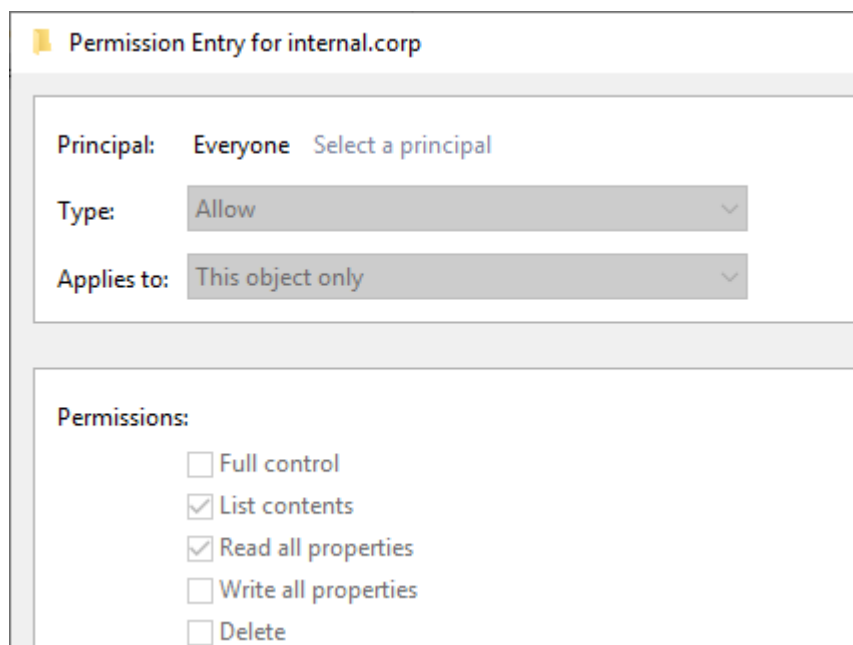


Getting in the Zone: dumping Active Directory DNS using adidnsdump

dirkjanm.io/getting-in-the-zone-dumping-active-directory-dns-with-adidnsdump

April 25, 2019



Permission Entry for internal.corp

Principal: Everyone [Select a principal](#)

Type: Allow

Applies to: This object only

Permissions:

- ☐ Full control
- ☒ List contents
- ☒ Read all properties
- ☐ Write all properties
- ☐ Delete

🕒 5 minute read

Zone transfers are a classical way of performing reconnaissance in networks (or even from the internet). They require an insecurely configured DNS server that allows anonymous users to transfer all records and gather information about host in the network. What not many people know however is that if Active Directory integrated DNS is used, any user can query all the DNS records by default. This blog introduces a tool to do this and describes a method to do this even for records normal users don't have read rights for.

Knowing where you are and where to go

Personally whenever I arrive at a new pentest or red team assignment I want to learn about the layout of the network, the software in use and where the interesting data is. If a company has non-descriptive server names or descriptions, tools like BloodHound or ldapdomaindump are not going to help much since `SRV00001.company.local` still doesn't tell me what runs on this host. Running discovery tools like EyeWitness on a large range of IP addresses often returns a lot of default Apache/IIS pages, since most sites are configured to listen on a DNS name and not on the IP address. Knowing that `gitlab.company.local` also points to the same IP as `SRV00001.company.local` tells me that this is an interesting server if I'm after source code. Having access to DNS entries for

AD is thus in my opinion quite valuable. Thus I wrote this small tool that can dump those records. You can either run it directly from a host within the network, or through a SOCKS tunnel using your favourite implant.

Prior work

This started when I was looking at Active Directory DNS, mostly inspired by [Kevin Robertson's work on ADIDNS](#). I tried to figure out how AD uses zones in LDAP for storing DNS records as I pulled up ADSI Edit and suddenly saw an overview of all the DNS records in the domain, using only a limited regular user. As I shared my surprise, Kevin pointed out to me that [mubix](#) already wrote about this [back in 2013](#). So there was already a PowerShell script that could do this, but it didn't do exactly what I wanted, so I decided to write a version in Python and add some options to enumerate more records than possible by default. **Edit:** [@3xocyte](#) also wrote a similar version in Python [here](#).

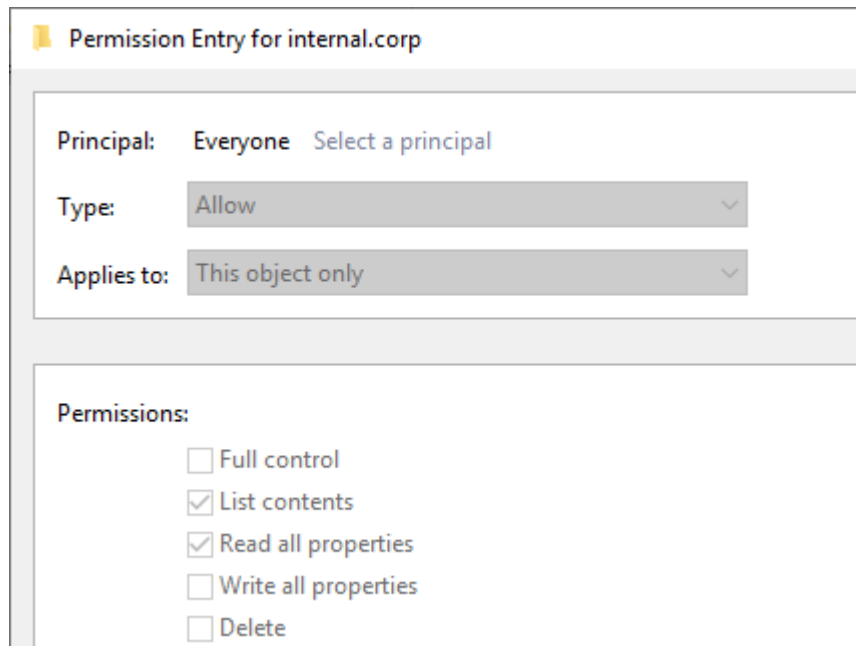
The “hidden” DNS records

The most obvious way to query for DNS records in LDAP would be to perform a query selecting all objects of the class `dnsNode`, which represent entries in the DNS zone. When I performed a query with the filter `(objectClass=dnsNode)`, this returned quite limited results, even though I could see several more records when manually browsing to the DNS zone:

DC=icorp-dc	dnsNode	DC=icorp-dc,DC=
DC=ICORP-W10		DC=ICORP-W10,I
DC=newhost	dnsNode	DC=newhost,DC=
DC=notinternal		DC=notinternal,D
DC=test	dnsNode	DC=test,DC=inter
DC=testpm	dnsNode	DC=testpm,DC=ii

As visible on the image above, for several objects the `objectClass` is not visible. This is because of the default permissions on computer DNS records (and I think on other records not created via the AD DNS gui as well), which don't allow all users to see the contents. Since the IP addresses are actually stored as a property of this object, it isn't possible to view the IP address for these records either.

But like any user can create new DNS records by default, any user can also list the child objects of a DNS zone by default. So we know a records is there, we just can't query it using LDAP.



Permission Entry for internal.corp

Principal: Everyone [Select a principal](#)

Type: Allow

Applies to: This object only

Permissions:

- ☐ Full control
- ☒ List contents
- ☒ Read all properties
- ☐ Write all properties
- ☐ Delete

Once we know a records exists by enumerating with LDAP, we can however query for it using DNS directly (since performing regular DNS queries doesn't require privileges). This way we can resolve all records in the zone.

Querying records with adidnsdump

With adidnsdump, which you can get [from my GitHub](#), it is possible to enumerate all records in the DNS zone. To get started, first display the zones in the domain where you are currently in with `--print-zones`. This will show which zones are present. Not all zones are interesting, for example forward, cache and stub zones don't contain all the records for that domain. If you find these zones, it's better to query the domain to which they actually belong. The output below shows that my test domain has only the default zones:

```
user@localhost:~/adidnsdump$ adidnsdump -u icorp\\testuser --print-zones icorp-dc.internal.corp
Password:
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Found 2 domain DNS zones:
    internal.corp
    RootDNSServers
[-] Found 2 forest DNS zones:
    ..TrustAnchors
    _msdcs.internal.corp
```

If we specify the zone to the tool (or leave it empty for the default zone), we will get a list of all the records. Records which can be listed but not read (so called "hidden" records) are shown but only with a question mark, as it is unknown which type of record is present and where it points to. The records are all saved to a file called `records.csv`.

```
(adidnsdump-4XiJn7UR) dirkjan@ubuntu:~/adidnsdump$ adidnsdump -u icorp\\testuser icorp-dc.internal.corp
Password:
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Querying zone for records
[+] Found 17 records
(adidnsdump-4XiJn7UR) dirkjan@ubuntu:~/adidnsdump$ head records.csv
type,name,ip
A,wpad,10.1.1.2
A,wpad,10.1.1.1
A,testpwm,192.168.111.12
A,testpm,192.168.111.12
A,test,10.0.0.10
?,notinternal,?
A,newhost,10.1.1.1
?,ICORP-W10,?
```

To resolve the unknown records, specify the `-r` flag, which will perform an `A` query for all unknown records (you can easily change this to `AAAA` in the code if you're in an IPv6 network). Several nodes which were blank before now suddenly have records:

```
(adidnsdump-4XiJn7UR) dirkjan@ubuntu:~/adidnsdump$ adidnsdump -u icorp\\testuser icorp-dc.internal.corp -r
Password:
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Querying zone for records
[-] Could not resolve node hoi (probably no A record assigned to name)
[+] Found 17 records
(adidnsdump-4XiJn7UR) dirkjan@ubuntu:~/adidnsdump$ head records.csv
type,name,ip
A,wpad,10.1.1.2
A,wpad,10.1.1.1
A,testpwm,192.168.111.12
A,testpm,192.168.111.12
A,test,10.0.0.10
A,notinternal,95.179.182.12
A,newhost,10.1.1.1
A,ICORP-W10,192.168.111.73
```

If you don't have a direct connection but are working via an agent, you can proxy the tool through socks and perform the DNS queries over TCP with the `--dns-tcp` flag.

Mitigations

You shouldn't really rely on secrecy of your DNS records for security. If you really want to hide this information, removing the "List contents" permission for "Everyone" and "Pre-Windows 2000 Compatible Access" does prevent regular users from querying the entries, but this does require disabling inheritance on the DNS zone and may break stuff, so I don't really recommend going that way. Monitoring for high volumes of DNS queries or enabling auditing on DNS zone listings may be a better way to deal with this, by detecting instead of blocking this kind of activity.

The tools

adidnsdump is available on [GitHub](#) and on PyPI (`pip install adidnsdump`). Right now the tool only dumps records to CSV files, but feel free to submit requests for alternate formats.

All blog content is available under the [Creative Commons BY 4.0 License](#) unless stated otherwise.

Powered by Jekyll and a modified version of the "Minimal Mistakes" theme.