Advanced Network Diagnostics & TCP/IP Troubleshooting Lab W/ Windows 10 & CLI

Author: Darnell Maiden | **Certifications:** CompTIA Security+, Google IT Support Professional Certificate, Google Cybersecurity Professional Certificate | **Target Environment:** Windows 10 Pro, VirtualBox / Hyper-V, Command Line Interface |
PORTFOLIO WEBSITE: https://www.saintmaiden.site/

EXECUTIVE SUMMARY:

PROBLEM:	SOLUTION:	IMPACT:
End users routinely report generic "internet is down" errors caused by localized Layer 3/7 network stack corruption, APIPA address assignments, and DNS server lookups timing out.	**Executed a systematic, command-line diagnostic ladder using `ipconfig`, `ping`, `nslookup`, and `netsh` on a Windows 10 client VM to isolate and remediate simulated TCP/IP stack failures.**	**Reduced mean time to resolution (MTTR) by eliminating reliance on GUI troubleshooters, restoring adapter connectivity via CLI, and documenting diagnostic standard operating procedures (SOPs).**

TECHNICAL TOPOLOGY/SPECIFICATIONS:

COMPONENT/ROLE	SYSTEM/SPACE NAME	NETWORK MODE & ACCESS	SERVICES & ROLES
Client Workstation	**WIN-CLIENT01**	**Virtual Adapter (NAT/Bridged)**	**Windows 10 Pro, CMD, PowerShell**
Diagnostic Toolkit	**CLI Utilities**	**Native Windows Environment**	`ipconfig`, `ping`, `nslookup`, `netsh`, `tracert`
Target Gateway/WAN	**Local Router/Public DNS**	**Subnet Gateway (e.g., 192.168.1.1), Public WAN (8.8.8.8)**	**DHCP Server, DNS Resolution** (`google.com`)

PHASE 1: BASELINE VERIFICATION & DIAGNOSTIC LADDER

ACTION: **Executed** ipconfig /all **and** netsh interface show interface **to map physical MAC addresses, active DHCP leases, default gateways, and exact interface names (e.g.,** "Ethernet"**). Ran the four-step OSI diagnostic ladder (**ping 127.0.0.1 → ping Gateway → ping 8.8.8.8 → ping google.com**).**

TROUBLESHOOTING ISSUE: **Standard** `ipconfig` **outputs were insufficient to identify active DNS servers or specific adapter interface names needed for CLI bindings. Resolved by elevating CMD to Administrator and executing** `ipconfig /all`**.**

WHY?: **Establishes a known healthy baseline across OSI Layers 1–7 and verifies local TCP/IP stack integrity before simulating network failures.**

```
Administrator: Windows PowerShell (x86)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : DESKTOP-B3S0UI2
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-65-23-07
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2601:c4:c200:1df0::94b7(Preferred)
   Lease Obtained. . . . . . . . . . : Thursday, October 1, 2026 10:31:10 PM
   Lease Expires . . . . . . . . . . : Friday, October 2, 2026 12:23:23 AM
   IPv6 Address. . . . . . . . . . . : 2601:c4:c200:1df0:7c27:9b0f:3fd3:199f(Preferred)
   Temporary IPv6 Address. . . . . . : 2601:c4:c200:1df0:702e:67f8:b988:97ec(Preferred)
   Link-local IPv6 Address . . . . . : fe80::7c27:9b0f:3fd3:199f%6(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.0.0.192(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Thursday, October 1, 2026 10:31:14 PM
   Lease Expires . . . . . . . . . . : Saturday, October 3, 2026 10:31:21 PM
   Default Gateway . . . . . . . . . : fe80::fad0:eff:fe39:8bd7%6
                                       10.0.0.1
   DHCP Server . . . . . . . . . . . : 10.0.0.1
   DHCPv6 IAID . . . . . . . . . . . : 101187623
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-32-50-8F-0F-08-00-27-65-23-07
   DNS Servers . . . . . . . . . . . : 2001:558:feed::1
                                       2001:558:feed::2
                                       75.75.75.75
                                       75.75.76.76
                                       2001:558:feed::1
                                       2001:558:feed::2
   NetBIOS over Tcpip. . . . . . . . : Enabled
PS C:\Windows\system32>
```

```
Administrator: Windows PowerShell (x86)
PS C:\Windows\system32> ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time=3ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 0ms
PS C:\Windows\system32> ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=16ms TTL=64
Reply from 10.0.0.1: bytes=32 time=4ms TTL=64
Reply from 10.0.0.1: bytes=32 time=5ms TTL=64
Reply from 10.0.0.1: bytes=32 time=4ms TTL=64

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 16ms, Average = 7ms
PS C:\Windows\system32> ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=53ms TTL=109
Reply from 8.8.8.8: bytes=32 time=14ms TTL=109
Reply from 8.8.8.8: bytes=32 time=33ms TTL=109
Reply from 8.8.8.8: bytes=32 time=29ms TTL=109

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 53ms, Average = 32ms
PS C:\Windows\system32> ping google.com

Pinging google.com [2607:f8b0:4002:c05::71] with 32 bytes o
Reply from 2607:f8b0:4002:c05::71: time=97ms
Reply from 2607:f8b0:4002:c05::71: time=64ms
Reply from 2607:f8b0:4002:c05::71: time=61ms
Reply from 2607:f8b0:4002:c05::71: time=73ms

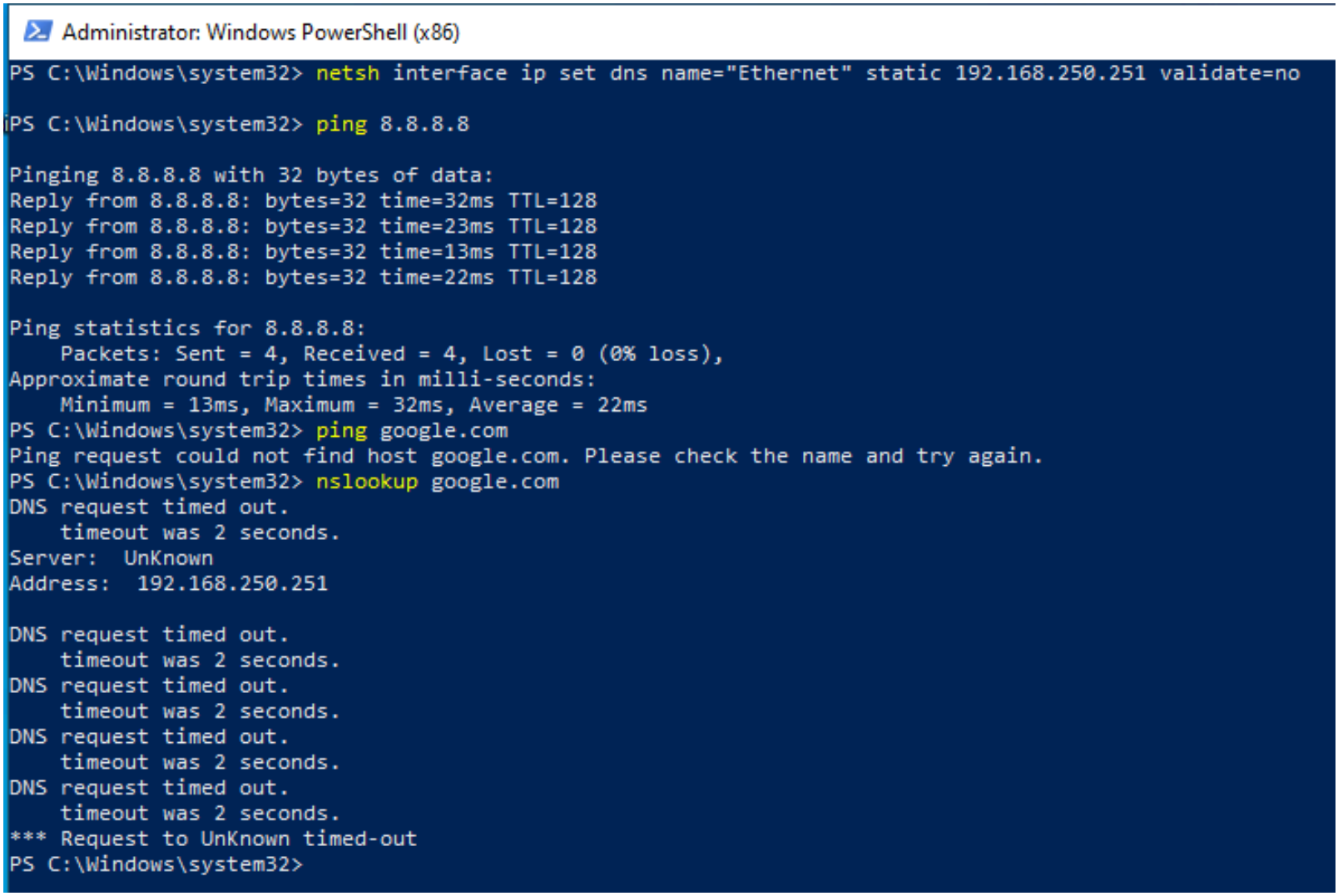
Ping statistics for 2607:f8b0:4002:c05::71:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 61ms, Maximum = 97ms, Average = 73ms
PS C:\Windows\system32>
```

PHASE 2: BROKEN DNS RESOLUTION & RESOLVER CACHE PURGE

ACTION: **Forced a static, non-existent DNS server IP (`192.168.250.251`) on the active adapter using `netsh interface ip set dns name="Ethernet" static 192.168.250.251 validate=no`. Verified failure state using `nslookup google.com` and restored settings via DHCP and cache flushing.**

TROUBLESHOOTING ISSUE: **`netsh` threw a DNS validation error during script execution. Mitigated by appending the `validate=no` parameter to force Windows to accept the static configuration without pre-testing server reachability.**

WHY?: **Isolates Layer 7 DNS failure from Layer 3 IP routing. Proves that raw IP pings (`ping 8.8.8.8`) can succeed while domain name resolution (`ping google.com`) completely fails.**

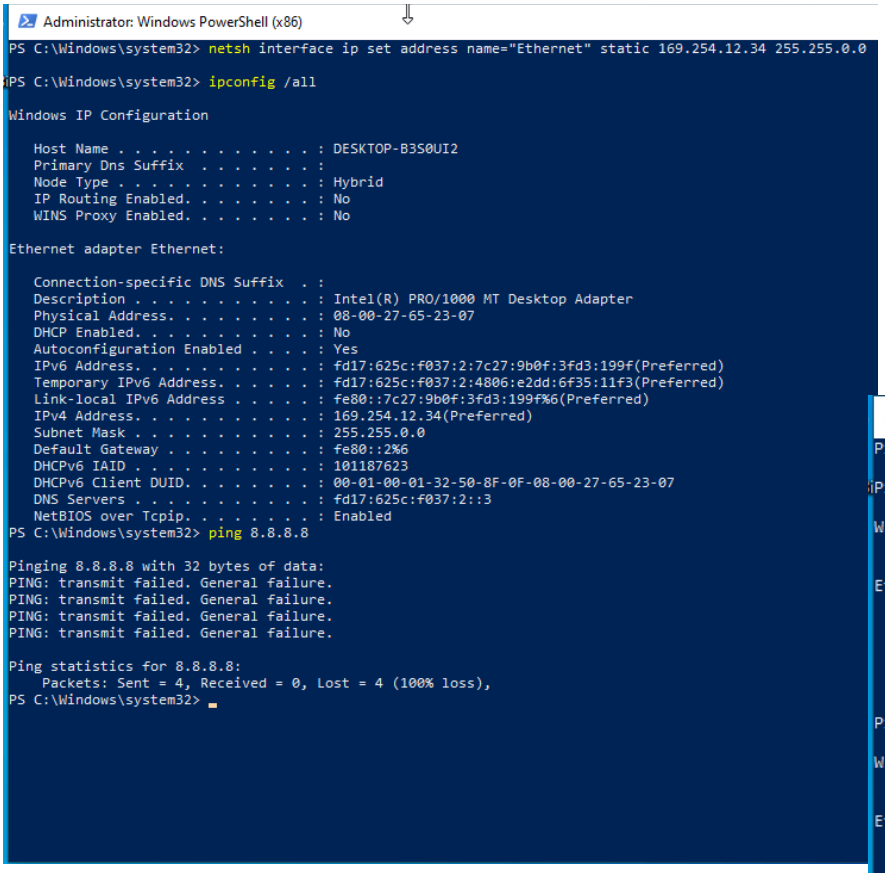


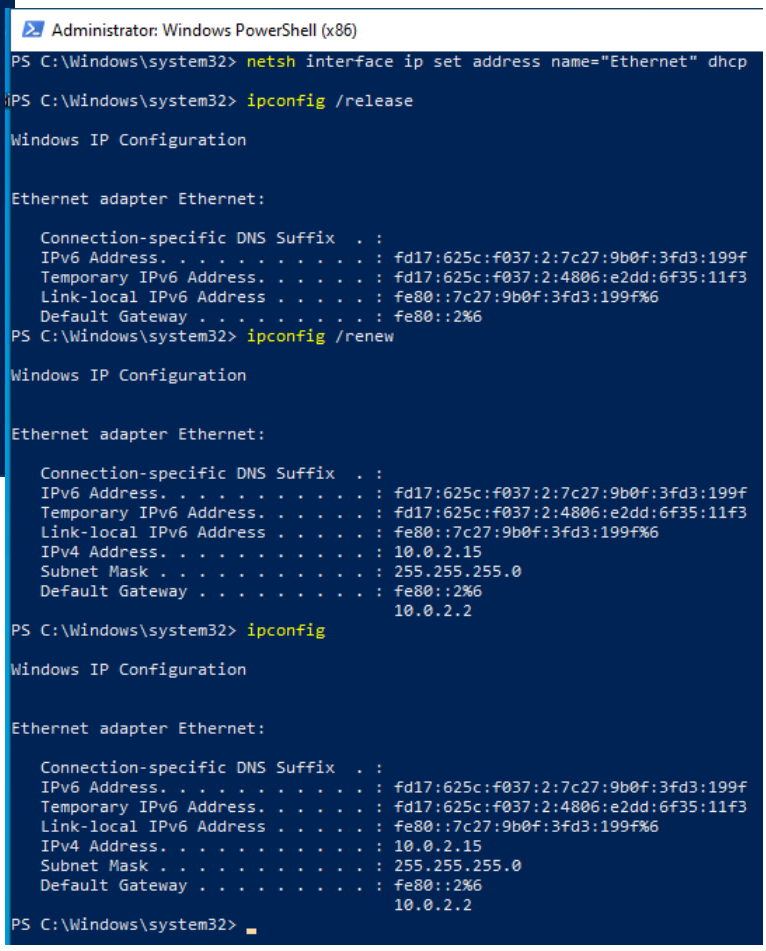
PHASE 3: DHCP FAILURE, APIPA REMEDIATION & DORA RENEGOTIATION

ACTION: **Simulated total DHCP server failure by forcing a static APIPA address (`169.254.12.34/16`). Identified missing default gateway parameters and restored automatic addressing using `netsh`, followed by an IP release and renew cycle.**

TROUBLESHOOTING ISSUE: **Client remained bound to stale APIPA parameters even after re-enabling DHCP. Resolved by explicitly issuing `ipconfig /release` followed by `ipconfig /renew` to force a complete DORA (Discover, Offer, Request, Acknowledge) broadcast sequence.**

WHY?: **Replicates real-world scenario where non-routable APIPA addresses (`169.254.0.0/16`) block internet traffic due to missing default gateways.**

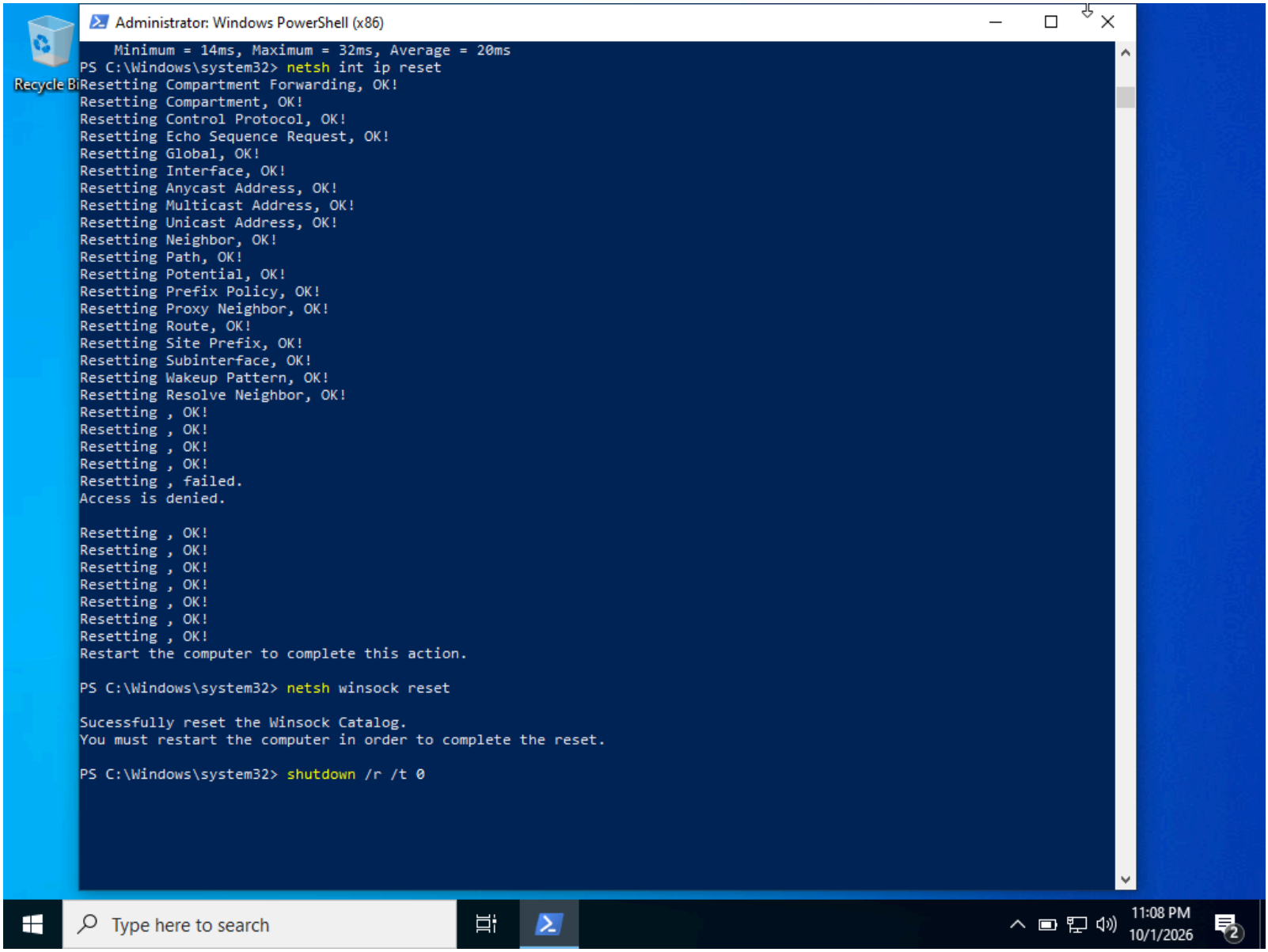




PHASE 4: LOW-LEVEL TCP/IP STACK & WINSOCK CATALOG RESET

ACTION: **Executed kernel-level network resets using `netsh int ip reset` and `netsh winsock reset` to clear corrupted registry parameter keys and rebuild the Windows Sockets catalog. Finalized changes via a system restart (`shutdown /r /t 0`).**

WHY?: **Clears rogue Layered Service Providers (LSPs), bad registry entries, or corrupted socket bindings caused by failed software/VPN uninstalls without requiring a full OS re-image.**



TECHNICAL SUMMARY:

SECURITY & NETWORKING DOMAIN	TECH & CONTROLS CONFIGURED
CLI DIAGNOSTICS	`ipconfig /all`, `ping`, `nslookup`, `tracert`, `netsh`
PROTOCOL LAYERS	**OSI Layer 3 (IP Routing) & Layer 7 (DNS Resolution)**
ADDRESS MANAGEMENT	**APIPA (`169.254.x.x`), Dynamic DHCP (DORA Sequence), Static IP/DNS Overrides**
STACK REMEDIATION	**TCP/IP Registry Rewrite, Winsock API Catalog Purge, DNS Resolver Flush (`/flushdns`)**
SYSTEM VALIDATION	**Loopback Testing (`127.0.0.1`), Dual-Verification Pings (IP vs Hostname)**

RESULT & BUSINESS IMPACT:

- **METHODICAL TROUBLESHOOTING:** **Mastered the four-step diagnostic ladder to quickly differentiate between physical link issues, routing failures, and DNS lookup timeouts.**
- **COMMAND-LINE EFFICIENCY:** **Restored total network functionality using administrative CLI utilities (`netsh`, `ipconfig`), bypassing slower GUI troubleshooters.**
- **PREVENTED UNNECESSARY ESCALATIONS:** **Demonstrated the ability to resolve APIPA and Winsock corruption at Tier 1, avoiding unnecessary device re-imaging or hardware replacements.**
- **VERIFIED DOCUMENTATION:** **Created a clear standard operating procedure (SOP) for isolating network outages across local, gateway, and WAN environments.**

Author: Darnell Maiden | **Certifications:** CompTIA Security+, Google IT Support Professional Certificate, Google Cybersecurity Professional Certificate | **Target Environment:** Windows 10 Pro, VirtualBox / Hyper-V, Command Line Interface | **PORTFOLIO WEBSITE:** https://www.saintmaiden.site/