Kali Linux VM Installation

Written By Jeffery Waldron

@VoidXDcypher

Block 1 – Project Title

Kali Linux 2025.2 Virtual Machine Installation & Verification

Block 2 – Objective

The goal of this project is to establish a clean and verified Kali Linux 2025.2 virtual machine as the first component of my home cybersecurity lab. I chose Kali as a starting point because it provides a pre-configured set of penetration testing tools, allowing me to focus on learning Linux fundamentals and testing workflows without spending time on complex installations. By verifying the image integrity and documenting the build, this project ensures a reliable baseline that I can expand, harden, and replicate for future projects.

Block 3 – Asset/Environment Details

- **Host Machine:** Acer Nitro 5
- **Host OS:** Windows 11
- **Virtualization Platform:** VMware Workstation Player 16
- **Guest OS:** Kali Linux 2025.2 (x64)
- **Download Source:** Official kali.org VM image

Block 4 – Build Procedure (Summary)

To begin this project, I removed my previous Kali VM to ensure a completely clean starting point. VMware Workstation Player 16, which I had already installed several months ago, served as the virtualization platform. I then navigated to the official kali.org website (See Fig.1) and downloaded the Kali Linux 2025.2 image. I chose Kali as my base environment because it comes preloaded with the penetration testing tools I need, which allows me to focus on learning workflows rather than spending time on initial tool installations.

The download process itself presented challenges due to unstable hotspot connectivity and limited data availability. The file took nearly a full day to complete, as I had to top up my data plan and repeatedly click “retry” to resume failed downloads. These interruptions made me acutely aware of how exposed I was to the risk of file tampering or corruption during transfer.

This realization led to a critical step in the build process: verifying the file's integrity before using it. I initially considered using the standard PowerShell `Get-FileHash` command, but it quickly became clear that manual verification would be tedious and prone to human error, especially when comparing long hash values by sight. I also recognized that this would not be the last time I needed to perform hash checks—this would become a regular part of my workflow whenever I download system images or critical files.

To solve this problem, I created a custom PowerShell script that automated the process (See Fig. 2 and 3). The script prompts for the file location, calculates its SHA256 hash, and then directly compares it to the official hash value from the vendor (See Fig. 4). If the two values match, the script confirms integrity; if not, it immediately flags a mismatch. This approach not only improves accuracy but also gives me a repeatable tool I can use for all future downloads.

Once the image was verified, I made a clean backup copy of the original file and saved it to a separate location as a backup (See Fig. 5). This ensures that I can always restore a pristine version of the VM image if needed. After creating the backup, I loaded the verified image into VMware Workstation Player and checked the original configurations for the virtual hardware settings (See Fig. 6). I then booted Kali Linux for the first time, completing the initial build phase. The VM is now running and prepared for further configuration, updates, and hardening (See Fig. 7).

Block 5 – Issues Encountered

The main challenge during this build process was related to the instability of my network connection. Because I was using a mobile hotspot with limited data, the Kali Linux image download frequently failed, requiring me to manually restart and resume the process several times. This added significant delays, extending the download time to nearly a full day.

Another issue I identified was the inherent security risk of downloading large OS images over an unreliable connection. I realized that without verification, I could not fully trust the integrity of the file I had downloaded. This awareness led directly to the creation of my custom PowerShell hash verification script, which resolved the problem of manually checking file integrity and eliminated the risk of overlooking a mismatch.

Lastly, this was my first time creating a cybersecurity project report outside of a military format. I found that I was initially frustrated by "blank page syndrome," and I realized how essential structured templates are to documenting technical work effectively. This realization led me to begin developing my own reusable report templates, which will improve the clarity and speed of future documentation such as this.

Block 6 – Key Evidence

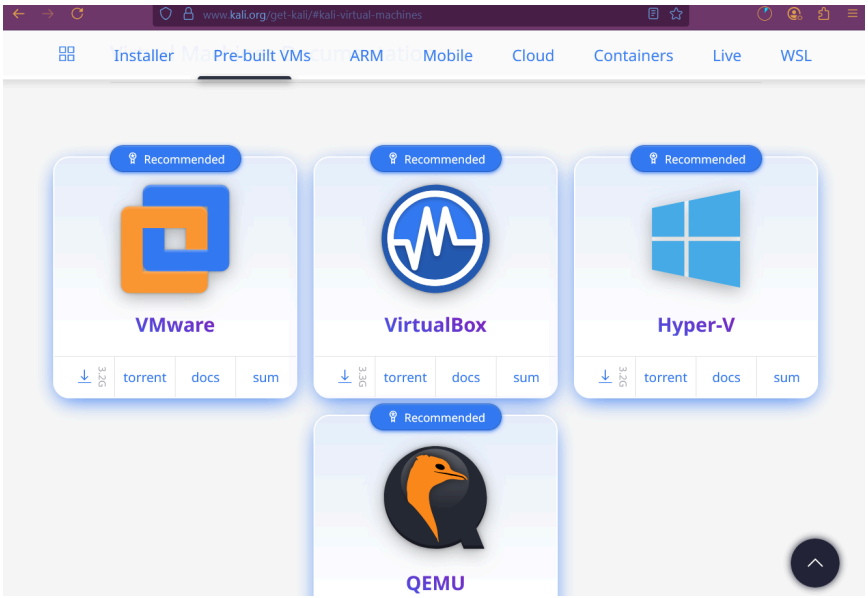

Fig. 1

```
    Write-Host "  .\chksum.ps1 'C:\Path\To\File.zip'"
    Write-Host "      (Displays the calculated SHA256 hash and then asks for a verification hash)"
    Write-Host ""
    Write-Host "  .\chksum.ps1 'C:\Path\To\File.zip' 5F2A7C3D..."
    Write-Host "      (Calculates and compares directly with the provided hash)"
    Write-Host ""
    Write-Host "Options:"
    Write-Host "  -h, --help      Show this help menu"
    exit
}

# If no file path was provided, ask for it
if (-not $FilePath) {
    $FilePath = Read-Host "Enter the full path to the file you want to check"
}

# Check if the file exists
if (-not (Test-Path $FilePath)) {
    Write-Host "Error: File not found at $FilePath"
    exit
}

# Calculate hash
$calculatedHash = (Get-FileHash -Path $FilePath -Algorithm SHA256).Hash
Write-Host "`nCalculated SHA256 Hash:"
Write-Host $calculatedHash "`n"

# If no expected hash was provided, ask for it
if (-not $ExpectedHash) {
    $ExpectedHash = Read-Host "Paste the verification hash here"
}

# Compare hashes
if ($calculatedHash.ToUpper() -eq $ExpectedHash.ToUpper()) {
    Write-Host "Hashes match. File integrity verified."
} else {
    Write-Host "Hash mismatch. The file may be corrupted or tampered with."
}
```

Fig. 2

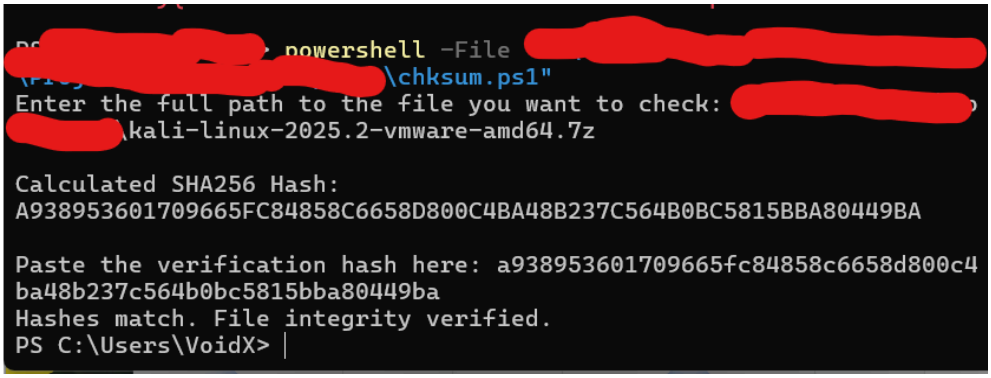

Fig. 3

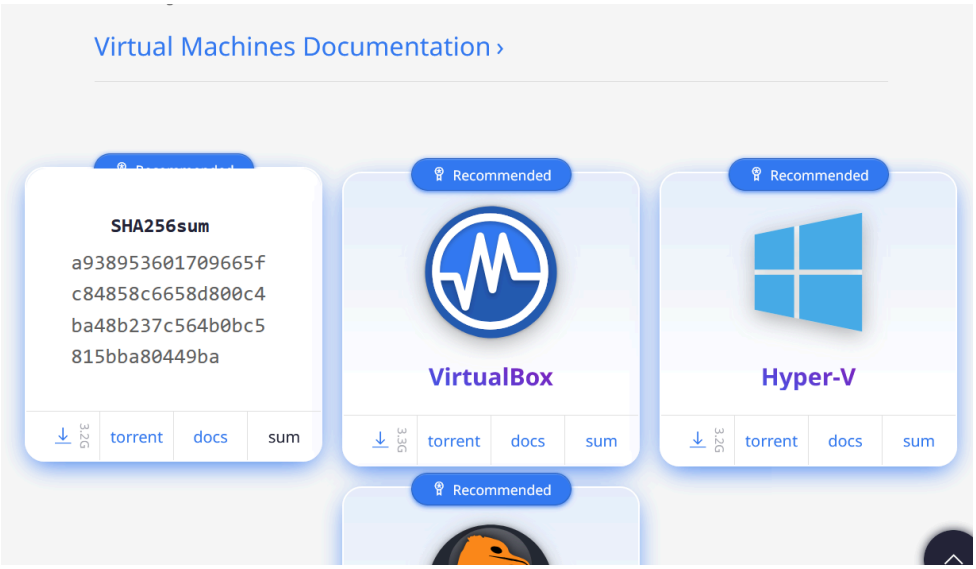

Fig. 4

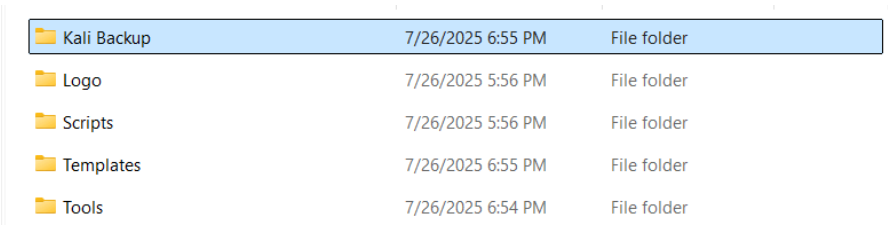

Fig. 5

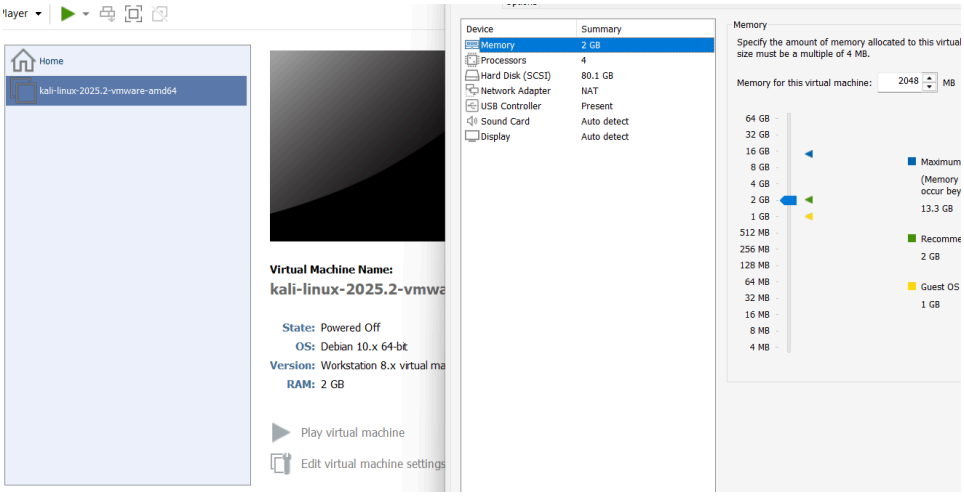

Fig. 6

FIg. 7

Block 7 – Results

The project successfully delivered a fully verified Kali Linux 2025.2 virtual machine, prepared for use as the foundation of my cybersecurity home lab. The OS image was validated through an automated hash comparison, ensuring its integrity and security before deployment. A clean backup copy of the verified image has been stored separately, providing a reliable baseline for restoration or replication as needed. The VM is now installed on VMware Workstation Player 16, successfully booted, and ready for tool updates, configuration, and hardening in subsequent phases.

Block 8 – Next Steps

1. **Perform a full system update** to ensure all Kali tools and packages are current.

2. **Install and configure UFW (Uncomplicated Firewall)** for initial hardening of the environment.

3. **Document UFW rules and configurations** as part of a follow-up report (BAR-2).

4. **Test VM performance and resource allocation** to confirm optimal settings.