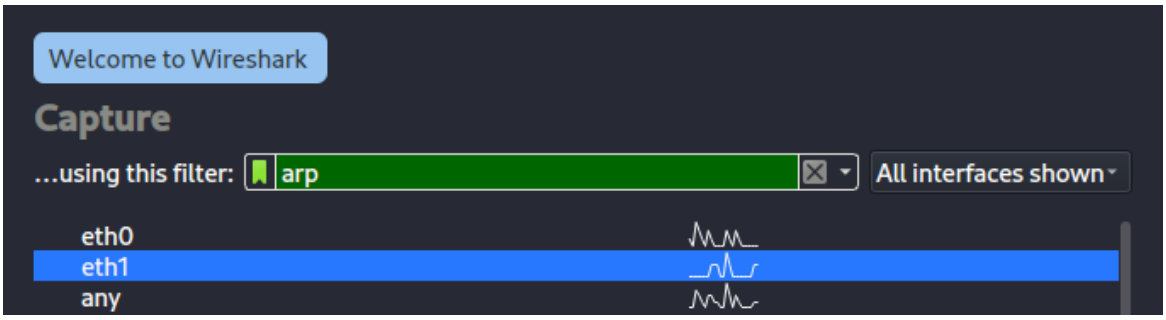
Intention: Stealthily map the subnet, identify valuable targets, and test a basic ARP spoof on Windows without moving into MITM.

Hours Spent: 7 Hours

RECON - Chosing a Target

so that there is no active probing of the network as a real attacker would i have chosen to go with a gentler approach as although an ARPsweep would be significantly quicker the entire network would see my linux machine asking who has 10.20.0.x which means that if the network is monitored that would give away that an attack is coming. instead using wireshark for passive discovery. the drawback here is that i will only get information if the machines communicate in some way which is unlikely unless i create simulated traffic from my other machines such as regular use.

first i will begin by starting a packet capture for arp on the linux machine



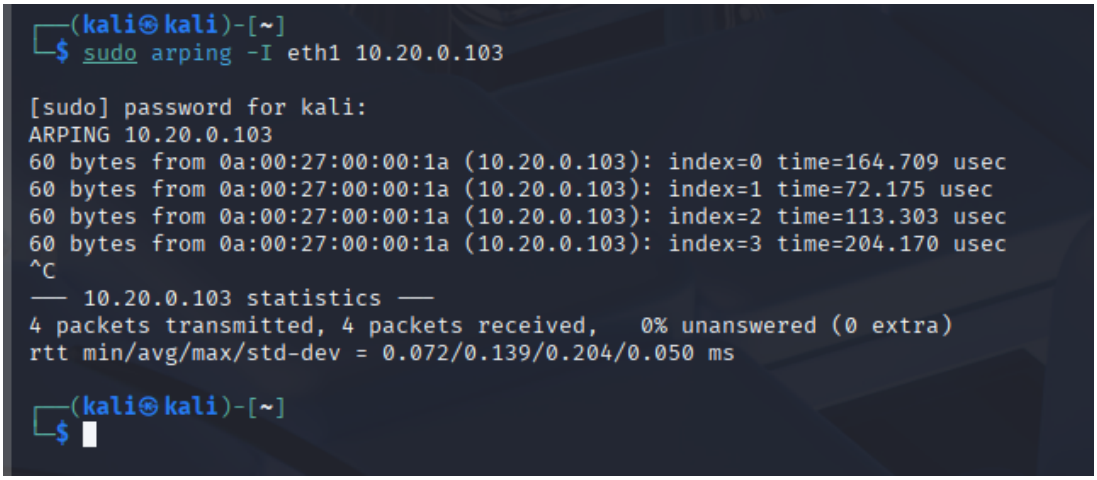
after running the packet capture and pinging the gateway 10.20.0.1 from both the ubuntu and windows VMs i found this output showing the IPs and Mac addresses of the machines on the networks. I am however unsure what machine owns .103 so i will do some checks to see if its another machine

7	2.551775966	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
8	3.552107410	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
9	3.814759056	0a:00:27:00:00:1a	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.103
10	4.551786723	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
11	4.597219362	0a:00:27:00:00:1a	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.103
12	5.550060154	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
13	5.596095772	0a:00:27:00:00:1a	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.103
14	6.547826184	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
15	7.544158942	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
16	8.541230616	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
17	9.537200165	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
18	10.532265521	PCSSystemtec_6f:c0:…	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.101
19	17.112490622	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
20	18.103924111	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
21	19.097674622	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
22	20.108534374	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
23	21.099419308	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
24	22.092477437	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
25	23.103793186	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
26	24.093865677	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102
27	25.096705267	VMware_09:7d:a4	Broadcast	ARP	60 Who has 10.20.0.1? Tell 10.20.0.102

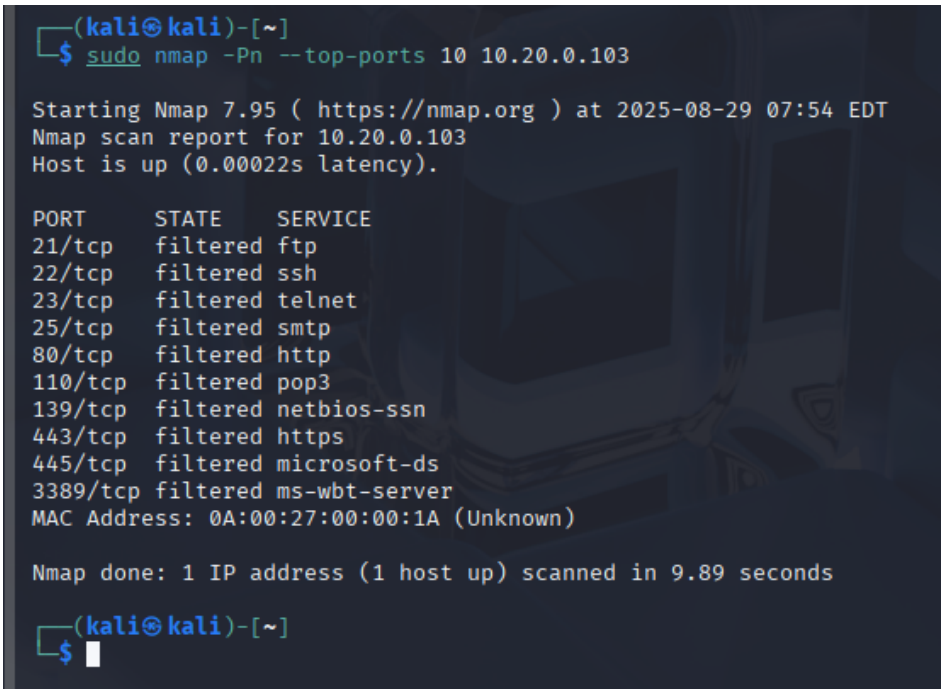
as i already know that .101 and .102 are windows and ubuntu i wont run test against them for this purpose for now as i want to rule out .103 first

---

Testing .103

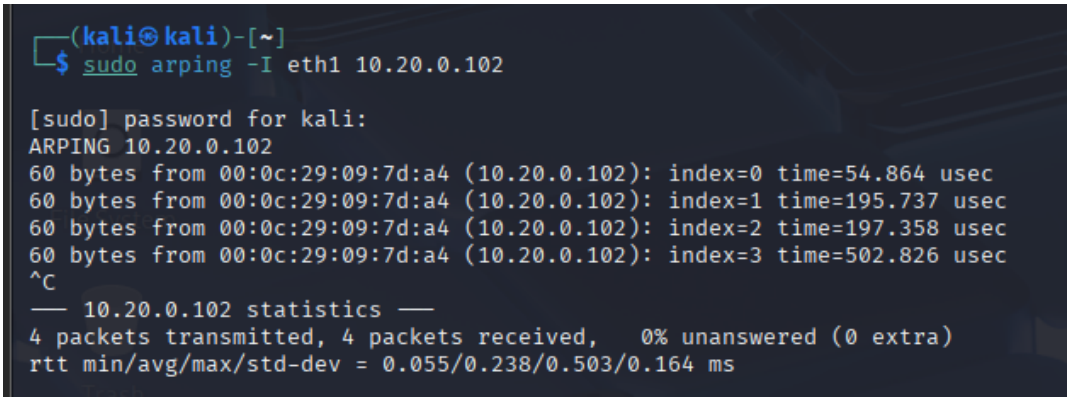


as the .103 has responded to the arp requests its safe to assume it is present at L2 and exists in the subnet with the MAC address : 0a:00:27:00:00:1a. i will now check if it exists at higher layers



I used Nmap with the -Pn flag to bypass host discovery and directly attempt connections on the ten most common ports. The scan returned results showing that the host was up, but all probed ports were in a filtered state. This means that packets were received but silently dropped, most likely by a firewall or restrictive configuration.This means its probably not the best target available on the network since lateral movement will be hard due to the lack of open services so as the "attacker" i would deprioritise this and move onto the other ones (.101.102)

Testing .102



10.20.0.102 responded to ARP consistently with MAC 00:0c:29:09:7d:a4. This confirms the host is alive at Layer 2 and present on the subnet.

```
┌──(kali㉿kali)-[~]
└─$ ping -c 3 10.20.0.102

PING 10.20.0.102 (10.20.0.102) 56(84) bytes of data.
64 bytes from 10.20.0.102: icmp_seq=1 ttl=64 time=0.912 ms
64 bytes from 10.20.0.102: icmp_seq=2 ttl=64 time=0.174 ms
64 bytes from 10.20.0.102: icmp_seq=3 ttl=64 time=0.218 ms

── 10.20.0.102 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.174/0.434/0.912/0.338 ms

┌──(kali㉿kali)-[~]
└─$
```

After verifying at Layer 2 that 10.20.0.102 responded to ARP, I tested for Layer 3 reachability using ICMP echo requests. The host responded consistently to all probes with TTL=64, which strongly suggests it is a Linux-based system. At this stage, the exact distribution is unknown (as the attacker), but the behavior confirms it is alive and reachable on the subnet.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn --top-ports 10 10.20.0.102

Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 08:03 EDT
Nmap scan report for 10.20.0.102
Host is up (0.00021s latency).

PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   open   ssh
23/tcp   closed telnet
25/tcp   closed smtp
80/tcp   open   http
110/tcp  closed pop3
139/tcp  closed netbios-ssn
443/tcp  closed https
445/tcp  closed microsoft-ds
3389/tcp closed ms-wbt-server
MAC Address: 00:0C:29:09:7D:A4 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 8.74 seconds
```

The host at 10.20.0.102 replied to ARP and ICMP normally. At L4, Nmap identified two open services: SSH (22/tcp) and HTTP (80/tcp). This confirms the host is not only alive but actively exposing both remote shell access and a web service. From an attacker's perspective, this makes .102 a valuable target with multiple attack surfaces unlike .103 which was filtered.

Testing .101

```
┌──(kali㉿kali)-[~]
└─$ sudo arping -I eth1 10.20.0.101

[sudo] password for kali:
ARPING 10.20.0.101
60 bytes from 08:00:27:6f:c0:e1 (10.20.0.101): index=0 time=361.075 usec
60 bytes from 08:00:27:6f:c0:e1 (10.20.0.101): index=1 time=270.466 usec
60 bytes from 08:00:27:6f:c0:e1 (10.20.0.101): index=2 time=299.958 usec
60 bytes from 08:00:27:6f:c0:e1 (10.20.0.101): index=3 time=281.522 usec
^C
— 10.20.0.101 statistics —
4 packets transmitted, 4 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.270/0.303/0.361/0.035 ms

┌──(kali㉿kali)-[~]
└─$
```

10.20.0.101 responded to ARP with MAC 08:00:27:6f:c0:e1. This confirms the host is active on the subnet and reachable at Layer 2.

```
┌──(kali㉿kali)-[~]
└─$ ping -c 3 10.20.0.101

PING 10.20.0.101 (10.20.0.101) 56(84) bytes of data.
64 bytes from 10.20.0.101: icmp_seq=1 ttl=128 time=0.688 ms
64 bytes from 10.20.0.101: icmp_seq=2 ttl=128 time=0.435 ms
64 bytes from 10.20.0.101: icmp_seq=3 ttl=128 time=0.458 ms

— 10.20.0.101 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.435/0.527/0.688/0.114 ms

┌──(kali㉿kali)-[~]
└─$
```

10.20.0.101 responded to ICMP echo requests with TTL=128, which is typical of Windows systems. This confirms the host is alive and reachable at Layer 3, and strongly suggests it is a Windows machine.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap --top-ports 10 10.20.0.101

Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 08:14 EDT
Nmap scan report for 10.20.0.101
Host is up (0.00030s latency).

PORT     STATE    SERVICE
21/tcp   filtered ftp
22/tcp   filtered ssh
23/tcp   filtered telnet
25/tcp   filtered smtp
80/tcp   filtered http
110/tcp  filtered pop3
139/tcp  filtered netbios-ssn
443/tcp  filtered https
445/tcp  open     microsoft-ds
3389/tcp filtered ms-wbt-server
MAC Address: 08:00:27:6F:C0:E1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 10.11 seconds
```

The host at 10.20.0.101 shows a single open port: 445/tcp (Microsoft-DS, SMB). All other common ports appeared filtered. This is consistent with a Windows system, as SMB is commonly exposed. From an attacker's perspective, the presence of SMB is significant because it has historically been a major attack vector for credential theft, relay attacks,

and exploitation of vulnerabilities like EternalBlue. --> https://www.avast.com/c-eternalblue as well as this vulnerabilities such as CVE-2020-0796 (SMBGhost), CVE-2020-1206 (SMBleed), MS08-068 are also exploitable on SMB

Target

Recon identified three active hosts on the subnet. 10.20.0.103 was confirmed at Layer 2 but remained silent at Layer 3 and exposed no services at Layer 4, making it a low-value target. 10.20.0.102 responded at all layers and revealed two open services, SSH (22/tcp) and HTTP (80/tcp), which present concrete avenues for exploitation. 10.20.0.101 was confirmed as a Windows host with SMB (445/tcp) open, a historically vulnerable service and therefore high value for further attacks. Based on this, both 10.20.0.102 and 10.20.0.101 are prioritized as primary targets for the next attack phase.

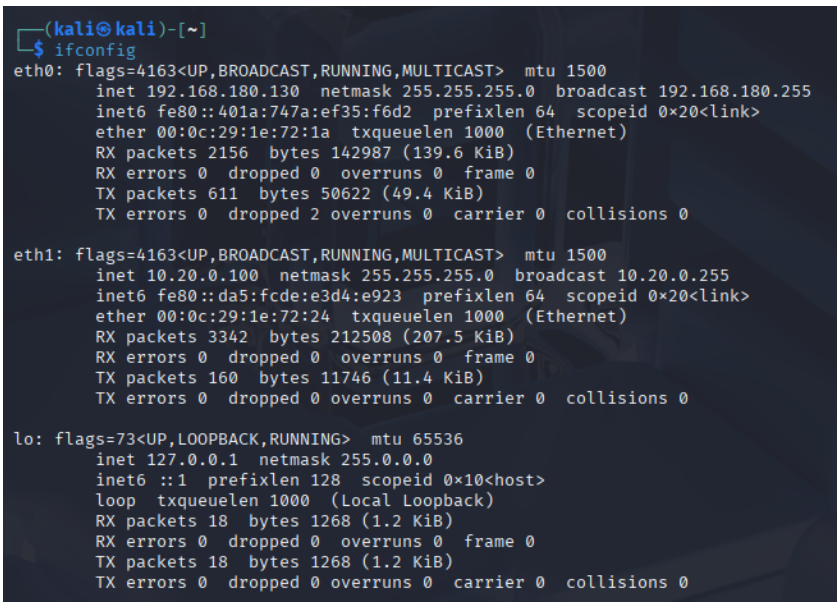
29/08/2025

Spoofing Windows

ARP Spoofing

I have chosen to target 10.20.0.101 (Windows) first. The host exposes SMB (445/tcp), a service that has historically been one of the most vulnerable and widely exploited on Windows systems. From an attacker's perspective, this makes it a high-value target. It is also easier to test whether SMB is still exploitable, as the attack primarily depends on whether the service has been patched or hardened.

first i must take note of the current kali details to ensure that i have record of the linux machine's details to cross reference later on



now to set the foundation i must run the arp -a command to see the arp table

```
C:\Users\vagrant.VAGRANT-2008R2>arp -a

Interface: 10.20.0.101 --- 0xb
  Internet Address      Physical Address      Type
  10.20.0.100           00-0c-29-1e-72-24     dynamic
  10.20.0.102           00-0c-29-09-7d-a4     dynamic
  10.20.0.255           ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\vagrant.VAGRANT-2008R2>S
```

after taking note of this i have run the spoofing command

```
┌──(kali㉿kali)-[~]
└─$ sudo arpspoof -i eth1 -t 10.20.0.101 10.20.0.102

[sudo] password for kali:
0:c:29:1e:72:24 8:0:27:6f:c0:e1 0806 42: arp reply 10.20.0.102 is-at 0:c:29:1e:72:24
0:c:29:1e:72:24 8:0:27:6f:c0:e1 0806 42: arp reply 10.20.0.102 is-at 0:c:29:1e:72:24
0:c:29:1e:72:24 8:0:27:6f:c0:e1 0806 42: arp reply 10.20.0.102 is-at 0:c:29:1e:72:24
```

this command sends out poison packets spoofing the attacker mac address to that of the ubuntu machine and we can see that upon running the arp -a command again on the windows machine

```
Interface: 10.20.0.101 --- 0xb
  Internet Address      Physical Address      Type
  10.20.0.100           00-0c-29-1e-72-24     dynamic
  10.20.0.102           00-0c-29-1e-72-24     dynamic
  10.20.0.255           ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\vagrant.VAGRANT-2008R2>S
```

as can be seen here the .102 MAC address is that of the .100 kali machine's

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.180.130  netmask 255.255.255.0  broadcast 192.168.180.255
        inet6 fe80::401a:747a:ef35:f6d2  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:1e:72:1a  txqueuelen 1000  (Ethernet)
        RX packets 2156  bytes 142987 (139.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 611  bytes 50622 (49.4 KiB)
        TX errors 0  dropped 2 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.20.0.100  netmask 255.255.255.0  broadcast 10.20.0.255
        inet6 fe80::da5:fcde:e3d4:e923  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:1e:72:24  txqueuelen 1000  (Ethernet)
        RX packets 3342  bytes 212508 (207.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 11746 (11.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 18  bytes 1268 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1268 (1.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

from the previous screenshot and the one documented we can confirm this kali MAC address was successfully used

30/08/2025

MAC Spoofing

Now that i have finished ARP spoofing its time to move onto other spoofing attacks, these can occur on different layers of the network. ARP cache poisoning takes place on the 2nd Layer (Data Link) by decieving only one victims ARP table, MAC spoofing can decieve the entire network by appearing as another hostdespite also operating on Layer 2, this is done by changing the attacker's hardware address rather than sending poisoned pakets. Both attacks have their value however although MAC spoofing can decieve an entire network and is stronger when trying to bypass MAC based access controls, impersonating a trusted device on a network or blend in as pre-authorised traffic it has its own drawbacks. Two major ones i found were that it can result in MAC conflicts if the spoofed device is also online which can make hosts and switches drop connections or become altert to a spoofed device, furthermore NAC and port secure switches can also make it even easier to detect the attacker machine. so both ARP and MAC have drawbacks which is why as the attacker its better to decide which is more suited to my situation

since the hardware address will change we must make note of the current one so that it can be restored when needed the previous ifconfig command shows [00:0c:29:1e:72:24]

from the recon stage we found the arp replies on wireshark showing the MAC address of the .102 machine however if there was no reply the attacker can also run the following and .102 will reply with its mac address

```
┌──(kali㉿kali)-[~]
└─$ sudo arping -I eth1 10.20.0.102

[sudo] password for kali:
ARPING 10.20.0.102
60 bytes from 00:0c:29:09:7d:a4 (10.20.0.102): index=0 time=306.132 usec
60 bytes from 00:0c:29:09:7d:a4 (10.20.0.102): index=1 time=241.877 usec
60 bytes from 00:0c:29:09:7d:a4 (10.20.0.102): index=2 time=171.850 usec
^X^C
── 10.20.0.102 statistics ──
3 packets transmitted, 3 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.172/0.240/0.306/0.055 ms

┌──(kali㉿kali)-[~]
└─$
```

here we can see the mac address for the .102 (ubuntu) machine is [00:0c:29:09:7d:a4]

04/09/2025

I had to stop here to finish off another assignment and prepare for a test and it will continue on in week 6 (today) since i was busy and i successfully did MAC spoofing but the ms3 licensing for windows server expired and i had to rebuild the machine from scratch but i will explain more in the week 6 notebook.

Week 6

Om Shah (osshah@uts.edu.au) - Sep 18, 2025, 4:59 PM GMT+10

group-effort presentation collaboration rebuild licensing virtualbox vmware-troubleshooting drivers display-fix spoofing arp validation

Week 6 -

Date: Tue 02/09/2025 (roadshow presentation)

Intention: Deliver the roadshow presentation with the team, confirm details with Megh, and present smoothly.

Hours Spent: ~3 hours (prep + class presentation)

What we did:

- Finalised the last details of the presentation with Megh.
- Double-checked slides and flow to make sure the content was consistent.
- Presented as a group during the roadshow session.
- Gave thoughtful constructive criticism to help improve other students' presentations

Outcome:

- Presentation delivered without issues.
- Collaboration with Megh ensured content was polished.
- Hopefully the group came across aligned and confident and the groundwork for the week 12 presentation is set well or else we can update with the feedback provided.

Tags: group-effort, presentation, collaboration
Keywords: roadshow, presentation, teamwork, polish

Date: Thu 04/09/2025 – Fri 05/09/2025

Intention: Rebuild Windows Server after license expiry, troubleshoot black screen and VMware Tools issues, and re-do MAC spoofing on the new VM for proof of concept (spoofing only, no MITM).

Hours Spent: 7 hours (10 mins rebuild, 2 hours black screen fix, 3.5 hours VMware Tools troubleshooting, 20 mins driver checks, 1 hour MAC spoofing redo)

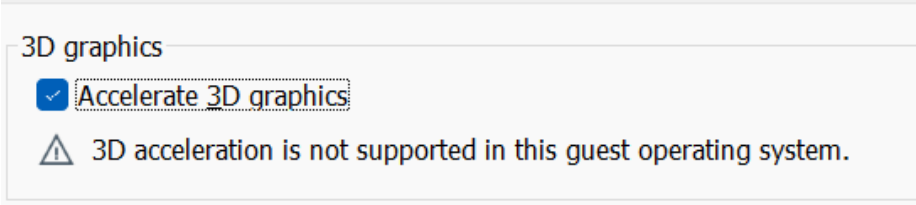
04/09/2025

Licensing Issues (around 10 minutes)

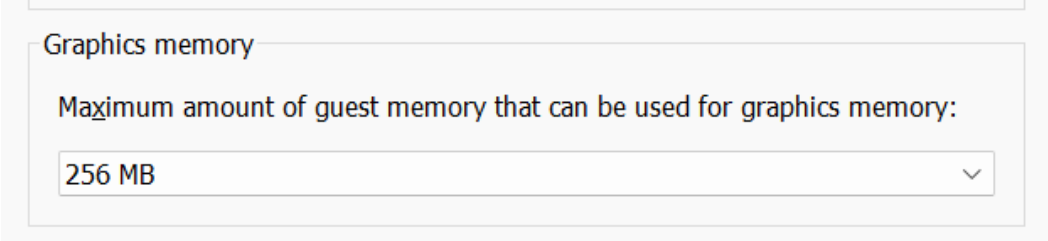
In week 5 I did MAC spoofing but had to cut it short due to other assignments but upon starting up the VMs again the ms3 Windows Server notified about licensing period elapsing and upon clicking ignore it showed that "not genuine software" indication which might interfere with the services running so i just rebuilt the machine as that seemed easier.

How I beat the black screen (2 hours)

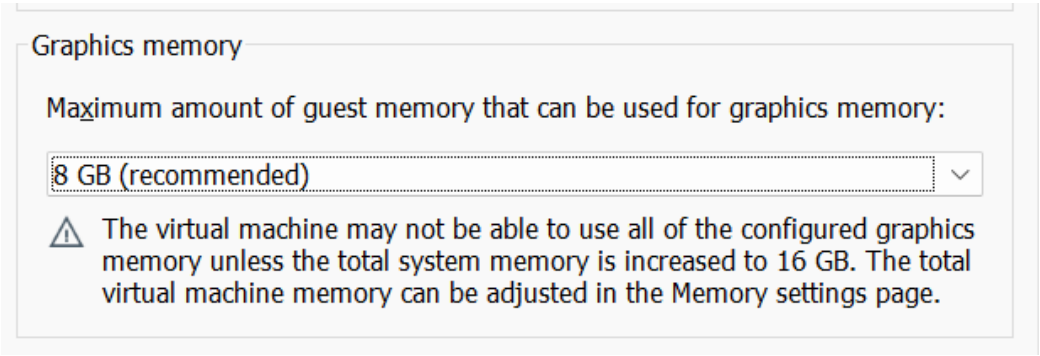
Upon installing i was encountering the black screen issue mentioned in week 4 under PDJ/Things that failed/Things that Failed (keywords vmware-loop), the black-loop screen is a common issue as mentioned before but i hadnt seen it in virtualbox until now so i tried out a few things to come to a solution. Firstly i checked if i had allocated enough RAM (i had given it 16GB) which should be enough to run this machine and tabs like user interface or storage which might be having issues initialising but the solution was video memory, apparently the default allocated isnt considered a valid configuration so i changed that to 53MB and that booted the VM as expected. so i tried to export the appliance as i did in week 4 and opened in VMWare only to realise that it doesnt have the display settings directly adjustable so i tried looking into another way as to how i might be able to change display settings. Then in the hardware tab under display it showed no way to make changes



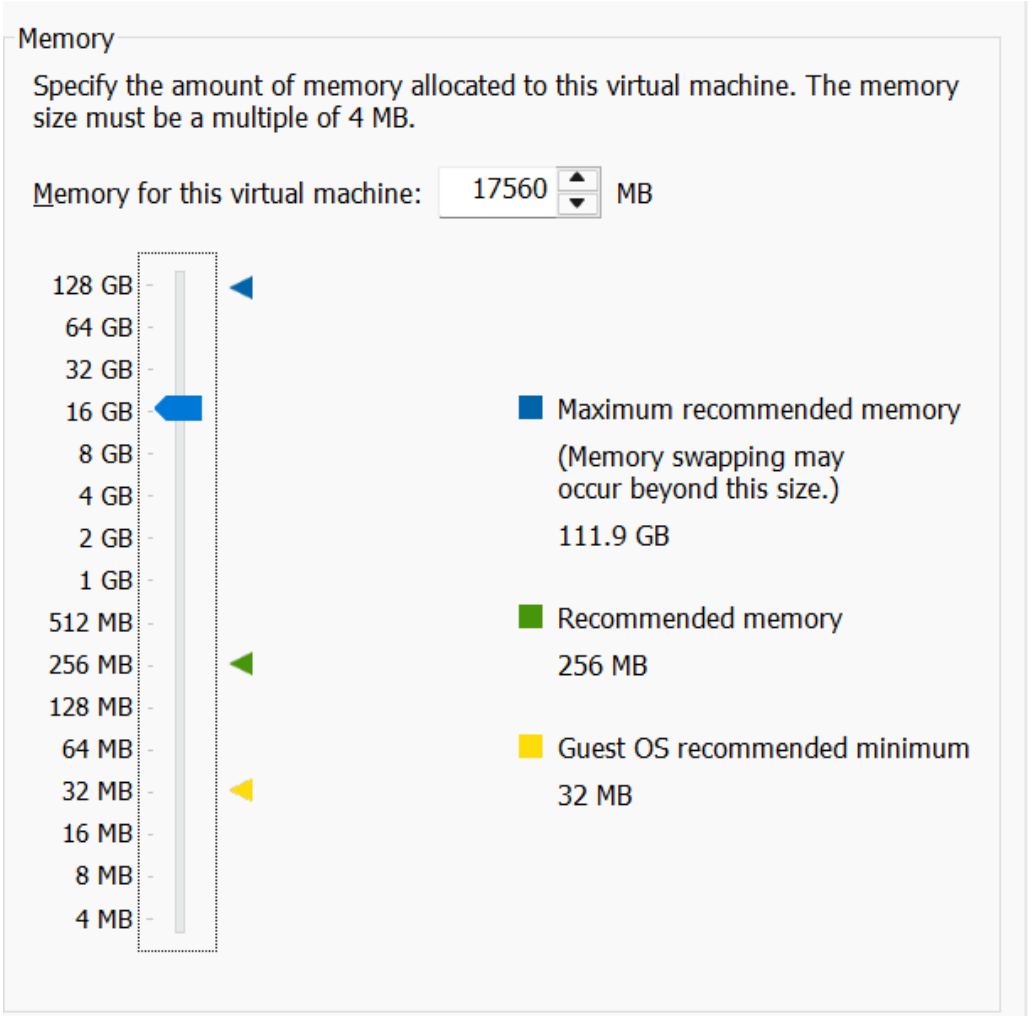
and



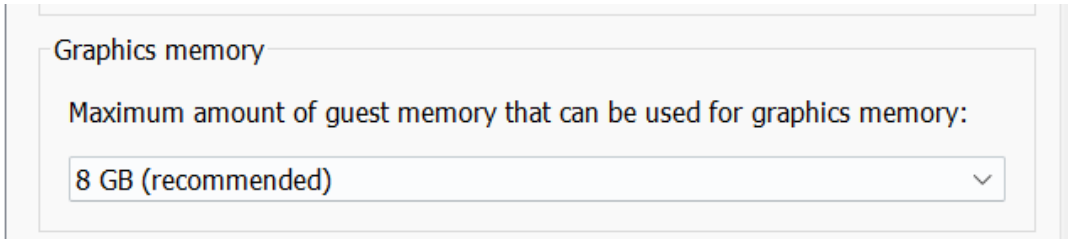
i changed this amount to the recommended 8GB



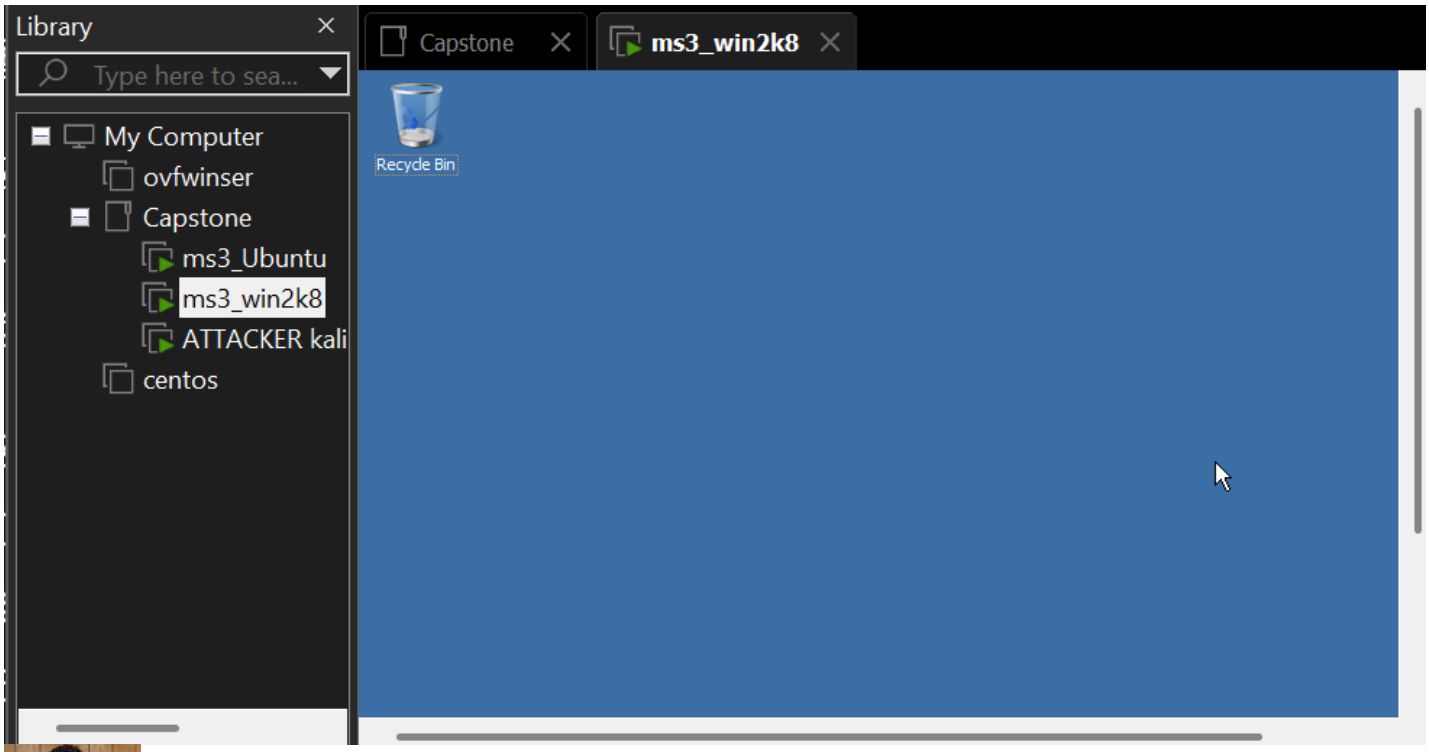
the had to allocate more resources



and that made the warning message go away

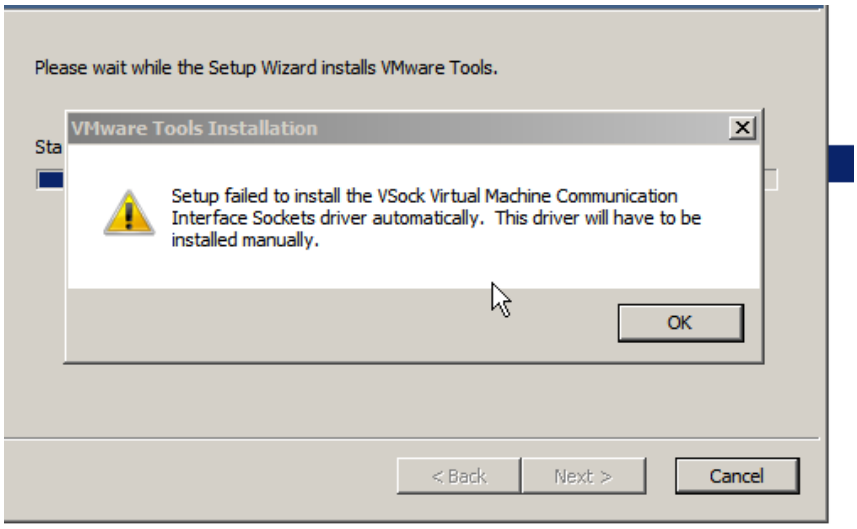


and VMWare can now run windows

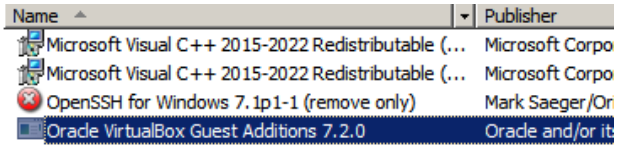


Tools Installation Issues (3.5 hours)

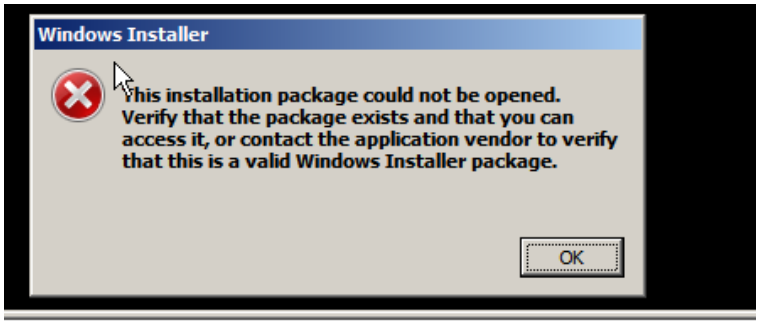
When I tried installing VMware Tools on the Windows Server 2008 R2 VM (imported from VirtualBox), I ran into multiple issues. At first, the old VirtualBox Guest Additions conflicted with VMware Tools, so I removed them. Even after that, the Tools installer kept failing with driver errors (VSock/VMCI, Memory Control/balloon, and Physical Disk Helper), which caused the setup to roll back every time. I then tried running the MSI directly from the Temp folder, but that failed with error 2711 because the extracted package was incomplete. When I looked at the mounted ISO, it turned out to be a “lite” version that only included disk (pvscsi) drivers and not the important SVGA (graphics) or VMXNet3 (network) drivers I needed. Combined with the quirks of an older OS (2008 R2) and unsigned driver prompts, the whole process stalled. In the end, the main blockers were using the wrong ISO and dealing with leftover VirtualBox hardware drivers.



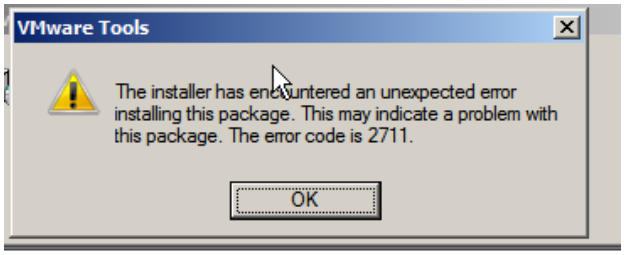
The installer repeatedly failed to load the VSock/VMCI and Memory Control (balloon) drivers, throwing errors like this



The VM still had Oracle Guest Additions installed, which clashed with VMware's drivers and contributed to repeated rollbacks.



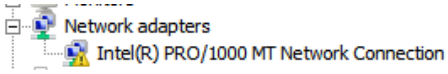
Running the MSI directly from the Temp folder resulted in "This installation package could not be opened"



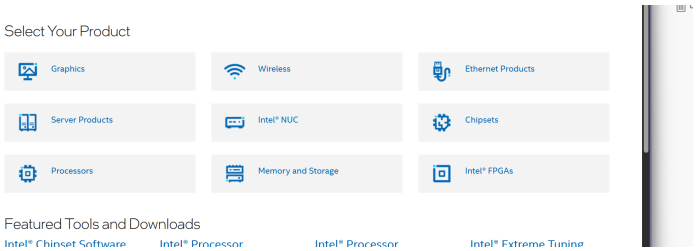
and when forced via msiexec, the process ended with error 2711 (feature missing in package).

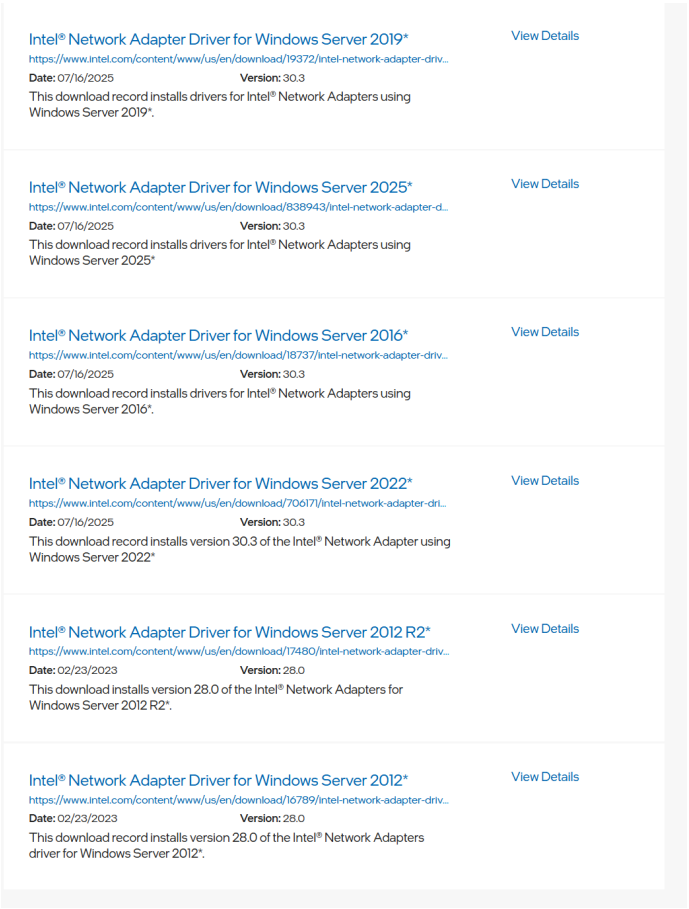
Driver Issues (around 20 minutes)

Since VMware didnt have the tools installed the drivers were having issues connecting to the internet



and the best solve was to install the tools but that wasnt an option so i tried to install a driver directly onto my host and move it to the VM but the win server 2008 drivers weren't available





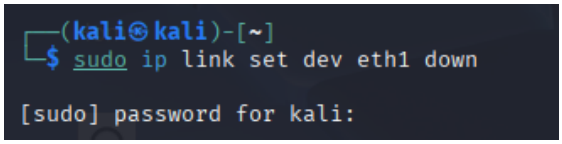
so sadly i had to just move back to virtualbox as everything was already working there and it wasnt worth the extra time trying to trouble shoot.

MAC Spoofing (1 hour)

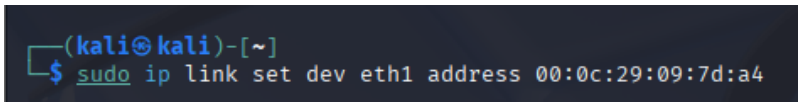
Since i already did a little bit about MAC Spoofing i already had an idea as to what to do but in the interest of continuity im going to replicate the process with the new VM since it has a different IP address of .105 rather than .101

First i must find the baseline and record it to ensure the original MAC address can be recovered

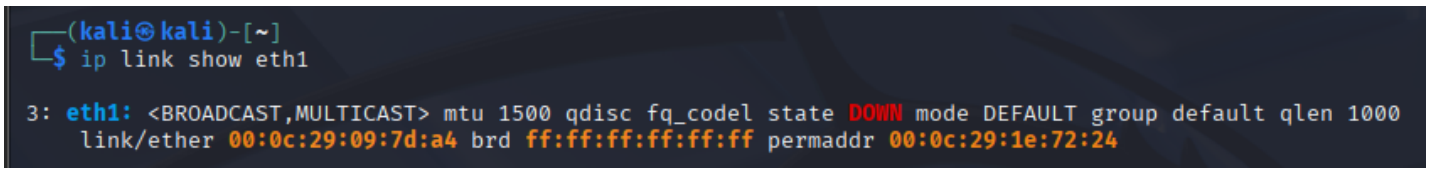
Then I set eth1 to down to allow for the spoof



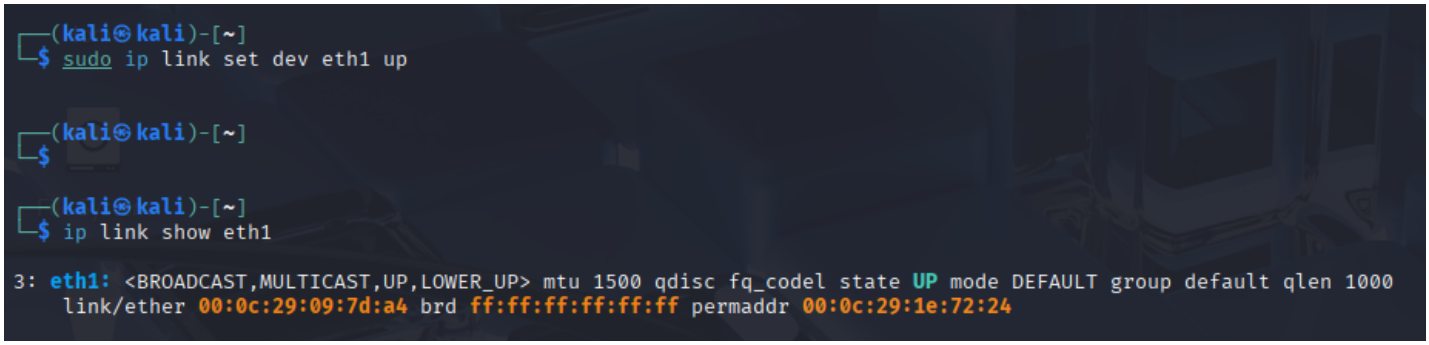
now for the MAC spoof i run



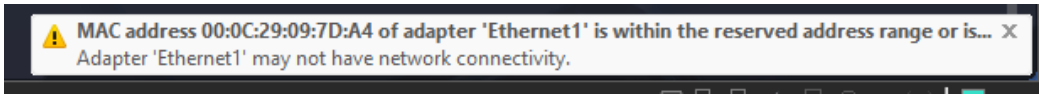
running the show eth1 command again shows the state as down and the mac address will be changed



and finally setting eth1 to up will change the state with the new MAC address



upon setting it to up this warning is given indicating another machine is claiming that same MAC address



05/09/2025

When two machines use the same MAC address, the network gets confused because it doesn't know which one is the "real" owner. As a result, some packets end up going to the wrong machine and get thrown away, which is why the ping from Windows to Kali timed out.

```
Pinging 10.20.0.100 with 32 bytes of data:
Request timed out.

Ping statistics for 10.20.0.100:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

C:\Users\vagrant>arp -d 10.20.0.100

C:\Users\vagrant>S
```

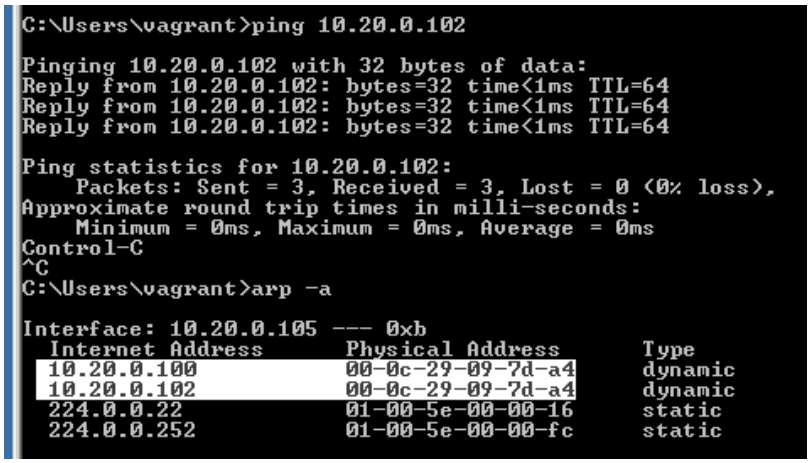
Since Windows never got a reply, it didn't bother saving an entry for kali in its arp table. To fix this and prove the spoof was working, I had kali send out a special kind of arp message called a gratuitous arp. This is basically kali shouting to everyone on the network, saying 10.20.0.100 is at this mac address. Even though it's unsolicited, windows accepts it and updates its arp table to show kali's ip with the spoofed MAC. That gave me visible proof that mac spoofing worked, even though two machines were alive on the same network with the same identity.

Here is the updated arp table with the linux ip corresponding the ubuntu machine's MAC to the linux IP.

```
C:\Users\vagrant>arp -a

Interface: 10.20.0.105 --- 0xb
  Internet Address      Physical Address      Type
  10.20.0.100           00-0c-29-09-7d-a4     dynamic
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static

C:\Users\vagrant>_
```

notice there is no entry for ubuntu this can be fixed by making the windows and ubuntu machine communicate. In the real world there is no incentive to make the two machine's communicate as the spoofing job ends when the arp table is poisoned but for demonstration purposes i could just ping the ubuntu machine from windows or for argument's sake if i wanted to update window's arp table with ubutnu's details from linux i could just force the source IP from ubuntu and perform the same command as before like telling arping to send an unsolicited or gratuitous arp reply by pretending my source is .102 (ubuntu). this would tell windows .102 is at (kali's MAC) which is actually ubuntu's MAC.

but to keep everything realistic im just going to ping ubuntu from windows



and now windows things the MAC address is the same for the two machines.

Week 7

Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 1:31 PM GMT+10

Week 7 spoofing DNS Spoofing ARP Spoofing DHCP Spoofing Penterrant Lab Kali Yersinia dnsmasq

Date: Wed 17/09/2025

Intention:
Test and implement spoofing attacks (DNS, DNS+ARP, and DHCP) in the lab environment as preparation for later MITM demonstrations.

Hours Spent (rough): ~7 hrs total

- DNS Spoofing setup and testing - 2 hrs
- DNS + ARP Spoofing attempt - 2 hrs
- DHCP Spoofing attempt - 3 hrs

What I did:

DNS Spoof

before we can start dns spoofing, we need to make sure that kali's lab interface really has the ip address we expect.

we need to check this because:

- the fake dns server must "bind" to a valid ip address.
- the victims (windows + ubuntu) will later be pointed at this ip as their dns server.
- if the ip isn't what we think (10.20.0.100/24), the attack won't work.

```
┌──(kali㉿kali)-[~]
└─$ ip addr show eth1

3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:1e:72:24 brd ff:ff:ff:ff:ff:ff
    inet 10.20.0.100/24 brd 10.20.0.255 scope global dynamic noprefixroute eth1
       valid_lft 5182563sec preferred_lft 5182563sec
    inet6 fe80::da5:fcde:e3d4:e923/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$
```

We're only doing dns spoofing right now, not a full man-in-the-middle. if ip forwarding is enabled, kali might start routing packets between victims and the gateway, which would turn this into an MITM scenario. We want to keep these two attacks separate and clean. To achieve this we need to disable IP forwarding

https://nordvpn.com/blog/ip-forwarding-linux/#:~:text=13%20min%20read-,How%20to%20disable%20IP%20forwarding%20on%20Linux,sudo%20sysctl%20%2Dp%E2%80%9D%20command.

```
┌──(kali㉿kali)-[~]
└─$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

before we can run our fake dns server, we need to make sure the dns port (53) on kali isn't already in use. dns servers always run on port 53, and if another service is already bound to it, our spoofing server won't even start.

```
┌──(kali㉿kali)-[~]
└─$ sudo ss -lntup
[sudo] password for kali:
Netid       State        Recv-Q        Send-Q              Local Address:Port              Peer Address:Port       Process
```

the ss tool lists sockets and lntup makes it show listening ports but running the command yeilded no results so i assume there are none listening but to be sure i also tried to grep port 53

```
┌──(kali㉿kali)-[~]
└─$ sudo ss -lntup | grep :53

┌──(kali㉿kali)-[~]
```

since this provided no output it means that it is not being used at the moment

to perform DNS spoofing, I need a DNS server running on Kali that will answer queries from the victims. I'll use dnsmasq because it's lightweight and easy to configure. I'll set it so that no matter what domain name is requested, it always responds with Kali's IP address (10.20.0.100).

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.2 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [51.8 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [120 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [326 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [200 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [911 kB]
Get:8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [11.3 kB]
Get:9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents (deb) [28.4 kB]
Fetched 74.7 MB in 15s (4,966 kB/s)
979 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

i refresh the package list

```
Installing:
  dnsmasq

Suggested packages:
  resolvconf

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 979
  Download size: 69.5 kB
  Space needed: 159 kB / 63.7 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 dnsmasq all 2.91-1 [69.5 kB]
Fetched 69.5 kB in 9s (7,875 B/s)
Selecting previously unselected package dnsmasq.
(Reading database ... 412840 files and directories currently installed.)
Preparing to unpack .../dnsmasq_2.91-1_all.deb ...
Unpacking dnsmasq (2.91-1) ...
Setting up dnsmasq (2.91-1) ...
update-rc.d: We have no instructions for the dnsmasq init script.
update-rc.d: It looks like a network service, we disable it.
dnsmasq.service is a disabled or a static unit, not starting it.
```

and install dnsmasq, now i need to tell it to only bind to kali's lab ip, listen on port 53 and respond to every dns query with kali's IP, this way no matter what the victim will always get my IP. to do this dnsmasq needs a small config file that tells it how to behave. instead of editing the big system file in /etc, I'll just make a small temporary one in /tmp so it's easy to throw away after the demo.

I open a new file called dnsmasq.conf under /tmp like this:

```
┌──(kali㉿kali)-[/tmp]
└─$ sudo nano /dnsmasq.conf
```

and configure it using these

```
File  Actions  Edit  View  Help
  GNU nano 8.4                                    /dnsmasq.conf *
port=53
listen-address=10.20.0.100
bind-interfaces
address=/#/10.20.0.100
```

the next step is to actually start dnsmasq so it begins answering DNS requests.

```
┌──(kali㉿kali)-[/tmp]
└─$ sudo dnsmasq -C /dnsmasq.conf

┌──(kali㉿kali)-[/tmp]
└─$
```

so it has began running in the background and that means on port 53 it is ready to respond to queries. whenever windows or ubuntu ask for a domain dnsmasq will respond with the attacker IP. to check if it works before interacting with the victims i need to first check with the kali machine

```
┌──(kali㉿kali)-[/tmp]
└─$ nslookup www.google.com 10.20.0.100

Server:         10.20.0.100
Address:        10.20.0.100#53

Name:   www.google.com
Address: 10.20.0.100
Name:   www.google.com
Address: 2404:6800:4006:80a::2004

┌──(kali㉿kali)-[/tmp]
└─$
```

this successfully shows that the dns spoof has worked however to make this realistic i would have to manually change the DNS server on the victim machines which is impractical so i think a better option is to merge two different spoofs to make this spoof successful

---

DNS + ARP Spoofing

To make Windows use my fake DNS without touching its settings, I need to trick it into thinking I am the default gateway. I do this with ARP spoofing: sending fake ARP replies that map the gateway IP to my MAC address.

first i need to find the default gateway of windows

```
Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . : localdomain
   IPv6 Address. . . . . . . . . . . : fd17:625c:f037:3:8020:d3e9:ea4c:770d
   Link-local IPv6 Address . . . . . : fe80::8020:d3e9:ea4c:770d%18
   IPv4 Address. . . . . . . . . . . : 10.0.3.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::2%18
                                       10.0.3.2

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::59dd:5b6f:a781:e36f%11
   IPv4 Address. . . . . . . . . . . : 10.20.0.106
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Tunnel adapter isatap.localdomain:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : localdomain

C:\Users\vagrant>S_
```

according to this its default gateway is 10.0.3.2 on the internet side

```
┌──(kali㉿kali)-[/tmp]
└─$ sudo arpspoof -i eth1 -t 10.20.0.106 10.0.3.2

[sudo] password for kali:
0:c:29:1e:72:24 8:0:27:f:ca:a4 0806 42: arp reply 10.0.3.2 is-at 0:c
:29:1e:72:24
```

i ran the arp spoof against 10.0.3.2

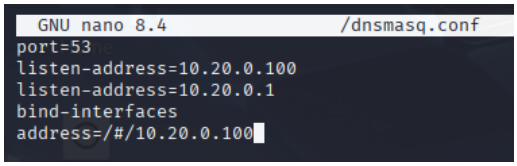
```
C:\Users\vagrant>nslookup www.google.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  10.20.0.1

DNS request timed out.
    timeout was 2 seconds.
```

however the nslookup failed here so i tried against 10.20.0.1 but that didnt work either

```
┌──(kali㉿kali)-[/tmp]
└─$ sudo arpspoof -i eth1 -t 10.20.0.106 10.20.0.1

0:c:29:1e:72:24 8:0:27:f:ca:a4 0806 42: arp reply 10.20.0.1 is-at 0:
c:29:1e:72:24
0:c:29:1e:72:24 8:0:27:f:ca:a4 0806 42: arp reply 10.20.0.1 is-at 0:
c:29:1e:72:24
0:c:29:1e:72:24 8:0:27:f:ca:a4 0806 42: arp reply 10.20.0.1 is-at 0:
c:29:1e:72:24
0:c:29:1e:72:24 8:0:27:f:ca:a4 0806 42: arp reply 10.20.0.1 is-at 0:
```

```
C:\Users\vagrant>nslookup www.google.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  10.20.0.1

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
*** Request to UnKnown timed-out

C:\Users\vagrant>_
```

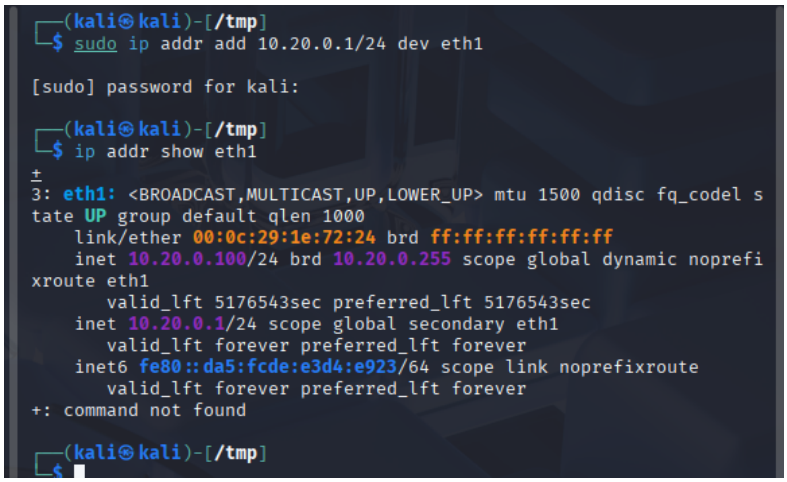
this is likely because dnsmasq is listening only on 10.20.0.100 not 10.20.0.1 so i am going to re-try this using 10.20.0.1 as a listen address



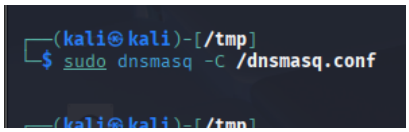
now i need to restart the dnsmasq

this however didnt work either

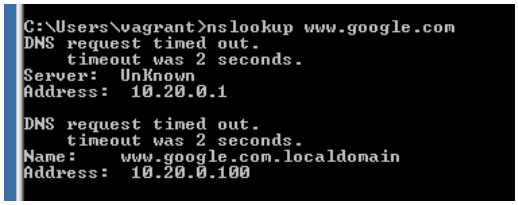
this is probably because dnsmasq can only bind to IPs on one of Kali's interfaces and at the moment kali has only 10.20.0.100 not 10.20.0.1 so i will add it in



this seemed to work



and successfully without touching the victim machines i performed the dns spoof



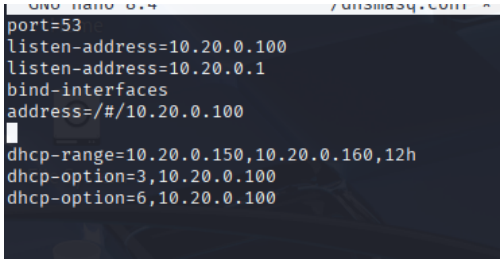
DHCP Spoof

Dynamic Host Configuration Protocol (DHCP) is responsible for automatically assigning network settings such as IP addresses, default gateways, and DNS servers to clients when they join a network. Because DHCP is based on broadcast messages and does not include authentication, any machine on the same subnet can reply with “offers.”

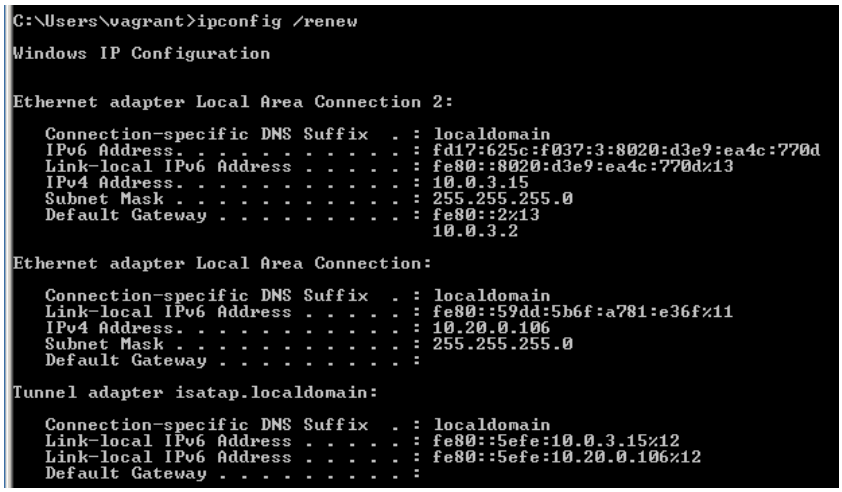
In a DHCP spoofing attack, I run a rogue DHCP server that responds to these requests faster than the legitimate server. By doing this, I can assign the victim an IP configuration of my choosing most importantly making my machine the default gateway and DNS server. This does not require any interaction with the victim: as soon as it renews its lease, it automatically accepts my fake settings.

The result is that all of the victim’s traffic flows through me, setting up a natural lead-in to Man-in-the-Middle (MITM) attacks.

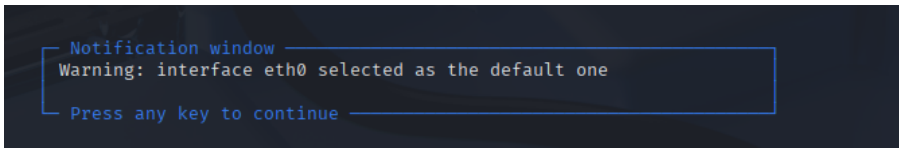
For this attack Yersina is a better option but im going to stick with dnsmasq because even though yersina has the ability to do dhcp starvation which blocked the legitimate dhcp server with requests it will be easier to continue with dhcpmasq



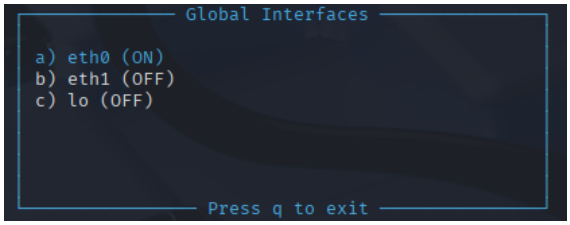
so what this does is it gives out IPs between .150 and .160 with 12 hour leases and option 3 and 6 are for default gateway and dns server respectively, what this does is it tells the victims i am their dns server and default gateway. and upon restarting the dnsmasq service it will run as both a DNS and DHCP server and when windows or ubuntu eventually refresh their DHCP lease they will accept my offer if it reaches them before the actual DHCP server.



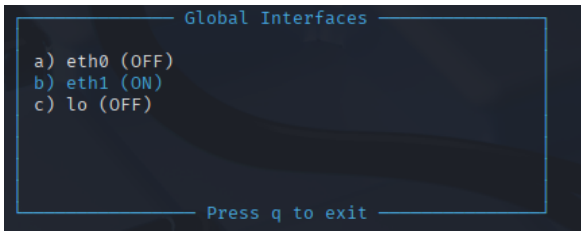
according to this the dhcp server is reaching the windows machine before my rogue server so i might have to use yersina after all



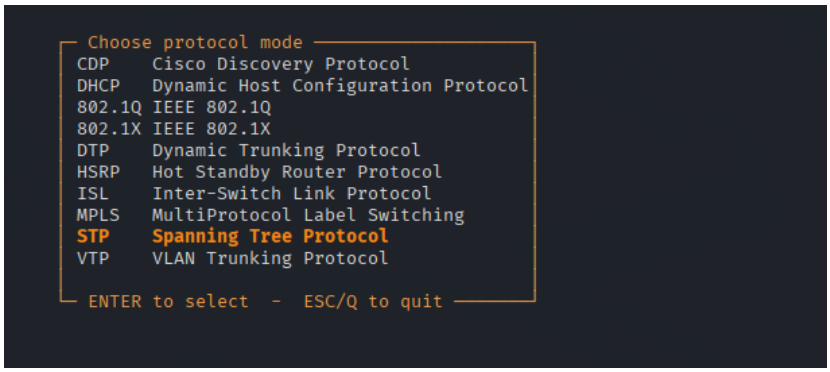
in the Yersina interactive window i see that eth0 is default selection and i need that to be eth1



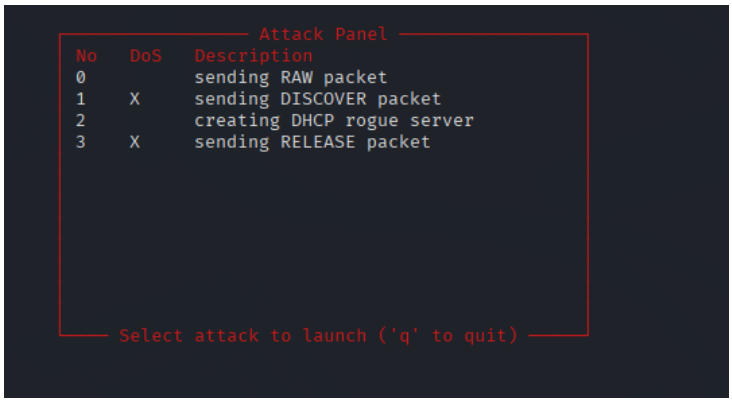
changed to



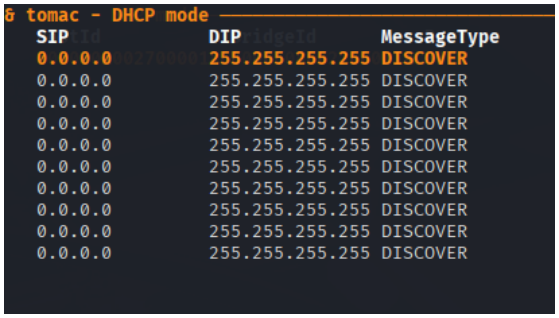
now i must chose a protocol mode



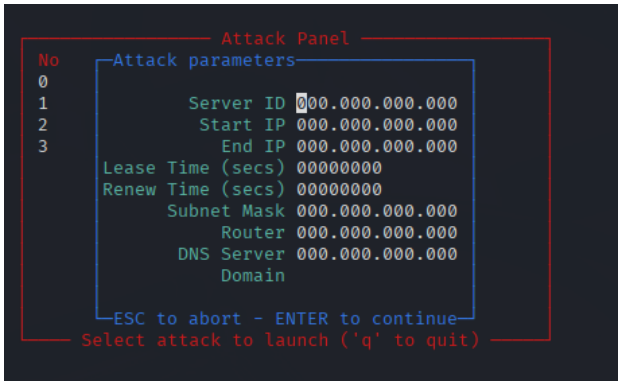
on this i selected DHCP after clicking "g" then "x" to open attack panel

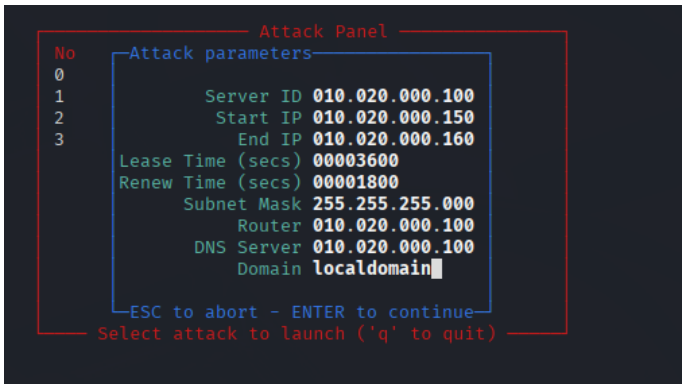


on selecting 1 i see this



and now to create the rogue server i selected 2 and will configure the attack parameters





This didnt seem to work because after running this the original dhcp server seemed to take over and the rogue one didnt which was a bit frustrating but i will continue this in week 8

Outcome:

- **DNS Spoofing** successfully demonstrated with `dnsmasq`.
- **DNS + ARP Spoofing** worked after IP aliasing, showing how DNS redirection can be achieved transparently.
- **DHCP Spoofing** setup attempted with both `dnsmasq` and Yersinia, but unsuccessful so far due to the legitimate DHCP server dominating responses. Further testing required.
- Established groundwork for MITM attacks in the following weeks.

Tags:
`Week 7, Spoofing, DNS Spoofing, ARP Spoofing, DHCP Spoofing, Penterrant Lab, Kali, Yersinia, dnsmasq`

Keywords:
network spoofing, DNS redirection, ARP spoofing, DHCP rogue server, Kali Linux, VMware lab, victim redirection, network security testing, MITM preparation, dnsmasq, Yersinia

Week 8

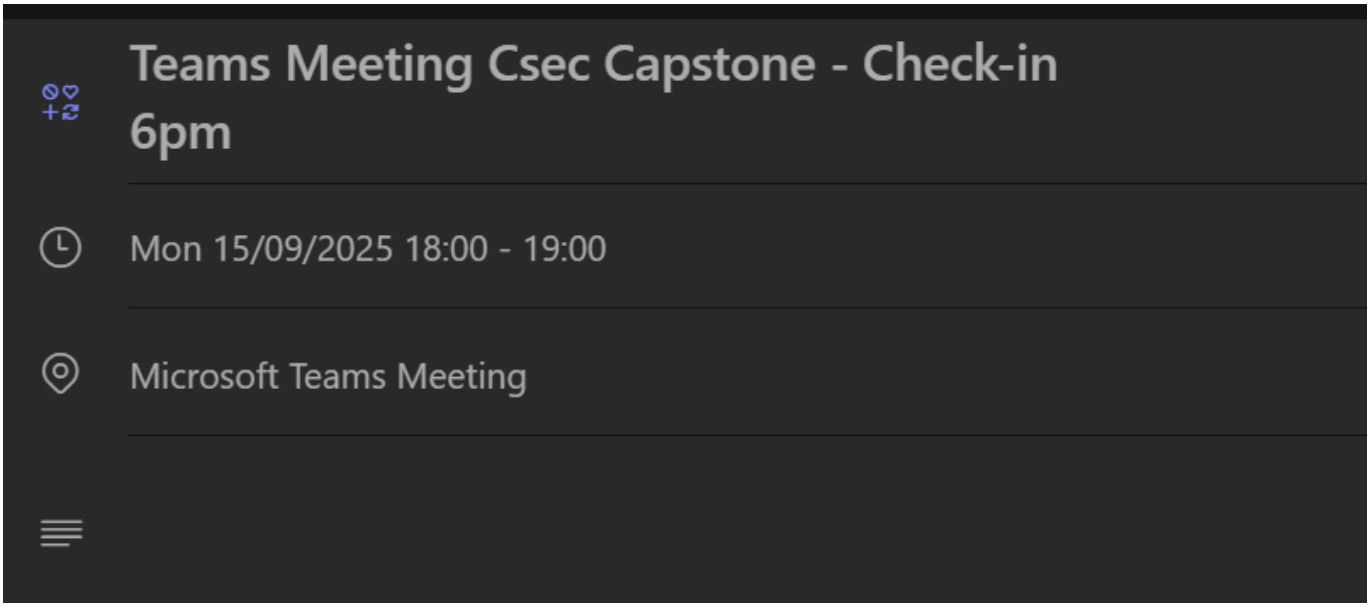
Om Shah (osshah@uts.edu.au) - Sep 19, 2025, 1:46 PM GMT+10

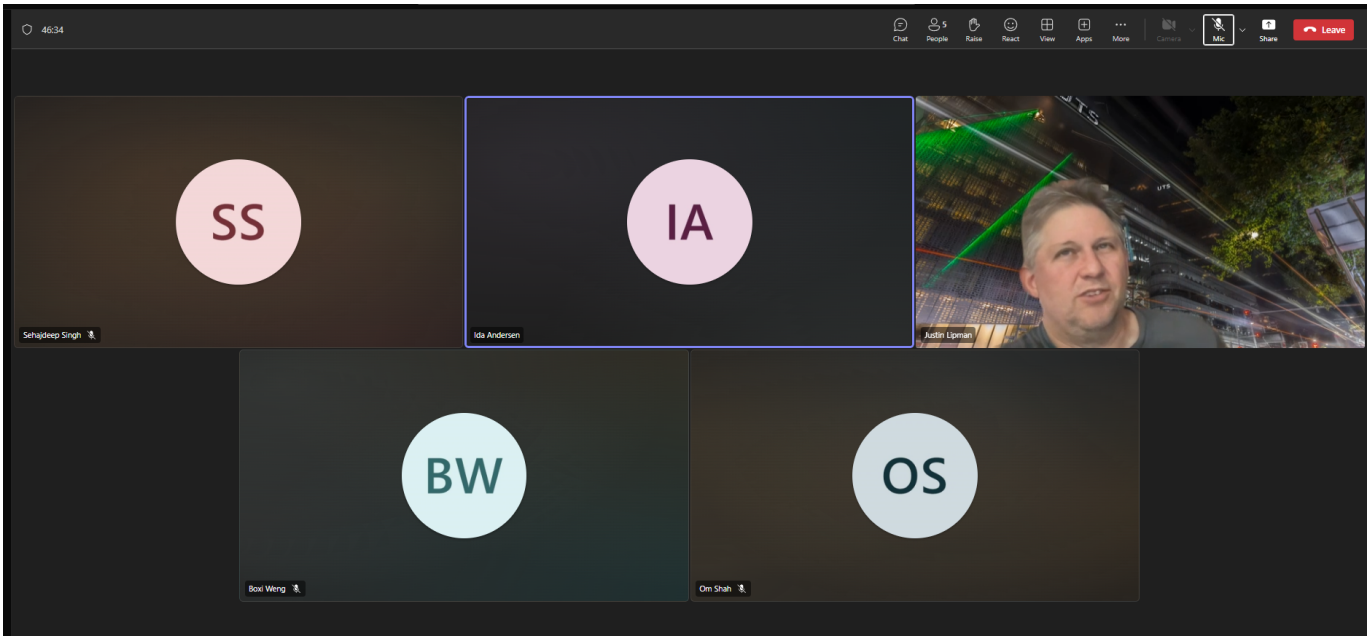
Python Sentinel Week 8 MVP Network Scanning Config Automation Kali VMware

Date: Mon 15/09/2025

Intention: Meeting with Product Owner to discuss progress

Hours Spent: 1.5 hours





What we did:

- Held the team meeting on Monday of Week 8.
- Discussed making a product available as part of the project.

- Suggested building an automation tool to handle scanning, attacking, and mitigation for Penterrant.
- Planned to first create a minimum viable product (MVP) due to other assignment and work commitments.
- Considered the timeline for finishing MITM work and follow-up tasks.

Outcome:

- Agreed that automation would be a strong candidate for the product component.
- MVP targeted for completion by the end of the week.
- MITM work scheduled for the following two weeks, leveraging prior experience.
- Remaining rubric items (mitigations and suggestions) planned for later stages.

Coding

Date: Wed 17/09/2025

Intention: Finish building the main parts of my Sentinel tool so it can scan a network safely, use a config file for settings, and show results in a clean way for demo use.

Total Hours Spent (Week 8): ~8 hrs

- Setting up shared folder & automation prep **2 hrs**
- Tools setup (libraries) **0.5 hr**
- Config (YAML + loader) **1 hr**
- Safety check (allowlist) **0.5 hr**
- Discovery (scanning logic) **2.5 hrs**
- Main program (entrypoint + CLI) **1 hr**
- Version control setup **0.5 hr**

What i did:

Setting up before automation (2 hours)

Firstly i needed to create a shared folder on linux purely for efficiency and this was the most effective way of ensuring my requirements i mentioned in week 7s planning which were that the code needs to:

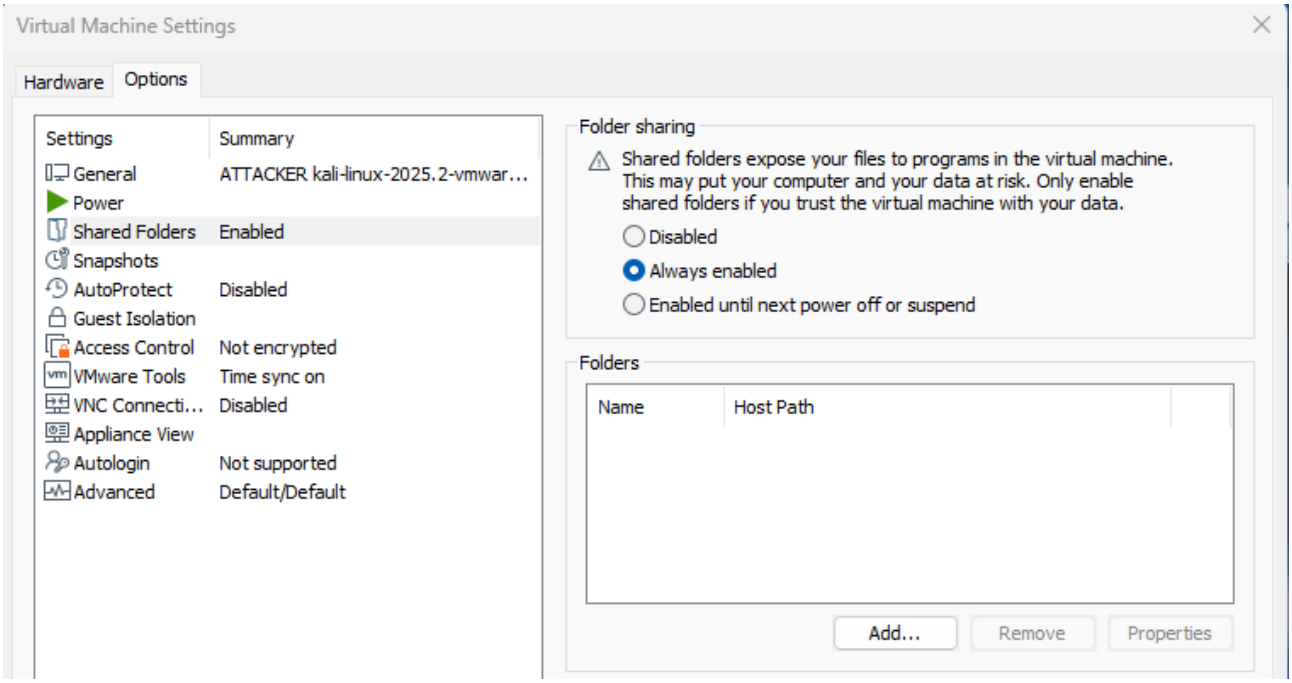
- run locally even without an internet connection

- run strictly on the configured subnet and not any other as that would be invasive

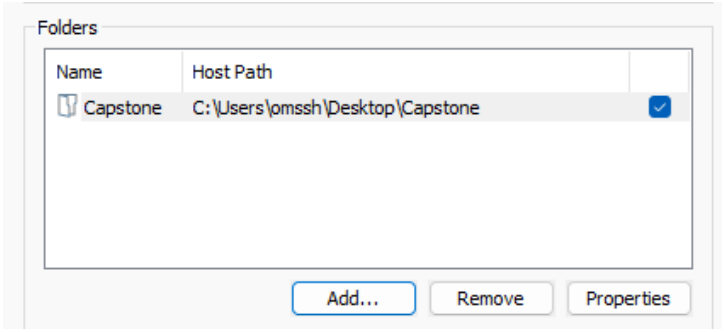
- output files that are easy to understand (3 files: technical, executive, and CLI based for debugging)

- be auto-updated without having to repeat any processes

so with this rough outline decided the following is the process i took of connecting a folder to a VM, i first went to the VM settings as i am pretty sure i have already downloaded the VMware tools on the kali VM then in shared folders i got the option to add a folder



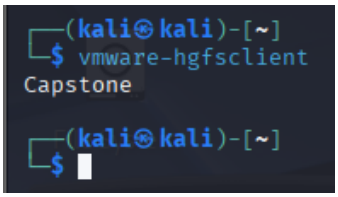
Then browsed to the location where i have the Capstone folder saved to allow access on the kali vm.



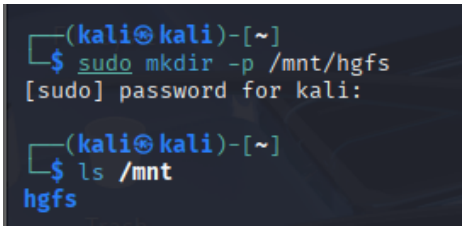
Now i needed to confirm that the file was correctly added and after some research i found that they would be stored in the mnt/hgfs directory

Liquid Web. (2023, November 27). *How to create and access a VMware shared folder*. Liquid Web. https://www.liquidweb.com/blog/create-vmware-shared-folder/

from the looks fo this the file wasnt mounted correctly as there is no subdirectory for this folder so i had to do some troublesooting here, firstly i had to check if the file was exposed by the VMware client which it was



so now i can deduce that there probably isnt a mount point so i just needed to add in the hgfs folder



okay so now that the hgfs folder was made but the hgfs folder is still empty so i need to add the shared folder manually

i wasnt entirely sure how to solve this part but i looked online and found a bit of information regarding it which mentioned using vmhgfs-fuse command

Superuser. (2019, March 5). *VMwared shared folder not working*. Super User. https://superuser.com/questions/1419274/vmwared-shared-folder-not-working

Launchpad. (2016). *Bug #1551558: open-vm-tools: Shared folders not working*. Launchpad. https://bugs.launchpad.net/ubuntu/+source/open-vm-tools/+bug/1551558

after a bit of reading i was still slightly confused so i used the --help tag which was interesting, the highlighted parts in particular so i looked up what .host:/ was in vmhgfs-fuse

so upon looking it up i found this command which is great since not only does it explain the other stuff but also allows the kali user to access it because im not 100% sure but i do know kali isnt root but kali has root privileges on using sudo and entering the code so its just easier to use the allow-other tag so i dont have to do too much trial and error

AI Overview

`.host:/` with `vmhgfs-fuse` refers to mounting all the host's shared folders within a VMware guest by using the `vmhgfs-fuse` command-line utility, which leverages the FUSE (Filesystem in Userspace) framework to provide a shared filesystem. You use it in a Linux guest to access files from the host by specifying `.host:/` as the source, a mount point on the guest, and various options like `allow_other` for permissions, for example: `sudo vmhgfs-fuse .host:/ /mnt/hgfs -o allow_other`.

How it Works

- `vmhgfs-fuse`: This is the user-mode FUSE client for VMware's Shared Folders feature. It replaces the older kernel-based HGFS module.
- `.host:/`: This is the path that tells the `vmhgfs-fuse` client to expose all folders shared by the VMware host on your guest.
- **Mount Point**: This is the local directory on your Linux guest where the host's shared files will become accessible (e.g., `/mnt/hgfs`).
- **Options**: You can pass options to control access and behavior, such as `allow_other` to grant all users access to the mount point.

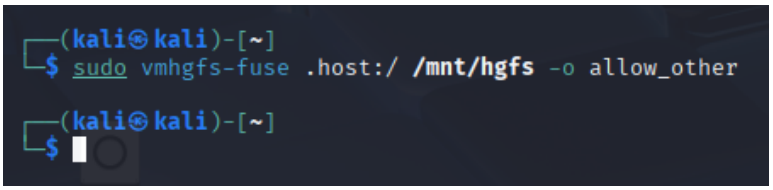
Example Command

To mount all host shared folders to `/mnt/hgfs` in your guest:

Code

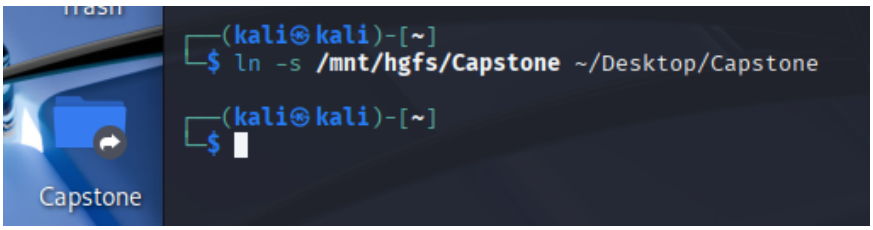
```
sudo vmhgfs-fuse .host:/ /mnt/hgfs -o allow_other
```

so i ran the command and it seems to have ran successfully



but i still need to double check if it put the folder itno /mnt/hgfs so i ran the ls command and its now there

another thing i wanted to do was make a symlink which is basically a shortcut from the desktop directly to the folder so i dont have to navigate everytime into the folder and now the file is visible from the desktop as a shortcut



and finally i wanted to ensure i dont have to repeat this process every time i want to use the folder so i need to add it to fstab according to stack exchange

Unix Stack Exchange. (2020, February 25). *Where to find the shared folder in Kali Linux?*. Stack Exchange. https://unix.stackexchange.com/questions/594080/where-to-find-the-shared-folder-in-kali-linux

Just add the following line to your `/etc/fstab` and reboot your machine:

```
vmhgfs-fuse /mnt/hgfs fuse defaults,allow_other 0 0
```

You will find your shared folder at `/mnt/hgfs/<FOLDER_NAME>` afterwards.

The first field (*fs_spec*).
This field describes the block special device, remote filesystem or filesystem image for loop device to be mounted or swap file or swap device to be enabled.

now other than the fs_spec and fs_vfstype fields as shown below this command **vmhgfs-fuse /mnt/hgfs fuse defaults,allow_other,nofail 0 0** should be correct to use so i needed to figure out how to use it properly

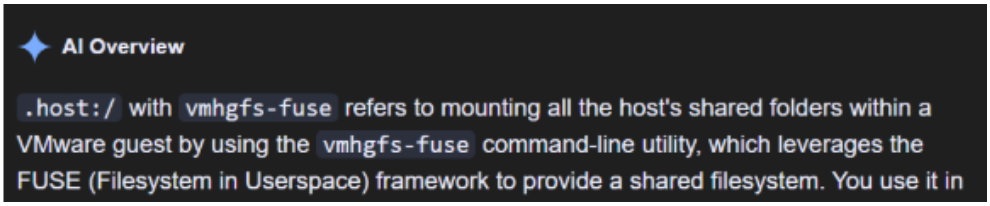
The third field (*fs_vfstype*).
This field describes the type of the filesystem. Linux supports many filesystem types: ext4, xfs, btrfs, f2fs, vfat, ntfs, hfsplus, tmpfs, sysfs, proc, iso9660, udf, squashfs, nfs, cifs, and many more. For more details, see mount(8).

An entry *swap* denotes a file or partition to be used for swapping, cf. swapon(8). An entry *none* is useful for bind or move mounts.

More than one type may be specified in a comma-separated list.

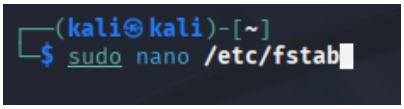
mount(8) and umount(8) support filesystem *subtypes*. The subtype is defined by '.subtype' suffix. For example 'fuse.sshfs'. It's recommended to use subtype notation rather than add any prefix to the first fstab field (for example 'sshfs#example.com' is deprecated).

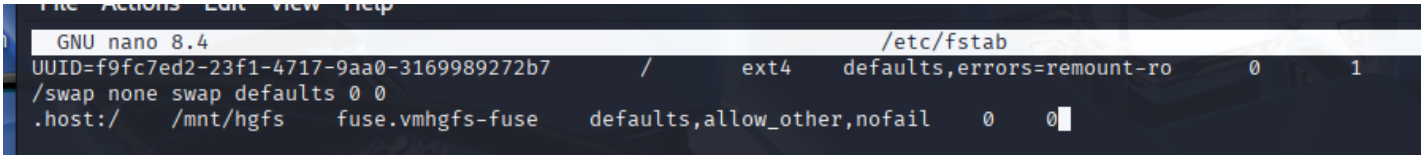
so basically what this is saying is that in /etc/fstab the first field is what i am mounting and for vmware that would be the .host/ which is the source that represents the host's shared folders as shown in the screenshot here



now the third field answers how to mount it so its where i ned to specify the driver or method. for a FUSE filestystem the normal way to use it is to use a subtype as the manual shows is to use a subtype like fuse.sshfs for SSHFS, for VMware shared folders that would be fuse.vmhgfs-fuse. So the stackexchange command differs from the manual in the sense that the vmhgfs-fuse isnt the thing being mounted its the method in which to mount it. so using this information the conclusion would be

.host:/ /mnt/hgfs fuse.vmhgfs-fuse defaults,allow_other,nofail 0 0





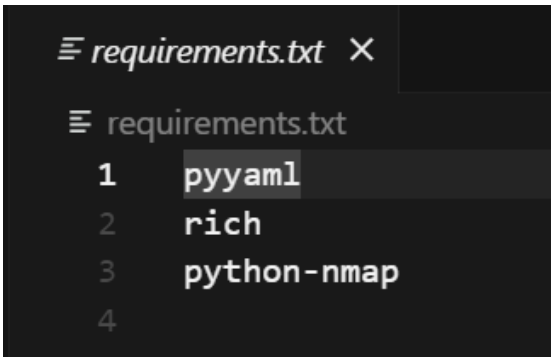
Tools Setup (30 mins)

I picked 3 main python packages which were as follows

pyyaml - to read settings from a config file

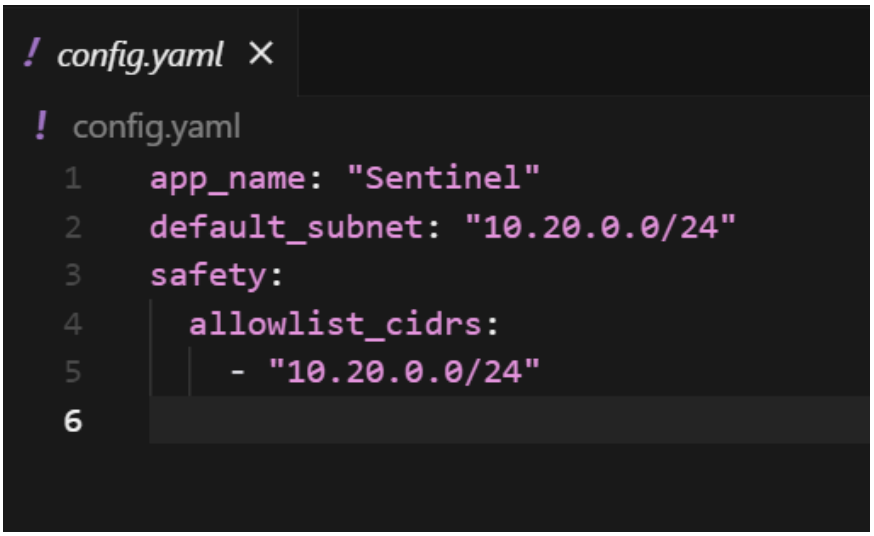
rich - to make the console output look nicer with tables and colours

python-nmap - to run scans and check open ports



Configuration (1 hour)

I made a config.yaml file to store the app name, default subnet, and safe networks in the allowlist this was because i didnt want the headache of going back and forth and parsing through the code to find the configurations, this set up makes it easier to edit stuff later so that if i need to demonstrate the automation outside of the environment at home on my PC and if the subnet or something changes then its easier for me to just change the config file rather than look for all the places i referenced the subnet and other info



then i made a config.py file so that the YAML file can be easily loaded and i dont have to hard-code anything so that stuff like IP ranges can be updated quickly, another benefit this has is that i dont have to swap out any logic in order to change the allowlist or subnet.

config.py

sentinel > core > config.py

```python
from pathlib import Path
import yaml


class AppConfig:

    def __init__(self, path: str = "config.yaml"):
        config_file = Path(path)
        config_text = config_file.read_text(encoding="utf-8")
        self._raw = yaml.safe_load(config_text)

    @property
    def raw(self):
        return self._raw

    @property
    def app_name(self):

        return self._raw.get("app_name", "Sentinel")

    @property
    def default_subnet(self):
        return self._raw["default_subnet"]

    @property
    def allowlist(self):
        return self._raw["safety"]["allowlist_cidrs"]

```

Safety Check (40 mins)

I then added a small helper in utils.py to stop scans running on networks outside of the allowlist which makes sure that i can only scan the lab network and dont accidentally touch anything else

sentinel > core > utils.py

```python
from ipaddress import ip_network

def subnet_in_allowlist(subnet: str, allowlist: list[str]) -> bool:
    try:

        target_network = ip_network(subnet, strict=False)

        for allowed_network in allowlist:
            allowed = ip_network(allowed_network, strict=False)
            if target_network.subnet_of(allowed):
                return True
        return False

    except Exception:
        return False

```

Discovery (2.5 hours)

I wrote the main scanning logic in discovery.py which uses nmap to scan a network to find computers and what services they're running.

This is the main function in the discovery file, it shows the core dependencies and function signature. Demonstrates that the code uses proper type hints (List[Dict]) and imports the essential libraries (nmap for scanning, rich for colored output). This establishes the technical foundation of the scanning module.

```python
from typing import List, Dict
import nmap
from rich import print

def scan_subnet(subnet: str) -> List[Dict]:
```

This part of the code shows how Nmap is integrated into the script. It uses the arguments -T4 --top-ports 20 -sV to run a fast scan on the 20 most common ports and identify service versions. The print statements provide clear progress updates so the user knows a scan is running and which command is being executed.

```python
def scan_subnet(subnet: str) -> List[Dict]:
    results: List[Dict] = []
    nm = nmap.PortScanner()

    scan_args = "-T4 --top-ports 20 -sV"
    print(f"[blue]Running FULL nmap scan: {scan_args}[/]")
    print(f"[blue]Full scan with service detection - may take 2-5 minutes...[/]")
    print(f"[dim]Full command: nmap {scan_args} {subnet}[/]")

    nm.scan(hosts=subnet, arguments=scan_args)
```

This code takes the raw Nmap scan results and organizes them into a structured dictionary. For each host, it records details like IP, port numbers, service names, products, and versions, making the data easier to use and display later.

```python
for i, host in enumerate(nm.all_hosts(), 1):
    print(f"[cyan]Analyzing host {i}/{host_count}: {host}[/]")
    host_dict = {"ip": host, "os": "unknown", "services": []}

    service_count = 0
    for proto in nm[host].all_protocols():
        for port in sorted(nm[host][proto].keys()):
            svc = nm[host][proto][port]
            port_state = svc.get("state", "unknown")

            host_dict["services"].append({
                "port": port,
                "proto": proto,
                "state": port_state,
                "name": svc.get("name", ""),
                "product": svc.get("product", ""),
                "version": svc.get("version", ""),
                "reason": svc.get("reason", "")
            })
            service_count += 1
```

And finally all put together the output looks like this

```
┌──(capstone-venv)─(kali㉿kali)-[/mnt/hgfs/Capstone]
└─$ python -m sentinel.main --mode scan --subnet 10.20.0.0/24

Starting scan of 10.20.0.0/24 ...
Discovering live hosts ...
Running FULL nmap scan: -T4 --top-ports 20 -sV
Full scan with service detection - may take 2-5 minutes ...
Full command: nmap -T4 --top-ports 20 -sV 10.20.0.0/24
Processing 5 discovered host(s) ...
Analyzing host 1/5: 10.20.0.100
  Found 20 service(s) on 10.20.0.100
Analyzing host 2/5: 10.20.0.102
  Found 20 service(s) on 10.20.0.102
Analyzing host 3/5: 10.20.0.103
  Found 20 service(s) on 10.20.0.103
Analyzing host 4/5: 10.20.0.106
  Found 20 service(s) on 10.20.0.106
Analyzing host 5/5: 10.20.0.200
  Found 20 service(s) on 10.20.0.200
Scan complete! Found 5 host(s)
```

Discovery Results (10.20.0.0/24)

IP	Services
10.20.0.100	tcp/21:ftp, tcp/22:ssh, tcp/23:telnet, tcp/25:smtp, tcp/53:domain, tcp/80:http, tcp/110:pop3, tcp/111:rpcbind, tcp/135:msrpc, tcp/139:netbios-ssn, tcp/143:imap, tcp/443:https, tcp/445:microsoft-ds, tcp/993:imaps, tcp/995:pop3s, tcp/1723:pptp, tcp/3306:mysql, tcp/3389:ms-wbt-server, tcp/5900:vnc, tcp/8080:http-proxy
10.20.0.102	tcp/21:ftp, tcp/22:ssh, tcp/23:telnet, tcp/25:smtp, tcp/53:domain, tcp/80:http, tcp/110:pop3, tcp/111:rpcbind, tcp/135:msrpc, tcp/139:netbios-ssn, tcp/143:imap, tcp/443:https, tcp/445:microsoft-ds, tcp/993:imaps, tcp/995:pop3s, tcp/1723:pptp, tcp/3306:mysql, tcp/3389:ms-wbt-server, tcp/5900:vnc, tcp/8080:http
10.20.0.103	tcp/21:ftp, tcp/22:ssh, tcp/23:telnet, tcp/25:smtp, tcp/53:domain, tcp/80:http, tcp/110:pop3, tcp/111:rpcbind, tcp/135:msrpc, tcp/139:netbios-ssn, tcp/143:imap, tcp/443:https, tcp/445:microsoft-ds, tcp/993:imaps, tcp/995:pop3s, tcp/1723:pptp, tcp/3306:mysql, tcp/3389:ms-wbt-server, tcp/5900:vnc, tcp/8080:http-proxy
10.20.0.106	tcp/21:ftp, tcp/22:ssh, tcp/23:telnet, tcp/25:smtp, tcp/53:domain, tcp/80:http, tcp/110:pop3, tcp/111:rpcbind, tcp/135:msrpc, tcp/139:netbios-ssn, tcp/143:imap, tcp/443:https, tcp/445:microsoft-ds, tcp/993:imaps, tcp/995:pop3s, tcp/1723:pptp, tcp/3306:mysql, tcp/3389:ms-wbt-server, tcp/5900:vnc, tcp/8080:http-proxy
10.20.0.200	tcp/21:ftp, tcp/22:ssh, tcp/23:telnet, tcp/25:smtp, tcp/53:domain, tcp/80:http, tcp/110:pop3, tcp/111:rpcbind, tcp/135:msrpc, tcp/139:netbios-ssn, tcp/143:imap, tcp/443:https, tcp/445:microsoft-ds, tcp/993:imaps, tcp/995:pop3s, tcp/1723:pptp, tcp/3306:mysql, tcp/3389:ms-wbt-server, tcp/5900:vnc, tcp/8080:http-proxy

---

Main Program (1 hour)

I then built a main.py as the starting point with the options for scan attack and mitigate right now only scan works and havent added any functionality for attack and mitigate

This following code defines the command-line interface for the tool. It lets users choose a subnet to scan and select a mode (scan, attack, or mitigate). The default mode is scanning, which runs the discovery function. This is the main entry point where users interact with the program.

```python
def main():
    cfg = AppConfig()
    parser = argparse.ArgumentParser(description="Sentinel — lab orchestrator")
    parser.add_argument("--subnet", default=cfg.default_subnet, help="Target subnet CIDR")
    parser.add_argument("--mode", choices=["scan","attack","mitigate"], default="scan")
    args = parser.parse_args()

    if args.mode == "scan":
        run_scan(cfg, args.subnet)
```

This code adds a safety check to ensure only approved subnets can be scanned. If the subnet isn't on the allowlist, the scan is blocked. Otherwise, it runs the scan and shows clear status updates so the user can follow the progress.

```python
def run_scan(cfg: AppConfig, subnet: str):
    if not subnet_in_allowlist(subnet, cfg.allowlist):
        print(f"[red]Safety block:[/] Subnet {subnet} not in allowlist {cfg.allowlist}")
        return

    print(f"[yellow]Starting scan of {subnet}...[/]")
    print("[yellow]Discovering live hosts...[/]")
    findings, nmap_args = scan_subnet(subnet)
    print(f"[green]Scan complete![/] Found {len(findings)} host(s)")

```

This code formats the scan results into a neat table. It lists each host's IP address along with the services found, making the output clear and easy to read for the user.

```python
    table = Table(title=f"Discovery Results ({subnet})")
    table.add_column("IP")
    table.add_column("Services")
    for h in findings:
        svcs = ", ".join([f"{s['proto']}/{s['port']}:{s.get('name','')}" for s in h.get("services",[])])
        table.add_row(h["ip"], svcs or "—")
    print(table)

```

Version Control (20 mins)

From my other classes even the ones this semester i learned how important this was especially since i am coding on both my PC and laptop and can easily get turned around so it was easier to just add a init.py file

sentinel > __init__.py

```python
__version__ = "0.4.2"

```

Outcomes:

- Sentinel MVP built with working **scan mode**.
- Shared folder integrated with VM and auto-mounted for smooth workflow.
- Config file + loader working → easy subnet/allowlist updates without code changes.
- Safety check ensures scans run only on the lab subnet.
- Discovery module successfully scans top 20 TCP ports and outputs structured host + service data.
- CLI interface (`main.py`) in place with scan, attack, mitigate modes (scan functional, others stubbed).
- Results displayed in clear tables using Rich library.
- Version control set up, enabling development across multiple machines.

Tags:

Sentinel, `Week 8`, `MVP`, `Network Scanning`, `Config`, `Automation`, `Kali` VMware

Keywords:

network discovery, Nmap, Python, YAML config, allowlist, VM shared folder, fuse.vmhgfs-fuse, CLI, Rich library, Penterrant, subnet scanning, MVP development, version control

Week 9

Week 10

Week 11

Week 12