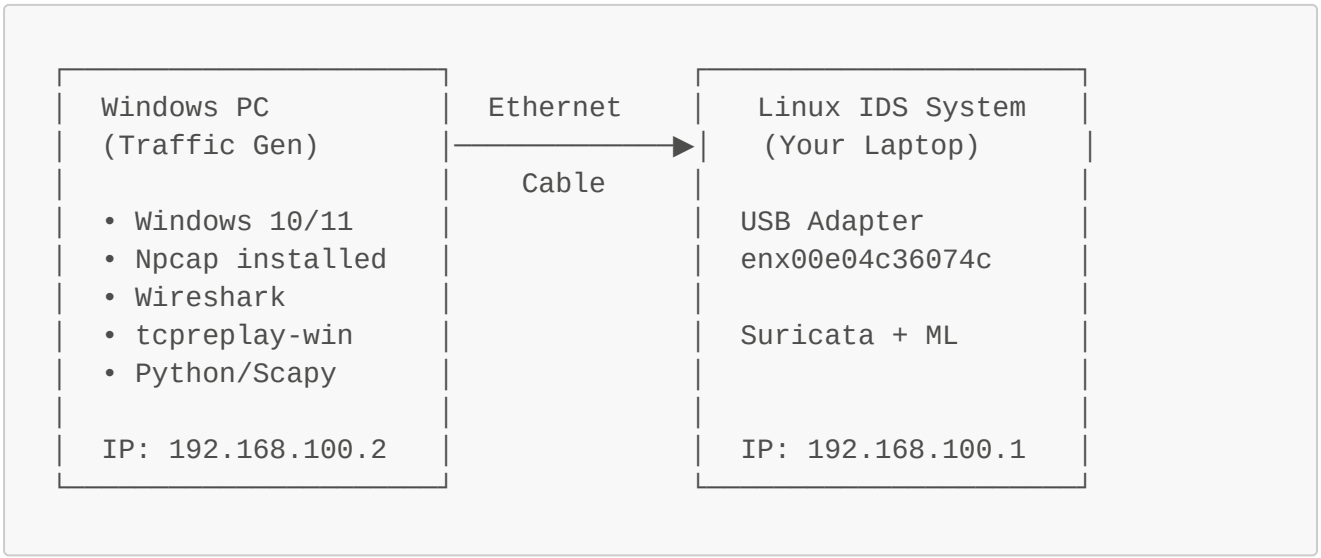
Windows External Device Setup Guide

🎯 Architecture Overview



🚀 Quick Start

Prerequisites on Windows

- Windows 10/11 (64-bit)
- Administrator access
- Ethernet port or USB Ethernet adapter
- Ethernet cable

📋 Step-by-Step Setup

PROF

Part 1: Windows Network Configuration

Step 1: Identify Your Network Interface

1. **Open PowerShell as Administrator**
   - Press `Win + X`
   - Select "Windows PowerShell (Admin)" or "Terminal (Admin)"
2. **List network adapters:**

```
Get-NetAdapter | Format-Table Name, InterfaceDescription, Status
```

3. **Note the adapter name** (e.g., "Ethernet", "Ethernet 2", etc.)

- This is usually "Ethernet" for wired connections
- If you have multiple adapters, identify the one you'll connect to the IDS

Step 2: Configure Static IP

Method A: Using PowerShell (Recommended)

```
# Replace "Ethernet" with your actual adapter name
$AdapterName = "Ethernet"

# Remove any existing IP configuration
Remove-NetIPAddress -InterfaceAlias $AdapterName -Confirm:$false -
ErrorAction SilentlyContinue
Remove-NetRoute -InterfaceAlias $AdapterName -Confirm:$false -
ErrorAction SilentlyContinue

# Set static IP
New-NetIPAddress -InterfaceAlias $AdapterName -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# Set DNS (optional, pointing to IDS)
Set-DnsClientServerAddress -InterfaceAlias $AdapterName -ServerAddresses
192.168.100.1

# Verify
Get-NetIPAddress -InterfaceAlias $AdapterName -AddressFamily IPv4
```

Method B: Using GUI

1. Open **Settings** → **Network & Internet** → **Ethernet**
2. Click on your Ethernet adapter
3. Click **Edit** next to "IP assignment"
4. Select **Manual**
5. Enable **IPv4**
6. Enter:
   - **IP address**: `192.168.100.2`
   - **Subnet prefix length**: `24`
   - **Gateway**: `192.168.100.1`
7. Click **Save**

PROF

Step 3: Test Connectivity

```
# Test ping to IDS system
ping 192.168.100.1

# Should get replies if IDS is configured and cable is connected
```

Part 2: Install Traffic Generation Tools

Option 1: Wireshark + tcpreplay-win (Recommended)

A. Install Npcap (Required for packet capture)

1. Download from: https://npcap.com/#download
2. Run installer as Administrator
3. **Important**: Check "Install Npcap in WinPcap API-compatible Mode"
4. Check "Support raw 802.11 traffic"
5. Complete installation
6. Restart Windows

B. Install Wireshark

1. Download from: https://www.wireshark.org/download.html
2. Run installer as Administrator
3. Accept all defaults
4. Complete installation

C. Install tcpreplay for Windows

1. Download from: https://github.com/appneta/tcpreplay/releases
    - Look for `tcpreplay-<version>-win64.zip`
2. Extract to `C:\tcpreplay\`
3. Add to PATH:

```
# Add tcpreplay to PATH
$env:Path += ";C:\tcpreplay"
[Environment]::SetEnvironmentVariable("Path", $env:Path,
[System.EnvironmentVariableTarget]::Machine)
```

4. Verify installation:

```
tcpreplay --version
```

Option 2: Python with Scapy

A. Install Python

1. Download Python 3.11+ from: https://www.python.org/downloads/
2. **Important**: Check "Add Python to PATH" during installation
3. Complete installation

PROF

B. Install Npcap (same as above)

C. Install Scapy

```
# Open PowerShell as Administrator
pip install scapy

# Verify
python -c "from scapy.all import *; print('Scapy installed
successfully')"
```

Option 3: hping (Windows Build)

1. Download from: https://github.com/antirez/hping/releases
2. Or use Windows Subsystem for Linux (WSL)

Option 4: Native Windows Tools

PowerShell with .NET (No extra software needed!)

```
# Built-in tools
# - Test-NetConnection (ping replacement)
# - Invoke-WebRequest (HTTP requests)
# - New-Object System.Net.Sockets.TcpClient (raw sockets)
```

Part 3: Traffic Generation Methods

Method 1: Replay PCAP with tcpreplay (Recommended)

PROF

```
# Basic replay
tcpreplay -i "\Device\NPF_{GUID}" capture.pcap

# Find your interface GUID:
getmac /v /fo list
# Look for your Ethernet adapter's Transport Name

# Or use Wireshark interface name
tcpreplay -i "Ethernet" capture.pcap

# Replay with speed control (10 Mbps)
tcpreplay -i "Ethernet" --mbps=10 capture.pcap

# Replay as fast as possible
tcpreplay -i "Ethernet" --topspeed capture.pcap
```

```
# Loop replay
tcpreplay -i "Ethernet" --loop=10 capture.pcap
```

Method 2: Python Scapy Scripts

Simple packet sender:

```
# save as send_traffic.py
from scapy.all import *
import time

# Target IDS
target = "192.168.100.1"
iface = "Ethernet"  # Your adapter name

print(f"Sending packets to {target}...")

# Send TCP SYN packets
for port in range(80, 100):
    packet = IP(dst=target)/TCP(dport=port, flags="S")
    send(packet, iface=iface, verbose=0)
    print(f"Sent SYN to port {port}")
    time.sleep(0.1)

print("Done!")
```

Replay PCAP:

```
# save as replay_pcap.py
from scapy.all import *
import sys

if len(sys.argv) < 2:
    print("Usage: python replay_pcap.py <pcap_file>")
    sys.exit(1)

pcap_file = sys.argv[1]
target = "192.168.100.1"
iface = "Ethernet"

print(f"Replaying {pcap_file} to {target}...")

# Read packets
packets = rdpcap(pcap_file)
print(f"Loaded {len(packets)} packets")

# Send packets
sendp(packets, iface=iface, verbose=1)
```

PROF

```
print("Replay complete!")
```

HTTP flood:

```
# save as http_flood.py
from scapy.all import *
import time

target = "192.168.100.1"
iface = "Ethernet"

print(f"Sending HTTP requests to {target}...")

for i in range(100):
    # HTTP GET request
    packet = IP(dst=target)/TCP(dport=80, flags="PA")/Raw(load="GET /
HTTP/1.0\r\n\r\n")
    send(packet, iface=iface, verbose=0)
    print(f"Sent request {i+1}")
    time.sleep(0.5)

print("Done!")
```

Run the scripts:

```
python send_traffic.py
python replay_pcap.py C:\path\to\capture.pcap
python http_flood.py
```

PROF

Method 3: PowerShell Native Scripts

TCP Connection Flood:

```
# save as tcp_flood.ps1
$target = "192.168.100.1"
$startPort = 80
$endPort = 100

Write-Host "Sending TCP connections to $target..."

for ($port = $startPort; $port -le $endPort; $port++) {
    try {
        $tcpClient = New-Object System.Net.Sockets.TcpClient
        $tcpClient.Connect($target, $port)
        Write-Host "Connected to port $port"
```

```
        $tcpClient.Close()
    } catch {
        Write-Host "Port $port - Connection failed (expected)"
    }
    Start-Sleep -Milliseconds 100
}

Write-Host "Done!"
```

HTTP Request Flood:

```
# save as http_flood.ps1
$target = "http://192.168.100.1"
$requests = 100

Write-Host "Sending HTTP requests to $target..."

for ($i = 1; $i -le $requests; $i++) {
    try {
        $response = Invoke-WebRequest -Uri $target -TimeoutSec 2 -
ErrorAction Stop
        Write-Host "Request $i - Status: $($response.StatusCode)"
    } catch {
        Write-Host "Request $i - Failed (expected)"
    }
    Start-Sleep -Milliseconds 500
}

Write-Host "Done!"
```

Port Scan Simulation:

PROF

```
# save as port_scan.ps1
$target = "192.168.100.1"
$ports = 1..1000

Write-Host "Scanning $target..."

foreach ($port in $ports) {
    $connection = Test-NetConnection -ComputerName $target -Port $port -
WarningAction SilentlyContinue
    if ($connection.TcpTestSucceeded) {
        Write-Host "Port $port is OPEN" -ForegroundColor Green
    } else {
        Write-Host "Port $port is closed" -ForegroundColor Red
    }
}
```

```
Write-Host "Scan complete!"
```

Run PowerShell scripts:

```
# Allow script execution (run once)
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

# Run scripts
.\tcp_flood.ps1
.\http_flood.ps1
.\port_scan.ps1
```

Method 4: Using nmap on Windows

Install nmap:

1. Download from: https://nmap.org/download.html
2. Run installer as Administrator
3. Complete installation

Run scans:

```
# Basic scan
nmap 192.168.100.1

# SYN scan (requires admin)
nmap -sS 192.168.100.1

# Full port scan
nmap -p- 192.168.100.1

# Aggressive scan
nmap -A 192.168.100.1

# OS detection
nmap -O 192.168.100.1
```

Method 5: Simple Traffic with curl/wget

Using curl (built into Windows 10+):

```
# Simple HTTP requests
for ($i=1; $i -le 100; $i++) {
    curl http://192.168.100.1
```

```
    Write-Host "Request $i sent"
    Start-Sleep -Milliseconds 100
}

# With different URLs (simulating web browsing)
$urls = @(
    "http://192.168.100.1/",
    "http://192.168.100.1/admin",
    "http://192.168.100.1/login",
    "http://192.168.100.1/api/users"
)

foreach ($url in $urls) {
    curl $url
    Write-Host "Accessed: $url"
    Start-Sleep -Seconds 1
}
```

Part 4: Attack Simulation Examples

1. DDoS Simulation (SYN Flood)

Using Scapy:

```
# syn_flood.py
from scapy.all import *
import random

target = "192.168.100.1"
target_port = 80
iface = "Ethernet"

print("Starting SYN flood...")

for i in range(1000):
    # Random source IP
    src_ip = f"192.168.{random.randint(1,254)}.{random.randint(1,254)}"

    # SYN packet
    packet = IP(src=src_ip, dst=target)/TCP(dport=target_port,
flags="S")
    send(packet, iface=iface, verbose=0)

    if i % 100 == 0:
        print(f"Sent {i} packets...")

print("Done!")
```

PROF

2. Port Scan

Using PowerShell:

```powershell
# fast_port_scan.ps1
$target = "192.168.100.1"
$ports = 1..10000

Write-Host "Scanning $target (fast mode)..."

$ports | ForEach-Object -Parallel {
    $tcpClient = New-Object System.Net.Sockets.TcpClient
    try {
        $tcpClient.ConnectAsync($using:target, $_).Wait(100)
        if ($tcpClient.Connected) {
            Write-Host "Port $_ is OPEN" -ForegroundColor Green
        }
    } catch {}
    $tcpClient.Close()
} -ThrottleLimit 100

Write-Host "Scan complete!"
```

3. HTTP Attack Patterns

SQL Injection Attempts:

```python
# sql_injection.py
from scapy.all import *
import urllib.parse

target = "192.168.100.1"
iface = "Ethernet"

# SQL injection payloads
payloads = [
    "' OR '1'='1",
    "admin'--",
    "1' OR 1=1--",
    "'; DROP TABLE users--"
]

print("Simulating SQL injection attempts...")

for payload in payloads:
    encoded = urllib.parse.quote(payload)
    http_request = f"GET /login?user={encoded} HTTP/1.0\r\n\r\n"

    packet = IP(dst=target)/TCP(dport=80,
```

PROF

```
flags="PA")/Raw(load=http_request)
    send(packet, iface=iface, verbose=0)
    print(f"Sent: {payload}")
    time.sleep(1)

print("Done!")
```

Using PowerShell:

```
# sql_injection.ps1
$target = "http://192.168.100.1/login"
$payloads = @(
    "' OR '1'='1",
    "admin'--",
    "1' OR 1=1--",
    "'; DROP TABLE users--"
)

foreach ($payload in $payloads) {
    $url = "$target?user=$([uri]::EscapeDataString($payload))"
    try {
        Invoke-WebRequest -Uri $url -TimeoutSec 2
    } catch {
        Write-Host "Sent SQL injection: $payload"
    }
    Start-Sleep -Seconds 1
}
```

4. Replay Attack PCAPs

Download sample attack PCAPs:

PROF

```
# Create directory for PCAPs
New-Item -ItemType Directory -Path "C:\attack_pcaps" -Force
cd C:\attack_pcaps

# Download sample PCAPs (examples)
# You can download from:
# - https://www.malware-traffic-analysis.net/
# - https://www.netresec.com/?page=PcapFiles
# - Your IDS project: ~/Programming/IDS/pcap_samples/

# Or transfer from your IDS system
# On IDS system: scp ~/Programming/IDS/pcap_samples/*.pcap user@windows-
ip:C:\attack_pcaps\
```

Replay with tcpreplay:

```
# Replay single PCAP
tcpreplay -i "Ethernet" C:\attack_pcaps\attack.pcap

# Replay all PCAPs in directory
Get-ChildItem C:\attack_pcaps\*.pcap | ForEach-Object {
    Write-Host "Replaying: $($_.Name)"
    tcpreplay -i "Ethernet" --mbps=10 $_.FullName
    Start-Sleep -Seconds 5
}
```

🔧 Troubleshooting

Issue: Can't ping IDS system (192.168.100.1)

Check Windows Firewall:

```
# Temporarily disable firewall for testing
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False

# Or allow ICMP
New-NetFirewallRule -DisplayName "Allow ICMPv4" -Direction Outbound -
Protocol ICMPv4 -Action Allow
New-NetFirewallRule -DisplayName "Allow ICMPv4" -Direction Inbound -
Protocol ICMPv4 -Action Allow

# Re-enable firewall when done
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True
```

Check cable connection:

PROF

```
Get-NetAdapter | Where-Object {$_.Name -eq "Ethernet"} | Select-Object
Status, LinkSpeed
# Should show: Status = Up, LinkSpeed = 1 Gbps or 100 Mbps
```

Check IP configuration:

```
Get-NetIPAddress -InterfaceAlias "Ethernet" -AddressFamily IPv4
# Should show: IPv4Address = 192.168.100.2, PrefixLength = 24
```

Issue: tcpreplay not working

Find correct interface name:

```
# List interfaces
getmac /v /fo list

# Use with tcpreplay
tcpreplay -i "\Device\NPF_{YOUR-GUID-HERE}" capture.pcap
```

Or use Npcap's dpkt:

```
# List interfaces with dpkt
"C:\Program Files\Npcap\dpcap.exe" -D
```

Issue: Scapy can't find interface

Verify Npcap is installed:

```
Get-Service npcap
# Should show: Status = Running
```

Check Scapy can see interfaces:

```
from scapy.all import *
conf.ifaces
# Should list all interfaces
```

Use correct interface name:

PROF

```
# Try these variations:
iface = "Ethernet"
# or
iface = r"\Device\NPF_{GUID}"
# or
iface = get_if_list()[0]  # Use first interface
```

Issue: Permission denied errors

Run PowerShell as Administrator:

1. Right-click PowerShell
2. Select "Run as Administrator"

For Python scripts:

```
# Run Python as admin
Start-Process python -ArgumentList "script.py" -Verb RunAs
```

Issue: No packets reaching IDS

Verify ARP table:

```
arp -a | Select-String "192.168.100"
# Should show entry for 192.168.100.1
```

Force ARP refresh:

```
arp -d  # Clear ARP cache
ping 192.168.100.1  # Repopulate ARP
```

Check with Wireshark:

1. Open Wireshark
2. Capture on "Ethernet" interface
3. Send traffic
4. Verify packets are leaving your Windows PC

📊 Monitoring on Windows

View Network Statistics

PROF

```
# Real-time packet stats
Get-NetAdapterStatistics -Name "Ethernet"

# Watch continuously
while ($true) {
    Clear-Host
    Get-NetAdapterStatistics -Name "Ethernet" | Format-List
    Start-Sleep -Seconds 1
}
```

Packet Capture with Wireshark

1. Open Wireshark
2. Select "Ethernet" interface
3. Start capture
4. Apply filter: `ip.dst == 192.168.100.1`

5. Verify your traffic is going to IDS

PowerShell Logging

```
# Log all sent packets
$logFile = "C:\traffic_log.txt"

# Add to your traffic generation scripts:
"[$(Get-Date)] Sent packet to $target" | Out-File -Append $logFile
```

🎮 Complete Workflow

Terminal 1: Configure Network

```
# Run as Administrator
$AdapterName = "Ethernet"

# Configure IP
New-NetIPAddress -InterfaceAlias $AdapterName -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# Test
ping 192.168.100.1
```

Terminal 2: Generate Traffic

Option A: Replay PCAP

```
tcpreplay -i "Ethernet" --mbps=10 C:\attack_pcaps\ddos.pcap
```

PROF

Option B: Run Python script

```
python http_flood.py
```

Option C: Run PowerShell script

```
.\port_scan.ps1
```

Terminal 3: Monitor (Optional)

```
# Watch network stats
while ($true) {
    Clear-Host
    Write-Host "Network Statistics:" -ForegroundColor Cyan
    Get-NetAdapterStatistics -Name "Ethernet" | Format-Table
    Start-Sleep -Seconds 2
}
```

📚 Sample Attack Scripts Collection

Save these to `C:\attack_scripts\`:

1. multi_attack.ps1 (All-in-one)

```
# Multi-attack simulation
$target = "192.168.100.1"

Write-Host "=== Multi-Attack Simulation ===" -ForegroundColor Cyan
Write-Host "Target: $target`n"

# 1. Port scan
Write-Host "[1] Port Scanning..." -ForegroundColor Yellow
1..100 | ForEach-Object {
    Test-NetConnection -ComputerName $target -Port $_ -WarningAction
SilentlyContinue | Out-Null
}

Start-Sleep -Seconds 5

# 2. HTTP flood
Write-Host "[2] HTTP Flood..." -ForegroundColor Yellow
1..50 | ForEach-Object {
    try { Invoke-WebRequest -Uri "http://$target" -TimeoutSec 1 } catch
{}
}

Start-Sleep -Seconds 5

# 3. SQL injection attempts
Write-Host "[3] SQL Injection Attempts..." -ForegroundColor Yellow
$sqli = @("' OR '1'='1", "admin'--", "1' OR 1=1--")
foreach ($payload in $sqli) {
    try {
        $url = "http://$target/login?
user=$([uri]::EscapeDataString($payload))"
        Invoke-WebRequest -Uri $url -TimeoutSec 1
    } catch {}
    Start-Sleep -Milliseconds 500
```

```
}

Write-Host "`n=== Attack simulation complete ===" -ForegroundColor Green
```

2. continuous_traffic.ps1 (Background traffic)

```
# Continuous background traffic
$target = "192.168.100.1"

Write-Host "Starting continuous traffic to $target..." -ForegroundColor
Cyan
Write-Host "Press Ctrl+C to stop`n"

while ($true) {
    # Random traffic type
    $rand = Get-Random -Minimum 1 -Maximum 4

    switch ($rand) {
        1 {
            # HTTP request
            try { Invoke-WebRequest -Uri "http://$target" -TimeoutSec 1
} catch {}
            Write-Host "." -NoNewline -ForegroundColor Green
        }
        2 {
            # TCP connection
            $port = Get-Random -Minimum 80 -Maximum 100
            try {
                $tcp = New-Object System.Net.Sockets.TcpClient
                $tcp.Connect($target, $port)
                $tcp.Close()
            } catch {}
            Write-Host "." -NoNewline -ForegroundColor Yellow
        }
        3 {
            # Ping
            Test-Connection -ComputerName $target -Count 1 -Quiet | Out-
Null
            Write-Host "." -NoNewline -ForegroundColor Cyan
        }
    }

    Start-Sleep -Milliseconds (Get-Random -Minimum 100 -Maximum 1000)
}
```

PROF

🎯 Testing Checklist

- [ ] Windows network adapter configured with 192.168.100.2

- [ ] Ethernet cable connected to IDS system
- [ ] Can ping 192.168.100.1
- [ ] Npcap installed (for packet tools)
- [ ] At least one traffic generation tool installed:
  - [ ] tcpreplay-win
  - [ ] Python + Scapy
  - [ ] PowerShell scripts ready
- [ ] Firewall configured to allow traffic
- [ ] IDS system is running and ready
- [ ] Test traffic reaches IDS (check with tcpdump on IDS)

🎓 Summary

Component	Configuration
Windows IP	192.168.100.2/24
Gateway	192.168.100.1 (IDS)
Interface	Ethernet (built-in or USB)
Required Software	Npcap
Optional Tools	Wireshark, tcpreplay, Python/Scapy, nmap
Easiest Method	PowerShell scripts (built-in!)

Quick Commands Reference

```
# Network setup
New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# Test connectivity
ping 192.168.100.1

# Simple traffic generation (no install needed!)
1..100 | ForEach-Object { curl "http://192.168.100.1" }

# With tcpreplay
tcpreplay -i "Ethernet" --mbps=10 attack.pcap

# With Python/Scapy
python replay_pcap.py attack.pcap
```

PROF

📞 Next Steps

1. ✅ Configure Windows network: Run PowerShell commands above

2. ✅ Connect Ethernet cable to IDS system
3. ✅ Test ping: `ping 192.168.100.1`
4. ✅ Choose traffic generation method
5. ✅ Start generating traffic
6. ✅ Watch IDS detect attacks!

Your Windows PC is now ready to generate attack traffic! 🚀