\*\*Thesis\*\* = narrative \*\*why\*\* + appendix \*\*how\*\*.

Thesis:

Chapter 4

4.1 Lab environment

“All experiments ran inside a GPU-enabled VS Code Dev Container (Docker Desktop 28.1, CUDA 12.5 image) to ensure full reproducibility.”

4.1.1 Nw topology

4.1.2 Host roles & vm specs

**4.1.3 Operating System Selection and Preparation**

\label{sec:os-choice}

The operating system used for initial development and evaluation was **Windows 10 Enterprise Evaluation**. This choice was driven by several technical and practical considerations that align with the goals of this project and the requirements of a realistic security lab environment.

Windows 10 remains the most widely deployed Windows version, with approximately 53% market share in April 2025 compared to 44% for Windows 11 (StatCounter Global Stats). Since the goal of this research is to develop methods applicable to real-world Security Operations Centers (SOCs), targeting the most commonly used OS version provides better external validity and easier reviewer acceptance. Most public Sysmon configurations (e.g., SwiftOnSecurity), Sigma rules, and labeled datasets have been developed or tested on Windows 10, making it faster to replicate existing research and reduce the need for immediate filter tuning.

Windows 10 Evaluation is also technically easier to deploy in nested virtualization environments. It boots without requiring TPM 2.0, UEFI, or Secure Boot—unlike Windows 11, which enforces these hardware checks. This avoids the need for ISO patching or registry edits in Week 1. Moreover, Windows 10 idles with lower memory usage and disk footprint, which allows for more efficient multi-VM testing on limited hardware (e.g., 6–8 VMs on a 32 GB host). This OS version also aligns with prior datasets such as the 2023 Sysmon Learning Malware dataset and other drift-IDS baselines, enhancing reproducibility.

Support is ensured throughout the duration of the project, as Windows 10 Enterprise LTSC receives security updates until 2029. While Windows 11 will be introduced later in the pipeline as a deliberate drift-inducing scenario, starting with Windows 10 allows the system to stabilize first.

The version chosen for the experiments is 64-bit due to its wide compatibility with Sysmon, common threat emulation tools like Mimikatz, and machine learning libraries such as NumPy and River. Modern hypervisors also default to 64-bit guest OS installation, which reduces additional configuration.

A comparison of key factors is summarized below:

| **Reason** | **Why Start with Windows 10** | **What Changes with Windows 11** | **Impact** |
| --- | --- | --- | --- |
| Market reality | Still holds the largest share | Models a minority slice | Better external validity |
| Tool maturity | Existing rules and configs | Extra noise, renamed logs | Easier replication |
| VM compatibility | No TPM/Secure Boot required | Enforces modern boot config | Fewer setup issues |
| Resource usage | Lower RAM and disk | Heavier UI | Fits on limited hardware |
| Literature alignment | Matches previous studies | Requires rebuilding baselines | Enables reuse |
| Support window | Patched through 2029 | Also supported | No urgency to migrate |

The base image was finalized after patching and minor hardening steps (e.g., installing Sysinternals Suite, enabling RDP, disabling telemetry). To ensure consistency and reduce snapshot size, a final cleanup step was performed using Windows' built-in tools. Specifically, cleanmgr /sageset:1 was used to select cleanup options, followed by cleanmgr /sagerun:1 to execute the cleanup. This removed approximately 2 GB of unnecessary files and reduced the base image to 16 GB, improving performance for subsequent snapshot cloning. The full cleanup command is included in Appendix A.

“Sysmon executables were ACL-locked (icacls …) so only SYSTEM retained write access.”

### 4.1.4 Logging and Instrumentation

[Add your actual content here related to how logging was set up and which instrumentation tools were used—this placeholder is kept to maintain structure continuity.]

### 4.1.5 Domain Controller

| **Why a DC matters** | **Impact on the artefacts you’ve scheduled** |
| --- | --- |
| **Realistic authentication telemetry** – Pass-the-Hash, Kerberoasting, lateral movement, and most modern blue-team ML papers assume Active Directory events (4624/4625, 4768/4769, 4688, etc.). | Weeks 5–8: a DC means your “benign” and “attack-poc” traces include domain logons, service-ticket requests, NetLogon RPC, replicated Sysvol reads, etc. Feature space is richer and closer to production. |
| **Centralised policy push** – You’re hardening Sysmon on two Windows 10 VMs. With a DC you can ship your sysmon-config.xml, audit-policy, registry tweaks, and future ETW providers once via Group Policy rather than touching each clone. | Week 2: replace “Install & harden Sysmon” manual step with a GPO‐based deployment. |
| **Consistent user/computer identities** – Feast/River features such as “same user SID authenticating from two countries in 1 h” need globally unique SIDs. Local accounts collide across hosts; domain SIDs do not. | Weeks 4–7: your feature schema can key on AccountDomain + AccountName instead of hashing (hostname, local-user) pairs. |
| **Attack surface for blue-team experiments** – Later weeks include drift detection and adaptive models. Having DC-specific events lets you test concept drift caused by e.g. introducing a new domain controller or elevating functional level. | Weeks 9–12: drift detectors will pick up schema shifts (extra Event ID 2889 after enabling LDAP channel binding, etc.). |

**4.1.6 Data Pipeline & Lab Build**

“All target hosts were configured with Windows Advanced Audit Policy. Success and failure auditing was enabled for every sub-category via auditpol /set /category:\* /success:enable /failure:enable. Log sizes were raised to 100 MB (*wevtutil … /ms:102400*) to prevent rollover during 24 h attack replays (Table 4-2). The complete policy backup is listed in Appendix B.”

Patch automation has been done on every VMs. The scripts you can find in Appendix B.

Sysmon Event ID 16 (configuration change) is forwarded to ELK and triggers a Grafana alert to ensure any tampering with the monitoring configuration is detected within one minute.

**External Tasks**

The following tasks should be completed outside this document:

1. Add \label{sec:os-choice} below the heading “Operating System Selection and Preparation” so it can be referenced in Chapter 7.
2. Include the full cleanup script block in **Appendix A** under a section like "."
3. Add one sentence in **Chapter 5** “Static Baselines” to reference that prior numbers from the literature are used directly due to OS alignment.
4. Include the cleaned-up snapshot size in your **Methods section** if you submit a journal article.
5. Add the cleanup script to your **GitHub repo** under docs/lab\_build.md or as scripts/cleanup.ps1.

There you paste the “Why Win 10 first?” paragraph plus the table above; later, Chapter 7 can reference \ref{sec:os-choice} when introducing the Win 11/LTSC drift experiments.

Operating-System Selection & Justification

Lower friction, larger real-world footprint, richer public tooling.  
We’ll add Win 11 later as a deliberate drift scenario once the pipeline is stable.

| **Why start with Windows 10 Enterprise Eval** | | | | **What changes with Windows 11** | | | **Impact on your 9-month plan** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1 Market reality** – Win 10 still holds the single-largest Windows share (≈ 53 %) vs 44 % for Win 11 as of April 2025 ([StatCounter Global Stats](https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/?utm_source=chatgpt.com" \o "Desktop Windows Version Market Share Worldwide | Statcounter ...)). Your PhD (and journal reviewers) care that you target what SOCs actually run today. | | | | You’d be modelling a minority slice first. | | | Better external validity straight away; reviewers recognise the dataset as realistic. | | |
| **2 Tool maturity** – Most public Sysmon configs (e.g. SwiftOnSecurity), Sigma rules, and LM research datasets were written/tested on 10. | | | | Win 11 introduces extra noise (Widgets, DevHome, Copilot), and some event channels are renamed; you’ll spend time re-tuning filters. | | | Faster to replicate papers and reuse existing feature schemas. | | |
| **3 Hardware & VM friction** – Win 10 Eval boots on BIOS or UEFI, no TPM/Secure Boot required. | | | | Win 11 Eval enforces TPM 2.0, UEFI, Secure Boot. Nested hypervisors (VirtualBox on macOS, many cloud labs) fail the check unless you script work-arounds ([Microsoft Learn](https://learn.microsoft.com/en-us/windows/whats-new/windows-11-requirements?utm_source=chatgpt.com)). | | | Saves you hours of patch-ISO / registry-hack yak-shaving in Weeks 1-2. | | |
| **4 Resource footprint** – Win 10 idles with ≈ 250 MB less RAM and a smaller VHD. Handy when you run 6-8 VMs in parallel for data generation. | | | | Heavier UI stack, extra services. | | | Lets you keep the whole lab on a 32 GB host without swapping. | | |
| **5 Reproducibility & literature alignment** – 2023 Sysmon LM dataset and most drift-IDS benchmarks use Win 10 ([Microsoft](https://www.microsoft.com/en-us/evalcenter/evaluate-windows-11-enterprise?utm_source=chatgpt.com)). | | | | You’d need to rebuild baselines to compare apples-to-apples. | | | Your Chapter 5 “Static Baselines” can cite prior numbers directly. | | |
| **6 Support window** – Win 10 Enterprise LTSC is patched until 2029 ([Wikipedia](https://en.wikipedia.org/wiki/Windows_10?utm_source=chatgpt.com)), easily covering your PhD timeline. | | | | Win 11 support is obviously fine too, but no urgency to migrate. | | | Security updates stay available the whole project. | | |
| **Reason** | | | **Impact** | |
| Modern hyper-visors install only 64-bit guest OSs by default. | | | Less fiddling with nested-VT settings. | |
| Sysmon, Mimikatz, many Python wheels (NumPy, River) ship only x64 builds. | | | No “DLL bad format” errors during Week-1 setup. | |
| You’ll run 3-5 VMs + ELK on one host—32-bit RAM cap (≈ 4 GB) is a deal-breaker. | | | Smoother multi-VM lab. | |
| **Feature** | **Enterprise (GA / SAC channel)** | | | **Enterprise LTSC (Long-Term Servicing Channel)** | | | |
| Feature updates | Yearly “22H2, 23H2…” releases until Oct 2025 ([Microsoft Learn](https://learn.microsoft.com/en-us/windows/release-health/release-information?utm_source=chatgpt.com)) | | | No feature updates; only monthly security patches ([Microsoft Learn](https://learn.microsoft.com/en-us/windows/deployment/update/waas-overview?utm_source=chatgpt.com)) | | | |
| Bundled apps | Full UI (Edge, Widgets, Copilot, etc.) | | | Many consumer/UWP apps removed; smaller image | | | |
| Noise in logs | **Higher** (extra Event ID chatter from consumer apps) | | | **Lower**—cleaner baseline | | | |
| Tool support (Sigma rules, public datasets) | Widely tested (default in most papers) | | | Occasionally missing GUI-dependent telemetry ([Microsoft Learn](https://learn.microsoft.com/en-us/windows/whats-new/ltsc/whats-new-windows-10-2021?utm_source=chatgpt.com)) | | | |
| Use case in this project | **Primary OS** for realistic enterprise behaviour | | | Nice second OS to model an *abrupt drift* (Win 10 GA → LTSC) or “minimal-baseline” scenario | | | |

and drop the paragraph verbatim; then add  
\label{sec:os-choice} so Ch 7 can cite it with \ref{sec:os-choice}

4.1.3. Operating System preparation

**Where (and how much) to mention the “Disk-Cleanup” step**

| **Document** | **Section / heading** | **Level of detail** | **Rationale** |
| --- | --- | --- | --- |
| **PhD dissertation** | **Chapter 4 – Data Pipeline & Dataset Contribution** § 4.1.3 Operating-System Preparation (or a bullet in “Host-Build Checklist”) | *One or two sentences* in the narrative ➜ “A final cleanmgr /sagerun:1 pass removed superseded update files (≈2 GB) to minimise snapshot size.” | Readers just need to know the image was slimmed for reproducibility; they don’t need the dialog screenshot. |
|  | **Appendix A (Lab-Build Scripts)** | *Full command block* (the cleanmgr /sageset:1 list or the DISM one-liner). | Keeps the main text light but preserves exact commands for anyone rebuilding. |
| **Journal article** | **Methods ▸ Experimental Environment** (usually ½-page) | One clause inside the OS paragraph: “…after patching, we ran cleanmgr /sagerun:1 to remove obsolete WinSxS components, yielding a 16 GB base VDI.” | Space is precious; a clause is enough. Put full script in supplement. |
| **GitHub repo** | docs/lab\_build.md (or scripts/cleanup.ps1) | Full script + brief comment header. | Reviewers can run it verbatim; links nicely from paper. |

**Sample wording for the dissertation (main text)**

\subsubsection{Operating-System Preparation}

The Windows 10 Enterprise guest was fully patched via Windows Update and

lightly hardened (Sysinternals Suite, RDP enabled, telemetry disabled).

Immediately before taking the \textit{Win10-gold--2025-05-22} snapshot we

ran the built-in Disk Cleanup tool:

\texttt{cleanmgr /sageset:1} followed by \texttt{/sagerun:1},

removing superseded update packages and temporary files (≈2 GB),

thereby reducing the base image to 16 GB and speeding subsequent clone

operations.\footnote{The exact cleanup script is listed in Appendix~A.}

**Appendix snippet (full command block)**

# Appendix A: VM Build Automation Snippets

# Windows Update cleanup before snapshot

cleanmgr /sageset:1 ^

/WindowsUpdateCleanup ^

/TempFiles ^

/DeliveryOptimizationFiles

cleanmgr /sagerun:1 # executes the preset

:: alternative DISM one-liner (commented)

:: Dism.exe /Online /Cleanup-Image /StartComponentCleanup /ResetBase

**Why you document it at all**

* **Reproducibility** – anyone rerunning your pipeline knows why their VDI might be 19 GB vs your 16 GB.
* **Performance context** – smaller images = faster clone/load times, which influences throughput metrics you later report.
* **Transparency** – shows reviewers you controlled for irrelevant disk-IO noise.

If you skip mentioning it, nothing breaks scientifically, but a reviewer might ask “exactly how was the image prepared?”—this pre-empts that question with one concise sentence and a script in the appendix/supplement.

| **Tick?** | **Item** | **Why** |
| --- | --- | --- |
| ✅ | **Windows Update Cleanup** | Frees 2–3 GB of superseded update packages you’ll never need inside a lab VM. |
| ✅ | **Downloaded Program Files** | Legacy ActiveX cache; safe to delete. |
| ✅ | **Temporary Internet Files** | Empty. |
| ✅ | **Delivery Optimization Files** | Peer-to-peer update chunks; not needed once patched. |
| ✅ | **Temporary Files** | Everything under %TEMP%. |
| ✅ | **Microsoft Defender Antivirus** (if present) | Deletes old signature versions only. |
| ➖ | **Device Driver Packages** | Skip **unless** you’re certain no guest drivers might roll back; usually safe to leave unticked. |
| ➖ | **Previous Windows Installations** | Will not appear on a clean Eval install; if you see it, tick it — but only if you’re 100 % sure you won’t need rollback. |
| ❌ | **Recycle Bin** | Only tick if you’ve already confirmed nothing important was binned. |
| ❌ | **Thumbnails / Game Statistics / Logs** | All small; doesn’t matter either way. |

4.1.4 Logging & instrumentation

# Appendix

## Appendix A - VM Build Automation Snippets

## Appendix B – Lab Build Scrips

[Full command block or PowerShell script; copy of auditpol\_backup.csv in listing form.]

Powershell

# enable full auditing

auditpol /set /category:\* /success:enable /failure:enable

# bump sizes

wevtutil sl Security /ms:102400

wevtutil sl "microsoft-windows-sysmon/Operational" /ms:102400

#

secedit /export /areas SECURITYPOLICY /cfg C:\Tools\secpol\_backup.inf

#

auditpol /backup /file:docs/auditpol\_backup.csv

**Patch automation**

Set-ExecutionPolicy RemoteSigned -Scope Process -Force

Install-PackageProvider NuGet -Force -Scope CurrentUser

Install-Module PSWindowsUpdate -Force -Scope CurrentUser

Import-Module PSWindowsUpdate

Get-WindowsUpdate -AcceptAll -Install -AutoReboot

Gold vm snapshot

Installed Sysmon v15.x with SwiftOnSecurity config (commit abc123, trimmed to EID 1/3/11/23).

Command: Sysmon64.exe -accepteula -i sysmonconfig.xml

Ds 1, 3, 11 & 23 (core for lateral-movement features).