Automated Certificate Management

Digital Certificates

Digital certificates are used by websites. They have two purposes:

1. Website application domain identification
2. Channel privacy between web browsers and web servers

A trusted digital certificate guarantees that the website being browsed isn't fake. The same certificate also securely encrypts all data exchanged between the web browser and the web server - thereby preventing eavesdropping by malicious third parties.

The Problem

Digital certificates have an expiration date. They must be replaced periodically. Replacing an enterprise certificate can be quite a chore.

Until recently it's been common for large enterprises to purchase a single "wildcard" certificate for all their application domains. The enterprise would do this every two to five years. The relatively high cost of a wildcard certificate was only part of the problem.

Whenever a certificate needed replacement, teams of technology professionals (system admins, project managers, app owners, developers, and QA testers) - typically dozens of them - would huddle together during a 1am conference call. Collectively they would manually distribute a new certificate to each of hundreds of servers, install it (often via a shared password – not very secure), and then bind it - sometimes to multiple IP addresses and ports per server.

This procedure would take hours and would be error prone. As a result, significant testing would also be required.

It was bad enough having to do every two years or so, but now - to limit the risk of a stolen certificate private key - no reputable certificate authority will sell a certificate that lasts for more than 13 months.

Finally, the latest security best practices dictate that wildcard certificates are no longer safe to use on the public Internet. That means if your enterprise has a hundred or more domains that previously shared a single wildcard certificate, the "Whenever a certificate needed replacement..." procedure would be performed a hundred or more times (once for each domain - with a different certificate specific to that domain only) – all repeated at least once per year.

Bottom line: *manual digital certificate management is now unmanageable*.

Features of AutoCert Automation

- Monitors certificate expiration status on every Windows server every day.
- Acquires certificates free-of-charge using industry standard ACME protocol.
- Each certificate supports up to 100 “subject alternative names” (the SAN list).
- Securely distributes certificates anywhere in the enterprise (including Linux [1]).
- Never shares certificate passwords with anyone.
- Installs and port binds new certificates in default and/or customizable ways.
- Securely archives new certificates locally for future use.
- Securely distributes new certificates to your F5 or any other load balancer.
- System self-corrects and retries intelligently when anything goes wrong.
- Lists of people can be notified if an issue requires human intervention.

Who Benefits?

Have you heard about well-known companies brought to their knees - in a very public way - after their digital certificate expired? [2]

Medium to large enterprises with Windows-based server farms benefit from ACM by no longer having to burden their already overworked IT teams with the esoteric and potentially error-prone task of periodically replacing digital certificates on hundreds of servers - and all that entails. This is especially true if the enterprise uses an SSO solution (single-sign-on, like ADFS) for its critical web applications.

The real benefit is not having to worry about expiring digital certificates ever again.

Competition & Risk

Companies offering similar services are some of the existing digital certificate authorities. As they lose paying customers to free certificates made available by consortiums like ISRG, they are offering more benefits with their paid certificate subscriptions. That said, their costs and their prices are much higher, yet they offer nowhere near the features and flexibility of **AutoCert**.

The main risk that comes with **AutoCert** is the possibility that **Let’s Encrypt** might someday decide to charge for its certificates. That seems highly unlikely since zero cost certificates are in the very first bullet of **Let’s Encrypt’s** statement of key principles. If that were to ever happen, **AutoCert** can be configured to use certificates provided by any other certificate authority.

One risk is minimized by limiting access to the enterprise-wide repository of digital certificates. This is accomplished by using a jointly secured **SCS** (Secure Certificate Service) appliance dedicated to the enterprise.

The "**AutoCert SCS Appliance**" is a virtual machine (VM) with encrypted storage that can run within the enterprise's virtualization infrastructure (whether in the cloud or on premises).

While the **SCS** VM is Internet accessible exclusively by **AutoCert**, it can also be secured within the confines of the enterprise's firewall. That means the enterprise has complete and strict control over the network traffic in and out of the **SCS** VM.[3]

Operating Model

AutoCert is a subscription service. The **AutoCert.exe** agent software is provided to the customer free-of-charge while **AutoCert** installs the **SCS** software on the customer's own virtual machine (VM) – a certificate repository VM that only **AutoCert** has internal access to.[3]

The customer may define as many domains as is needed by the enterprise (ie. separate unique certificates). The customer may also install **AutoCert.exe** on any number of corporate servers as needed by the enterprise.

This, plus the daily health monitoring of the **AutoCert SCS** - and related **AutoCert** servers - all comes with a single annual flat fee.

Track Record

AutoCert is currently in use by a major Medicaid technology provider for one of the several state governments it contracts for.

The system consists of 104 domains and 139 servers. Some certificates in their non-production domains get turned over weekly while most get turned over monthly. Their non-load balanced production certificates get turned over every 60 days while their load balanced production certificates get turned over every 30 days, all completely automatically.

Solution Details

Daily check-ins

Each morning, during the enterprise defined maintenance window (typically between midnight and 4am), **AutoCert.exe** checks in with the **AutoCert SCS** certificate repository. That's how **AutoCert** knows the **AutoCert.exe** agent running on a particular server is still alive and well.

Enterprise application owners are notified if **AutoCert.exe** fails to check-in with the **SCS** 3 days in a row.

After the **SCS** check-in, **AutoCert.exe** interrogates the server's current certificate expiration status. If the certificate is not soon-to-expire (typically more than 30 days out), **AutoCert.exe** checks-out, exits and does nothing more - until the next day.

If **AutoCert.exe** determines that the server's current certificate ***is*** soon-to-expire, it shifts into high gear and proceeds with the "get certificate" process.

Get Certificate

First, **AutoCert.exe** checks for the availability of a new certificate on the **SCS**. If it's already there ***and*** its binding date is today (as determined by the **SCS**) then **AutoCert.exe** proceeds to install and bind the new certificate.

If the **SCS** informs **AutoCert.exe** that no new certificate is available, **AutoCert.exe** then proceeds to acquire a new certificate from **Let's Encrypt**. Once acquired, the new certificate is uploaded to the **SCS** for use on all the servers in the same farm. This occurs on the following day (or thereafter) to guarantee that all servers bind the new certificate at the same time (ie. on the "certificate binding date").

Send Certificate to Load Balancer

Assuming a load balancer is defined for the new certificate's domain, the day before the certificate binding date, the new certificate is transmitted to the load balancer for use there on the same certificate binding date. This transfer is handled by a PowerShell script snippet provided by the enterprise and called by **AutoCert.exe** at the appropriate time.

Bind Certificate (IIS)

On the binding date, each server in the server farm will first install the new certificate, then, by default, use the **AutoCert.exe** built-in **IIS** (Internet Information Services) binding procedure. That entails binding the new certificate to all the same IP addresses and ports already bound to the old certificate.

Bind Certificate (non-IIS)

If a particular server needs to use a non-standard binding procedure (eg. java binding), in addition to the standard **IIS** binding (or instead of), that option is available. The application owner needing the special binding simply supplies a PowerShell script snippet with all the coded binding details. **AutoCert.exe** will then run the given script snippet at the appropriate time during the overall binding process.

Archive Certificate

After a new certificate is successfully installed and bound, it can optionally be archived anywhere on the current server. Archived certificates are typically needed later for a new server added to the server farm. Archived certificates are created with a GUID based filename and written to any enterprise-defined location. A random password of at least 10 to 20 characters in length is used for the archive certificate's PFX file (a certificate archive password is used only once). It can be obtained from **AutoCert** as needed.[4]

HTTP Challenges

By default, the ACME HTTP-01 challenge protocol is used for proving domain name ownership to **Let's Encrypt**. That's the technique **AutoCert.exe** uses out-of-the-box.

The ACME HTTP-01 challenge requires the certificate domain name to be visible on the public Internet. Most enterprises have many internal (ie. non-public) domain names as well. To handle non-public (as well as public) domain names, the ACME DNS-01 challenge must be used instead.

DNS Challenges

Unlike HTTP challenges, DNS challenges require credentials to the enterprise's DNS provider account (eg. a DNS provider that provides an automation API, like DNSimple); this is accomplished via another PowerShell script snippet called by **AutoCert.exe**. This way **AutoCert** does not need to know enterprise DNS credentials.

AutoCert.exe Credentials

Regarding credentials, none are required for **AutoCert.exe** to communicate with the **AutoCert SCS** service. **AutoCert.exe** is secured using the same certificate used by the server's websites to identify itself to the **SCS**. That means once the **AutoCert.exe** agent software is installed and running, there is no risk that agent credentials or an agent security token can be compromised, or that they will expire or need updating – simply because no such credentials or tokens are needed.

Handling SSO Certificate Thumbprint Swaps

In addition to handling certificate changes for the server on which it is installed, **AutoCert.exe** also handles changing SSO certificate thumbprint references in website configuration files (any number of them) on the current server. This is done in tandem with certificate changes that occur on SSO servers elsewhere in the enterprise. This step is critical for SSO reliant web applications that simply will not function if they don't have the current SSO certificate thumbprint in their configuration.

Daily Backups and Log Files

As part of its daily processing, **AutoCert.exe** does extensive logging to easy-to-read dated text-based log files. The subsystem that runs **AutoCert.exe** every day also backs it up and cleans up its log files on a regular schedule.

Extensive Customization

In addition to the various customizable PowerShell script snippets that are available, almost every aspect of **AutoCert.exe**'s behavior can be controlled via a text-editable configuration file and\or command-line arguments and switches.

Executive Summary

In the recent past an enterprise could purchase a single “wildcard” digital certificate that would last from 2 to 5 years before expiring. Today a certificate will expire annually – if not sooner. Likewise, “wildcard” certificates are no longer considered safe to use on the Internet. That means a hundred or more certificates are now needed (potentially) - and much more frequently - when previously a single certificate would have sufficed for years at a time.

In 2013 the Internet Security Research Group (ISRG [5] – a consortium of major technology companies including Amazon, Google, IBM, and Meta - among others) established the ACME standard protocol for automated certificate management.

ISRG also started the **Let’s Encrypt** certificate authority to supply digital certificates worldwide – free of charge.

Let’s Encrypt makes it easy to automate the acquisition of a single certificate for a single server.

AutoCert takes a free certificate acquired from **Let’s Encrypt** and makes it available to any number of servers in a server farm - typically behind a load balancer.

The **AutoCert** agent (**AutoCert.exe**) is installed and configured on each server that requires a new digital certificate. Once installed it runs automatically on a daily basis during any enterprise defined maintenance window.

Everything previously handled manually during the "Whenever a certificate needed replacement..." procedure (see above) is now managed automatically by **AutoCert.exe**:

- Check the current certificate. If it’s not soon-to-expire, do nothing.
- Otherwise, download a new certificate from **Let’s Encrypt**.
- Upload the new certificate to the **AutoCert** repository (the **SCS**).
- Make the new certificate available to all servers in the farm.
- Also make the new certificate available to the load balancer.
- Securely archive the new certificate (for future use) on any local servers.
- Install the certificate on all servers while preventing user access to its private key.
- Bind the new certificate to each IP address and port bound to the old certificate.
- Finally, swap SSO certificate thumbprints - if the server is single-sign-on reliant.

One final important detail. The **AutoCert.exe** agent software requires no credentials on your servers. So, you won’t be trading the old problem of managing expiring digital certificates across your enterprise with a new problem of managing expiring tokens/passwords for agents running on hundreds of your servers. The same continually changing certificate that identifies your website also identifies the **AutoCert.exe** agent running on the same server.

Appendix

AutoCert.exe Configuration Options

```
-AcmeAccountFile='Account.xml'

    This is the name of the ACME account file returned by the certificate
    provider. If it's deleted, it will be replaced during the next run.

-AcmeAccountKeyFile='AccountKey.xml'

    This is the name of the ACME account key file returned by the
    certificate provider. If it's deleted, it will be replaced during
    the next run.

-AcmePsModuleUseGallery=False

    Set this switch True and the 'PowerShell Gallery' version of 'ACME-PS'
    will be used in lieu of the version embedded in the EXE (see -AcmePsPath
    below).

-AcmePsPath='ACME-PS'

    'ACME-PS' is the tool used by this utility to communicate with the
    "Let's Encrypt" certificate network. By default, this key is set to the 'ACME-PS'
    folder which, with no absolute path given, will be expected to be found within
    the folder containing 'AutoCert.exe.config'. Set -AcmePsModuleUseGallery=True
    (see above) and the OS will look to find 'ACME-PS' in its usual place as a
    module from the PowerShell gallery.

-AcmePsWorkPath='AcmeState'

    This is where temporary ACME state data is cached. It may be useful
    while troubleshooting issues. This data is automatically cleaned-up
    after each run, unless there is an error or unless -CleanupAcmeWork
    is set False (see below).

-AcmeSystemWide=''

    This is a script snippet that is prepended to every ACME script
    (see -ScriptStage1 below). This snippet may be useful to resolve
    certain issues that can arise during ACME script execution.

    Here's an example:

        -AcmeSystemWide='[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12'

        The above forces the use of the TLS 1.2 protocol with every ACME
        call. This is sometimes necessary when the network enforces rules
        that are not supported, by default, in older operating systems.

-AllowSsoThumbprintUpdatesAnytime=True

    Set this switch False to limit SSO thumbprint replacements to occur only
    during the domain defined maintenance window.

-Auto=False

    Set this switch True to run this utility one time (with no interactive UI)
    then shutdown automatically upon completion. This switch is useful if the
    software is run in a batch process or by a server job scheduler.
```

-CertificateDomainName= NO DEFAULT VALUE

    This is the subject name (ie. DNS name) of the certificate returned.

-CertificatePrivateKeyExportable=False

    Set this switch True (not recommended) to allow certificate private keys
    to be exportable from the local certificate store to a local disk file. Any
    SA with server access will then have access to the certificate's private key.

-CertificateRequestWaitSecs=10

    These are the seconds of wait time before a certificate request
    is submitted. This allows time for the order to be completed by
    the certificate provider network before a request can be accepted.

-CleanupAcmeWork=True

    Set this switch False to preserve temporary ACME state data, even after
    a successful run.

-ContactEmailAddress= NO DEFAULT VALUE

    This is the contact email address the certificate network uses to send
    certificate expiration notices.

-CreateSanSitesForCertGet=True

    If a SAN specific website does not yet exist in IIS, it will be created
    automatically during the first run of the 'get certificate' process for
    that SAN value. Set this switch False to have all SAN challenges routed
    through the IIS default website (such challenges will typically fail). If
    they do fail, you will need to create your SAN specific sites manually.

    Note: each SAN value must be challenged and therefore SAN challenges
          must be routed through your web server just like your primary domain.
          That means every SAN value must have a corresponding entry in the
          global internet DNS database.

          When a new website is created in IIS for a new SAN value, by default
          it is setup to use the same physical path as the primary domain.

-CreateSanSitesForBinding=False

    Set this switch True to have SAN sites created (if they don't already exist)
    during the certificate binding phase. The primary certificate name on the
    SAN list (see -CertificateDomainName above) will be bound to the default
    website as a last resort (if this value is False).

-DefaultPhysicalPath='%SystemDrive%\inetpub\wwwroot'

    This is the default physical storage path used during the creation of
    new IIS websites that can't otherwise be associated with an existing
    site (see -CreateSanSitesFor* above).

-DoStagingTests=True

    Initial testing is done with the certificate provider staging network.
    Set this switch False to use the live production certificate network.

-FetchSource=False

    Set this switch True to fetch the source code for this utility from the EXE.
    Look in the containing folder for a ZIP file with the full project sources.

-Help= SEE PROFILE FOR DEFAULT VALUE

    This help text.

-KillProcessForcedWaitMS=1000

    This is the maximum milliseconds to wait while force killing a process.

-KillProcessOrderlyWaitSecs=10

    This is the maximum number of seconds given to a process after a 'close'
    command is given before the process is forcibly terminated.

-LoadBalancerReleaseCert

    This switch indicates the new certificate has been released by the load
    balancer administrator or process.

    This switch would typically never appear in a profile file. It's meant to be
    used on the command-line only (in a script or a shortcut).

    Note: this switch is ignored when -UseStandAloneMode is True.

-LogCertificateStatus=False

    Set this switch True to see in detail how the AutoCert.exe app's identifying
    certificate is selected. This may be useful to troubleshoot setup issues.

    Note: this switch is ignored when -UseStandAloneMode is True.

-LogEntryDateTimeFormatPrefix='yyyy-MM-dd hh:mm:ss:fff tt '

    This format string is used to prepend a timestamp prefix to each log entry
    in the process log file (see -LogPathFile below).

-LogFileDateFormat='-yyyy-MM-dd'

    This format string is used to form the variable part of each log file output
    filename (see -LogPathFile below). It is inserted between the filename and
    the extension.

-LogPathFile='Logs\Log.txt'

    This is the output path\file that will contain the process log. The profile
    filename will be prepended to the default filename (see above) and the current
    date (see -LogFileDateFormat above) will be inserted between the filename and
    the extension.

-MaxCertRenewalLockDelaySecs=300

    Wait a random period each cycle (at most this many seconds) to allow different
    clients the opportunity to lock the certificate renewal (ie. only one client
    at a time per domain can communicate with the certificate provider network).

-NonIISBindingScript= SEE PROFILE FOR DEFAULT VALUE

    This is the PowerShell script that binds a new certificate when IIS is not in
    use or the standard IIS binding procedure does not work for whatever reason.

-NoPrompts=False

    Set this switch True and all pop-up prompts will be suppressed. Messages
    are written to the log instead (see -LogPathFile above). You must use this
    switch whenever the software is run via a server computer batch job or job
    scheduler (ie. where no user interaction is permitted).

-PowerScriptPathFile=PowerScript.ps1

    This is the path\file location of the current (ie. temporary) PowerShell script
    file.

-PowerScriptSleepMS=200

    This is the number of sleep milliseconds between loops while waiting for the
    PowerShell script process to complete.

-PowerScriptTimeoutSecs=300

    This is the maximum number of seconds allocated to any PowerShell script
    process to run prior to throwing a timeout exception.

-PowerShellExeArgs=-NoProfile -ExecutionPolicy unrestricted -File "{0}" "{1}"

    These are the arguments passed to the Windows PowerShell EXE (see below).

-PowerShellExePathFile=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

    This is the path\file location of the Windows PowerShell EXE.

-RegexDnsNamePrimary='^([a-zA-Z0-9\-]+\.)+[a-zA-Z]{2,18}$'

    This regular expression is used to validate -CertificateDomainName (see above).

-RegexDnsNameSanList='^([a-zA-Z0-9\-]+\.)*([a-zA-Z0-9\-]+\.)([a-zA-Z]{2,18}|)$'

    This regular expression is used to validate -SanList names (see below).

    Note: -RegexDnsNameSanList and -RegexDnsNamePrimary are both used to validate
          SAN list names. A match of either pattern will pass validation.

-RegexEmailAddress='^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9\-]+\.)*([a-zA-Z0-9\-]+\.)[a-zA-Z]{2,18}$'

    This regular expression is used to validate -ContactEmailAddress (see above).

-RemoveReplacedCert=False

    Set this switch True and the old (ie. previously bound) certificate will be
    removed whenever a newly retrieved certificate is bound to replace it.

    Note: this switch is ignored when -UseStandAloneMode is False.

-RenewalDateOverride= NO DEFAULT VALUE

    Set this date value to override the date calculation that subtracts
    -RenewalDaysBeforeExpiration days (see below) from the current certificate
    expiration date to know when to start fetching a new certificate.

    Note: this parameter is ignored when -UseStandAloneMode is False. It will
          be removed from the profile after a certificate has been successfully
          retrieved from the certificate provider network.

-RenewalDaysBeforeExpiration=30

    This is the number of days until certificate expiration before automated
    gets of the next new certificate are attempted.

-ResetStagingLogs=True

    Having to wade through several log sessions during testing can be cumbersome.
    So the default behavior is to clear the log file after it is uploaded to the SCS
    server (following each test). Setting this switch False will retain all previous
    log sessions on the client during testing.

    Note: this switch is ignored when -UseStandAloneMode is True.

-RetainNewCertAfterError=False

    Set this switch True to prevent removal of new certificates after errors. A new
    certificate should typically not persist after a failure. This prevents multiple
    versions of essentially the 'same' certificate from piling up.

    Note: this parameter will be removed from the profile after every run
          (whether used or not). If you need several uses, it might make more
          sense to pass this parameter as a command-line argument instead.

-SanList= SEE PROFILE FOR DEFAULT VALUE

    This is the SAN list (subject alternative names) to appear on the certificate
    when it is generated. It will consist of -CertificateDomainName by default (see
    above). This list can be edited here directly or through the 'SAN List' button
    in the AutoCert.exe UI. Click the 'SAN List' button to see the proper format here.

    Here's a command-line example:

    -SanList="-Domain='MyDomain.com' -Domain='www.MyDomain.com'"

    Note: the AutoCert.exe UI limits you to 100 SAN values (the certificate provider
          does the same). If you add more names than this limit, an error results and
          no certificate will be generated.

-SaveProfile=True

    Set this switch False to prevent saving to the profile file by this utility
    software itself. This is not recommended since process status information is
    written to the profile after each run.

-SaveSansCmdLine=True

    Set this switch False to allow merged command-lines to be written to
    the profile file (ie. 'AutoCert.exe.config'). When True, everything
    but command-line keys will be saved.

-ScriptBindingDone=''

    This is the PowerShell script snippet that handles any final processing after
    the successful binding of a new certificate.

-ScriptCertAcquired=''

    This is the PowerShell script snippet that handles any final processing after
    the successful acquisition of a new certificate from the certificate provider.

-ScriptSSO= SEE PROFILE FOR DEFAULT VALUE

    This is the PowerShell script that updates SSO servers with new certificates.

-ScriptStage1= SEE PROFILE FOR DEFAULT VALUE

    There are multiple stages involved with the process of getting a certificate
    from the certificate provider network. Each stage has an associated PowerShell
    script snippet. The stages are represented in this profile by -ScriptStage1
    through -ScriptStage7.

-ServiceNameLive='LetsEncrypt'

    This is the name mapped to the live production certificate network service URL.

-ServiceNameStaging='LetsEncrypt-Staging'

    This is the name mapped to the non-production (ie. 'staging') certificate
    network service URL.

-SetDefaultBinding=True

    Set this switch False to prevent even a default binding from being applied
    after no binding is found referencing the 'old' certificate.

-Setup=False

    Set this switch True to ignore the usual constraints during the acquisition
    of new certificates. For example, you can get a new certificate even if the
    old one is not about to expire, like when the software is run in interactive
    mode (eg. during initial setup).

    This switch is typically passed as a command-line argument:

        AutoCert.exe -Auto -Setup

-ShowProfile=False

    Set this switch True to immediately display the entire contents of the profile
    file at startup in command-line format. This may be helpful as a diagnostic.

-SingleSessionEnabled=False

    Set this switch True to run all PowerShell scripts in a single global session.

-SkipPreviousStore=False

    The 'AutoCertPrevious' certificate store is used to house the previously bound
    'old' certificate (ie. one most recently removed from the 'Personal' store).
    Set this switch False to simply delete old certificates directly.

-SkipSsoServer=False

    When a client is on an SSO domain (ie. it's a member of an SSO server farm),
    it will automatically attempt to update SSO services with each new certificate
    retrieved. Set this switch True to disable SSO updates (for this client only).

-SkipSsoThumbprintUpdates=False

    When a client's domain references an SSO domain, the client will automatically
    attempt to update configuration files with each new SSO certificate thumbprint.
    Set this switch True to disable SSO thumbprint configuration updates (for this
    client only). See -SsoThumbprintFiles below.

-SsoMaxRenewalSleepSecs=60

    This is the maximum seconds an SSO server will wait for another
    SSO server (in the same farm) to renew the SSO certificates.

-SsoProxySleepSecs=60

    An SSO proxy server will wait this many seconds before
    polling the SSO server again for a certificate change.

-SsoProxyTimeoutMins=30

    An SSO proxy server will wait this many minutes for the SSO server
    to change its certificates before throwing a timeout exception.

-SsoThumbprintFiles='C:\inetpub\wwwroot\web.config'

    This is the path and filename of files that will have their SSO certificate
    thumbprint replaced whenever the related SSO certificate changes. Each file
    with the same name at all levels of the directory hierarchy will be updated,
    starting with the given base path, if the old SSO certificate thumbprint is
    found there. See -SkipSsoThumbprintUpdates above.

    Note: This key may appear any number of times in the profile and wildcards
          can be used in the filename.

-SsoThumbprintReplacementArgs=''

    Whatever is provided here will be passed to the SSO thumbprint replacement
    sub-process during each run. This makes it easier to adjust the SSO thumbprint
    replacement sub-process without having to edit its configuration separately.

```
-SubmissionRetries=42

    Pending submissions to the certificate provider network will be retried until
    they succeed or fail, at most this many times. By default, the  process will
    retry for 7 minutes (-SubmissionRetries times -SubmissionWaitSecs, see below)
    for challenge status updates as well as certificate request status updates.

-SubmissionWaitSecs=5

    These are the seconds of wait time after an ACME challenge request has been
    submitted to the certificate network as well as after a certificate request
    has been submitted. This is the amount of time during which the request should
    transition from a 'pending' state to anything other than 'pending'.

-UseNonIISBindingAlso=False

    Set this switch True to use the typical IIS binding procedures
    on this machine as well as the -NonIISBindingScript (see above).

-UseNonIISBindingOnly=False

    Set this switch True to disable the usual IIS binding procedures on
    this machine and use the -NonIISBindingScript instead (see above).

-UseStandAloneMode=True

    Set this switch False and the software will use the AutoCert Secure Certificate
    Service (see 'AutoCert.com') to manage certificates between several servers
    in a server farm, on SSO servers, SSO integrated application servers and load
    balancers.
```

Endnotes

1. **AutoCert.exe** can install and bind new certificates on remote Linux servers via SSH calls (see PowerShell remoting over SSH).

2. https://www.encryptionconsulting.com/consequences-of-expired-digital-certificate-extend-to-tech-titans-microsoft-and-cisco/

3. A customer supplied VM is the most cost-effective approach. Alternatively, an **SCS** appliance VM - dedicated to the enterprise (ie. its certificate repository) - can be hosted and managed by **AutoCert** within its own cloud of **SCS** VMs.

4. Random certificate passwords are intentionally exceedingly difficult to obtain. A small fee is required for such requests which must go through a strict security protocol to be fulfilled.

5. ISRG - https://www.abetterinternet.org/about/