Using Let's Encrypt for Active Directory Domain Controller Certificates

If you've ever had to setup an HTTPS website in the past couple years, you've most likely heard of Let's Encrypt which is arguably the largest public certificate authority in the world. Not only are their certificates free, the entire ordering and renewal process can be completely automated using a recently finalized protocol standard known as ACME (RFC 8555).

But there are endless apps and services other than HTTP based web sites that can also use TLS certificates. On Windows specifically, there are things like Remote Desktop (RDP), SQL Server, WinRM, Exchange, and Active Directory. Many folks don't realize that the certificates you get from Let's Encrypt can be used for these other services as well and they don't even need to be exposed to the Internet as long as the domain name is real and obtained from a public domain registrar. This can be a huge operational win for smaller organizations who don't have the resources or expertise to design, implement, and maintain an internal Public Key Infrastructure (PKI).

Active Directory is historically a bit more picky about certs than some other services and ever since I wrote Posh-ACME, I've been curious if Let's Encrypt certs would work with it. As it turns out, they work great with a couple minor caveats.

Certificate Transparency and Security Through Obscurity

All publicly trusted CAs in 2019 including Let's Encrypt must now adhere to the Certificate Transparency standard which is a "cryptographically assured, publicly auditable, append-only record" of issued certificates. In other words, all public certificates are now logged and searchable by the general public. So if your security team demands that your internal server or domain names remain a secret, you simply can't use any public CA for internal certificates. Be sure to get approval from those folks before continuing with this if you have any doubt.

Even if you can't or won't use public certs for your production AD, they're still super handy for dev/test/lab domains.

Using Public Certs for Internal Services

In order to get a certificate from a public CA like Let's Encrypt, the FQDN in the cert must be part of a domain that was obtained from an ICANN recognized domain registrar. If your internal domains end in TLDs like `.local` or `.int`, you're out of luck. You're also more likely to run into future problems like everyone who was using `.dev` until Google purchased it. So if you're doing that, stop it.

It might seem obvious, but the CA will also need you to prove control/ownership of the domain or names in your certificate. Generally, this means being able to modify the public DNS records for it. For Let's Encrypt and any other ACME capable CA, you need the ability to create TXT records.

Let's say your domain is `example.com` and your internal service is on a machine with the FQDN `dc.ad.example.com`. In order to get a certificate for that name, you would have to create a TXT record called `_acme-challenge.dc.ad.example.com` with a special value in the *public facing* DNS zone for `example.com`. Any machine *outside your network* should be able to do a standard DNS query like this and have the special value returned.

```
C:\>nslookup -q=TXT _acme-challenge.dc.ad.example.com.
```

Finally, the machine running the ACME client must be able to reach the ACME server on the Internet. If your environment blocks outbound Internet access, you'll need to either move the ACME client to a whitelisted host or add exceptions for the ACME server URLs to your outbound firewall or proxy server. For Let's Encrypt, that would be `acme-v02.api.letsencrypt.org` for the Production endpoint and `acme-staging-v02.api.letsencrypt.org` for the Staging endpoint. Having both available is very useful for troubleshooting.

Active Directory and Certificates

Adding TLS certificates to your Active Directory domain controllers has been a recommended practice for a long while now. One of the primary benefits is enabling LDAPS (LDAP over SSL) which prevents exposing cleartext credentials on the wire for legacy applications who still need to use basic BINDs. This will become increasingly important in 2020 when Microsoft changes the LDAP defaults to require channel binding and signing. Even for clients who use more modern BIND methods like Kerberos with SASL, it will protect the confidentiality of the LDAP query traffic which standard LDAP does not.

Here is Microsoft's official guidance on obtaining domain controller certificates from a third-party CA and enabling LDAP over SSL.

- Requirements for domain controller certificates from a third-party CA
- How to enable LDAP over SSL with a third-party certification authority

There are two main things we care about from those docs:

- Each DC's cert must contain its own FQDN (dc.example.com) and the domain's FQDN (example.com).
- The cert should be installed in the local computer's Personal certificate store

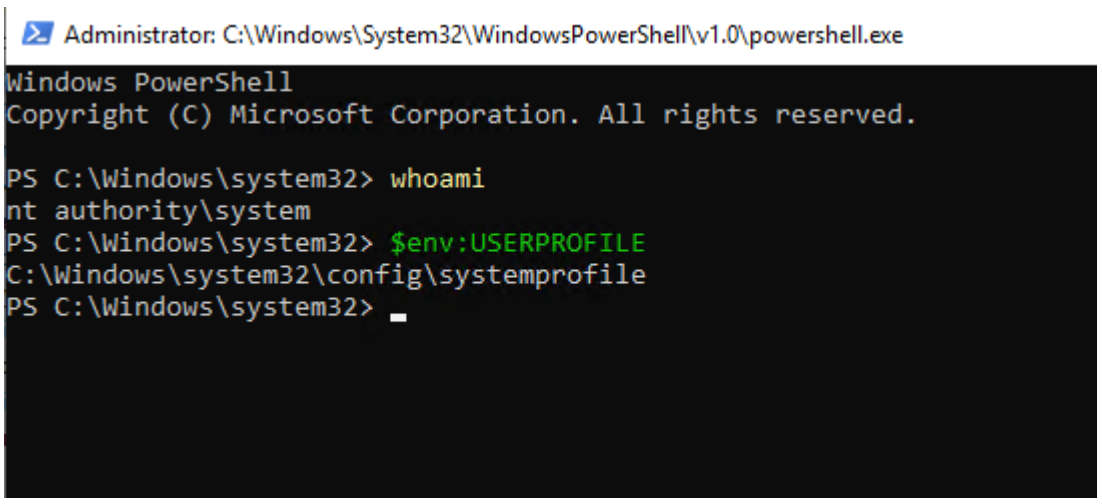
Domain Controller Prep

For this demo, we'll be using a freshly installed Windows Server 2019 domain controller, `dcle`, in a domain called `ad.poshacme.online`. Server 2019 comes pre-installed with the necessary Posh-ACME prerequisites. But if you're on an earlier OS, make sure you have PowerShell 5.1 and .NET 4.7.1 or later. Posh-ACME has been installed into the default system-wide module path, `C:\Program Files\WindowsPowerShell\Modules`. See the readme for help with installation.

We're going to run Posh-ACME in the context of the local SYSTEM account so it has permission to modify the local computer's certificate store and restart services. *In practice, any administrative account will do. Just make sure to run your PowerShell session elevated.*

Using PsExec, open PowerShell as `NT AUTHORITY\SYSTEM` and verify your context using `whoami` and `$env:USERPROFILE`.

```
.\PsExec64.exe -i -h -s powershell.exe
```



Getting the Certificate

If you're not familiar with how Posh-ACME works, I'd suggest going through the tutorial first. This guide will assume you're already familiar with the basics of getting a cert using a DNS plugin.

I currently host the public facing DNS zone for this `poshacme.online` domain at Digital Ocean, so I'll be using the `DOcean` DNS plugin for this order. Any plugin will work, but the Manual plugin will prevent automatic renewal because it requires human interaction.

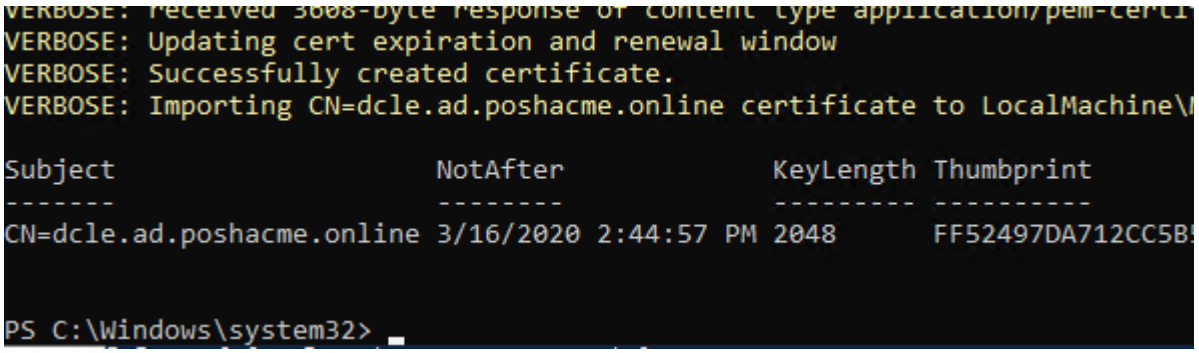
Prep the plugin arguments for the DNS plugin, generate the list of names we'll need in the cert, and set the email address for expiration notification.

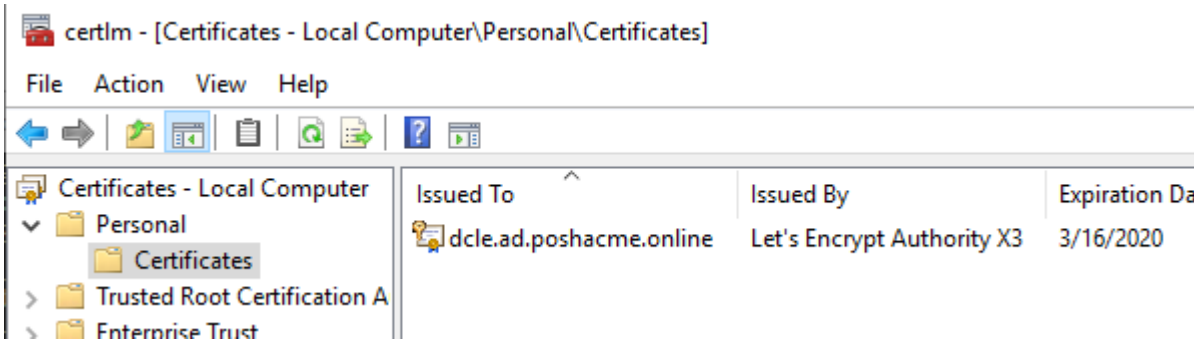
```
# Digital Ocean requires a simple API token
$pArgs = @{DOToken='xxxxxxxxxxx'}

# The ActiveDirectory PowerShell module is installed by default on DCs
$dc = Get-ADDomainController $env:COMPUTERNAME
$certNames = @($dc.HostName, $dc.Domain)

# This is optional, but usually a good idea.
$notifyEmail = 'myaddress@poshacme.online'
```

Build a hashtable of parameters and use them with `New-PACertificate` via splatting. Make sure you include the `-Install` parameter so the resulting certificate gets installed to the local computer's certificate store.

```
$certParams = @{
    Domain = $certNames
    DnsPlugin = 'DOcean'
    PluginArgs = $pArgs
    AcceptTOS = $true
    Install = $true
    Contact = $notifyEmail  # optional
    Verbose = $true         # optional
}

New-PACertificate @certParams
```

If all went well, you should see the new cert details in the PowerShell output and the certificate should show up in the `Local Computer\Personal` certificate snap-in. Open it by running `certlm.msc`.

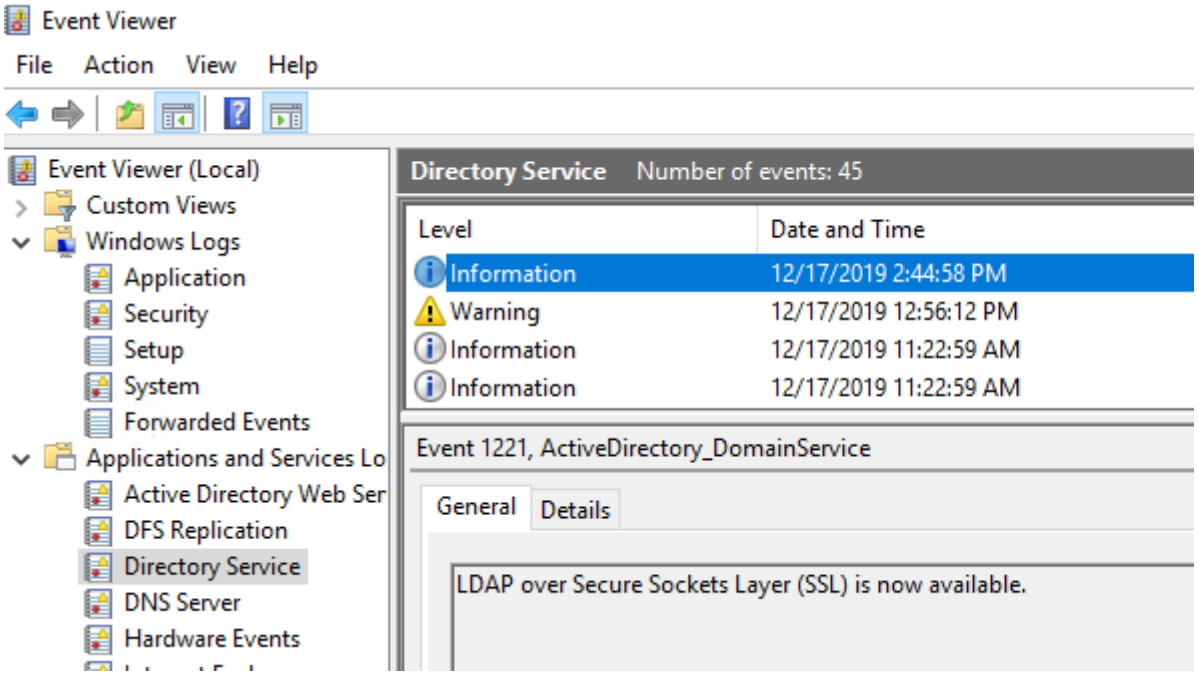




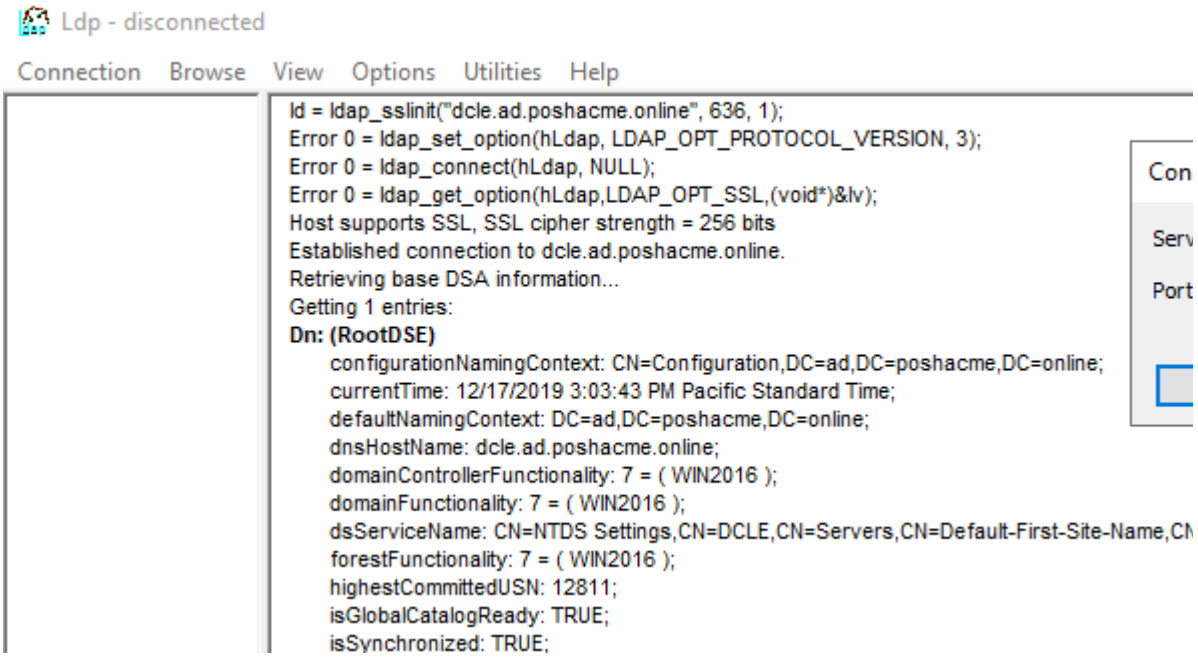
Verifying LDAPS and ADWS

The nice thing about domain controller certs is that LDAPS should immediately be functional with no service restarts. But how do we know it's working?

If the DC did not previously have a cert, the `Directory Service` event log should contain an Event ID 1221 confirming LDAPS is now working. However, no events are generated on a cert renewal or replacement.

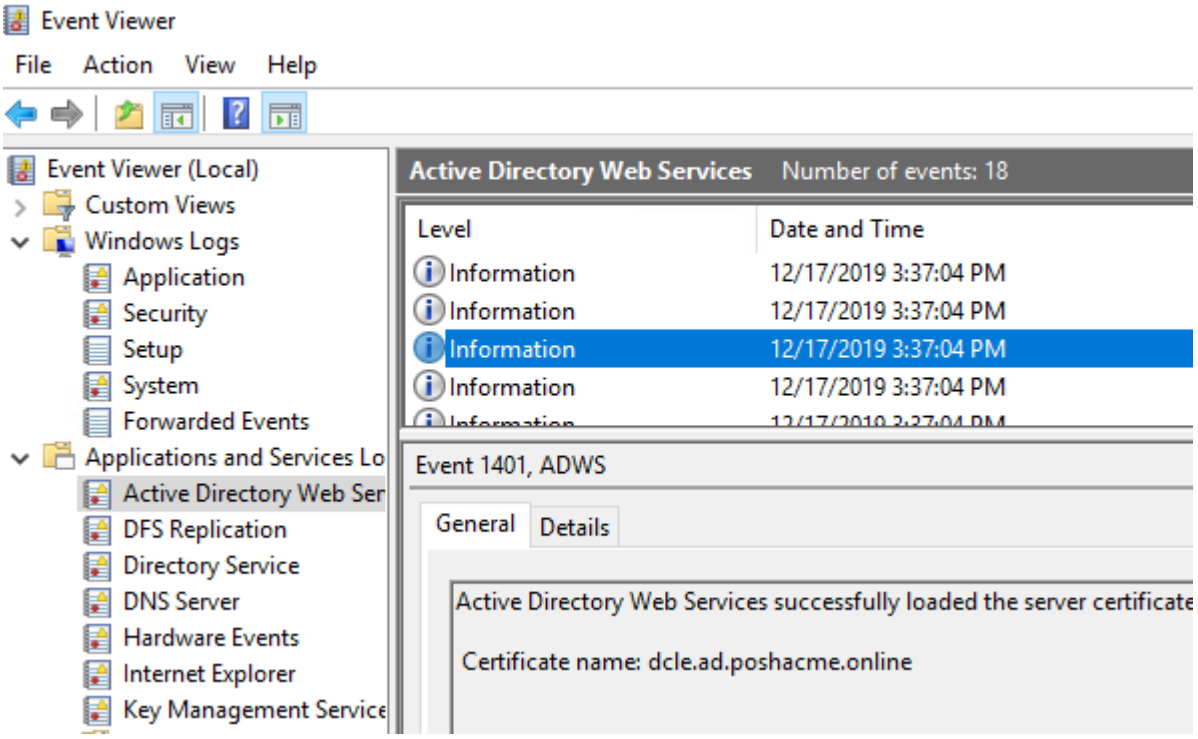


Another good way to test is using the native `ldp.exe` utility. Run it and select `Connection - Connect` specifying the name of your DC, 636 as the port, and check the SSL box. Then press OK. You should see some successful connection details and then a bunch of LDAP info from the RootDSE object.



If you want to see the specific details of the certificate being used, there are scripts such as this one that will attempt a connection to your DC and parse those details from the TLS conversation.

In addition to LDAPS, Active Directory Web Services (ADWS) will also use this new certificate. The PowerShell ActiveDirectory module (among other things) uses this service rather than raw LDAP to communicate with AD. However, it requires a service restart to recognize and use the new certificate. You can do that via the Services MMC snap-in or run `Restart-Service ADWS`. When done, you should find Event ID 1401 in the associated event log which confirms it has successfully loaded the certificate.



Expiration and Renewal

Let's Encrypt certificates are only valid for 90 days. While you could manually repeat this process shortly before your cert expires every 70-80 days, it's much

less hassle to setup a scheduled task that will renew the certificate automatically. If you configured a notification email with the certificate order, Let's Encrypt will email you if the cert hasn't been renewed starting about 20 days before it expires. So once you have the renewal process automated, you can largely forget about it.

Posh-ACME's `Submit-Renewal` is designed to be run on a regular (daily) basis. It will only act when the suggested renewal window has been reached for a certificate and it will return the details for the new certificate if successful. There are a few things we want to accomplish in our scheduled task.

- Run `Submit-Renewal`
- If we got a new cert:
  - Delete the old certificate
  - Restart the ADWS service
- Log everything for future diagnostic purposes

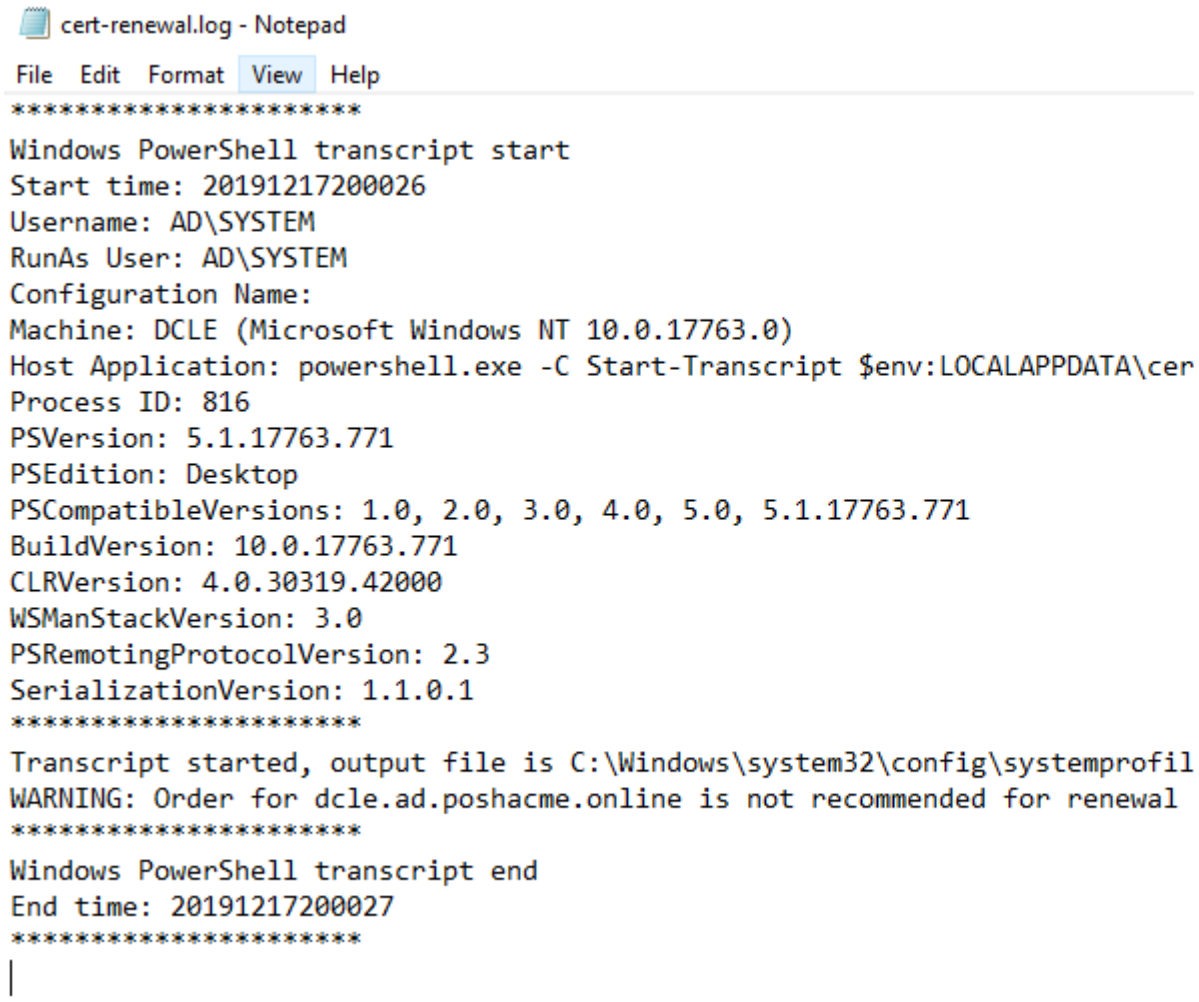
Assuming no other Posh-ACME certificates have been ordered on this machine, our renewal script might look something like this.

```
Start-Transcript $env:LOCALAPPDATA\cert-renewal.log -Append
$hostname = (Get-ADDomainController $env:COMPUTERNAME).HostName
$oldCert = Get-ChildItem Cert:\LocalMachine\My |
    Where-Object { $_.Subject -eq "CN=$hostname" } |
    Sort-Object -Descending NotAfter |
    Select-Object -First 1
if (Submit-Renewal -Verbose) {
    $oldCert | Remove-Item
    Restart-Service ADWS
}
Stop-Transcript
```

Now we'll throw a slightly minified version of that code into a scheduled task and we're good to go.

```
$taskname = "Renew DC Certificate"
$taskdesc = "Renews the Let's Encrypt certificate installed on this domain contro
$actionArg = '-C "Start-Transcript $env:LOCALAPPDATA\cert-renewal.log -Append; $na
$action = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument $actionArg
$trigger =  New-ScheduledTaskTrigger -Daily -At 2am -RandomDelay (New-TimeSpan -M
$settings = New-ScheduledTaskSettingsSet -ExecutionTimeLimit (New-TimeSpan -Minute
Register-ScheduledTask $taskname -Action $action -Trigger $trigger -User 'System'
```

After running your new task, you should find the `cert-renewal.log` in `C:\Windows\System32\config\systemprofile\AppData\Local` unless you customized the path in the `Start-Transcript` call.



The transcript logging is optional. Some may want to skip it in favor of sending an email instead when things successfully renew or add additional error handling and only send an email if there's an error. It's all just PowerShell, so the possibilities are endless.

Wrapping Up

I realize this solution is probably not realistic for an environment with more than a handful of DCs. Large environments are better suited to spin up a real PKI solution and use manually installed certs with multi-year expirations or native Windows auto-enrollment if the internal CA is Microsoft based. An internal CA also has the benefit of being able to use alternative name fields like IP address and GUID which are useful for certain edge cases.

However, there are certainly ways to scale this if you really wanted to. And the resource and maintenance benefits of not having to run an internal CA are huge for small teams. I wouldn't be surprised if there were small orgs already doing this with traditional purchased certificates.

Regardless, I had a lot of fun exploring this topic and hope it shed some light on the versatility of Posh-ACME and alternative use cases for Let's Encrypt certificates. Let's Encrypt is an amazing organization that is truly making the web a more secure place for everyone. Consider becoming a donor or sponsor if you'd like to help them with that goal.